DAV

DEUTSCHE
AKTUARVEREINIGUNG e.V.

Praktische Prüfung im Vertiefungswissen

Actuarial Data Science Immersion

gemäß Prüfungsordnung 4
der Deutschen Aktuarvereinigung e. V.

Zeitraum: 17.04.2023 - 17.05.2023

Hinweise:

- Die Gesamtpunktzahl beträgt 180 Punkte. Die Prüfung ist bestanden, wenn mindestens 90 Punkte erreicht werden.
- Bitte prüfen Sie die Ihnen vorliegenden Prüfungsunterlagen auf Vollständigkeit. Die Prüfung besteht aus den zwei Blöcken A und B und beinhaltet insgesamt 18 Seiten. Zusätzliche Materialien (Datensätze, Datenbeschreibung, Anhang 1 und 2 zu Block A) sollten Sie mit der Prüfung erhalten haben. Auf diese wird in den Blöcken eingegangen.
- Für jeden Block ist ein PDF-Dokument einzureichen. Falls ein Notebook zu erstellen ist, dann ist dieses zusätzlich einzureichen. Der Output erstellter Notebooks muss mindestens im eingereichten PDF-Dokument, falls möglich auch im Notebook sichtbar sein. Aufgabenteile mit erforderlichem, aber fehlendem Output werden mit 0 Punkten bewertet.
- Für jeden Block sind die Aufgaben in der vorgegebenen Reihenfolge zu bearbeiten. Die Lösung muss direkt nach der Aufgabenstellung in einer strukturierten und übersichtlichen Weise dargestellt werden. Nichtbeachtung dieser Vorgabe führt zu Punkteabzug.
- Mit der Einreichung von Ergebnis-PDF-Dokument und -Notebook erklärt die zu prüfende Person, dass der eingereichte Ergebnisbericht inklusive Code- und Beschreibungsanteilen eigenständig erzeugt worden ist. Verstöße gegen diese Grundlagen können vom Prüfungsausschuss sanktioniert werden und zum Ausschluss von der Prüfung führen.

Lösungsvorschlag zur Prüfungsaufgabe Immersion, Block A

Analyse von Kreditrisiken [75 Punkte]

Aufgabe A-0: Einbinden von Modulen, Packages, Funktionen etc. & Reproduzierbarkeit [Lernziel 2.2, 3.3, 3.4; 4 Punkte]

Alle für die weiteren Aufgaben benötigten Module, Packages, (selbstgeschriebene) Funktionen etc. sind zentral an dieser Stelle einzubinden.

Erläutern Sie zwei Maßnahmen, welche sinnvoll sind für die Reproduzierbarkeit des Notebooks, in jeweils wenigen Sätzen. Diese Maßnahmen sind im weiteren Verlauf der Prüfung umzusetzen.

Lösungsvorschlag:

Zunächst werden alle für die nachfolgenden Aufgaben benötigten Bibliotheken eingeladen.

In [1]:
```
# Benötigte Bibliotheken einbinden

import time

import pandas as pd
pd.set_option('display.max_rows', 25)
pd.set_option('display.max_columns', None)

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
```

```
import warnings
warnings.filterwarnings('ignore')
```

Es folgen die Funktionen, die im weiteren Verlauf des Notebooks angewendet werden.

In [2]:
```
# Die Funktion `nan_df_create` erzeugt für einen gegebenen Dataframe einen weiteren Dataframe,
# welcher aus allen Spaltennamen und deren (prozentualen) Anteilen an fehlenden Werten besteht,
# wobei nur Spaltennamen mit mindestens einem fehlenden Wert aufgenommen werden.
def nan_df_create(data):
    nan_percentages = data.isna().sum() * 100 / len(data)
    df = pd.DataFrame({'Spalte' : nan_percentages.index, 'Prozent' : nan_percentages.values})
    df = df[df['Prozent'] > 0]
    df.sort_values(by = 'Prozent', ascending = False, inplace = True)
    return df
```

In [3]:
```
# Die Funktion `plot_nan_percent` erzeugt für einen gegebenen Dataframe einen Barplot bestehend aus
# allen Spaltennamen mit fehlenden Werten und dem jeweiligen prozentualen Anteil an fehlenden Werten.
def plot_nan_percent(df_nan, title_name, tight_layout = True, figsize = (20,8), grid = False, rotation = 90):
    if df_nan.Prozent.sum() != 0:
        print(f"Anzahl der Spalten mit NaNs: {df_nan[df_nan['Prozent'] != 0].shape[0]} Spalten")
        plt.figure(figsize = figsize, tight_layout = tight_layout)
        sns.barplot(x= 'Spalte', y = 'Prozent', data = df_nan[df_nan['Prozent'] > 0])
        plt.xticks(rotation = rotation)
        plt.xlabel('Spaltenname')
        plt.ylabel('Prozentanteil der NaN')
        plt.title(f'prozentualer Anteil der NaN in {title_name}')
        if grid:
            plt.grid()
        plt.show()
    else:
        print(f"Der Datensatz {title_name} enthält keine NaNs.")
```

In [4]:
```
# Die Funktion `plot_continuous_variables` erzeugt für eine gegebene numerische Spalte eines gegebenen
# Dataframes einen Plot bestehend aus zwei Kerndichteschätzern, je einen für die Werte der Spalte, für
# welche der `TARGET`-Wert des Dataframes gleich 0 sind, und einen für die Werte der Spalte, für welche
# der `TARGET`-Wert des Dataframes gleich 1 ist.
def plot_continuous_variables(data, column_name, figsize = (20,8)):
    data_to_plot = data.copy()
    number_of_subplots = 2
    plt.figure(figsize = figsize)
    sns.set_style('whitegrid')
    sns.distplot(data_to_plot[column_name][data['TARGET'] == 0].dropna(), label='Non-Defaulters', color='red')
    sns.distplot(data_to_plot[column_name][data['TARGET'] == 1].dropna(), label='Defaulters', color='black')
    plt.xlabel(column_name)
    plt.ylabel('Probability Density')
    plt.legend(fontsize='large')
    plt.title("Dist-Plot of {}".format(column_name))
    plt.show()
```

In [5]:
```
# Die Funktion `print_column_datatypes` gibt für einen gegebenen Dataframe die Anzahl an Spalten mit
# numerischen Werten, die Anzahl an Spalten mit kategoriellen Werten sowie die Anzahl an Spalten insgesamt aus.
def print_column_datatypes(data):
    kategoriell = ['category']
    numerisch = ['int16', 'int32', 'int64', 'float16','float32','float64']
    print("Spalten mit numerischen Werten: " + str(len(data.select_dtypes(include=numerisch).columns.tolist())))
    print("Spalten mit kategoriellen Werten: " + str(len(data.select_dtypes(include=kategoriell).columns.tolist())))
    print("Insgesamt " + str(len(data.select_dtypes(include=numerisch).columns.tolist()) +
                             len(data.select_dtypes(include=kategoriell).columns.tolist())) + " Spalten")
```

In [6]:
```
# Die Funktion `plot_cat` erzeugt für einen gegebenen Dataframe für jedes kategorielle Merkmal des
# Dataframes eine gemäß den Werten einer gegebenen Zielspalte gestapelte Säulengrafik.
def plot_cat(data, target, n_cols, n_rows, title):
    cols = data.select_dtypes(include='category').columns.tolist()
    fg,ax = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=(n_rows*7, n_cols*5), squeeze=False)
    plt.suptitle(title, fontsize=28, y=0.95)
    j=0;
    i=0;

    for col in cols:
        data.groupby([col, target]).size().unstack().fillna(0).plot(kind='bar', stacked=True, color=['steelblue', 'red'], ax=ax[i][j])
        i += 1
        if(i>=n_rows):
            i = 0
            j += 1

    # Lösche Axen die nicht benötigt werden (hier nur für den Spezialfall dass die letzte Spalte leere Plots enthält)
    while (i<n_rows):
        fg.delaxes(ax[i,j])
        i += 1
```

In [7]:
```
# Die Funktion `plot_cat_proz` erzeugt für einen gegebenen Dataframe für jedes kategorielle Merkmal des
# Dataframes eine gemäß den Werten einer gegebenen Zielspalte gestapelte Säulengrafik, die sich je
# Ausprägung des kategoriellen Merkmals zu 100% ergänzen.
def plot_cat_proz(data, target, n_cols, n_rows, title):
    cols = data.select_dtypes(include='category').columns.tolist()
    fg,ax = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=(n_rows*7, n_cols*5), squeeze=False)
    plt.suptitle(title, fontsize=28, y=0.95)
    j=0;
    i=0;

    for col in cols:
        (data.groupby([col, target]).size()/data.groupby([col]).size()).unstack().fillna(0).plot(kind='bar', stacked=True,
                                                                                                 color=['steelblue', 'red'], ax=ax[i][j])
```

```
        i += 1
        if(i>=n_rows):
            i = 0
            j += 1

    # Lösche Axen die nicht benötigt werden (hier nur für den Spezialfall dass die letzte Spalte leere Plots enthält)
    while (i<n_rows):
        fg.delaxes(ax[i,j])
        i += 1
```

In [8]:
```
# Die Funktion `plot_num` erzeugt für einen gegebenen Dataframe für jedes numerische Merkmal des
# Dataframes einen gemäß den Werten einer gegebenen Zielspalte unterschiedene Kerndichteschätzer-Plot.
def plot_num(data, target, n_cols,n_rows, title):
    cols = data.select_dtypes(exclude='category').columns.tolist()
    cols.remove('SK_ID_CURR')
    fg,ax = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=(3*n_rows, 18*n_cols), squeeze=False)
    plt.suptitle(title, fontsize=18, y=0.95)
    j=0;
    i=0;
    for col in cols:
        sns.kdeplot(data.loc[data['TARGET'] == 1, col], label='target == 1', color='r', ax=ax[i,j])
        sns.kdeplot(data.loc[data['TARGET'] == 0, col], label='target == 0', color='b', ax=ax[i,j])
        i += 1
        if(i>=n_rows):
            i = 0
            j += 1
```

Die folgende Funktion berechnet den Modus einer Spalte, wie in Aufgabe A-4 benötigt.

In [9]:
```
# Die Funktion `modus` berechnet den Modus (= am häufigsten auftretender Wert) von eingegebenen Werten
def modus(x):
    m = pd.Series.mode(x);
    return m.values[0] if not m.empty else np.nan
```

Die folgende Funktion (aus dem Anhang zur Prüfungsaufgabe) dient zur Berechnung der Information Values.

In [10]:
```
# Die Funktion `iv_woe` berechnet für einen gebenen Dataframe und die vorgegebene Zielspalte
# dieses Dataframes die Information Values sowie die Weights of Evidence.
def iv_woe(data, target, bins=10, show_woe=False):

    # erzeuge leere Dataframes
    newDF,woeDF = pd.DataFrame(), pd.DataFrame()

    # extrahiere die Spaltennamen des Input-Dataframes
    cols = data.columns

    # berechne die Information Values sowie die Weigths of Evidence
    for ivars in cols[~cols.isin([target])]:
        if (data[ivars].dtype.kind in 'bifc') and (len(np.unique(data[ivars]))>10):
            binned_x = pd.qcut(data[ivars], bins,  duplicates='drop')
            d0 = pd.DataFrame({'x': binned_x, 'y': data[target]})
        else:
            d0 = pd.DataFrame({'x': data[ivars], 'y': data[target]})
        d0 = d0.astype({"x": str})
        d = d0.groupby("x", as_index=False, dropna=False).agg({"y": ["count", "sum"]})
        d.columns = ['Cutoff', 'N', 'Events']
        d['% of Events'] = np.maximum(d['Events'], 0.5) / d['Events'].sum()
        d['Non-Events'] = d['N'] - d['Events']
        d['% of Non-Events'] = np.maximum(d['Non-Events'], 0.5) / d['Non-Events'].sum()
        d['WoE'] = np.log(d['% of Non-Events']/d['% of Events'])
        d['IV'] = d['WoE'] * (d['% of Non-Events']-d['% of Events'])
        d.insert(loc=0, column='Variable', value=ivars)
        #print("Information value of " + ivars + " is " + str(round(d['IV'].sum(),6)))
        temp =pd.DataFrame({"Variable" : [ivars], "IV" : [d['IV'].sum()]}, columns = ["Variable", "IV"])
        newDF=pd.concat([newDF,temp], axis=0)
        woeDF=pd.concat([woeDF,d], axis=0)

        # falls beim Aufruf der Funktion angegeben, werden die Weights of Evidence ausgegeben
        if show_woe == True:
            print(d)

    # gebe die Dataframes zu den Information Values und den Weights of Evidence zurück
    return newDF, woeDF
```

Bezüglich der Reproduzierbarkeit des Notebook wären folgende Punkte zu nennen:

- Setzen eines Seeds im Falle des Verwendens von Zufallszahlen
- Angabe der verwendeten System-/Bibliotheks-/Sprachversionen
- Weitere sinnvolle Antworten sind möglich

Aufgabe A-1: Einlesen und Aufbereiten der Datensätze [Lernziel 3.4; 5 Punkte]

Die Datensätze `application_train.csv`, `previous_application.csv`, `bureau.csv`, `application_test.csv` und `application_test_target.csv` sind passend einzulesen.

Für die weitere Verarbeitung sind die Datensätze aufzubereiten. Dafür sind die folgenden Schritte auszuführen:

- **Die Informationen zu den Testdaten sind über die beiden Dateien `application_test.csv` und `application_test_target.csv` verteilt. Diese sind mit einer geeigneten Methode zusammenzufügen. Der resultierende Datensatz ist mit einer zusätzlichen Spalte TYPE mit Inhalt 'TEST' zu versehen.**
- **Der Trainingsdatensatz `application_train` ist analog mit einer Spalte TYPE mit Inhalt 'TRAIN' zu versehen.**
- **Beide aus den vorherigen Schritten erstellten Datensätze sind zu einem gemeinsamen Datensatz zusammenzufassen.**

Lösungsvorschlag:

Zum Einlesen der drei Datensätze verwenden wir die Funktion `read_csv` des `pandas`-Packages.

In [11]:
```
# Anmerkung: csv ist mit Dezimaltrennzeichen Komma und Trennzeichen Semikolon abgespeichert.
# Dies muss beim Einlesen berücksichtigt werden
dfapptrain_raw      = pd.read_csv('../../1_Datensatz/application_train.csv',       delimiter=';', decimal=',')
dfprevapp_raw       = pd.read_csv('../../1_Datensatz/previous_application.csv',    delimiter=';', decimal=',')
dfbureau_raw        = pd.read_csv('../../1_Datensatz/bureau.csv',                  delimiter=';', decimal=',')
dfapptest_raw       = pd.read_csv('../../1_Datensatz/application_test.csv',        delimiter=';', decimal=',')
dfapptesttarget_raw = pd.read_csv('../../1_Datensatz/application_test_target.csv', delimiter=';', decimal=',')
```

In [12]:
```
dfapptrain_raw.shape
```

Out[12]:
```
(160440, 49)
```

In [13]:
```
dfapptest_raw.shape
```

Out[13]:
```
(30856, 48)
```

Der Trainingsdatensatz umfasst 160.440 Zeilen und 49 Spalten, der Testdatensatz enthält 30.856 Zeilen und 48 Spalten. Die Differenz in der Spaltenanzahl ist die Zielvariable TARGET, die im Testdatensatz nicht enthalten ist.

In [14]:
```
dfapptesttarget_raw_tmp = pd.merge(dfapptest_raw, dfapptesttarget_raw, on = "SK_ID_CURR", how = "left")
dfapptesttarget_raw_tmp['TYPE'] = 'TEST'

dfapptrain_raw_tmp = dfapptrain_raw.copy()
dfapptrain_raw_tmp['TYPE'] = 'TRAIN'

dfapptraintest_raw = pd.concat([dfapptesttarget_raw_tmp, dfapptrain_raw_tmp])
```

Aufgabe A-2: Analysieren der Datensätze [Lernziel 3.4; 10 Punkte]

Folgende Tätigkeiten sind für den kombinierten Datensatz mit Trainings- und Testdaten aus der vorherigen Aufgabe sowie für die Datensätze `previous_application` und `bureau` jeweils durchzuführen:

- **Zu bestimmen sind die Anzahl an Zeilen und Spalten des Datensatzes. Die ersten fünf Zeilen des Datensatzes sind auszugeben.**
- **Die Datentypen der Spalten sind auf Korrektheit zu prüfen. Die Anzahl an Spalten mit numerischen sowie die Anzahl an Spalten mit kategoriellen Werten ist anzugeben. Dabei ist zu berücksichtigen, dass alle Spalten mit 10 oder weniger Wertausprägungen als kategoriell gelten (Ausnahmen: Die Spalten beginnend mit `AMT_REQ_CREDIT_BUREAU` sowie die Spalte `CNT_CREDIT_PROLONG` bleiben numerisch und `OCCUPATION_TYPE` ist als kategoriell zu werten). Darüber hinaus sind Spalten mit Textcodierung unabhängig von der Anzahl der Wertausprägungen als kategoriell anzusehen. Künstlich hinzugefügte Spalten sollen dabei nicht berücksichtigt werden.**
- **Eine Tabelle bestehend aus den Spaltennamen und dem jeweiligen Anteil an fehlenden Werten ist auszugeben. Die Ergebnisse sind mit einem Barplot zu visualisieren. Bei dem kombinierten Datensatz aus Test- und Trainingsdaten soll diese Auswertung nur auf den Trainingsdaten erfolgen.**

Was kann für die Datensätze `previous_application` und `bureau` bzgl. der Spalte `SK_ID_CURR` ausgesagt werden?

Lösungsvorschlag:

Kombinierter Datensatz aus Trainings- und Testdaten

In [15]:
```
dfapptraintest_raw.shape
```

Out[15]:
```
(191296, 50)
```

In [16]:
```
print(dfapptraintest_raw.head())
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
0	200004	C	M	N	Y
1	200011	C	F	N	Y
2	200015	C	M	Y	N
3	200026	C	F	N	Y
4	200028	C	M	N	Y

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
0	1	405	204	22.0	180.0
1	0	430	1186	61.0	900.0
2	0	540	830	46.0	594.0
3	0	270	1279	55.0	1017.0
4	1	540	384	41.0	360.0

	NAME_TYPE_SUITE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS
0	U	C	H	M
1	U	C	S	C
2	U	W	I	C
3	U	P	S	W
4	F	C	H	M

	NAME_HOUSING_TYPE	REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_EMPLOYED
0	H	0.046	-10525	-621
1	H	0.020	-19606	-1782
2	H	0.023	-11057	-668
3	H	0.019	-23307	365243
4	H	0.046	-10503	-895

	DAYS_REGISTRATION	DAYS_ID_PUBLISH	OWN_CAR_AGE	FLAG_MOBIL
0	-612	-3039	NaN	1
1	-5601	-3166	NaN	1
2	-408	-3738	1.0	1
3	-12897	-4624	NaN	1
4	-4419	-566	NaN	1

	FLAG_EMP_PHONE	FLAG_WORK_PHONE	FLAG_CONT_MOBILE	FLAG_PHONE	FLAG_EMAIL
0	1	0	1	0	0
1	1	0	1	0	0
2	1	1	1	0	0
3	0	0	1	1	1
4	1	0	1	0	0

	OCCUPATION_TYPE	CNT_FAM_MEMBERS	REGION_RATING_CLIENT
0	H	3.0	1
1	C	2.0	3
2	L	2.0	2
3	NaN	1.0	2
4	NaN	3.0	1

	REGION_RATING_CLIENT_W_CITY	REG_REGION_NOT_LIVE_REGION
0	1	0
1	3	0
2	2	0
3	2	0
4	1	0

	REG_REGION_NOT_WORK_REGION	LIVE_REGION_NOT_WORK_REGION
0	0	0
1	0	0
2	1	1
3	0	0
4	0	0

	REG_CITY_NOT_LIVE_CITY	REG_CITY_NOT_WORK_CITY	LIVE_CITY_NOT_WORK_CITY
0	0	0	0
1	0	0	0
2	1	1	1
3	0	0	0
4	0	1	1

	ORGANIZATION_TYPE	EXT_SOURCE_1	EXT_SOURCE_2	EXT_SOURCE_3
0	B	0.605	0.631	0.671
1	S	NaN	0.466	0.351
2	I	NaN	0.225	0.354
3	X	NaN	0.764	0.703
4	B	NaN	0.286	0.281

	DAYS_LAST_PHONE_CHANGE	AMT_REQ_CREDIT_BUREAU_HOUR
0	0.0	0.0
1	-437.0	0.0
2	-972.0	0.0
3	-1706.0	0.0
4	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_WEEK
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_MON	AMT_REQ_CREDIT_BUREAU_QRT
0	0.0	0.0
1	0.0	0.0
2	0.0	2.0
3	0.0	0.0

```
4                       0.0                       0.0

   AMT_REQ_CREDIT_BUREAU_YEAR  TARGET  TYPE
0                         2.0       0  TEST
1                         0.0       0  TEST
2                         2.0       0  TEST
3                         1.0       0  TEST
4                         1.0       0  TEST
```

In [17]:
```
dfapptraintest_raw.dtypes
```

Out[17]:
```
SK_ID_CURR                    int64
NAME_CONTRACT_TYPE           object
CODE_GENDER                  object
FLAG_OWN_CAR                 object
FLAG_OWN_REALTY              object
                              ...
AMT_REQ_CREDIT_BUREAU_MON   float64
AMT_REQ_CREDIT_BUREAU_QRT   float64
AMT_REQ_CREDIT_BUREAU_YEAR  float64
TARGET                        int64
TYPE                         object
Length: 50, dtype: object
```

In [18]:
```
for col in dfapptraintest_raw.columns:
    print(f"Spalte: {col} Wertausprägungen: {len(dfapptraintest_raw.loc[:,col].unique())}")
```

```
Spalte: SK_ID_CURR Wertausprägungen: 191296
Spalte: NAME_CONTRACT_TYPE Wertausprägungen: 2
Spalte: CODE_GENDER Wertausprägungen: 2
Spalte: FLAG_OWN_CAR Wertausprägungen: 2
Spalte: FLAG_OWN_REALTY Wertausprägungen: 2
Spalte: CNT_CHILDREN Wertausprägungen: 13
Spalte: AMT_INCOME_TOTAL Wertausprägungen: 783
Spalte: AMT_CREDIT Wertausprägungen: 2742
Spalte: AMT_ANNUITY Wertausprägungen: 285
Spalte: AMT_GOODS_PRICE Wertausprägungen: 714
Spalte: NAME_TYPE_SUITE Wertausprägungen: 7
Spalte: NAME_INCOME_TYPE Wertausprägungen: 7
Spalte: NAME_EDUCATION_TYPE Wertausprägungen: 5
Spalte: NAME_FAMILY_STATUS Wertausprägungen: 5
Spalte: NAME_HOUSING_TYPE Wertausprägungen: 6
Spalte: REGION_POPULATION_RELATIVE Wertausprägungen: 29
Spalte: DAYS_BIRTH Wertausprägungen: 17359
Spalte: DAYS_EMPLOYED Wertausprägungen: 11136
Spalte: DAYS_REGISTRATION Wertausprägungen: 15056
Spalte: DAYS_ID_PUBLISH Wertausprägungen: 6069
Spalte: OWN_CAR_AGE Wertausprägungen: 59
Spalte: FLAG_MOBIL Wertausprägungen: 2
Spalte: FLAG_EMP_PHONE Wertausprägungen: 2
Spalte: FLAG_WORK_PHONE Wertausprägungen: 2
Spalte: FLAG_CONT_MOBILE Wertausprägungen: 2
Spalte: FLAG_PHONE Wertausprägungen: 2
Spalte: FLAG_EMAIL Wertausprägungen: 2
Spalte: OCCUPATION_TYPE Wertausprägungen: 12
Spalte: CNT_FAM_MEMBERS Wertausprägungen: 16
Spalte: REGION_RATING_CLIENT Wertausprägungen: 3
Spalte: REGION_RATING_CLIENT_W_CITY Wertausprägungen: 3
Spalte: REG_REGION_NOT_LIVE_REGION Wertausprägungen: 2
Spalte: REG_REGION_NOT_WORK_REGION Wertausprägungen: 2
Spalte: LIVE_REGION_NOT_WORK_REGION Wertausprägungen: 2
Spalte: REG_CITY_NOT_LIVE_CITY Wertausprägungen: 2
Spalte: REG_CITY_NOT_WORK_CITY Wertausprägungen: 2
Spalte: LIVE_CITY_NOT_WORK_CITY Wertausprägungen: 2
Spalte: ORGANIZATION_TYPE Wertausprägungen: 17
Spalte: EXT_SOURCE_1 Wertausprägungen: 935
Spalte: EXT_SOURCE_2 Wertausprägungen: 824
Spalte: EXT_SOURCE_3 Wertausprägungen: 646
Spalte: DAYS_LAST_PHONE_CHANGE Wertausprägungen: 3720
Spalte: AMT_REQ_CREDIT_BUREAU_HOUR Wertausprägungen: 5
Spalte: AMT_REQ_CREDIT_BUREAU_DAY Wertausprägungen: 10
Spalte: AMT_REQ_CREDIT_BUREAU_WEEK Wertausprägungen: 10
Spalte: AMT_REQ_CREDIT_BUREAU_MON Wertausprägungen: 21
Spalte: AMT_REQ_CREDIT_BUREAU_QRT Wertausprägungen: 12
Spalte: AMT_REQ_CREDIT_BUREAU_YEAR Wertausprägungen: 12
Spalte: TARGET Wertausprägungen: 2
Spalte: TYPE Wertausprägungen: 2
```

In [19]:
```
# Anpassungen an den Datentypen
dfapptraintest = dfapptraintest_raw.copy(deep=True)

# Nach den Vorgaben werden folgende Variablen als kategoriell betrachtet
cat = ['TARGET', 'NAME_CONTRACT_TYPE','CODE_GENDER','FLAG_OWN_CAR','FLAG_OWN_REALTY','NAME_TYPE_SUITE','NAME_INCOME_TYPE',
       'NAME_EDUCATION_TYPE','NAME_FAMILY_STATUS','NAME_HOUSING_TYPE','OCCUPATION_TYPE','ORGANIZATION_TYPE',
       'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY','REG_CITY_NOT_WORK_CITY','LIVE_CITY_NOT_WORK_CITY','FLAG_MOBIL',
       'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'TYPE']
dfapptraintest[cat]=dfapptraintest[cat].astype('category')
```

In [20]:
```
# Kontrolle der angepassten Datentypen
dfapptraintest.dtypes
```

Out[20]:
```
SK_ID_CURR                    int64
NAME_CONTRACT_TYPE         category
CODE_GENDER                category
FLAG_OWN_CAR               category
FLAG_OWN_REALTY            category
                              ...
AMT_REQ_CREDIT_BUREAU_MON   float64
AMT_REQ_CREDIT_BUREAU_QRT   float64
AMT_REQ_CREDIT_BUREAU_YEAR  float64
TARGET                     category
TYPE                       category
Length: 50, dtype: object
```

In [21]:
```
print_column_datatypes(dfapptraintest.loc[:, dfapptraintest.columns != 'TYPE'])
```

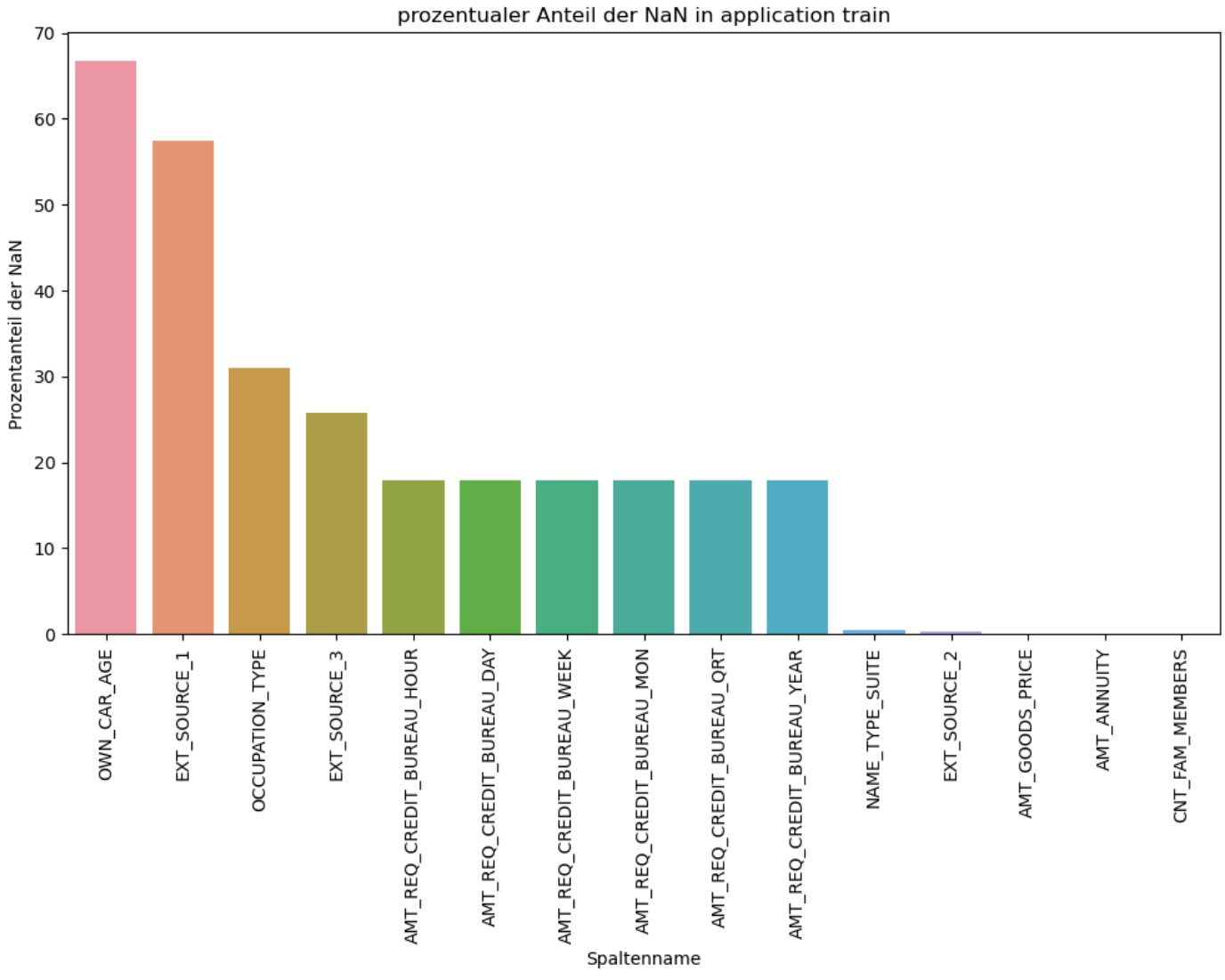
```
Spalten mit numerischen Werten: 23
Spalten mit kategoriellen Werten: 26
Insgesamt 49 Spalten
```

In [22]:
```
# NaN berechnen und plotten
df = nan_df_create(dfapptraintest[dfapptraintest['TYPE'] == 'TRAIN'])
print(df.to_string())
```

```
                      Spalte    Prozent
20                OWN_CAR_AGE  66.702817
38                EXT_SOURCE_1  57.504363
27             OCCUPATION_TYPE  30.981052
40                EXT_SOURCE_3  25.805909
42  AMT_REQ_CREDIT_BUREAU_HOUR  17.853403
43   AMT_REQ_CREDIT_BUREAU_DAY  17.853403
44  AMT_REQ_CREDIT_BUREAU_WEEK  17.853403
45   AMT_REQ_CREDIT_BUREAU_MON  17.853403
46   AMT_REQ_CREDIT_BUREAU_QRT  17.853403
47  AMT_REQ_CREDIT_BUREAU_YEAR  17.853403
10              NAME_TYPE_SUITE   0.483670
39                EXT_SOURCE_2   0.277362
9              AMT_GOODS_PRICE   0.095986
8                  AMT_ANNUITY   0.001870
28             CNT_FAM_MEMBERS   0.001247
```

In [23]:
```
plot_nan_percent(df, "application train", tight_layout = True, figsize = (10,8), grid = False, rotation = 90)
```

```
Anzahl der Spalten mit NaNs: 15 Spalten
```



Daten aus previous_application

In [24]:
```
dfprevapp_raw.shape
```

Out[24]:
```
(517957, 17)
```

In [25]:
```
print(dfprevapp_raw.head())
```

```
   SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY  AMT_APPLICATION  AMT_CREDIT  \
0      200001                 C         37.0              450         540
1      200002                 C         39.0              270         216
2      200002                 C         23.0              230         207
3      200003                 C         10.0               84          94
4      200004                 C         27.0              277         274

   AMT_DOWN_PAYMENT  AMT_GOODS_PRICE NAME_CONTRACT_STATUS  DAYS_DECISION  \
0               NaN            450.0                    A           -602
1              54.0            270.0                    A           -523
2              23.0            230.0                    A           -839
3               0.0             84.0                    A           -359
4              28.0            277.0                    A          -1232

  NAME_PAYMENT_TYPE CODE_REJECT_REASON NAME_TYPE_SUITE NAME_CLIENT_TYPE  \
0                 C                  X            NaN                N
1                 C                  X            NaN                R
2                 C                  X            NaN                N
3                 C                  X            NaN                N
4                 C                  X              U                N

  NAME_GOODS_CATEGORY  CNT_PAYMENT NAME_YIELD_GROUP  NFLAG_INSURED_ON_APPROVAL
0                   X         36.0               h                        1.0
1                   A          6.0               l                        0.0
2                   F         10.0               l                        0.0
3                   A         12.0               m                        0.0
4                   C         12.0               l                        0.0
```

In [26]:
```
dfprevapp_raw.dtypes
```

Out[26]:
```
SK_ID_CURR                    int64
NAME_CONTRACT_TYPE           object
AMT_ANNUITY                 float64
AMT_APPLICATION               int64
AMT_CREDIT                    int64
AMT_DOWN_PAYMENT            float64
AMT_GOODS_PRICE             float64
NAME_CONTRACT_STATUS         object
DAYS_DECISION                 int64
NAME_PAYMENT_TYPE            object
CODE_REJECT_REASON           object
NAME_TYPE_SUITE              object
NAME_CLIENT_TYPE             object
NAME_GOODS_CATEGORY          object
CNT_PAYMENT                 float64
NAME_YIELD_GROUP             object
NFLAG_INSURED_ON_APPROVAL   float64
dtype: object
```

In [27]:
```
for col in dfprevapp_raw.columns:
    print(f"Spalte: {col} Wertausprägungen: {len(dfprevapp_raw.loc[:,col].unique())}")
```

```
Spalte: SK_ID_CURR Wertausprägungen: 177056
Spalte: NAME_CONTRACT_TYPE Wertausprägungen: 3
Spalte: AMT_ANNUITY Wertausprägungen: 340
Spalte: AMT_APPLICATION Wertausprägungen: 2054
Spalte: AMT_CREDIT Wertausprägungen: 3003
Spalte: AMT_DOWN_PAYMENT Wertausprägungen: 565
Spalte: AMT_GOODS_PRICE Wertausprägungen: 2055
Spalte: NAME_CONTRACT_STATUS Wertausprägungen: 4
Spalte: DAYS_DECISION Wertausprägungen: 2922
Spalte: NAME_PAYMENT_TYPE Wertausprägungen: 3
Spalte: CODE_REJECT_REASON Wertausprägungen: 6
Spalte: NAME_TYPE_SUITE Wertausprägungen: 7
Spalte: NAME_CLIENT_TYPE Wertausprägungen: 3
Spalte: NAME_GOODS_CATEGORY Wertausprägungen: 17
Spalte: CNT_PAYMENT Wertausprägungen: 40
Spalte: NAME_YIELD_GROUP Wertausprägungen: 4
Spalte: NFLAG_INSURED_ON_APPROVAL Wertausprägungen: 3
```

In [28]:
```
# Anpassungen an den Datentypen
dfprevapp = dfprevapp_raw.copy(deep=True)

# Nach den Vorgaben werden folgende Variablen als kategoriell betrachtet
cat = ['NAME_CONTRACT_TYPE','NAME_CONTRACT_STATUS','NAME_PAYMENT_TYPE','CODE_REJECT_REASON','NAME_TYPE_SUITE','NAME_CLIENT_TYPE',
       'NAME_GOODS_CATEGORY','NAME_YIELD_GROUP','NFLAG_INSURED_ON_APPROVAL']

dfprevapp[cat]=dfprevapp[cat].astype('category')
```

In [29]:
```
dfprevapp.dtypes
```

Out[29]:
```
SK_ID_CURR                    int64
NAME_CONTRACT_TYPE         category
AMT_ANNUITY                 float64
AMT_APPLICATION               int64
AMT_CREDIT                    int64
AMT_DOWN_PAYMENT            float64
AMT_GOODS_PRICE             float64
NAME_CONTRACT_STATUS       category
DAYS_DECISION                 int64
NAME_PAYMENT_TYPE          category
CODE_REJECT_REASON         category
NAME_TYPE_SUITE            category
NAME_CLIENT_TYPE           category
NAME_GOODS_CATEGORY        category
CNT_PAYMENT                 float64
NAME_YIELD_GROUP           category
NFLAG_INSURED_ON_APPROVAL  category
dtype: object
```

In [30]:
```
print_column_datatypes(dfprevapp)
```

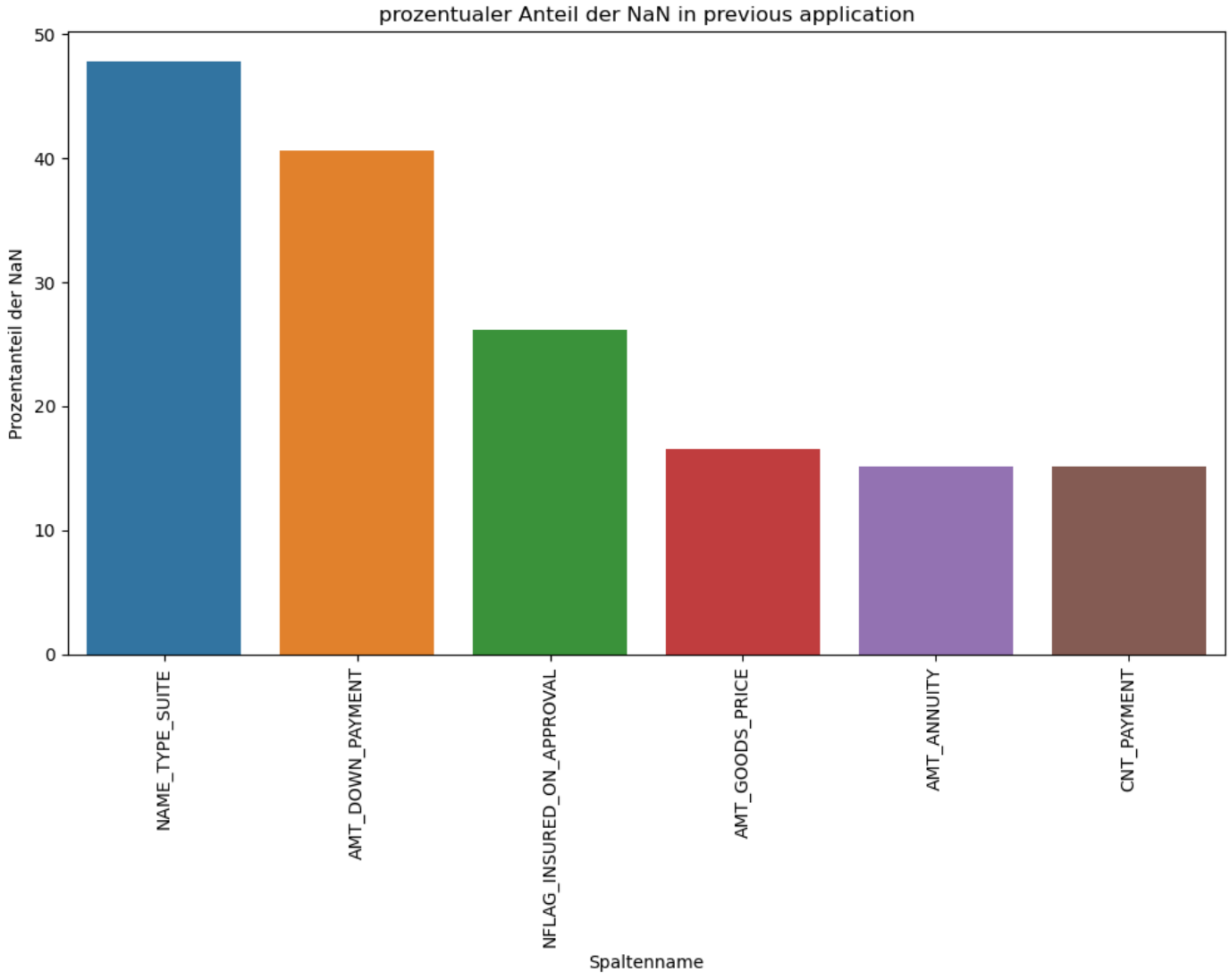
```
Spalten mit numerischen Werten: 8
Spalten mit kategoriellen Werten: 9
Insgesamt 17 Spalten
```

In [31]:
```
# NaN berechnen und plotten
df = nan_df_create(dfprevapp)
print(df.to_string())
```

```
                        Spalte    Prozent
11            NAME_TYPE_SUITE  47.811112
5           AMT_DOWN_PAYMENT  40.625959
16  NFLAG_INSURED_ON_APPROVAL  26.110276
6             AMT_GOODS_PRICE  16.500404
2                 AMT_ANNUITY  15.154540
14                CNT_PAYMENT  15.153575
```

In [32]:
```
plot_nan_percent(df, "previous application", tight_layout = True, figsize = (10,8), grid = False, rotation = 90)
```

```
Anzahl der Spalten mit NaNs: 6 Spalten
```



Daten aus bureau

In [33]:
```
dfbureau_raw.shape
```

Out[33]:
```
(535341, 14)
```

In [34]:
```
print(dfbureau_raw.head())
```

```
   SK_ID_CURR CREDIT_ACTIVE  DAYS_CREDIT  CREDIT_DAY_OVERDUE  \
0      200001             C        -1900                   0
1      200001             C        -1489                   0
2      200001             A         -163                   0
3      200001             A        -1037                   0
4      200002             C         -429                   0

   DAYS_CREDIT_ENDDATE  DAYS_ENDDATE_FACT  AMT_CREDIT_MAX_OVERDUE  \
0              -1170.0            -1292.0                     NaN
1              -1123.0            -1033.0                     NaN
2                 98.0                NaN                     NaN
3                786.0                NaN               17053.965
4               -187.0             -215.0                     NaN

   CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  AMT_CREDIT_SUM_DEBT  \
0                   0           804.0                  0.0
1                   0           404.0                  0.0
2                   0           450.0                457.0
3                   0           918.0                909.0
4                   0           234.0                  0.0

   AMT_CREDIT_SUM_LIMIT  AMT_CREDIT_SUM_OVERDUE  DAYS_CREDIT_UPDATE  \
0                   0.0                       0               -1287
1                   0.0                       0               -1033
2                   0.0                       0                -133
3                   0.0                       0                  -5
4                   NaN                       0                -193

   AMT_ANNUITY
0          0.0
1          0.0
2          NaN
3        115.0
4          NaN
```

In [35]:
```
dfbureau_raw.dtypes
```

Out[35]:
```
SK_ID_CURR                int64
CREDIT_ACTIVE            object
DAYS_CREDIT               int64
CREDIT_DAY_OVERDUE        int64
DAYS_CREDIT_ENDDATE     float64
DAYS_ENDDATE_FACT       float64
AMT_CREDIT_MAX_OVERDUE  float64
CNT_CREDIT_PROLONG        int64
AMT_CREDIT_SUM          float64
AMT_CREDIT_SUM_DEBT     float64
AMT_CREDIT_SUM_LIMIT    float64
AMT_CREDIT_SUM_OVERDUE    int64
DAYS_CREDIT_UPDATE        int64
AMT_ANNUITY             float64
dtype: object
```

In [36]:
```
for col in dfbureau_raw.columns:
    print(f"Spalte: {col} Wertausprägungen: {len(dfbureau_raw.loc[:,col].unique())}")
```

```
Spalte: SK_ID_CURR Wertausprägungen: 155163
Spalte: CREDIT_ACTIVE Wertausprägungen: 4
Spalte: DAYS_CREDIT Wertausprägungen: 2923
Spalte: CREDIT_DAY_OVERDUE Wertausprägungen: 509
Spalte: DAYS_CREDIT_ENDDATE Wertausprägungen: 12125
Spalte: DAYS_ENDDATE_FACT Wertausprägungen: 2876
Spalte: AMT_CREDIT_MAX_OVERDUE Wertausprägungen: 27650
Spalte: CNT_CREDIT_PROLONG Wertausprägungen: 7
Spalte: AMT_CREDIT_SUM Wertausprägungen: 8334
Spalte: AMT_CREDIT_SUM_DEBT Wertausprägungen: 8934
Spalte: AMT_CREDIT_SUM_LIMIT Wertausprägungen: 16624
Spalte: AMT_CREDIT_SUM_OVERDUE Wertausprägungen: 135
Spalte: DAYS_CREDIT_UPDATE Wertausprägungen: 2890
Spalte: AMT_ANNUITY Wertausprägungen: 675
```

In [37]:
```
# Anpassungen an den Datentypen
dfbureau = dfbureau_raw.copy(deep=True)

# Nach den Vorgaben werden folgende Variablen als kategoriell betrachtet
cat = ['CREDIT_ACTIVE']
dfbureau[cat]=dfbureau[cat].astype('category')
```

In [38]:
```
dfbureau.dtypes
```

Out[38]:
```
SK_ID_CURR                int64
CREDIT_ACTIVE          category
DAYS_CREDIT               int64
CREDIT_DAY_OVERDUE        int64
DAYS_CREDIT_ENDDATE     float64
DAYS_ENDDATE_FACT       float64
AMT_CREDIT_MAX_OVERDUE  float64
CNT_CREDIT_PROLONG        int64
AMT_CREDIT_SUM          float64
AMT_CREDIT_SUM_DEBT     float64
AMT_CREDIT_SUM_LIMIT    float64
AMT_CREDIT_SUM_OVERDUE    int64
DAYS_CREDIT_UPDATE        int64
AMT_ANNUITY             float64
dtype: object
```

In [39]:
```
print_column_datatypes(dfbureau)
```

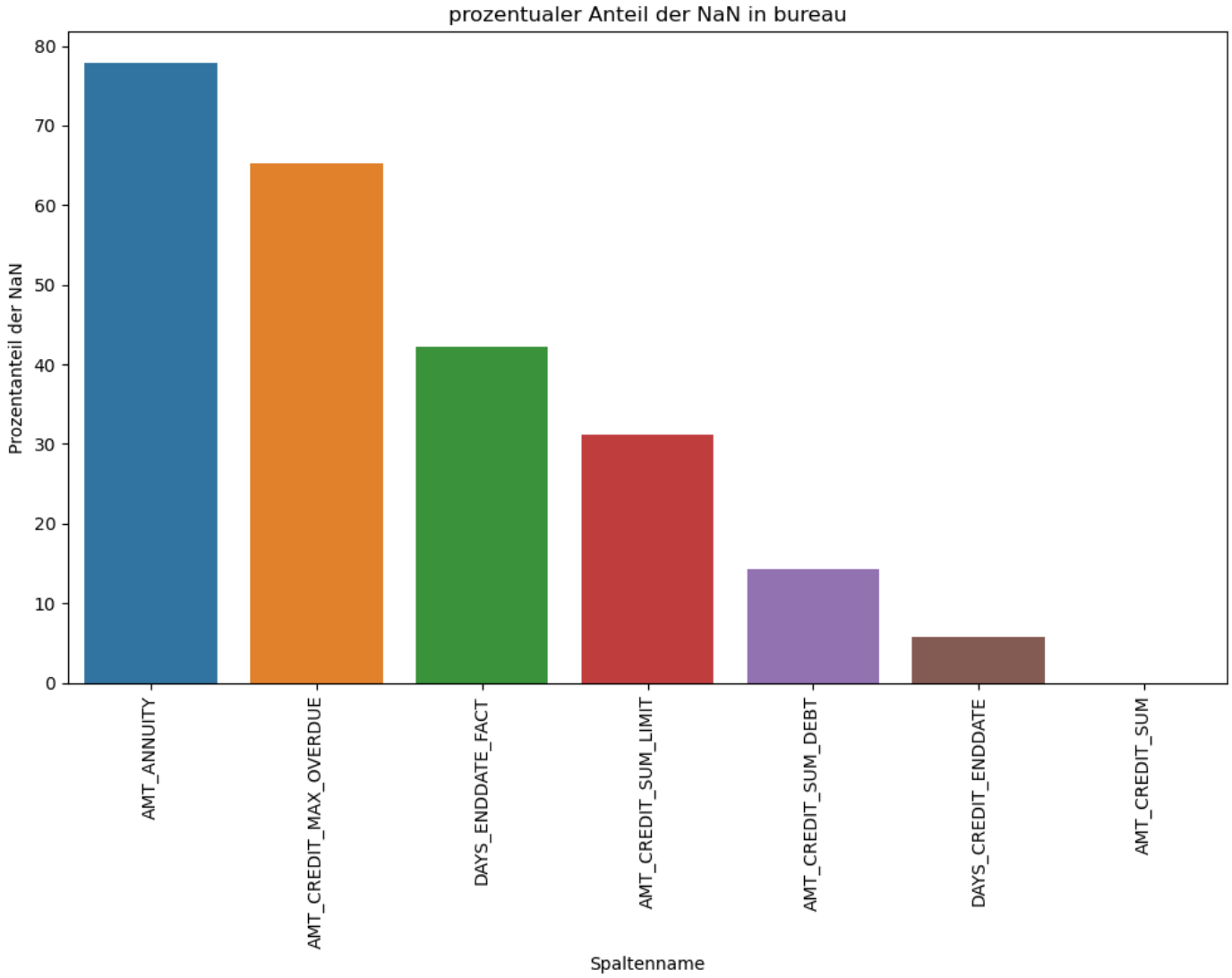
```
Spalten mit numerischen Werten: 13
Spalten mit kategoriellen Werten: 1
Insgesamt 14 Spalten
```

In [40]:
```
# Anteil NaN berechnen und plotten
df = nan_df_create(dfbureau)
print(df.to_string())
```

```
                  Spalte    Prozent
13            AMT_ANNUITY  77.881575
6   AMT_CREDIT_MAX_OVERDUE  65.254856
5        DAYS_ENDDATE_FACT  42.212907
10    AMT_CREDIT_SUM_LIMIT  31.147063
9      AMT_CREDIT_SUM_DEBT  14.370467
4      DAYS_CREDIT_ENDDATE   5.848609
8           AMT_CREDIT_SUM   0.000374
```

In [41]:
```
plot_nan_percent(df, "bureau", tight_layout = True, figsize = (10,8), grid = False, rotation = 90)
```

```
Anzahl der Spalten mit NaNs: 7 Spalten
```



Für die beiden Datensätze `dfprevapp` und `dfbureau` wird das Auftreten der jeweiligen IDs gezählt:

In [42]:
```
print(dfprevapp['SK_ID_CURR'].value_counts().to_frame())
```

```
        SK_ID_CURR
286171           6
317491           6
300014           6
260270           6
317483           6
...            ...
333991           1
333993           1
333997           1
246810           1
200001           1

[177056 rows x 1 columns]
```

In [43]:
```
print(dfbureau['SK_ID_CURR'].value_counts().to_frame())
```

```
        SK_ID_CURR
376637           7
356840           7
254635           7
306944           7
212186           7
...            ...
333777           1
333781           1
333783           1
333785           1
391298           1

[155163 rows x 1 columns]
```

Interpretation: Sowohl in der Tabelle previous_application als auch in der Tabelle bureau gibt es mehrere Einträge pro ID. Diese müssen vor dem Zusammenführen mit den application-Datensätzen geeignet aggregiert werden.

Aufgabe A-3: Datensätze visualisieren und aufbereiten [Lernziel 3.3.2, 3.3.4; 20 Punkte]

---

a) Für die Trainingsdaten sollen die kategoriellen Merkmale jeweils mittels Säulengrafiken visualisiert werden. Dafür sind gestapelte ("stacked") Säulengrafiken bzgl. des Targets zu verwenden, die auf der Y-Achse die jeweiligen Anzahlen der Target-Werte je Ausprägung zeigen. Weiter sollen gestapelte ("stacked") Säulengrafiken bzgl. des Targets erstellt werden, die auf der Y-Achse die jeweiligen Anteile der Target-Werte je Ausprägung des Merkmals zeigen. Welche Schlussfolgerungen ergeben sich aus diesen Grafiken?

b) Für die Trainingsdaten sollen die numerischen Merkmale jeweils mittels Kerndichteschätzungs-Plots visualisiert werden. Dafür sind für jedes Merkmal eine Grafik mit je zwei (gemäß den Target-Werten 0 und 1 unterschiedenen) geeigneten Kerndichteschätzungen zu erstellen. Welche Schlussfolgerungen ergeben sich aus diesen Grafiken?

c) Für die Trainingsdaten sollen die Merkmale `DAYS_BIRTH` und `DAYS_EMPLOYED` nun genauer untersucht werden. Zu diesem Zweck sind folgende Schritte auszuführen:

- **Rechnen Sie beide Merkmale in Jahre um und visualisieren Sie deren Verteilungen in jeweils einer Grafik und unterschieden nach Personen mit ausgefallenen (Target=1) und nicht ausgefallenen (Target=0) Krediten.**
- **Kommentieren Sie Auffälligkeiten und ersetzen Sie ggf. auffällige Werte durch den speziellen Wert für fehlende Werte (Python: `NaN`, R: `NA`).**

d) Für Kredite, die länger als 50 Jahre zurückliegen, sind die Werte in den Spalten `DAYS_CREDIT_ENDDATE`, `DAYS_ENDDATE_FACT` und `DAYS_CREDIT_UPDATE` durch 0 zu ersetzen.

Anmerkung: Temporär hinzugefügte Spalten sind nach Abschluss der Arbeiten aus dem Datensatz zu entfernen.

Lösungsvorschlag:

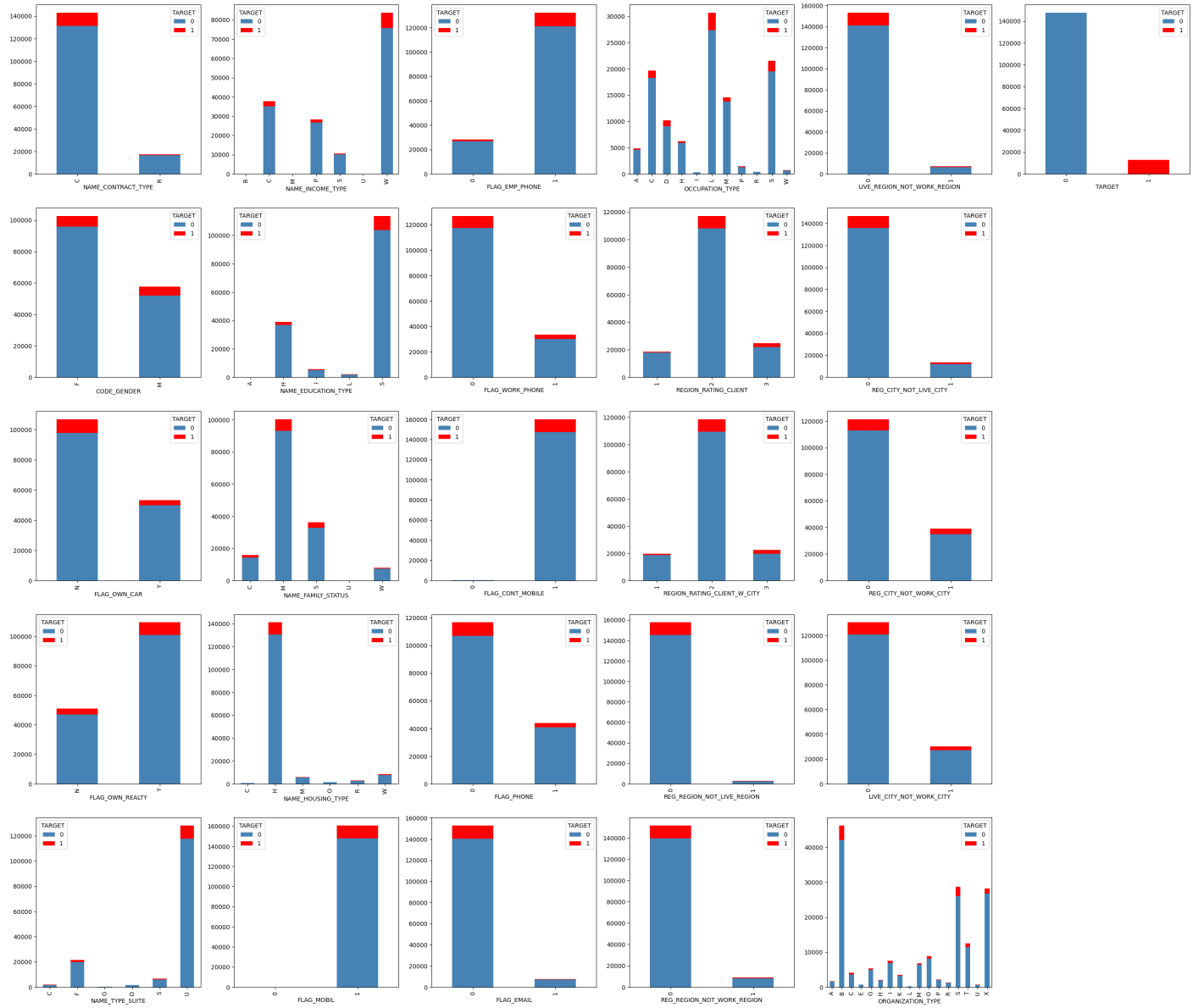
Teil a)

In [44]:
```
dfapptrain = dfapptraintest[dfapptraintest['TYPE'] == 'TRAIN']
dfapptrain = dfapptrain.loc[:, dfapptrain.columns != 'TYPE']

print(dfapptrain.describe())
```

```
          SK_ID_CURR   CNT_CHILDREN  AMT_INCOME_TOTAL     AMT_CREDIT  \
count  160440.000000  160440.000000     160440.000000  160440.000000
mean   295695.239653       0.420332        321.357617    1188.626303
std     55219.988044       0.723498        212.341860     812.138961
min    200001.000000       0.000000         52.000000      90.000000
25%    247905.750000       0.000000        207.000000     540.000000
50%    295748.500000       0.000000        270.000000    1017.000000
75%    343485.250000       1.000000        405.000000    1618.000000
max    391298.000000      19.000000      36001.000000    8100.000000

         AMT_ANNUITY  AMT_GOODS_PRICE  REGION_POPULATION_RELATIVE  \
count  160437.000000    160286.000000               160440.000000
mean       54.529198      1066.831114                    0.021178
std        29.577592       743.972618                    0.014353
min         4.000000        90.000000                    0.000000
25%        33.000000       468.000000                    0.010000
50%        50.000000       900.000000                    0.019000
75%        70.000000      1359.000000                    0.029000
max       517.000000      8100.000000                    0.073000

          DAYS_BIRTH  DAYS_EMPLOYED  DAYS_REGISTRATION  DAYS_ID_PUBLISH  \
count  160440.000000  160440.000000      160440.000000    160440.000000
mean   -15774.055329   62425.506862       -4989.590482     -2913.411232
std      4440.451270  139910.094187        3519.124202      1521.928071
min    -25229.000000  -17139.000000      -23738.000000     -6383.000000
25%    -19478.000000   -2513.000000       -7491.000000     -4264.000000
50%    -15461.000000   -1096.000000       -4495.000000     -3106.000000
75%    -12028.750000    -268.000000       -2027.000000     -1590.000000
max     -7489.000000  365243.000000           0.000000         0.000000

        OWN_CAR_AGE  CNT_FAM_MEMBERS  EXT_SOURCE_1   EXT_SOURCE_2  \
count  53422.000000    160438.000000  68180.000000  159995.000000
mean      12.204335         2.144772      0.490488       0.513548
std       11.776800         0.918340      0.212465       0.192060
min        0.000000         1.000000      0.015000       0.000000
25%        5.000000         2.000000      0.319000       0.388000
50%        9.000000         2.000000      0.490000       0.565000
75%       15.000000         3.000000      0.664000       0.664000
max       91.000000        20.000000      0.963000       0.855000

        EXT_SOURCE_3  DAYS_LAST_PHONE_CHANGE  AMT_REQ_CREDIT_BUREAU_HOUR  \
count  119037.000000           160440.000000                131796.000000
mean        0.536615             -867.770107                     0.006222
std         0.191917              818.139219                     0.083227
min         0.001000            -4292.000000                     0.000000
25%         0.401000            -1407.000000                     0.000000
50%         0.567000             -628.000000                     0.000000
75%         0.689000             -210.000000                     0.000000
max         0.896000                0.000000                     3.000000

       AMT_REQ_CREDIT_BUREAU_DAY  AMT_REQ_CREDIT_BUREAU_WEEK  \
count              131796.000000               131796.000000
mean                    0.006905                    0.031306
std                     0.111064                    0.195531
min                     0.000000                    0.000000
25%                     0.000000                    0.000000
50%                     0.000000                    0.000000
75%                     0.000000                    0.000000
max                     9.000000                    8.000000

       AMT_REQ_CREDIT_BUREAU_MON  AMT_REQ_CREDIT_BUREAU_QRT  \
count              131796.000000              131796.000000
mean                    0.194376                   0.220743
std                     0.727423                   0.899616
min                     0.000000                   0.000000
25%                     0.000000                   0.000000
50%                     0.000000                   0.000000
75%                     0.000000                   0.000000
max                    19.000000                 261.000000

       AMT_REQ_CREDIT_BUREAU_YEAR
count               131796.000000
mean                     1.376195
std                      1.427731
min                      0.000000
25%                      0.000000
50%                      1.000000
75%                      2.000000
max                     10.000000
```

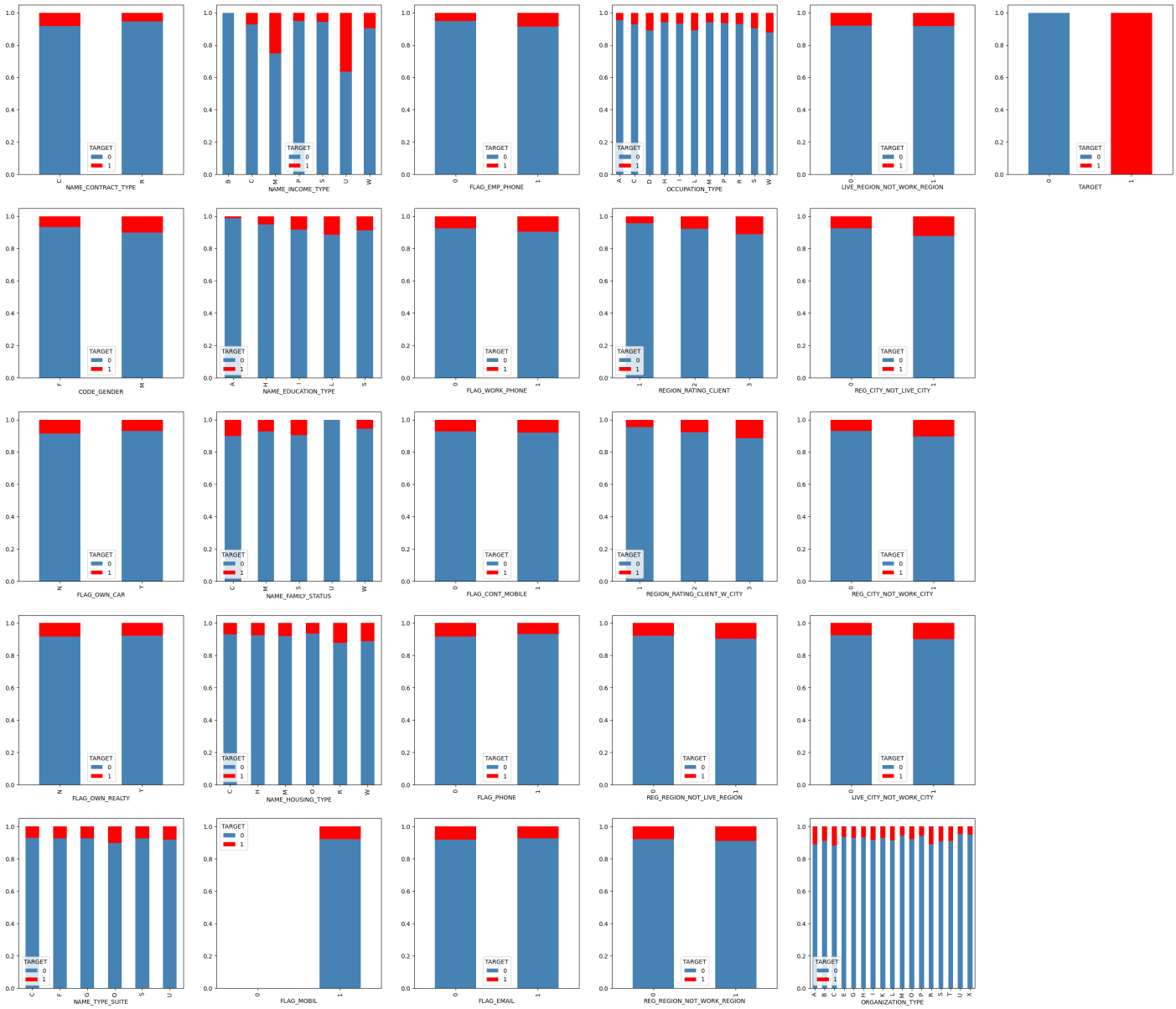
In [45]:
```
plot_cat(dfapptrain, 'TARGET', 6, 5, "Countplots für kategorielle Variablen im Datensatz application_train")
```

Countplots für kategorielle Variablen im Datensatz application_train



In [46]:
```
plot_cat_proz(dfapptrain, 'TARGET', 6, 5, "Prozentuale Stacked-Barplots für kategorielle Variablen im Datensatz application_train")
```

Prozentuale Stacked-Barplots für kategorielle Variablen im Datensatz application_train



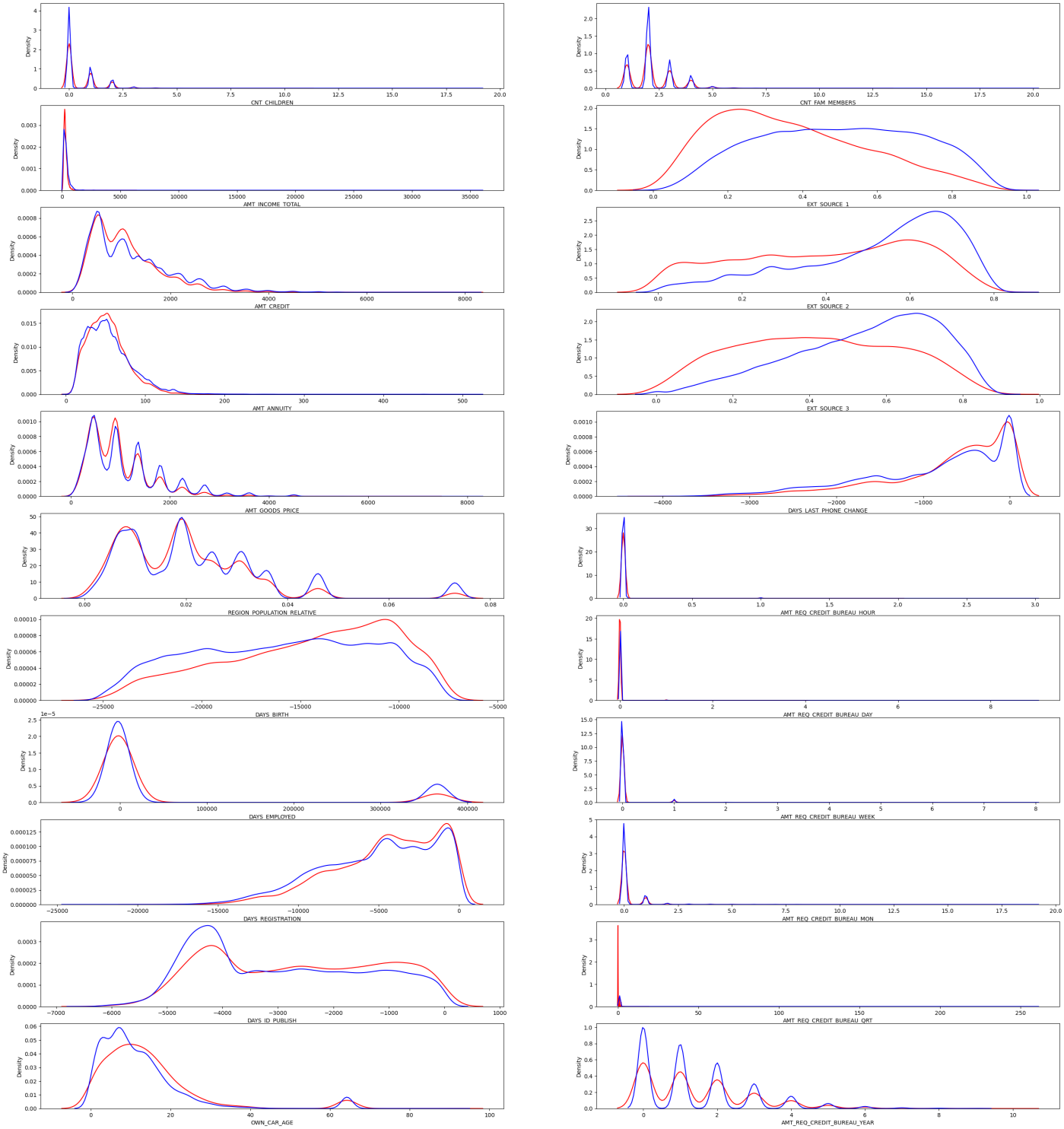
Interpretation:

Aus den Grafiken ergeben sich unter anderem die folgenden Beobachtungen (Auswahl):

- Die Zielvariable TARGET ist unbalanciert: Es gibt deutlich weniger ausgefallene als nicht ausgefallene Kredite.
- Männliche Personen scheinen eine höhere Wahrscheinlichkeit eines Kreditausfalls zu haben als weibliche Personen.
- Bei dem Merkmal OCCUPATION_TYPE scheint die Klasse W eine höhere Ausfallwahrscheinlichkeit zu haben als die restlichen Klassen.
- Bei dem Merkmal NAME_EDUCATION_TYPE scheint die Klasse L eine höhere Ausfallwahrscheinlichkeit zu haben als die restlichen Klassen.

Teil b)

In [47]:
```
plot_num(dfapptrain, 'TARGET', 2, 11, "Numerische Variablen im Datensatz application_train")
```



Interpretation:

Aus den Grafiken ergeben sich unter anderem die folgenden Beobachtungen (Auswahl):

- Die Scores (EXT_SOURCE_1, EXT_SOURCE_2 und EXT_SOURCE_3) scheinen bereits eine gute Differenzierung zu bieten.
- Das Merkmal DAYS_BIRTH scheint im rechts liegenden Bereich ebenfalls eine gute Differenzierung zu bieten.
- Für das Merkmal DAYS_EMPLOYED sieht man einen fragwürden Verlauf im rechten Wertebereich. Das wird im Teil c) genauer analysiert.

Teil c)

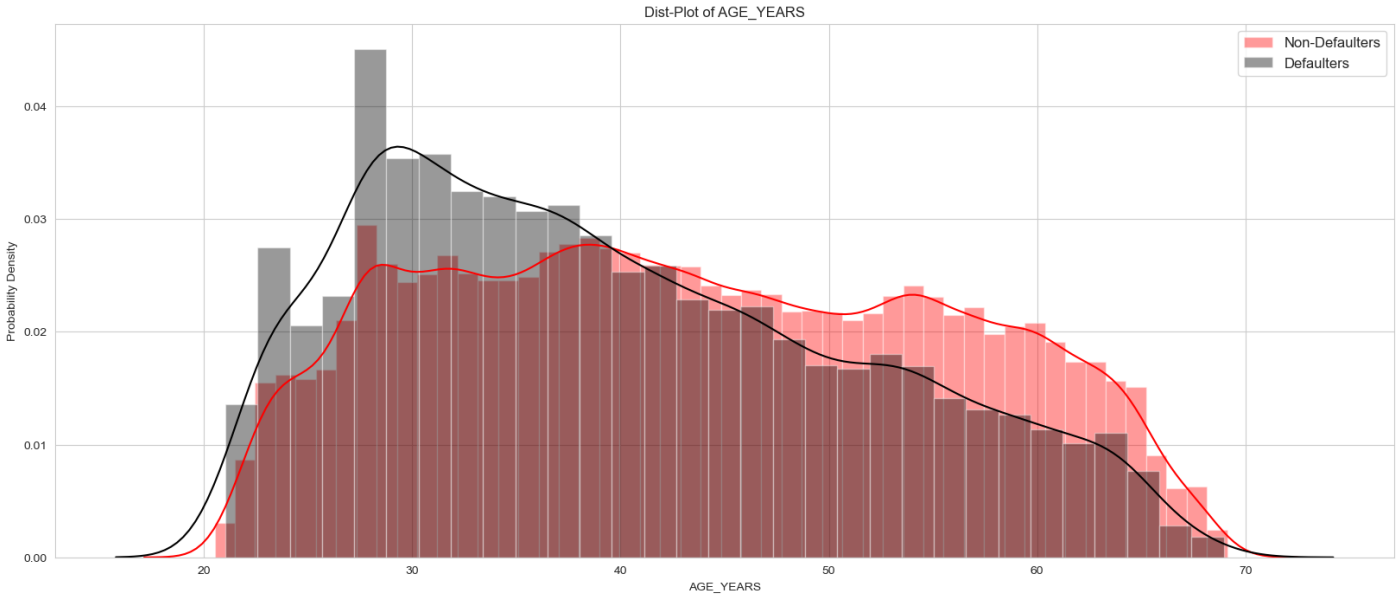
Das Merkmal **DAYS_BIRTH** beschreibt das Alter in Tagen vor dem Antrag. Zur Visualisierung rechnen wir dieses wie gefordert in Jahre um.

In [48]:
```
dfapptrain['AGE_YEARS'] = dfapptraintest[dfapptraintest['TYPE'] == 'TRAIN']['DAYS_BIRTH'] * -1 / 365
```

In [49]:
```
dfapptrain['AGE_YEARS'].describe()
```

Out[49]:
```
count    160440.000000
mean         43.216590
std          12.165620
min          20.517808
25%          32.955479
50%          42.358904
75%          53.364384
max          69.120548
Name: AGE_YEARS, dtype: float64
```

In [50]:
```
plot_continuous_variables(dfapptrain, 'AGE_YEARS')
```



Interpretation: Die Antragsteller sind zwischen 20 und 70 Jahren alt, was plausibel ist. Man erkennt ein Muster bei den Personen mit einem Kreditausfall: Dieser ist sehr häufig bei jungen Personen (Alter < 40 Jahren) zu beobachten. Ab diesem Alter überwiegen die Personen ohne Kreditausfall. Eine Anpassung des Features ist nicht notwenig.

In [51]:
```
# Zusätzliche Spalte wieder entfernen
dfapptrain.pop('AGE_YEARS');
```

In [52]:
```
# Zur Besseren Einschätzung rechnen wir wieder in Jahre um
dfapptrain['YEARS_EMPLOYED'] = dfapptrain['DAYS_EMPLOYED'] * -1 / 365
```

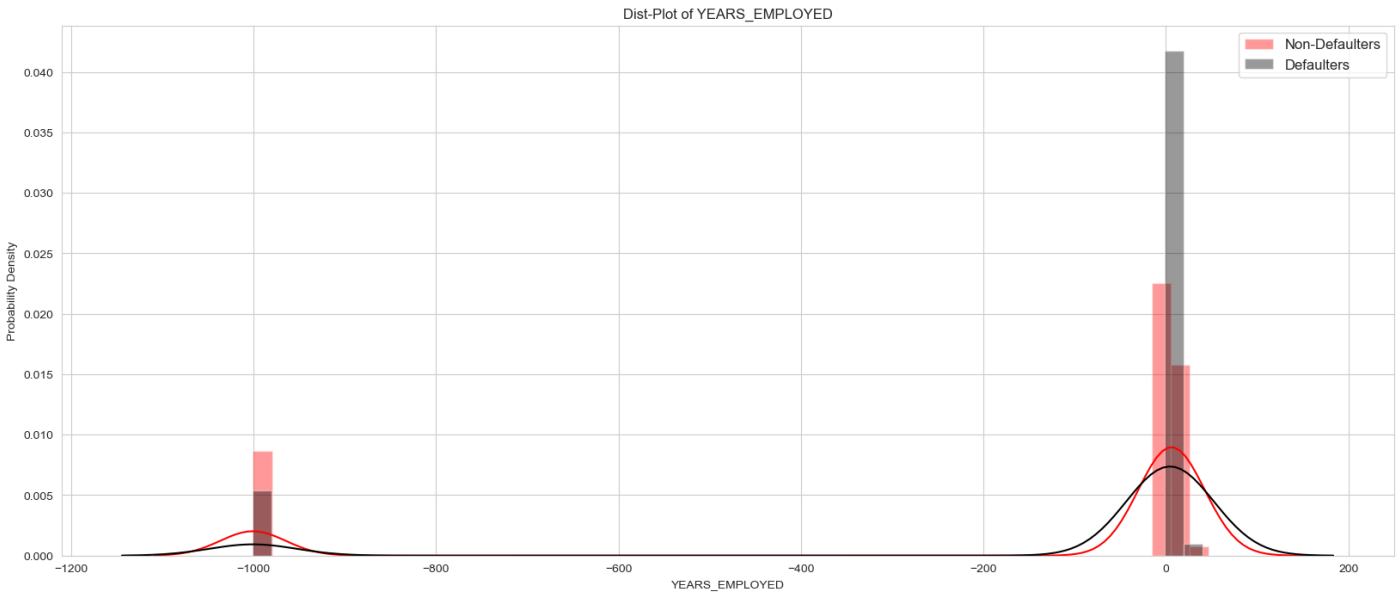
In [53]:
```
dfapptrain['YEARS_EMPLOYED'].describe()
```

Out[53]:
```
count    160440.000000
mean       -171.028786
std         383.315327
min       -1000.665753
25%           0.734247
50%           3.002740
75%           6.884932
max          46.956164
Name: YEARS_EMPLOYED, dtype: float64
```

In [54]:
```
dfapptrain["YEARS_EMPLOYED"][dfapptrain["YEARS_EMPLOYED"] < 0].count()
```

Out[54]:
```
28219
```

In [55]:
```
plot_continuous_variables(dfapptrain, 'YEARS_EMPLOYED')
```

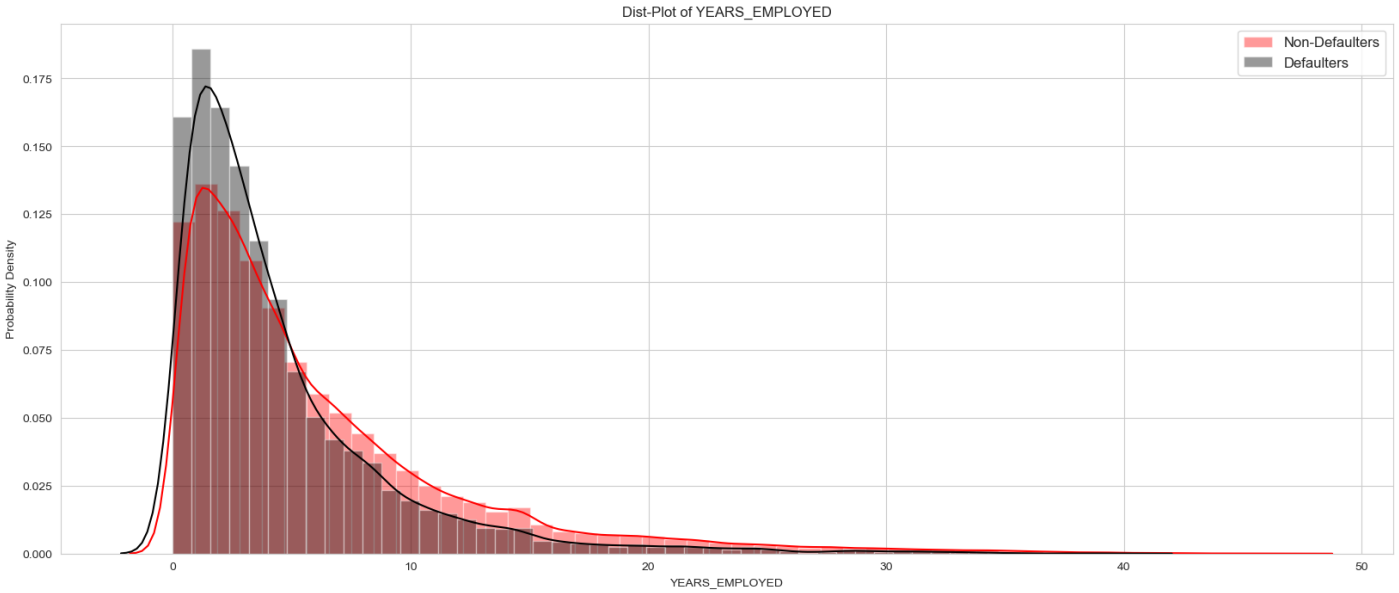


Interpretation: Das Minimum ist auffällig: Es liegt bei -1000 Jahren. Das kann auf eine besondere Markierung in den Daten hinweisen. Diese Auffälligkeit wird wie in der Aufgabenstellung gefordert dadurch behoben, dass diese Werte durch NaN ersetzt werden. Insgesamt gibt es 28.219 solcher Einträge. Die Grafik zeigt zudem, dass für diesen Zeitpunkt die Personen ohne Kreditausfall überwiegen.

In [56]:
```
# Aktionen durchführen
dfapptrain['DAYS_EMPLOYED'].replace(365243, np.nan, inplace= True)
dfapptraintest['DAYS_EMPLOYED'].replace(365243, np.nan, inplace= True)
```

In [57]:
```
# Zur Besseren Einschätzung rechnen wir wieder in Jahre um
dfapptrain['YEARS_EMPLOYED'] = dfapptrain['DAYS_EMPLOYED'] * -1 / 365
```

In [58]:
```
plot_continuous_variables(dfapptrain, 'YEARS_EMPLOYED')
```



Ohne die negativen Werte sehen die Daten vernünftig aus.

In [59]:
```
#Zusätzliche Spalte wieder entfernen
dfapptrain.pop('YEARS_EMPLOYED');
```

Teil d)

Für die geforderten Spalten werden die Werte, die sich auf einen Zeitpunkt vor über 50 Jahren beziehen auf 0 gesetzt.

In [60]:
```
dfbureau['DAYS_CREDIT_ENDDATE'][dfbureau['DAYS_CREDIT_ENDDATE'] < -50*365] = 0
dfbureau['DAYS_ENDDATE_FACT'][dfbureau['DAYS_CREDIT_ENDDATE'] < -50*365] = 0
dfbureau['DAYS_CREDIT_UPDATE'][dfbureau['DAYS_CREDIT_ENDDATE'] < -50*365] = 0
```

Aufgabe A-4: Feature Engineering [Lernziel 3.3, 3.4; 10 Punkte]

In dieser Aufgabe sollen die Informationen aus den Datensätzen `previous_application`, `bureau` und dem kombinierten Datensatz aus Trainings- und Testdaten aufbereitet werden.

a) Es sind die folgenden Tätigkeiten für jeden der drei genannten Datensätze durchzuführen:

- **Bei numerischen Merkmalen sollen fehlende Werte durch 0 ersetzt werden.**
- **Bei kategoriellen Merkmalen sind fehlende Werte durch die neue Kategorie `Kategorie_NaN` zu ersetzen.**
- **Zusätzlich sollen die in Anhang 1 angegebenen domänenspezifischen Merkmale in den betrachteten Datensatz neu mit aufgenommen werden.**
- **Der Datensatz ist so zu aggregieren, dass pro ID (gemäß `SK_ID_CURR`) genau eine Zeile übrigbleibt. Dabei sollen folgende Punkte berücksichtigt werden:**
  - **Vor der Aggregation sind die Datensätze nach `SK_ID_CURR` und `DAYS_DECISION` (bei `previous_application`) bzw. nach `SK_ID_CURR` und `DAYS_CREDIT_ENDDATE` (bei `bureau`) aufsteigend zu sortieren.**
  - **Bei numerischen Merkmalen sollen die vier Aggregationen `sum` (Summe), `mean` (Mittelwert), `max` (Maximum) und `min` (Minimum) verwendet werden. Folglich wird eine numerische Spalte durch vier Spalten zu den entsprechenden Aggregationen ersetzt.**
  - **Bei kategoriellen Merkmalen sollen die drei Aggregationen `mode` (Modus), `first` (erster auftretender Wert) und `last` (letzter auftretender Wert) verwendet werden. Folglich wird eine kategorielle Spalte durch drei Spalten zu den entsprechenden Aggregationen ersetzt.**

b) Was sollte beim Hinzufügen von neuen Merkmalen, die (ähnlich wie diejenigen in Anhang 1) auf Domain Knowledge basieren, generell beachtet werden? Welche Maßnahmen kann man ergreifen, um die Qualität der zusätzlichen Merkmale sicherzustellen?

Lösungsvorschlag:

In [61]:
```
aggregations_num = ['sum', 'mean', 'max', 'min']
aggregations_cat = [modus, 'first', 'last']
```

4 a)

PREVIOUS APPLICATION

Wie im Teil 1) dargestellt, enthält der Datensatz die folgenden numerischen Spalten mit fehlenden Werten: AMT_DOWN_PAYMENT, AMT_GOODS_PRICE, AMT_ANNUITY und CNT_PAYMENT. Diese werden durch 0 ersetzt.

Die kategoriellen Variablen NAME_TYPE_SUITE und NFLAG_INSURED_ON_APPROVAL enthalten ebenfalls fehlende Werte.

In [62]:
```
print(dfprevapp.head())
```

```
   SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY  AMT_APPLICATION  AMT_CREDIT  \
0      200001                 C         37.0              450         540
1      200002                 C         39.0              270         216
2      200002                 C         23.0              230         207
3      200003                 C         10.0               84          94
4      200004                 C         27.0              277         274

   AMT_DOWN_PAYMENT  AMT_GOODS_PRICE NAME_CONTRACT_STATUS  DAYS_DECISION  \
0               NaN            450.0                    A           -602
1              54.0            270.0                    A           -523
2              23.0            230.0                    A           -839
3               0.0             84.0                    A           -359
4              28.0            277.0                    A          -1232

  NAME_PAYMENT_TYPE CODE_REJECT_REASON NAME_TYPE_SUITE NAME_CLIENT_TYPE  \
0                 C                  X            NaN                N
1                 C                  X            NaN                R
2                 C                  X            NaN                N
3                 C                  X            NaN                N
4                 C                  X              U                N

  NAME_GOODS_CATEGORY  CNT_PAYMENT NAME_YIELD_GROUP NFLAG_INSURED_ON_APPROVAL
0                   X         36.0                h                       1.0
1                   A          6.0                l                       0.0
2                   F         10.0                l                       0.0
3                   A         12.0                m                       0.0
4                   C         12.0                l                       0.0
```

In [63]:
```
# Bevor die fehlenden Werte im Datensatz ersetzt werden, wird die Anzahl dieser Werte noch gezählt
dfprevapp['PreDK_MISSING_VALUES_TOTAL_PREV'] = dfprevapp.isna().sum(axis = 1)
```

In [64]:
```
# Ersetzen der NaNs in den numerischen Spalten durch den Wert `0`
numerical_columns = ['AMT_DOWN_PAYMENT','AMT_GOODS_PRICE','AMT_ANNUITY','CNT_PAYMENT']
dfprevapp[numerical_columns] = dfprevapp[numerical_columns].fillna(0)
```

In [65]:
```
# Ersetzen der NaNs in den kategoriellen Spalten durch den Wert `Kategorie_NaN`
dfprevapp['NAME_TYPE_SUITE'] = dfprevapp['NAME_TYPE_SUITE'].cat.add_categories("Kategorie_NaN").fillna("Kategorie_NaN")
dfprevapp['NFLAG_INSURED_ON_APPROVAL'] = dfprevapp['NFLAG_INSURED_ON_APPROVAL'].cat.add_categories("Kategorie_NaN").fillna("Kategorie_NaN")
```

In [66]:
```
# Basierend auf Domain-Wissen werden die geforderten Merkmale hinzugefügt.
# Alle neu hinzugefügten (manuellen) Features erhalten den Vorsatz PreDK

# Differenz zwischen Beantragung und tatsächlicher Kredit
dfprevapp['PreDK_AMT_DECLINED'] = dfprevapp['AMT_APPLICATION'] - dfprevapp['AMT_CREDIT']

# Verhältnis aus Kreditbetrag und zu finanzierendem Gut
dfprevapp['PreDK_AMT_CREDIT_GOODS_RATIO'] = dfprevapp['AMT_CREDIT'] / dfprevapp['AMT_GOODS_PRICE']

# Differenz aus Kreditbetrag und zu finanzierendem Gut
dfprevapp['PreDK_AMT_CREDIT_GOODS_DIFF'] = dfprevapp['AMT_CREDIT'] - dfprevapp['AMT_GOODS_PRICE']

# Quotient aus beantragtem Kreditbetrag und Kreditbetrag
dfprevapp['PreDK_AMT_CREDIT_APPLICATION_RATIO'] = dfprevapp['AMT_APPLICATION'] / dfprevapp['AMT_CREDIT']

# Schätzung Zinszahlung
dfprevapp['PreDK_AMT_INTEREST'] = dfprevapp['CNT_PAYMENT'] * dfprevapp['AMT_ANNUITY'] - dfprevapp['AMT_CREDIT']

# Quotient Rate zu Kredit
dfprevapp['PreDK_ANNUITY_CREDIT_RATIO'] = dfprevapp['AMT_ANNUITY'] / dfprevapp['AMT_CREDIT']

# Falls Divisionen durch Null auftreten, so werden die resultierenden `inf`-Werte durch `NaN` ersetzt
dfprevapp[dfprevapp == np.infty] = np.nan
dfprevapp[dfprevapp == -np.infty] = np.nan
```

In [67]:
```
aggregations_for_previous_application = {
            #numerisch
            'AMT_ANNUITY' :               aggregations_num,
            'AMT_APPLICATION' :           aggregations_num,
            'AMT_CREDIT' :                aggregations_num,
            'AMT_DOWN_PAYMENT' :          aggregations_num,
            'AMT_GOODS_PRICE' :           aggregations_num,
            'DAYS_DECISION' :             aggregations_num,
            'CNT_PAYMENT' :               aggregations_num,
            #domain knowledge
            'PreDK_MISSING_VALUES_TOTAL_PREV' :    aggregations_num,
            'PreDK_AMT_DECLINED' :                 aggregations_num,
```

```
            'PreDK_AMT_CREDIT_GOODS_RATIO' :       aggregations_num,
            'PreDK_AMT_CREDIT_GOODS_DIFF' :        aggregations_num,
            'PreDK_AMT_CREDIT_APPLICATION_RATIO' : aggregations_num,
            'PreDK_AMT_INTEREST' :                 aggregations_num,
            'PreDK_ANNUITY_CREDIT_RATIO':          aggregations_num,
            #kategorisch
            'NAME_CONTRACT_TYPE' :        aggregations_cat,
            'NAME_CONTRACT_STATUS' :      aggregations_cat,
            'NAME_PAYMENT_TYPE' :         aggregations_cat,
            'CODE_REJECT_REASON' :        aggregations_cat,
            'NAME_TYPE_SUITE' :           aggregations_cat,
            'NAME_CLIENT_TYPE' :          aggregations_cat,
            'NAME_GOODS_CATEGORY' :       aggregations_cat,
            'NAME_YIELD_GROUP' :          aggregations_cat,
            'NFLAG_INSURED_ON_APPROVAL' : aggregations_cat,
            }
```

In [68]:
```
# Sortierung und Durchführung der Aggregationen unter Berücksichtigung aller zugehörigen Anträge
tic = time.time()

dfprevapp = dfprevapp.sort_values(by = ['SK_ID_CURR','DAYS_DECISION'])
dfprevapp_agg = dfprevapp.groupby('SK_ID_CURR').agg(aggregations_for_previous_application)

# Spaltennamen korrigieren
dfprevapp_agg.columns = ['_'.join(col) for col in dfprevapp_agg.columns.values]

print("time (sec):" + "%6.0f" % (time.time() - tic))
```

```
time (sec):   182
```

In [69]:
```
print(dfprevapp_agg.head())
```

SK_ID_CURR	AMT_ANNUITY_sum	AMT_ANNUITY_mean	AMT_ANNUITY_max	AMT_ANNUITY_min	AMT_APPLICATION_sum	AMT_APPLICATION_mean	AMT_APPLICATION_max	AMT_APPLICATION_min	AMT_CREDIT_sum	AMT_CREDIT_mean	AMT_CREDIT_max	AMT_CREDIT_min	AMT_DOWN_PAYMENT_sum	AMT_DOWN_PAYMENT_mean	AMT_DOWN_PAYMENT_max	AMT_DOWN_PAYMENT_min	AMT_GOODS_PRICE_sum	AMT_GOODS_PRICE_mean	AMT_GOODS_PRICE_max	AMT_GOODS_PRICE_min	DAYS_DECISION_sum	DAYS_DECISION_mean	DAYS_DECISION_max	DAYS_DECISION_min	CNT_PAYMENT_sum	CNT_PAYMENT_mean	CNT_PAYMENT_max	CNT_PAYMENT_min	PreDK_MISSING_VALUES_TOTAL_PREV_sum	PreDK_MISSING_VALUES_TOTAL_PREV_mean	PreDK_MISSING_VALUES_TOTAL_PREV_max
200001	37.0	37.0	37.0	37.0	450	450.0	450	450	540	540.0	540	540	0.0	0.0	0.0	0.0	450.0	450.0	450.0	450.0	-602	-602.0	-602	-602	36.0	36.0	36.0	36.0	2	2.0	2
200002	62.0	31.0	39.0	23.0	500	250.0	270	230	423	211.5	216	207	77.0	38.5	54.0	23.0	500.0	250.0	270.0	230.0	-1362	-681.0	-523	-839	16.0	8.0	10.0	6.0	2	1.0	1
200003	10.0	10.0	10.0	10.0	84	84.0	84	84	94	94.0	94	94	0.0	0.0	0.0	0.0	84.0	84.0	84.0	84.0	-359	-359.0	-359	-359	12.0	12.0	12.0	12.0	1	1.0	1
200004	151.0	30.2	46.0	15.0	938	187.6	277	67	987	197.4	274	70	28.0	5.6	28.0	0.0	938.0	187.6	277.0	67.0	-3662	-732.4	-399	-1232	42.0	8.4	12.0	6.0	4	0.8	2
200005	67.0	13.4	18.0	7.0	1043	208.6	360	44	1070	214.0	360	42	12.0	2.4	7.0	0.0	1043.0	208.6	360.0	44.0	-10489	-2097.8	-709	-2754	136.0	27.2	48.0	6.0	5	1.0	2

SK_ID_CURR	PreDK_MISSING_VALUES_TOTAL_PREV_min	PreDK_AMT_DECLINED_sum	PreDK_AMT_DECLINED_mean	PreDK_AMT_DECLINED_max	PreDK_AMT_DECLINED_min	PreDK_AMT_CREDIT_GOODS_RATIO_sum	PreDK_AMT_CREDIT_GOODS_RATIO_mean	PreDK_AMT_CREDIT_GOODS_RATIO_max	PreDK_AMT_CREDIT_GOODS_RATIO_min	PreDK_AMT_CREDIT_GOODS_DIFF_sum	PreDK_AMT_CREDIT_GOODS_DIFF_mean	PreDK_AMT_CREDIT_GOODS_DIFF_max	PreDK_AMT_CREDIT_GOODS_DIFF_min	PreDK_AMT_CREDIT_APPLICATION_RATIO_sum	PreDK_AMT_CREDIT_APPLICATION_RATIO_mean	PreDK_AMT_CREDIT_APPLICATION_RATIO_max	PreDK_AMT_CREDIT_APPLICATION_RATIO_min	PreDK_AMT_INTEREST_sum
200001	2	-90	-90.0	-90	-90	1.200000	1.200000	1.200000	1.200000	90.0	90.0	90.0	90.0	0.833333	0.833333	0.833333	0.833333	792.0
200002	1	77	38.5	54	23	1.700000	0.850000	0.900000	0.800000	-77.0	-38.5	-23.0	-54.0	2.361111	1.180556	1.250000	1.111111	41.0
200003	1	-10	-10.0	-10	-10	1.119048	1.119048	1.119048	1.119048	10.0	10.0	10.0	10.0	0.893617	0.893617	0.893617	0.893617	26.0
200004	0	-49	-9.8	3	-29	5.298684	1.059737	1.161111	0.989170	49.0	9.8	29.0	-3.0	4.735439	0.947088	1.010949	0.861244	213.0
200005	0	-27	-5.4	2	-30	5.072006	1.014401	1.133333	0.954545	27.0	5.4	30.0	-2.0	4.946101	0.989220	1.047619	0.882353	1112.0

SK_ID_CURR	PreDK_AMT_INTEREST_mean	PreDK_AMT_INTEREST_max	PreDK_AMT_INTEREST_min	PreDK_ANNUITY_CREDIT_RATIO_sum	PreDK_ANNUITY_CREDIT_RATIO_mean	PreDK_ANNUITY_CREDIT_RATIO_max	PreDK_ANNUITY_CREDIT_RATIO_min	NAME_CONTRACT_TYPE_modus	NAME_CONTRACT_TYPE_first	NAME_CONTRACT_TYPE_last	NAME_CONTRACT_STATUS_modus	NAME_CONTRACT_STATUS_first	NAME_CONTRACT_STATUS_last	NAME_PAYMENT_TYPE_modus	NAME_PAYMENT_TYPE_first	NAME_PAYMENT_TYPE_last	CODE_REJECT_REASON_modus	CODE_REJECT_REASON_first	CODE_REJECT_REASON_last	NAME_TYPE_SUITE_modus	NAME_TYPE_SUITE_first	NAME_TYPE_SUITE_last	NAME_CLIENT_TYPE_modus	NAME_CLIENT_TYPE_first	NAME_CLIENT_TYPE_last
200001	792.0	792.0	792.0	0.068519	0.068519	0.068519	0.068519	C	C	C	A	A	A	C	C	C	X	X	X	Kategorie_NaN	Kategorie_NaN	Kategorie_NaN	N	N	N
200002	20.5	23.0	18.0	0.291667	0.145833	0.180556	0.111111	C	C	C	A	A	A	C	C	C	X	X	X	Kategorie_NaN	Kategorie_NaN	Kategorie_NaN	N	N	R
200003	26.0	26.0	26.0	0.106383	0.106383	0.106383	0.106383	C	C	C	A	A	A	C	C	C	X	X	X	Kategorie_NaN	Kategorie_NaN	Kategorie_NaN	N	N	N
200004	42.6	55.0	20.0	0.820429	0.164086	0.214286	0.098540	C	C	C	A	A	A	C	C	N	X	X	X	Kategorie_NaN	Kategorie_NaN	Kategorie_NaN	R	N	R
200005	222.4	465.0	6.0	0.469623	0.093925	0.190476	0.047222	C	C	C	A	A	A	X	X	X	X	X	X	Kategorie_NaN	Kategorie_NaN	Kategorie_NaN	R	N	R

```
           NAME_GOODS_CATEGORY_modus NAME_GOODS_CATEGORY_first  \
SK_ID_CURR
200001                             X                        X
200002                             A                        F
200003                             A                        A
200004                             C                        C
200005                             X                        C

           NAME_GOODS_CATEGORY_last NAME_YIELD_GROUP_modus  \
SK_ID_CURR
200001                            X                     h
200002                            A                     l
200003                            A                     m
200004                            M                     h
200005                            X                     m

           NAME_YIELD_GROUP_first NAME_YIELD_GROUP_last  \
SK_ID_CURR
200001                         h                    h
200002                         l                    l
200003                         m                    m
200004                         l                    h
200005                         l                    m

           NFLAG_INSURED_ON_APPROVAL_modus NFLAG_INSURED_ON_APPROVAL_first  \
SK_ID_CURR
200001                                 1.0                             1.0
200002                                 0.0                             0.0
200003                                 0.0                             0.0
200004                                 0.0                             0.0
200005                                 1.0                             1.0

           NFLAG_INSURED_ON_APPROVAL_last
SK_ID_CURR
200001                                1.0
200002                                0.0
200003                                0.0
200004                                0.0
200005                                1.0
```

BUREAU

Dieser Datensatz enthält in den folgenden numerischen Spalten fehlende Werte: AMT_ANNUITY, AMT_CREDIT_MAX_OVERDUE, DAYS_ENDDATE_FACT, AMT_CREDIT_SUM_LIMIT, AMT_CREDIT_SUM_DEBT, DAYS_CREDIT_ENDDATE und AMT_CREDIT_SUM. Diese werden durch 0 ersetzt.

In [70]:
```
print(dfbureau.head())
```

```
   SK_ID_CURR CREDIT_ACTIVE  DAYS_CREDIT  CREDIT_DAY_OVERDUE  \
0      200001             C        -1900                   0
1      200001             C        -1489                   0
2      200001             A         -163                   0
3      200001             A        -1037                   0
4      200002             C         -429                   0

   DAYS_CREDIT_ENDDATE  DAYS_ENDDATE_FACT  AMT_CREDIT_MAX_OVERDUE  \
0              -1170.0            -1292.0                     NaN
1              -1123.0            -1033.0                     NaN
2                 98.0                NaN                     NaN
3                786.0                NaN               17053.965
4               -187.0             -215.0                     NaN

   CNT_CREDIT_PROLONG  AMT_CREDIT_SUM  AMT_CREDIT_SUM_DEBT  \
0                   0           804.0                  0.0
1                   0           404.0                  0.0
2                   0           450.0                457.0
3                   0           918.0                909.0
4                   0           234.0                  0.0

   AMT_CREDIT_SUM_LIMIT  AMT_CREDIT_SUM_OVERDUE  DAYS_CREDIT_UPDATE  \
0                   0.0                       0               -1287
1                   0.0                       0               -1033
2                   0.0                       0                -133
3                   0.0                       0                  -5
4                   NaN                       0                -193

   AMT_ANNUITY
0          0.0
1          0.0
2          NaN
3        115.0
4          NaN
```

In [71]:
```
numerical_columns = ['AMT_ANNUITY','AMT_CREDIT_MAX_OVERDUE','DAYS_ENDDATE_FACT','AMT_CREDIT_SUM_LIMIT',
                     'AMT_CREDIT_SUM_DEBT','DAYS_CREDIT_ENDDATE','AMT_CREDIT_SUM']

dfbureau[numerical_columns] = dfbureau[numerical_columns].fillna(0)
```

In [72]:
```
# Basierend auf Domain-Wissen werden die geforderten Merkmale hinzugefügt.
# Alle neu hinzugefügten (manuellen) Features erhalten den Vorsatz BurDK

dfbureau['BurDK_CREDIT_DURATION'] = np.abs(dfbureau['DAYS_CREDIT'] - dfbureau['DAYS_CREDIT_ENDDATE'])
dfbureau['BurDK_FLAG_OVERDUE_RECENT'] = [0 if ele == 0 else 1 for ele in dfbureau['CREDIT_DAY_OVERDUE']]
dfbureau['BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO'] = dfbureau['AMT_CREDIT_SUM_OVERDUE'] / dfbureau['BurDK_CREDIT_DURATION']
dfbureau['BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO'] = dfbureau['AMT_CREDIT_SUM_OVERDUE'] / dfbureau['DAYS_CREDIT_ENDDATE']
```

```
dfbureau['BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL'] = dfbureau['CNT_CREDIT_PROLONG'] * dfbureau['AMT_CREDIT_MAX_OVERDUE']
dfbureau['BurDK_CNT_PROLONGED_DURATION_RATIO'] = dfbureau['CNT_CREDIT_PROLONG'] / dfbureau['BurDK_CREDIT_DURATION']
dfbureau['BurDK_CURRENT_DEBT_TO_CREDIT_RATIO'] = dfbureau['AMT_CREDIT_SUM_DEBT'] / dfbureau['AMT_CREDIT_SUM']
dfbureau['BurDK_CURRENT_CREDIT_DEBT_DIFF'] = dfbureau['AMT_CREDIT_SUM'] - dfbureau['AMT_CREDIT_SUM_DEBT']
dfbureau['BurDK_AMT_ANNUITY_CREDIT_RATIO'] = dfbureau['AMT_ANNUITY'] / dfbureau['AMT_CREDIT_SUM']
dfbureau['BurDK_CREDIT_ENDDATE_UPDATE_DIFF'] = np.abs(dfbureau['DAYS_CREDIT_UPDATE'] - dfbureau['DAYS_CREDIT_ENDDATE'])

# Falls Divisionen durch Null auftreten, so werden die resultierenden `inf`-Werte durch `NaN` ersetzt
dfbureau[dfbureau == np.infty]  = np.nan
dfbureau[dfbureau == -np.infty] = np.nan
```

In [73]:
```
aggregations_for_bureau = {
            #numerisch
            'DAYS_CREDIT' :            aggregations_num,
            'CREDIT_DAY_OVERDUE' :     aggregations_num,
            'DAYS_CREDIT_ENDDATE' :    aggregations_num,
            'DAYS_ENDDATE_FACT' :      aggregations_num,
            'AMT_CREDIT_MAX_OVERDUE' : aggregations_num,
            'CNT_CREDIT_PROLONG':      aggregations_num,
            'AMT_CREDIT_SUM':          aggregations_num,
            'AMT_CREDIT_SUM_DEBT':     aggregations_num,
            'AMT_CREDIT_SUM_LIMIT':    aggregations_num,
            'AMT_CREDIT_SUM_OVERDUE':  aggregations_num,
            'DAYS_CREDIT_UPDATE':      aggregations_num,
            'AMT_ANNUITY':             aggregations_num,
            #Domain knowledge
            'BurDK_CREDIT_DURATION' :                   aggregations_num,
            'BurDK_FLAG_OVERDUE_RECENT' :               aggregations_num,
            'BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO' : aggregations_num,
            'BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO' :    aggregations_num,
            'BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL' :      aggregations_num,
            'BurDK_CNT_PROLONGED_DURATION_RATIO' :       aggregations_num,
            'BurDK_CURRENT_DEBT_TO_CREDIT_RATIO' :       aggregations_num,
            'BurDK_CURRENT_CREDIT_DEBT_DIFF' :           aggregations_num,
            'BurDK_AMT_ANNUITY_CREDIT_RATIO' :           aggregations_num,
            'BurDK_CREDIT_ENDDATE_UPDATE_DIFF' :         aggregations_num,
            #kategorisch
            'CREDIT_ACTIVE' : aggregations_cat
            }
```

In [74]:
```
# Sortieren der Daten und Gruppieren nach SK_ID_CURR
dfbureau = dfbureau.sort_values(by = ['SK_ID_CURR','DAYS_CREDIT_ENDDATE'])
dfbureau_agg = dfbureau.groupby('SK_ID_CURR').agg(aggregations_for_bureau)
```

In [75]:
```
# Spaltennamen korrigieren
dfbureau_agg.columns = ['_'.join(col) for col in dfbureau_agg.columns.values]
```

In [76]:
```
print(dfbureau_agg.head())
```

SK_ID_CURR	DAYS_CREDIT_sum	DAYS_CREDIT_mean	DAYS_CREDIT_max	DAYS_CREDIT_min	CREDIT_DAY_OVERDUE_sum	CREDIT_DAY_OVERDUE_mean	CREDIT_DAY_OVERDUE_max	CREDIT_DAY_OVERDUE_min
200001	-4589	-1147.25	-163	-1900	0	0.0	0	0
200002	-429	-429.00	-429	-429	0	0.0	0	0
200003	-11250	-1875.00	-455	-2664	0	0.0	0	0
200004	-1780	-1780.00	-1780	-1780	0	0.0	0	0
200006	-1003	-1003.00	-1003	-1003	0	0.0	0	0

SK_ID_CURR	DAYS_CREDIT_ENDDATE_sum	DAYS_CREDIT_ENDDATE_mean	DAYS_CREDIT_ENDDATE_max	DAYS_CREDIT_ENDDATE_min
200001	-1409.0	-352.250000	786.0	-1170.0
200002	-187.0	-187.000000	-187.0	-187.0
200003	-4385.0	-730.833333	1371.0	-2223.0
200004	-1415.0	-1415.000000	-1415.0	-1415.0
200006	-819.0	-819.000000	-819.0	-819.0

SK_ID_CURR	DAYS_ENDDATE_FACT_sum	DAYS_ENDDATE_FACT_mean	DAYS_ENDDATE_FACT_max	DAYS_ENDDATE_FACT_min
200001	-2325.0	-581.25	0.0	-1292.0
200002	-215.0	-215.00	-215.0	-215.0
200003	-5514.0	-919.00	0.0	-2252.0
200004	-1444.0	-1444.00	-1444.0	-1444.0
200006	-910.0	-910.00	-910.0	-910.0

SK_ID_CURR	AMT_CREDIT_MAX_OVERDUE_sum	AMT_CREDIT_MAX_OVERDUE_mean	AMT_CREDIT_MAX_OVERDUE_max	AMT_CREDIT_MAX_OVERDUE_min
200001	17053.965	4263.49125	17053.965	0.0
200002	0.000	0.00000	0.000	0.0
200003	0.000	0.00000	0.000	0.0
200004	0.000	0.00000	0.000	0.0
200006	0.000	0.00000	0.000	0.0

SK_ID_CURR	CNT_CREDIT_PROLONG_sum	CNT_CREDIT_PROLONG_mean	CNT_CREDIT_PROLONG_max	CNT_CREDIT_PROLONG_min
200001	0	0.0	0	0
200002	0	0.0	0	0
200003	0	0.0	0	0
200004	0	0.0	0	0
200006	0	0.0	0	0

SK_ID_CURR	AMT_CREDIT_SUM_sum	AMT_CREDIT_SUM_mean	AMT_CREDIT_SUM_max
200001	2576.0	644.000000	918.0
200002	234.0	234.000000	234.0
200003	3710.0	618.333333	1777.0
200004	450.0	450.000000	450.0
200006	140.0	140.000000	140.0

SK_ID_CURR	AMT_CREDIT_SUM_min	AMT_CREDIT_SUM_DEBT_sum	AMT_CREDIT_SUM_DEBT_mean	AMT_CREDIT_SUM_DEBT_max	AMT_CREDIT_SUM_DEBT_min	AMT_CREDIT_SUM_LIMIT_sum	AMT_CREDIT_SUM_LIMIT_mean	AMT_CREDIT_SUM_LIMIT_max	AMT_CREDIT_SUM_LIMIT_min	AMT_CREDIT_SUM_OVERDUE_sum	AMT_CREDIT_SUM_OVERDUE_mean	AMT_CREDIT_SUM_OVERDUE_max	AMT_CREDIT_SUM_OVERDUE_min	DAYS_CREDIT_UPDATE_sum	DAYS_CREDIT_UPDATE_mean	DAYS_CREDIT_UPDATE_max	DAYS_CREDIT_UPDATE_min	AMT_ANNUITY_sum	AMT_ANNUITY_mean	AMT_ANNUITY_max	AMT_ANNUITY_min	BurDK_CREDIT_DURATION_sum	BurDK_CREDIT_DURATION_mean	BurDK_CREDIT_DURATION_max	BurDK_CREDIT_DURATION_min	BurDK_FLAG_OVERDUE_RECENT_sum
200001	404.0	1366.0	341.500000	909.0	0.0	0.00	0.00	0.00	0.0	0	0.0	0	0	-2458	-614.500000	-5	-1287	115.0	28.75	115.0	0.0	3180.0	795.000000	1823.0	261.0	0
200002	234.0	0.0	0.000000	0.0	0.0	0.00	0.00	0.00	0.0	0	0.0	0	0	-193	-193.000000	-193	-193	0.0	0.00	0.0	0.0	242.0	242.000000	242.0	242.0	0
200003	90.0	1504.0	250.666667	1504.0	0.0	225039.24	37506.54	225039.24	0.0	0	0.0	0	0	-6124	-1020.666667	-29	-2252	0.0	0.00	0.0	0.0	6865.0	1144.166667	1899.0	213.0	0
200004	450.0	0.0	0.000000	0.0	0.0	0.00	0.00	0.00	0.0	0	0.0	0	0	-386	-386.000000	-386	-386	0.0	0.00	0.0	0.0	365.0	365.000000	365.0	365.0	0
200006	140.0	0.0	0.000000	0.0	0.0	0.00	0.00	0.00	0.0	0	0.0	0	0	-908	-908.000000	-908	-908	0.0	0.00	0.0	0.0	184.0	184.000000	184.0	184.0	0

SK_ID_CURR	BurDK_FLAG_OVERDUE_RECENT_mean	BurDK_FLAG_OVERDUE_RECENT_max	BurDK_FLAG_OVERDUE_RECENT_min
200001	0.0	0	0
200002	0.0	0	0
200003	0.0	0	0
200004	0.0	0	0
200006	0.0	0	0

SK_ID_CURR	BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_sum	BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_mean	BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_max	BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_min
200001	0.0	0.0	0.0	0.0
200002	0.0	0.0	0.0	0.0
200003	0.0	0.0	0.0	0.0
200004	0.0	0.0	0.0	0.0
200006	0.0	0.0	0.0	0.0

SK_ID_CURR	BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_sum	BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_mean	BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_max	BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_min
200001	0.0	0.0	-0.0	-0.0
200002	0.0	0.0	-0.0	-0.0
200003	0.0	0.0	-0.0	-0.0
200004	0.0	0.0	-0.0	-0.0
200006	0.0	0.0	-0.0	-0.0

SK_ID_CURR	BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_sum	BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_mean
200001	0.0	0.0
200002	0.0	0.0
200003	0.0	0.0
200004	0.0	0.0
200006	0.0	0.0

SK_ID_CURR	BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_max	BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_min	BurDK_CNT_PROLONGED_DURATION_RATIO_sum	BurDK_CNT_PROLONGED_DURATION_RATIO_mean	BurDK_CNT_PROLONGED_DURATION_RATIO_max	BurDK_CNT_PROLONGED_DURATION_RATIO_min	BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_sum	BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_mean	BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_max	BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_min	BurDK_CURRENT_CREDIT_DEBT_DIFF_sum	BurDK_CURRENT_CREDIT_DEBT_DIFF_mean
200001	0.0	0.0	0.0	0.0	0.0	0.0	2.005752	0.501438	1.015556	0.0	1210.0	302.500000
200002	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	234.0	234.000000
200003	0.0	0.0	0.0	0.0	0.0	0.0	0.846370	0.141062	0.846370	0.0	2206.0	367.666667
200004	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	450.0	450.000000
200006	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	140.0	140.000000

```
            BurDK_CURRENT_CREDIT_DEBT_DIFF_max  \
SK_ID_CURR
200001                                   804.0
200002                                   234.0
200003                                  1080.0
200004                                   450.0
200006                                   140.0

            BurDK_CURRENT_CREDIT_DEBT_DIFF_min  \
SK_ID_CURR
200001                                    -7.0
200002                                   234.0
200003                                    90.0
200004                                   450.0
200006                                   140.0

            BurDK_AMT_ANNUITY_CREDIT_RATIO_sum  \
SK_ID_CURR
200001                                0.125272
200002                                0.000000
200003                                0.000000
200004                                0.000000
200006                                0.000000

            BurDK_AMT_ANNUITY_CREDIT_RATIO_mean  \
SK_ID_CURR
200001                                 0.031318
200002                                 0.000000
200003                                 0.000000
200004                                 0.000000
200006                                 0.000000

            BurDK_AMT_ANNUITY_CREDIT_RATIO_max  \
SK_ID_CURR
200001                                0.125272
200002                                0.000000
200003                                0.000000
200004                                0.000000
200006                                0.000000

            BurDK_AMT_ANNUITY_CREDIT_RATIO_min  \
SK_ID_CURR
200001                                   0.0
200002                                   0.0
200003                                   0.0
200004                                   0.0
200006                                   0.0

            BurDK_CREDIT_ENDDATE_UPDATE_DIFF_sum  \
SK_ID_CURR
200001                                  1229.0
200002                                     6.0
200003                                  1811.0
200004                                  1029.0
200006                                    89.0

            BurDK_CREDIT_ENDDATE_UPDATE_DIFF_mean  \
SK_ID_CURR
200001                                307.250000
200002                                  6.000000
200003                                301.833333
200004                               1029.000000
200006                                 89.000000

            BurDK_CREDIT_ENDDATE_UPDATE_DIFF_max  \
SK_ID_CURR
200001                                   791.0
200002                                     6.0
200003                                  1400.0
200004                                  1029.0
200006                                    89.0

            BurDK_CREDIT_ENDDATE_UPDATE_DIFF_min CREDIT_ACTIVE_modus  \
SK_ID_CURR
200001                                    90.0                  A
200002                                     6.0                  C
200003                                     0.0                  C
200004                                  1029.0                  C
200006                                    89.0                  C

           CREDIT_ACTIVE_first CREDIT_ACTIVE_last
SK_ID_CURR
200001                       C                 A
200002                       C                 C
200003                       C                 A
200004                       C                 C
200006                       C                 C
```

APPLICATION_TRAIN

Die fehlenden numerischen Werte werden durch 0 ersetzt.

Die beiden kategoriellen Merkmale (OCCUPATION_TYPE und NAME_TYPE_SUITE) enthalten eine eigene Kategorie.

In diesem Datensatz ist die ID SK_ID_CURR eindeutig, Aggregationen werden daher nicht vorgenommen.

In [77]:
```
print(dfapptraintest.head())
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
0	200004	C	M	N	Y
1	200011	C	F	N	Y
2	200015	C	M	Y	N
3	200026	C	F	N	Y
4	200028	C	M	N	Y

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
0	1	405	204	22.0	180.0
1	0	430	1186	61.0	900.0
2	0	540	830	46.0	594.0
3	0	270	1279	55.0	1017.0
4	1	540	384	41.0	360.0

	NAME_TYPE_SUITE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS
0	U	C	H	M
1	U	C	S	C
2	U	W	I	C
3	U	P	S	W
4	F	C	H	M

	NAME_HOUSING_TYPE	REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_EMPLOYED
0	H	0.046	-10525	-621.0
1	H	0.020	-19606	-1782.0
2	H	0.023	-11057	-668.0
3	H	0.019	-23307	NaN
4	H	0.046	-10503	-895.0

	DAYS_REGISTRATION	DAYS_ID_PUBLISH	OWN_CAR_AGE	FLAG_MOBIL	FLAG_EMP_PHONE
0	-612	-3039	NaN	1	1
1	-5601	-3166	NaN	1	1
2	-408	-3738	1.0	1	1
3	-12897	-4624	NaN	1	0
4	-4419	-566	NaN	1	1

	FLAG_WORK_PHONE	FLAG_CONT_MOBILE	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE
0	0	1	0	0	H
1	0	1	0	0	C
2	1	1	0	0	L
3	0	1	1	1	NaN
4	0	1	0	0	NaN

	CNT_FAM_MEMBERS	REGION_RATING_CLIENT	REGION_RATING_CLIENT_W_CITY
0	3.0	1	1
1	2.0	3	3
2	2.0	2	2
3	1.0	2	2
4	3.0	1	1

	REG_REGION_NOT_LIVE_REGION	REG_REGION_NOT_WORK_REGION
0	0	0
1	0	0
2	0	1
3	0	0
4	0	0

	LIVE_REGION_NOT_WORK_REGION	REG_CITY_NOT_LIVE_CITY	REG_CITY_NOT_WORK_CITY
0	0	0	0
1	0	0	0
2	1	1	1
3	0	0	0
4	0	0	1

	LIVE_CITY_NOT_WORK_CITY	ORGANIZATION_TYPE	EXT_SOURCE_1	EXT_SOURCE_2
0	0	B	0.605	0.631
1	0	S	NaN	0.466
2	1	I	NaN	0.225
3	0	X	NaN	0.764
4	1	B	NaN	0.286

	EXT_SOURCE_3	DAYS_LAST_PHONE_CHANGE	AMT_REQ_CREDIT_BUREAU_HOUR
0	0.671	0.0	0.0
1	0.351	-437.0	0.0
2	0.354	-972.0	0.0
3	0.703	-1706.0	0.0
4	0.281	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_WEEK
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_MON	AMT_REQ_CREDIT_BUREAU_QRT
0	0.0	0.0
1	0.0	0.0
2	0.0	2.0
3	0.0	0.0
4	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_YEAR	TARGET	TYPE
0	2.0	0	TEST
1	0.0	0	TEST
2	2.0	0	TEST

```
3                       1.0      0  TEST
4                       1.0      0  TEST
```

In [78]:
```
# Zusätzlich zu den in Aufgabe A-2 gefundenen Spalten mit NaNs muss auch noch bei `DAYS_EMPLOYED` imputiert werden, da wir hier
# den Wert 365243 durch NaN ersetzt hatten
numerical_columns = ['OWN_CAR_AGE','AMT_GOODS_PRICE','AMT_ANNUITY','DAYS_EMPLOYED', 'DAYS_LAST_PHONE_CHANGE', 'EXT_SOURCE_1',
                     'EXT_SOURCE_3','EXT_SOURCE_2','AMT_REQ_CREDIT_BUREAU_HOUR','AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CREDIT_BUREAU_WEEK',
                     'AMT_REQ_CREDIT_BUREAU_MON','AMT_REQ_CREDIT_BUREAU_QRT','AMT_REQ_CREDIT_BUREAU_YEAR','CNT_FAM_MEMBERS']
dfapptraintest[numerical_columns] = dfapptraintest[numerical_columns].fillna(0)

dfapptraintest['OCCUPATION_TYPE'] = dfapptraintest['OCCUPATION_TYPE'].cat.add_categories("Kategorie_NaN").fillna("Kategorie_NaN")
dfapptraintest['NAME_TYPE_SUITE'] = dfapptraintest['NAME_TYPE_SUITE'].cat.add_categories("Kategorie_NaN").fillna("Kategorie_NaN")
```

In [79]:
```
# Basierend auf Domain-Wissen werden die geforderten Merkmale hinzugefügt.
# Alle neu hinzugefügten (manuellen) Features erhalten den Vorsatz AppDK

# Verhältnis zu AMT_CREDIT
dfapptraintest['AppDK_ANNUITY_CREDIT_RATIO'] = dfapptraintest['AMT_ANNUITY'] / dfapptraintest['AMT_CREDIT']
dfapptraintest['AppDK_GOODS_CREDIT_RATIO']   = dfapptraintest['AMT_GOODS_PRICE'] / dfapptraintest['AMT_CREDIT']

# Verhältnis zu AMT_INCOME_TOTAL
dfapptraintest['AppDK_ANNUITY_INCOME_RATIO'] = dfapptraintest['AMT_ANNUITY'] / dfapptraintest['AMT_INCOME_TOTAL']
dfapptraintest['AppDK_CREDIT_INCOME_RATIO']  = dfapptraintest['AMT_CREDIT'] / dfapptraintest['AMT_INCOME_TOTAL']
dfapptraintest['AppDK_GOODS_INCOME_RATIO']   = dfapptraintest['AMT_GOODS_PRICE'] / dfapptraintest['AMT_INCOME_TOTAL']
dfapptraintest['AppDK_CNT_FAM_INCOME_RATIO'] = dfapptraintest['AMT_INCOME_TOTAL'] / dfapptraintest['CNT_FAM_MEMBERS']

# Verhältnisse zu DAYS_BIRTH, DAYS_EMPLOYED
dfapptraintest['AppDK_EMPLOYED_BIRTH_RATIO']  = dfapptraintest['DAYS_EMPLOYED'] / dfapptraintest['DAYS_BIRTH']
dfapptraintest['AppDK_INCOME_EMPLOYED_RATIO'] = dfapptraintest['AMT_INCOME_TOTAL'] / dfapptraintest['DAYS_EMPLOYED']
dfapptraintest['AppDK_INCOME_BIRTH_RATIO']    = dfapptraintest['AMT_INCOME_TOTAL'] / dfapptraintest['DAYS_BIRTH']
dfapptraintest['AppDK_CAR_BIRTH_RATIO']       = dfapptraintest['OWN_CAR_AGE'] / dfapptraintest['DAYS_BIRTH']
dfapptraintest['AppDK_CAR_EMPLOYED_RATIO']    = dfapptraintest['OWN_CAR_AGE'] / dfapptraintest['DAYS_EMPLOYED']

# Falls Divisionen durch Null auftreten, so werden die resultierenden `inf`-Werte durch `NaN` ersetzt
dfapptraintest[dfapptraintest == np.infty]  = np.nan
dfapptraintest[dfapptraintest == -np.infty] = np.nan
```

4 b)

Lösungsvorschlag:

Man sollte darauf achten, Merkmale hinzuzufügen, die einen zusätzlichen Informationsgehalt haben und die nicht zu stark mit bestehenden Merkmalen korrelieren, um Overfitting zu vermeiden. Auch die Erklärbarkeit von Modellen kann leiden, wenn die eingeführten neuen Merkmale keinen eigenen Informationsgehalt haben. Dies gilt es zu vermeiden.

Mit Korrelationsmatrizen ist es möglich, die neu eingefügten Merkmale qualitätszusichern.

Aufgabe A-5: Encoding [Lernziel 3.3, 3.4; 5 Punkte]

Erläutern Sie den Nutzen von Encoding-Verfahren im Rahmen des überwachten maschinellen Lernens. Beschreiben Sie sowohl das Verfahren „One-Hot-Encoding“ als auch das Verfahren „Label encoding“ und gehen Sie dabei für beide Verfahren auf je einen Vor- und Nachteil ein.

Anschließend sollen die kategoriellen Merkmale der drei aggregierten Datensätze aus der vorherigen Aufgabe wie folgt aufbereitet werden:

- **Für alle Features mit höchstens zwei Kategorien soll ein Label-Encoding durchgeführt werden.**
- **Für alle Features mit mehr als zwei Kategorien soll das One-Hot-Encoding durchgeführt werden.**

Lösungsvorschlag:

Viele Verfahren des maschinellen Lernens benötigen für die Verarbeitung numerische Werte (eine Ausnahme ist z.B. LightGBM). Daher müssen die kategoriellen Werte vor der Verarbeitung numerisch kodiert werden. Dafür gibt es mehrere Verfahren:

- One-hot encoding: Für jede Kategorie eines Features wird eine eigene Spalte angelegt. Nimmt ein Feature z.B. den Wert der Kategorie A an, so wird in der zugehörigen Spalte eine 1 eingesetzt und eine 0 in allen anderen Spalten.
- Label encoding: Jede Kategorie wird mit einem Integer versehen. Sind die beiden Kategorien eines Features z.B. A, B und C, dann könnte man z.B. A mit 0, B mit 1 und C mit 2 versehen.
- ...

Ein Vorteil des Label encoding ist die Einfachheit und die Tatsache, dass für die Codierung nur eine zusätzliche Spalte benötigt wird. Allerdings ist der zugewiesene Wert willkürlich. Zudem kann ein darauf basierendes Modells den hohen Kategorien (z.B. C mit der Codierung 2) entsprechende Gewichtungen zuordnen, was ggf. nicht gewünscht ist. Mit der One-hot Codierung wird dieses Problem umgangen. Problematisch ist allerdings die Tatsache, dass die Anzahl der Features sehr schnell sehr groß werden kann. Daher werden typischerweise im Anschluss Verfahren zur Dimensionsreduktion (z.B. PCA) druchgeführt.

In Teil 2 wurden bereits die Anzahlen der kategoriellen Variablen in den drei Datensätzen ermittelt: 11 (application_train), 8 (previous_application) und 1 (bureau).

In [80]:
```
print(dfapptraintest.head())
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
0	200004	C	M	N	Y
1	200011	C	F	N	Y
2	200015	C	M	Y	N
3	200026	C	F	N	Y
4	200028	C	M	N	Y

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
0	1	405	204	22.0	180.0
1	0	430	1186	61.0	900.0
2	0	540	830	46.0	594.0
3	0	270	1279	55.0	1017.0
4	1	540	384	41.0	360.0

	NAME_TYPE_SUITE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS
0	U	C	H	M
1	U	C	S	C
2	U	W	I	C
3	U	P	S	W
4	F	C	H	M

	NAME_HOUSING_TYPE	REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_EMPLOYED
0	H	0.046	-10525	-621.0
1	H	0.020	-19606	-1782.0
2	H	0.023	-11057	-668.0
3	H	0.019	-23307	0.0
4	H	0.046	-10503	-895.0

	DAYS_REGISTRATION	DAYS_ID_PUBLISH	OWN_CAR_AGE	FLAG_MOBIL	FLAG_EMP_PHONE
0	-612	-3039	0.0	1	1
1	-5601	-3166	0.0	1	1
2	-408	-3738	1.0	1	1
3	-12897	-4624	0.0	1	0
4	-4419	-566	0.0	1	1

	FLAG_WORK_PHONE	FLAG_CONT_MOBILE	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE
0	0	1	0	0	H
1	0	1	0	0	C
2	1	1	0	0	L
3	0	1	1	1	Kategorie_NaN
4	0	1	0	0	Kategorie_NaN

	CNT_FAM_MEMBERS	REGION_RATING_CLIENT	REGION_RATING_CLIENT_W_CITY
0	3.0	1	1
1	2.0	3	3
2	2.0	2	2
3	1.0	2	2
4	3.0	1	1

	REG_REGION_NOT_LIVE_REGION	REG_REGION_NOT_WORK_REGION
0	0	0
1	0	0
2	0	1
3	0	0
4	0	0

	LIVE_REGION_NOT_WORK_REGION	REG_CITY_NOT_LIVE_CITY	REG_CITY_NOT_WORK_CITY
0	0	0	0
1	0	0	0
2	1	1	1
3	0	0	0
4	0	0	1

	LIVE_CITY_NOT_WORK_CITY	ORGANIZATION_TYPE	EXT_SOURCE_1	EXT_SOURCE_2
0	0	B	0.605	0.631
1	0	S	0.000	0.466
2	1	I	0.000	0.225
3	0	X	0.000	0.764
4	1	B	0.000	0.286

	EXT_SOURCE_3	DAYS_LAST_PHONE_CHANGE	AMT_REQ_CREDIT_BUREAU_HOUR
0	0.671	0.0	0.0
1	0.351	-437.0	0.0
2	0.354	-972.0	0.0
3	0.703	-1706.0	0.0
4	0.281	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_WEEK
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_MON	AMT_REQ_CREDIT_BUREAU_QRT
0	0.0	0.0
1	0.0	0.0
2	0.0	2.0
3	0.0	0.0
4	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_YEAR	TARGET	TYPE	AppDK_ANNUITY_CREDIT_RATIO
0	2.0	0	TEST	0.107843
1	0.0	0	TEST	0.051433
2	2.0	0	TEST	0.055422
3	1.0	0	TEST	0.043002

```
4                       1.0      0  TEST                   0.106771

   AppDK_GOODS_CREDIT_RATIO  AppDK_ANNUITY_INCOME_RATIO  \
0                  0.882353                    0.054321
1                  0.758853                    0.141860
2                  0.715663                    0.085185
3                  0.795152                    0.203704
4                  0.937500                    0.075926

   AppDK_CREDIT_INCOME_RATIO  AppDK_GOODS_INCOME_RATIO  \
0                   0.503704                  0.444444
1                   2.758140                  2.093023
2                   1.537037                  1.100000
3                   4.737037                  3.766667
4                   0.711111                  0.666667

   AppDK_CNT_FAM_INCOME_RATIO  AppDK_EMPLOYED_BIRTH_RATIO  \
0                       135.0                    0.059002
1                       215.0                    0.090891
2                       270.0                    0.060414
3                       270.0                   -0.000000
4                       180.0                    0.085214

   AppDK_INCOME_EMPLOYED_RATIO  AppDK_INCOME_BIRTH_RATIO  \
0                    -0.652174                 -0.038480
1                    -0.241302                 -0.021932
2                    -0.808383                 -0.048838
3                          NaN                 -0.011585
4                    -0.603352                 -0.051414

   AppDK_CAR_BIRTH_RATIO  AppDK_CAR_EMPLOYED_RATIO
0               -0.00000                 -0.000000
1               -0.00000                 -0.000000
2               -0.00009                 -0.001497
3               -0.00000                       NaN
4               -0.00000                 -0.000000
```

In [81]:
```python
# application_traintest:
labelenc = LabelEncoder()
count = 0
columns_to_exclude = ['TARGET', 'TYPE']

columns_OHE = []
for col in dfapptraintest:
    if (dfapptraintest[col].dtype == 'category') and (col not in columns_to_exclude):
        if len(list(dfapptraintest[col].unique())) <= 2:
                labelenc.fit(dfapptraintest[col])
                dfapptraintest[col] = labelenc.transform(dfapptraintest[col])
                count += 1
        else:
            columns_OHE.append(col)

print('Label encoding: Insgesamt für ' + str(count) + ' Spalten.')

dfapptraintest = pd.get_dummies(dfapptraintest, columns=columns_OHE)
```

```
Label encoding: Insgesamt für 16 Spalten.
```

In [82]:
```python
print(dfapptraintest.head())
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
0	200004	0	1	0	1
1	200011	0	0	0	1
2	200015	0	1	1	0
3	200026	0	0	0	1
4	200028	0	1	0	1

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
0	1	405	204	22.0	180.0
1	0	430	1186	61.0	900.0
2	0	540	830	46.0	594.0
3	0	270	1279	55.0	1017.0
4	1	540	384	41.0	360.0

	REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_EMPLOYED	DAYS_REGISTRATION
0	0.046	-10525	-621.0	-612
1	0.020	-19606	-1782.0	-5601
2	0.023	-11057	-668.0	-408
3	0.019	-23307	0.0	-12897
4	0.046	-10503	-895.0	-4419

	DAYS_ID_PUBLISH	OWN_CAR_AGE	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE
0	-3039	0.0	1	1	0
1	-3166	0.0	1	1	0
2	-3738	1.0	1	1	1
3	-4624	0.0	1	0	0
4	-566	0.0	1	1	0

	FLAG_CONT_MOBILE	FLAG_PHONE	FLAG_EMAIL	CNT_FAM_MEMBERS
0	1	0	0	3.0
1	1	0	0	2.0
2	1	0	0	2.0
3	1	1	1	1.0
4	1	0	0	3.0

	REG_REGION_NOT_LIVE_REGION	REG_REGION_NOT_WORK_REGION
0	0	0
1	0	0
2	0	1
3	0	0
4	0	0

	LIVE_REGION_NOT_WORK_REGION	REG_CITY_NOT_LIVE_CITY
0	0	0
1	0	0
2	1	1
3	0	0
4	0	0

	REG_CITY_NOT_WORK_CITY	LIVE_CITY_NOT_WORK_CITY	EXT_SOURCE_1
0	0	0	0.605
1	0	0	0.000
2	1	1	0.000
3	0	0	0.000
4	1	1	0.000

	EXT_SOURCE_2	EXT_SOURCE_3	DAYS_LAST_PHONE_CHANGE
0	0.631	0.671	0.0
1	0.466	0.351	-437.0
2	0.225	0.354	-972.0
3	0.764	0.703	-1706.0
4	0.286	0.281	0.0

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_WEEK	AMT_REQ_CREDIT_BUREAU_MON
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_QRT	AMT_REQ_CREDIT_BUREAU_YEAR	TARGET	TYPE
0	0.0	2.0	0	TEST
1	0.0	0.0	0	TEST
2	2.0	2.0	0	TEST
3	0.0	1.0	0	TEST
4	0.0	1.0	0	TEST

	AppDK_ANNUITY_CREDIT_RATIO	AppDK_GOODS_CREDIT_RATIO
0	0.107843	0.882353
1	0.051433	0.758853
2	0.055422	0.715663
3	0.043002	0.795152
4	0.106771	0.937500

	AppDK_ANNUITY_INCOME_RATIO	AppDK_CREDIT_INCOME_RATIO
0	0.054321	0.503704
1	0.141860	2.758140
2	0.085185	1.537037
3	0.203704	4.737037

```
4                   0.075926                  0.711111

   AppDK_GOODS_INCOME_RATIO  AppDK_CNT_FAM_INCOME_RATIO  \
0                  0.444444                       135.0
1                  2.093023                       215.0
2                  1.100000                       270.0
3                  3.766667                       270.0
4                  0.666667                       180.0

   AppDK_EMPLOYED_BIRTH_RATIO  AppDK_INCOME_EMPLOYED_RATIO  \
0                    0.059002                    -0.652174
1                    0.090891                    -0.241302
2                    0.060414                    -0.808383
3                   -0.000000                          NaN
4                    0.085214                    -0.603352

   AppDK_INCOME_BIRTH_RATIO  AppDK_CAR_BIRTH_RATIO  AppDK_CAR_EMPLOYED_RATIO  \
0                 -0.038480               -0.00000                 -0.000000
1                 -0.021932               -0.00000                 -0.000000
2                 -0.048838               -0.00009                 -0.001497
3                 -0.011585               -0.00000                       NaN
4                 -0.051414               -0.00000                 -0.000000

   NAME_TYPE_SUITE_C  NAME_TYPE_SUITE_F  NAME_TYPE_SUITE_G  NAME_TYPE_SUITE_O  \
0                  0                  0                  0                  0
1                  0                  0                  0                  0
2                  0                  0                  0                  0
3                  0                  0                  0                  0
4                  0                  1                  0                  0

   NAME_TYPE_SUITE_S  NAME_TYPE_SUITE_U  NAME_TYPE_SUITE_Kategorie_NaN  \
0                  0                  1                              0
1                  0                  1                              0
2                  0                  1                              0
3                  0                  1                              0
4                  0                  0                              0

   NAME_INCOME_TYPE_B  NAME_INCOME_TYPE_C  NAME_INCOME_TYPE_M  \
0                   0                   1                   0
1                   0                   1                   0
2                   0                   0                   0
3                   0                   0                   0
4                   0                   1                   0

   NAME_INCOME_TYPE_P  NAME_INCOME_TYPE_S  NAME_INCOME_TYPE_U  \
0                   0                   0                   0
1                   0                   0                   0
2                   0                   0                   0
3                   1                   0                   0
4                   0                   0                   0

   NAME_INCOME_TYPE_W  NAME_EDUCATION_TYPE_A  NAME_EDUCATION_TYPE_H  \
0                   0                      0                      1
1                   0                      0                      0
2                   1                      0                      0
3                   0                      0                      0
4                   0                      0                      1

   NAME_EDUCATION_TYPE_I  NAME_EDUCATION_TYPE_L  NAME_EDUCATION_TYPE_S  \
0                      0                      0                      0
1                      0                      0                      1
2                      1                      0                      0
3                      0                      0                      1
4                      0                      0                      0

   NAME_FAMILY_STATUS_C  NAME_FAMILY_STATUS_M  NAME_FAMILY_STATUS_S  \
0                     0                     1                     0
1                     1                     0                     0
2                     1                     0                     0
3                     0                     0                     0
4                     0                     1                     0

   NAME_FAMILY_STATUS_U  NAME_FAMILY_STATUS_W  NAME_HOUSING_TYPE_C  \
0                     0                     0                    0
1                     0                     0                    0
2                     0                     0                    0
3                     0                     1                    0
4                     0                     0                    0

   NAME_HOUSING_TYPE_H  NAME_HOUSING_TYPE_M  NAME_HOUSING_TYPE_O  \
0                    1                    0                    0
1                    1                    0                    0
2                    1                    0                    0
3                    1                    0                    0
4                    1                    0                    0

   NAME_HOUSING_TYPE_R  NAME_HOUSING_TYPE_W  OCCUPATION_TYPE_A  \
0                    0                    0                  0
1                    0                    0                  0
2                    0                    0                  0
3                    0                    0                  0
4                    0                    0                  0

   OCCUPATION_TYPE_C  OCCUPATION_TYPE_D  OCCUPATION_TYPE_H  OCCUPATION_TYPE_I  \
0                  0                  0                  1                  0
1                  1                  0                  0                  0
```

```
2                 0                 0                 0                 0
3                 0                 0                 0                 0
4                 0                 0                 0                 0

   OCCUPATION_TYPE_L  OCCUPATION_TYPE_M  OCCUPATION_TYPE_P  OCCUPATION_TYPE_R  \
0                  0                  0                  0                  0
1                  0                  0                  0                  0
2                  1                  0                  0                  0
3                  0                  0                  0                  0
4                  0                  0                  0                  0

   OCCUPATION_TYPE_S  OCCUPATION_TYPE_W  OCCUPATION_TYPE_Kategorie_NaN  \
0                  0                  0                              0
1                  0                  0                              0
2                  0                  0                              0
3                  0                  0                              1
4                  0                  0                              1

   REGION_RATING_CLIENT_1  REGION_RATING_CLIENT_2  REGION_RATING_CLIENT_3  \
0                       1                       0                       0
1                       0                       0                       1
2                       0                       1                       0
3                       0                       1                       0
4                       1                       0                       0

   REGION_RATING_CLIENT_W_CITY_1  REGION_RATING_CLIENT_W_CITY_2  \
0                              1                              0
1                              0                              0
2                              0                              1
3                              0                              1
4                              1                              0

   REGION_RATING_CLIENT_W_CITY_3  ORGANIZATION_TYPE_A  ORGANIZATION_TYPE_B  \
0                              0                   0                   1
1                              1                   0                   0
2                              0                   0                   0
3                              0                   0                   0
4                              0                   0                   1

   ORGANIZATION_TYPE_C  ORGANIZATION_TYPE_E  ORGANIZATION_TYPE_G  \
0                    0                    0                    0
1                    0                    0                    0
2                    0                    0                    0
3                    0                    0                    0
4                    0                    0                    0

   ORGANIZATION_TYPE_H  ORGANIZATION_TYPE_I  ORGANIZATION_TYPE_K  \
0                    0                    0                    0
1                    0                    0                    0
2                    0                    1                    0
3                    0                    0                    0
4                    0                    0                    0

   ORGANIZATION_TYPE_L  ORGANIZATION_TYPE_M  ORGANIZATION_TYPE_O  \
0                    0                    0                    0
1                    0                    0                    0
2                    0                    0                    0
3                    0                    0                    0
4                    0                    0                    0

   ORGANIZATION_TYPE_P  ORGANIZATION_TYPE_R  ORGANIZATION_TYPE_S  \
0                    0                    0                    0
1                    0                    0                    1
2                    0                    0                    0
3                    0                    0                    0
4                    0                    0                    0

   ORGANIZATION_TYPE_T  ORGANIZATION_TYPE_U  ORGANIZATION_TYPE_X
0                    0                    0                    0
1                    0                    0                    0
2                    0                    0                    0
3                    0                    0                    1
4                    0                    0                    0
```

In [83]:

```
# previous_application:
labelenc = LabelEncoder()
count = 0

for col in dfprevapp_agg:
    if dfprevapp_agg[col].dtype == 'category':
        if len(list(dfprevapp_agg[col].unique())) <= 2:
                labelenc.fit(dfprevapp_agg[col])
                dfprevapp_agg[col] = labelenc.transform(dfprevapp_agg[col])
                count += 1

print('Es wurden insgesamt ' + str(count) + ' Spalten codiert')

# One-hot encoding für die restlichen kategoriellen Variablen:
dfprevapp_agg = pd.get_dummies(dfprevapp_agg)

print(dfprevapp_agg.head())
```

Es wurden insgesamt 0 Spalten codiert

SK_ID_CURR	AMT_ANNUITY_sum	AMT_ANNUITY_mean	AMT_ANNUITY_max
200001	37.0	37.0	37.0
200002	62.0	31.0	39.0
200003	10.0	10.0	10.0
200004	151.0	30.2	46.0
200005	67.0	13.4	18.0

SK_ID_CURR	AMT_ANNUITY_min	AMT_APPLICATION_sum	AMT_APPLICATION_mean
200001	37.0	450	450.0
200002	23.0	500	250.0
200003	10.0	84	84.0
200004	15.0	938	187.6
200005	7.0	1043	208.6

SK_ID_CURR	AMT_APPLICATION_max	AMT_APPLICATION_min	AMT_CREDIT_sum
200001	450	450	540
200002	270	230	423
200003	84	84	94
200004	277	67	987
200005	360	44	1070

SK_ID_CURR	AMT_CREDIT_mean	AMT_CREDIT_max	AMT_CREDIT_min
200001	540.0	540	540
200002	211.5	216	207
200003	94.0	94	94
200004	197.4	274	70
200005	214.0	360	42

SK_ID_CURR	AMT_DOWN_PAYMENT_sum	AMT_DOWN_PAYMENT_mean	AMT_DOWN_PAYMENT_max
200001	0.0	0.0	0.0
200002	77.0	38.5	54.0
200003	0.0	0.0	0.0
200004	28.0	5.6	28.0
200005	12.0	2.4	7.0

SK_ID_CURR	AMT_DOWN_PAYMENT_min	AMT_GOODS_PRICE_sum	AMT_GOODS_PRICE_mean
200001	0.0	450.0	450.0
200002	23.0	500.0	250.0
200003	0.0	84.0	84.0
200004	0.0	938.0	187.6
200005	0.0	1043.0	208.6

SK_ID_CURR	AMT_GOODS_PRICE_max	AMT_GOODS_PRICE_min	DAYS_DECISION_sum
200001	450.0	450.0	-602
200002	270.0	230.0	-1362
200003	84.0	84.0	-359
200004	277.0	67.0	-3662
200005	360.0	44.0	-10489

SK_ID_CURR	DAYS_DECISION_mean	DAYS_DECISION_max	DAYS_DECISION_min
200001	-602.0	-602	-602
200002	-681.0	-523	-839
200003	-359.0	-359	-359
200004	-732.4	-399	-1232
200005	-2097.8	-709	-2754

SK_ID_CURR	CNT_PAYMENT_sum	CNT_PAYMENT_mean	CNT_PAYMENT_max
200001	36.0	36.0	36.0
200002	16.0	8.0	10.0
200003	12.0	12.0	12.0
200004	42.0	8.4	12.0
200005	136.0	27.2	48.0

SK_ID_CURR	CNT_PAYMENT_min	PreDK_MISSING_VALUES_TOTAL_PREV_sum
200001	36.0	2
200002	6.0	2
200003	12.0	1
200004	6.0	4
200005	6.0	5

SK_ID_CURR	PreDK_MISSING_VALUES_TOTAL_PREV_mean
200001	2.0
200002	1.0
200003	1.0
200004	0.8
200005	1.0

SK_ID_CURR	PreDK_MISSING_VALUES_TOTAL_PREV_max
200001	2
200002	1
200003	1
200004	2
200005	2

```
            PreDK_MISSING_VALUES_TOTAL_PREV_min  PreDK_AMT_DECLINED_sum  \
SK_ID_CURR
200001                                        2                     -90
200002                                        1                      77
200003                                        1                     -10
200004                                        0                     -49
200005                                        0                     -27

            PreDK_AMT_DECLINED_mean  PreDK_AMT_DECLINED_max  \
SK_ID_CURR
200001                        -90.0                    -90
200002                         38.5                     54
200003                        -10.0                    -10
200004                         -9.8                      3
200005                         -5.4                      2

            PreDK_AMT_DECLINED_min  PreDK_AMT_CREDIT_GOODS_RATIO_sum  \
SK_ID_CURR
200001                         -90                         1.200000
200002                          23                         1.700000
200003                         -10                         1.119048
200004                         -29                         5.298684
200005                         -30                         5.072006

            PreDK_AMT_CREDIT_GOODS_RATIO_mean  \
SK_ID_CURR
200001                               1.200000
200002                               0.850000
200003                               1.119048
200004                               1.059737
200005                               1.014401

            PreDK_AMT_CREDIT_GOODS_RATIO_max  \
SK_ID_CURR
200001                              1.200000
200002                              0.900000
200003                              1.119048
200004                              1.161111
200005                              1.133333

            PreDK_AMT_CREDIT_GOODS_RATIO_min  PreDK_AMT_CREDIT_GOODS_DIFF_sum  \
SK_ID_CURR
200001                              1.200000                            90.0
200002                              0.800000                           -77.0
200003                              1.119048                            10.0
200004                              0.989170                            49.0
200005                              0.954545                            27.0

            PreDK_AMT_CREDIT_GOODS_DIFF_mean  PreDK_AMT_CREDIT_GOODS_DIFF_max  \
SK_ID_CURR
200001                                  90.0                            90.0
200002                                 -38.5                           -23.0
200003                                  10.0                            10.0
200004                                   9.8                            29.0
200005                                   5.4                            30.0

            PreDK_AMT_CREDIT_GOODS_DIFF_min  \
SK_ID_CURR
200001                                 90.0
200002                                -54.0
200003                                 10.0
200004                                 -3.0
200005                                 -2.0

            PreDK_AMT_CREDIT_APPLICATION_RATIO_sum  \
SK_ID_CURR
200001                                    0.833333
200002                                    2.361111
200003                                    0.893617
200004                                    4.735439
200005                                    4.946101

            PreDK_AMT_CREDIT_APPLICATION_RATIO_mean  \
SK_ID_CURR
200001                                     0.833333
200002                                     1.180556
200003                                     0.893617
200004                                     0.947088
200005                                     0.989220

            PreDK_AMT_CREDIT_APPLICATION_RATIO_max  \
SK_ID_CURR
200001                                    0.833333
200002                                    1.250000
200003                                    0.893617
200004                                    1.010949
200005                                    1.047619

            PreDK_AMT_CREDIT_APPLICATION_RATIO_min  PreDK_AMT_INTEREST_sum  \
SK_ID_CURR
200001                                    0.833333                  792.0
200002                                    1.111111                   41.0
200003                                    0.893617                   26.0
200004                                    0.861244                  213.0
200005                                    0.882353                 1112.0
```

SK_ID_CURR	PreDK_AMT_INTEREST_mean	PreDK_AMT_INTEREST_max	PreDK_AMT_INTEREST_min	PreDK_ANNUITY_CREDIT_RATIO_sum	PreDK_ANNUITY_CREDIT_RATIO_mean	PreDK_ANNUITY_CREDIT_RATIO_max	PreDK_ANNUITY_CREDIT_RATIO_min	NAME_CONTRACT_TYPE_modus_C	NAME_CONTRACT_TYPE_modus_R	NAME_CONTRACT_TYPE_modus_X	NAME_CONTRACT_TYPE_first_C	NAME_CONTRACT_TYPE_first_R	NAME_CONTRACT_TYPE_first_X	NAME_CONTRACT_TYPE_last_C	NAME_CONTRACT_TYPE_last_R	NAME_CONTRACT_TYPE_last_X	NAME_CONTRACT_STATUS_modus_A	NAME_CONTRACT_STATUS_modus_C	NAME_CONTRACT_STATUS_modus_R	NAME_CONTRACT_STATUS_modus_U	NAME_CONTRACT_STATUS_first_A	NAME_CONTRACT_STATUS_first_C	NAME_CONTRACT_STATUS_first_R	NAME_CONTRACT_STATUS_first_U
200001	792.0	792.0	792.0	0.068519	0.068519	0.068519	0.068519	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0
200002	20.5	23.0	18.0	0.291667	0.145833	0.180556	0.111111	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0
200003	26.0	26.0	26.0	0.106383	0.106383	0.106383	0.106383	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0
200004	42.6	55.0	20.0	0.820429	0.164086	0.214286	0.098540	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0
200005	222.4	465.0	6.0	0.469623	0.093925	0.190476	0.047222	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0

SK_ID_CURR	NAME_CONTRACT_STATUS_last_A	NAME_CONTRACT_STATUS_last_C	\
200001	1	0	
200002	1	0	
200003	1	0	
200004	1	0	
200005	1	0	

SK_ID_CURR	NAME_CONTRACT_STATUS_last_R	NAME_CONTRACT_STATUS_last_U	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_PAYMENT_TYPE_modus_C	NAME_PAYMENT_TYPE_modus_N	\
200001	1	0	
200002	1	0	
200003	1	0	
200004	1	0	
200005	0	0	

SK_ID_CURR	NAME_PAYMENT_TYPE_modus_X	NAME_PAYMENT_TYPE_first_C	\
200001	0	1	
200002	0	1	
200003	0	1	
200004	0	1	
200005	1	0	

SK_ID_CURR	NAME_PAYMENT_TYPE_first_N	NAME_PAYMENT_TYPE_first_X	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	1	

SK_ID_CURR	NAME_PAYMENT_TYPE_last_C	NAME_PAYMENT_TYPE_last_N	\
200001	1	0	
200002	1	0	
200003	1	0	
200004	0	1	
200005	0	0	

SK_ID_CURR	NAME_PAYMENT_TYPE_last_X	CODE_REJECT_REASON_modus_C	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	1	0	

SK_ID_CURR	CODE_REJECT_REASON_modus_H	CODE_REJECT_REASON_modus_L	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	CODE_REJECT_REASON_modus_S	CODE_REJECT_REASON_modus_V	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	CODE_REJECT_REASON_modus_X	CODE_REJECT_REASON_first_C	\
200001	1	0	
200002	1	0	
200003	1	0	
200004	1	0	
200005	1	0	

SK_ID_CURR	CODE_REJECT_REASON_first_H	CODE_REJECT_REASON_first_L	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	CODE_REJECT_REASON_first_S	CODE_REJECT_REASON_first_V	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	CODE_REJECT_REASON_first_X	CODE_REJECT_REASON_last_C
200001	1	0
200002	1	0
200003	1	0
200004	1	0
200005	1	0

SK_ID_CURR	CODE_REJECT_REASON_last_H	CODE_REJECT_REASON_last_L
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	CODE_REJECT_REASON_last_S	CODE_REJECT_REASON_last_V
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	CODE_REJECT_REASON_last_X	NAME_TYPE_SUITE_modus_C
200001	1	0
200002	1	0
200003	1	0
200004	1	0
200005	1	0

SK_ID_CURR	NAME_TYPE_SUITE_modus_F	NAME_TYPE_SUITE_modus_G
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_TYPE_SUITE_modus_Kategorie_NaN	NAME_TYPE_SUITE_modus_O
200001	1	0
200002	1	0
200003	1	0
200004	1	0
200005	1	0

SK_ID_CURR	NAME_TYPE_SUITE_modus_S	NAME_TYPE_SUITE_modus_U
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_TYPE_SUITE_first_C	NAME_TYPE_SUITE_first_F
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_TYPE_SUITE_first_G	NAME_TYPE_SUITE_first_O
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_TYPE_SUITE_first_S	NAME_TYPE_SUITE_first_U
200001	0	0
200002	0	0
200003	0	0
200004	0	1
200005	0	1

SK_ID_CURR	NAME_TYPE_SUITE_first_Kategorie_NaN	NAME_TYPE_SUITE_last_C
200001	1	0
200002	1	0
200003	1	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_TYPE_SUITE_last_F	NAME_TYPE_SUITE_last_G
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_TYPE_SUITE_last_O	NAME_TYPE_SUITE_last_S	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_TYPE_SUITE_last_U	NAME_TYPE_SUITE_last_Kategorie_NaN	\
200001	0	1	
200002	0	1	
200003	0	1	
200004	0	1	
200005	0	1	

SK_ID_CURR	NAME_CLIENT_TYPE_modus_N	NAME_CLIENT_TYPE_modus_R	\
200001	1	0	
200002	1	0	
200003	1	0	
200004	0	1	
200005	0	1	

SK_ID_CURR	NAME_CLIENT_TYPE_modus_X	NAME_CLIENT_TYPE_first_N	\
200001	0	1	
200002	0	1	
200003	0	1	
200004	0	1	
200005	0	1	

SK_ID_CURR	NAME_CLIENT_TYPE_first_R	NAME_CLIENT_TYPE_first_X	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_CLIENT_TYPE_last_N	NAME_CLIENT_TYPE_last_R	\
200001	1	0	
200002	0	1	
200003	1	0	
200004	0	1	
200005	0	1	

SK_ID_CURR	NAME_CLIENT_TYPE_last_X	NAME_GOODS_CATEGORY_modus_A	\
200001	0	0	
200002	0	1	
200003	0	1	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_modus_C	NAME_GOODS_CATEGORY_modus_D	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	1	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_modus_E	NAME_GOODS_CATEGORY_modus_F	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_modus_G	NAME_GOODS_CATEGORY_modus_H	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_modus_I	NAME_GOODS_CATEGORY_modus_J	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_modus_M	NAME_GOODS_CATEGORY_modus_O	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_modus_P	NAME_GOODS_CATEGORY_modus_S	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_modus_T	NAME_GOODS_CATEGORY_modus_V	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_modus_W	NAME_GOODS_CATEGORY_modus_X	\
200001	0	1	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	1	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_A	NAME_GOODS_CATEGORY_first_C	\
200001	0	0	
200002	0	0	
200003	1	0	
200004	0	1	
200005	0	1	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_D	NAME_GOODS_CATEGORY_first_E	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_F	NAME_GOODS_CATEGORY_first_G	\
200001	0	0	
200002	1	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_H	NAME_GOODS_CATEGORY_first_I	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_J	NAME_GOODS_CATEGORY_first_M	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_O	NAME_GOODS_CATEGORY_first_P	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_S	NAME_GOODS_CATEGORY_first_T	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_V	NAME_GOODS_CATEGORY_first_W	\
200001	0	0	
200002	0	0	
200003	0	0	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_first_X	NAME_GOODS_CATEGORY_last_A	\
200001	1	0	
200002	0	1	
200003	0	1	
200004	0	0	
200005	0	0	

SK_ID_CURR	NAME_GOODS_CATEGORY_last_C	NAME_GOODS_CATEGORY_last_D
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_GOODS_CATEGORY_last_E	NAME_GOODS_CATEGORY_last_F
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_GOODS_CATEGORY_last_G	NAME_GOODS_CATEGORY_last_H
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_GOODS_CATEGORY_last_I	NAME_GOODS_CATEGORY_last_J
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_GOODS_CATEGORY_last_M	NAME_GOODS_CATEGORY_last_O
200001	0	0
200002	0	0
200003	0	0
200004	1	0
200005	0	0

SK_ID_CURR	NAME_GOODS_CATEGORY_last_P	NAME_GOODS_CATEGORY_last_S
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_GOODS_CATEGORY_last_T	NAME_GOODS_CATEGORY_last_V
200001	0	0
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_GOODS_CATEGORY_last_W	NAME_GOODS_CATEGORY_last_X
200001	0	1
200002	0	0
200003	0	0
200004	0	0
200005	0	1

SK_ID_CURR	NAME_YIELD_GROUP_modus_X	NAME_YIELD_GROUP_modus_h
200001	0	1
200002	0	0
200003	0	0
200004	0	1
200005	0	0

SK_ID_CURR	NAME_YIELD_GROUP_modus_l	NAME_YIELD_GROUP_modus_m
200001	0	0
200002	1	0
200003	0	1
200004	0	0
200005	0	1

SK_ID_CURR	NAME_YIELD_GROUP_first_X	NAME_YIELD_GROUP_first_h
200001	0	1
200002	0	0
200003	0	0
200004	0	0
200005	0	0

SK_ID_CURR	NAME_YIELD_GROUP_first_l	NAME_YIELD_GROUP_first_m
200001	0	0
200002	1	0
200003	0	1
200004	1	0
200005	1	0

```
            NAME_YIELD_GROUP_last_X  NAME_YIELD_GROUP_last_h  \
SK_ID_CURR
200001                            0                        1
200002                            0                        0
200003                            0                        0
200004                            0                        1
200005                            0                        0

            NAME_YIELD_GROUP_last_l  NAME_YIELD_GROUP_last_m  \
SK_ID_CURR
200001                            0                        0
200002                            1                        0
200003                            0                        1
200004                            0                        0
200005                            0                        1

            NFLAG_INSURED_ON_APPROVAL_modus_0.0  \
SK_ID_CURR
200001                                        0
200002                                        1
200003                                        1
200004                                        1
200005                                        0

            NFLAG_INSURED_ON_APPROVAL_modus_1.0  \
SK_ID_CURR
200001                                        1
200002                                        0
200003                                        0
200004                                        0
200005                                        1

            NFLAG_INSURED_ON_APPROVAL_modus_Kategorie_NaN  \
SK_ID_CURR
200001                                                  0
200002                                                  0
200003                                                  0
200004                                                  0
200005                                                  0

            NFLAG_INSURED_ON_APPROVAL_first_0.0  \
SK_ID_CURR
200001                                        0
200002                                        1
200003                                        1
200004                                        1
200005                                        0

            NFLAG_INSURED_ON_APPROVAL_first_1.0  \
SK_ID_CURR
200001                                        1
200002                                        0
200003                                        0
200004                                        0
200005                                        1

            NFLAG_INSURED_ON_APPROVAL_first_Kategorie_NaN  \
SK_ID_CURR
200001                                                  0
200002                                                  0
200003                                                  0
200004                                                  0
200005                                                  0

            NFLAG_INSURED_ON_APPROVAL_last_0.0  \
SK_ID_CURR
200001                                      0
200002                                      1
200003                                      1
200004                                      1
200005                                      0

            NFLAG_INSURED_ON_APPROVAL_last_1.0  \
SK_ID_CURR
200001                                      1
200002                                      0
200003                                      0
200004                                      0
200005                                      1

            NFLAG_INSURED_ON_APPROVAL_last_Kategorie_NaN
SK_ID_CURR
200001                                                0
200002                                                0
200003                                                0
200004                                                0
200005                                                0
```

In [84]:

```
# Achtung: Labelencoding auf dem aggregierten Datensatz! Ggf. Abweichung zu test erwartbar!
# bureau:
labelenc = LabelEncoder()
count = 0

for col in dfbureau_agg:
    if dfbureau_agg[col].dtype == 'category':
```

```
        if len(list(dfbureau_agg[col].unique())) <= 2:
                labelenc.fit(dfbureau_agg[col])
                dfbureau_agg[col] = labelenc.transform(dfbureau_agg[col])
                count += 1

print('Es wurden insgesamt ' + str(count) + ' Spalten codiert')

# One-hot encoding für die restlichen kategoriellen Variablen:
dfbureau_agg = pd.get_dummies(dfbureau_agg)

print(dfbureau_agg.head())
```

Es wurden insgesamt 0 Spalten codiert

SK_ID_CURR	DAYS_CREDIT_sum	DAYS_CREDIT_mean	DAYS_CREDIT_max	DAYS_CREDIT_min	CREDIT_DAY_OVERDUE_sum	CREDIT_DAY_OVERDUE_mean	CREDIT_DAY_OVERDUE_max	CREDIT_DAY_OVERDUE_min	DAYS_CREDIT_ENDDATE_sum	DAYS_CREDIT_ENDDATE_mean	DAYS_CREDIT_ENDDATE_max	DAYS_CREDIT_ENDDATE_min	DAYS_ENDDATE_FACT_sum	DAYS_ENDDATE_FACT_mean	DAYS_ENDDATE_FACT_max	DAYS_ENDDATE_FACT_min	AMT_CREDIT_MAX_OVERDUE_sum	AMT_CREDIT_MAX_OVERDUE_mean	AMT_CREDIT_MAX_OVERDUE_max	AMT_CREDIT_MAX_OVERDUE_min	CNT_CREDIT_PROLONG_sum	CNT_CREDIT_PROLONG_mean	CNT_CREDIT_PROLONG_max	CNT_CREDIT_PROLONG_min	AMT_CREDIT_SUM_sum	AMT_CREDIT_SUM_mean	AMT_CREDIT_SUM_max
200001	-4589	-1147.25	-163	-1900	0	0.0	0	0	-1409.0	-352.250000	786.0	-1170.0	-2325.0	-581.25	0.0	-1292.0	17053.965	4263.49125	17053.965	0.0	0	0.0	0	0	2576.0	644.000000	918.0
200002	-429	-429.00	-429	-429	0	0.0	0	0	-187.0	-187.000000	-187.0	-187.0	-215.0	-215.00	-215.0	-215.0	0.000	0.00000	0.000	0.0	0	0.0	0	0	234.0	234.000000	234.0
200003	-11250	-1875.00	-455	-2664	0	0.0	0	0	-4385.0	-730.833333	1371.0	-2223.0	-5514.0	-919.00	0.0	-2252.0	0.000	0.00000	0.000	0.0	0	0.0	0	0	3710.0	618.333333	1777.0
200004	-1780	-1780.00	-1780	-1780	0	0.0	0	0	-1415.0	-1415.000000	-1415.0	-1415.0	-1444.0	-1444.00	-1444.0	-1444.0	0.000	0.00000	0.000	0.0	0	0.0	0	0	450.0	450.000000	450.0
200006	-1003	-1003.00	-1003	-1003	0	0.0	0	0	-819.0	-819.000000	-819.0	-819.0	-910.0	-910.00	-910.0	-910.0	0.000	0.00000	0.000	0.0	0	0.0	0	0	140.0	140.000000	140.0

SK_ID_CURR	AMT_CREDIT_SUM_min	AMT_CREDIT_SUM_DEBT_sum
200001	404.0	1366.0
200002	234.0	0.0
200003	90.0	1504.0
200004	450.0	0.0
200006	140.0	0.0

SK_ID_CURR	AMT_CREDIT_SUM_DEBT_mean	AMT_CREDIT_SUM_DEBT_max
200001	341.500000	909.0
200002	0.000000	0.0
200003	250.666667	1504.0
200004	0.000000	0.0
200006	0.000000	0.0

SK_ID_CURR	AMT_CREDIT_SUM_DEBT_min	AMT_CREDIT_SUM_LIMIT_sum
200001	0.0	0.00
200002	0.0	0.00
200003	0.0	225039.24
200004	0.0	0.00
200006	0.0	0.00

SK_ID_CURR	AMT_CREDIT_SUM_LIMIT_mean	AMT_CREDIT_SUM_LIMIT_max
200001	0.00	0.00
200002	0.00	0.00
200003	37506.54	225039.24
200004	0.00	0.00
200006	0.00	0.00

SK_ID_CURR	AMT_CREDIT_SUM_LIMIT_min	AMT_CREDIT_SUM_OVERDUE_sum
200001	0.0	0
200002	0.0	0
200003	0.0	0
200004	0.0	0
200006	0.0	0

SK_ID_CURR	AMT_CREDIT_SUM_OVERDUE_mean	AMT_CREDIT_SUM_OVERDUE_max
200001	0.0	0
200002	0.0	0
200003	0.0	0
200004	0.0	0
200006	0.0	0

SK_ID_CURR	AMT_CREDIT_SUM_OVERDUE_min	DAYS_CREDIT_UPDATE_sum
200001	0	-2458
200002	0	-193
200003	0	-6124
200004	0	-386
200006	0	-908

SK_ID_CURR	DAYS_CREDIT_UPDATE_mean	DAYS_CREDIT_UPDATE_max
200001	-614.500000	-5
200002	-193.000000	-193
200003	-1020.666667	-29
200004	-386.000000	-386
200006	-908.000000	-908

SK_ID_CURR	DAYS_CREDIT_UPDATE_min	AMT_ANNUITY_sum	AMT_ANNUITY_mean
200001	-1287	115.0	28.75
200002	-193	0.0	0.00
200003	-2252	0.0	0.00
200004	-386	0.0	0.00
200006	-908	0.0	0.00

SK_ID_CURR	AMT_ANNUITY_max	AMT_ANNUITY_min	BurDK_CREDIT_DURATION_sum
200001	115.0	0.0	3180.0
200002	0.0	0.0	242.0
200003	0.0	0.0	6865.0
200004	0.0	0.0	365.0
200006	0.0	0.0	184.0

SK_ID_CURR	BurDK_CREDIT_DURATION_mean	BurDK_CREDIT_DURATION_max
200001	795.000000	1823.0
200002	242.000000	242.0
200003	1144.166667	1899.0
200004	365.000000	365.0
200006	184.000000	184.0

SK_ID_CURR	BurDK_CREDIT_DURATION_min	BurDK_FLAG_OVERDUE_RECENT_sum
200001	261.0	0
200002	242.0	0
200003	213.0	0
200004	365.0	0
200006	184.0	0

```
            BurDK_FLAG_OVERDUE_RECENT_mean  BurDK_FLAG_OVERDUE_RECENT_max  \
SK_ID_CURR
200001                                 0.0                              0
200002                                 0.0                              0
200003                                 0.0                              0
200004                                 0.0                              0
200006                                 0.0                              0

            BurDK_FLAG_OVERDUE_RECENT_min  \
SK_ID_CURR
200001                                  0
200002                                  0
200003                                  0
200004                                  0
200006                                  0

            BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_sum  \
SK_ID_CURR
200001                                               0.0
200002                                               0.0
200003                                               0.0
200004                                               0.0
200006                                               0.0

            BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_mean  \
SK_ID_CURR
200001                                                0.0
200002                                                0.0
200003                                                0.0
200004                                                0.0
200006                                                0.0

            BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_max  \
SK_ID_CURR
200001                                               0.0
200002                                               0.0
200003                                               0.0
200004                                               0.0
200006                                               0.0

            BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_min  \
SK_ID_CURR
200001                                               0.0
200002                                               0.0
200003                                               0.0
200004                                               0.0
200006                                               0.0

            BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_sum  \
SK_ID_CURR
200001                                          0.0
200002                                          0.0
200003                                          0.0
200004                                          0.0
200006                                          0.0

            BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_mean  \
SK_ID_CURR
200001                                           0.0
200002                                           0.0
200003                                           0.0
200004                                           0.0
200006                                           0.0

            BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_max  \
SK_ID_CURR
200001                                         -0.0
200002                                         -0.0
200003                                         -0.0
200004                                         -0.0
200006                                         -0.0

            BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_min  \
SK_ID_CURR
200001                                         -0.0
200002                                         -0.0
200003                                         -0.0
200004                                         -0.0
200006                                         -0.0

            BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_sum  \
SK_ID_CURR
200001                                       0.0
200002                                       0.0
200003                                       0.0
200004                                       0.0
200006                                       0.0

            BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_mean  \
SK_ID_CURR
200001                                        0.0
200002                                        0.0
200003                                        0.0
200004                                        0.0
200006                                        0.0
```

SK_ID_CURR	BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_max	BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_min	BurDK_CNT_PROLONGED_DURATION_RATIO_sum	BurDK_CNT_PROLONGED_DURATION_RATIO_mean	BurDK_CNT_PROLONGED_DURATION_RATIO_max	BurDK_CNT_PROLONGED_DURATION_RATIO_min	BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_sum	BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_mean	BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_max	BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_min	BurDK_CURRENT_CREDIT_DEBT_DIFF_sum	BurDK_CURRENT_CREDIT_DEBT_DIFF_mean
200001	0.0	0.0	0.0	0.0	0.0	0.0	2.005752	0.501438	1.015556	0.0	1210.0	302.500000
200002	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	234.0	234.000000
200003	0.0	0.0	0.0	0.0	0.0	0.0	0.846370	0.141062	0.846370	0.0	2206.0	367.666667
200004	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	450.0	450.000000
200006	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.0	140.0	140.000000

```
            BurDK_CURRENT_CREDIT_DEBT_DIFF_max  \
SK_ID_CURR
200001                                   804.0
200002                                   234.0
200003                                  1080.0
200004                                   450.0
200006                                   140.0

            BurDK_CURRENT_CREDIT_DEBT_DIFF_min  \
SK_ID_CURR
200001                                    -7.0
200002                                   234.0
200003                                    90.0
200004                                   450.0
200006                                   140.0

            BurDK_AMT_ANNUITY_CREDIT_RATIO_sum  \
SK_ID_CURR
200001                                0.125272
200002                                0.000000
200003                                0.000000
200004                                0.000000
200006                                0.000000

            BurDK_AMT_ANNUITY_CREDIT_RATIO_mean  \
SK_ID_CURR
200001                                 0.031318
200002                                 0.000000
200003                                 0.000000
200004                                 0.000000
200006                                 0.000000

            BurDK_AMT_ANNUITY_CREDIT_RATIO_max  \
SK_ID_CURR
200001                                0.125272
200002                                0.000000
200003                                0.000000
200004                                0.000000
200006                                0.000000

            BurDK_AMT_ANNUITY_CREDIT_RATIO_min  \
SK_ID_CURR
200001                                   0.0
200002                                   0.0
200003                                   0.0
200004                                   0.0
200006                                   0.0

            BurDK_CREDIT_ENDDATE_UPDATE_DIFF_sum  \
SK_ID_CURR
200001                                  1229.0
200002                                     6.0
200003                                  1811.0
200004                                  1029.0
200006                                    89.0

            BurDK_CREDIT_ENDDATE_UPDATE_DIFF_mean  \
SK_ID_CURR
200001                                307.250000
200002                                  6.000000
200003                                301.833333
200004                               1029.000000
200006                                 89.000000

            BurDK_CREDIT_ENDDATE_UPDATE_DIFF_max  \
SK_ID_CURR
200001                                   791.0
200002                                     6.0
200003                                  1400.0
200004                                  1029.0
200006                                    89.0

            BurDK_CREDIT_ENDDATE_UPDATE_DIFF_min  CREDIT_ACTIVE_modus_A  \
SK_ID_CURR
200001                                   90.0                     1
200002                                    6.0                     0
200003                                    0.0                     0
200004                                 1029.0                     0
200006                                   89.0                     0

            CREDIT_ACTIVE_modus_C  CREDIT_ACTIVE_modus_S  \
SK_ID_CURR
200001                          0                     0
200002                          1                     0
200003                          1                     0
200004                          1                     0
200006                          1                     0

            CREDIT_ACTIVE_first_A  CREDIT_ACTIVE_first_B  \
SK_ID_CURR
200001                          0                     0
200002                          0                     0
200003                          0                     0
200004                          0                     0
200006                          0                     0
```

```
            CREDIT_ACTIVE_first_C  CREDIT_ACTIVE_first_S  \
SK_ID_CURR
200001                          1                     0
200002                          1                     0
200003                          1                     0
200004                          1                     0
200006                          1                     0

            CREDIT_ACTIVE_last_A  CREDIT_ACTIVE_last_B  CREDIT_ACTIVE_last_C  \
SK_ID_CURR
200001                         1                    0                    0
200002                         0                    0                    1
200003                         1                    0                    0
200004                         0                    0                    1
200006                         0                    0                    1

            CREDIT_ACTIVE_last_S
SK_ID_CURR
200001                         0
200002                         0
200003                         0
200004                         0
200006                         0
```

Aufgabe A-6: Zusammenfassen des Hauptdatensatzes [Lernziel 6.1.1; 5 Punkte]

In diesem Abschnitt ist der Datensatz für die Modellierung zu erstellen. Dabei ist unter Verwendung der encodierten Datensätze aus der vorherigen Aufgabe wie folgt vorzugehen:

- **Die Daten aus bureau (in aggregierter Form) sind per Left-Join dem kombinierten Datensatz mit Trainings- und Testdaten hinzuzufügen.**
- **Im Anschluss sind die Daten aus previous_application (in aggregierter Form) den Daten aus dem vorherigen Schritt hinzuzufügen.**
- **Schließlich sind die Anzahl der Zeilen und Spalten sowie fünf zufällige Spalten des finalen Datensatzes auszugeben.**

Durch das Zusammenfügen des Datensatzes entstehen erneut 'Missing Values'. Diese sollen mit dem Wert 0 befüllt werden.

Abschließend soll der Datensatz in Trainings- und Testdatensatz getrennt werden. Eingeführte Hilfsvariablen sind danach zu entfernen.

Lösungsvorschlag:

In [85]:
```
# Hinzufügen von dfbureau
dfapptraintest_tmp1 = dfapptraintest.join(dfbureau_agg, how='left', on='SK_ID_CURR', rsuffix='_bureau')

# Hinzufügen von dfprevapp
df = dfapptraintest_tmp1.join(dfprevapp_agg, how='left', on='SK_ID_CURR', rsuffix='_prev')
```

In [86]:
```
# Anzeigen der Zeilen und Spalten des Datensatzes
df.shape
```

Out[86]:
```
(191296, 422)
```

Der zusammengefügte Datensatz enthält 191.296 Zeilen und 422 Spalten.

In [87]:
```
# Ausgabe von fünf zufälligen Zeilen des Datensatzes
print(df.sample(5))
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR
136998	363426	0	0	0
10058	262193	0	0	1
126057	350288	0	1	1
98957	318153	0	0	0
25278	356598	0	0	0

	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT
136998	1	0	450	1800
10058	1	0	270	747
126057	0	0	360	2115
98957	0	0	225	1350
25278	1	1	144	396

	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	DAYS_BIRTH
136998	93.0	1800.0	0.046	-18178
10058	46.0	567.0	0.031	-10767
126057	113.0	2115.0	0.031	-15457
98957	44.0	1350.0	0.010	-14925
25278	32.0	396.0	0.015	-12701

	DAYS_EMPLOYED	DAYS_REGISTRATION	DAYS_ID_PUBLISH	OWN_CAR_AGE
136998	-4896.0	-12222	-1721	0.0
10058	-1717.0	-3478	-1909	9.0
126057	-4394.0	-6911	-4659	3.0
98957	-7826.0	-2571	-4035	0.0
25278	-337.0	-1298	-4510	0.0

	FLAG_MOBIL	FLAG_EMP_PHONE	FLAG_WORK_PHONE	FLAG_CONT_MOBILE
136998	1	1	0	1
10058	1	1	0	1
126057	1	1	0	1
98957	1	1	0	1
25278	1	1	0	1

	FLAG_PHONE	FLAG_EMAIL	CNT_FAM_MEMBERS	REG_REGION_NOT_LIVE_REGION
136998	1	0	1.0	0
10058	0	1	2.0	0
126057	0	0	2.0	0
98957	1	1	2.0	0
25278	0	0	2.0	0

	REG_REGION_NOT_WORK_REGION	LIVE_REGION_NOT_WORK_REGION
136998	0	0
10058	0	0
126057	0	0
98957	0	0
25278	0	0

	REG_CITY_NOT_LIVE_CITY	REG_CITY_NOT_WORK_CITY
136998	0	0
10058	0	0
126057	0	1
98957	0	1
25278	0	0

	LIVE_CITY_NOT_WORK_CITY	EXT_SOURCE_1	EXT_SOURCE_2	EXT_SOURCE_3
136998	0	0.683	0.799	0.546
10058	0	0.600	0.551	0.592
126057	1	0.275	0.464	0.648
98957	1	0.621	0.571	0.705
25278	0	0.403	0.351	0.728

	DAYS_LAST_PHONE_CHANGE	AMT_REQ_CREDIT_BUREAU_HOUR
136998	-1963.0	0.0
10058	-735.0	0.0
126057	-142.0	0.0
98957	-2635.0	0.0
25278	-1863.0	0.0

	AMT_REQ_CREDIT_BUREAU_DAY	AMT_REQ_CREDIT_BUREAU_WEEK
136998	0.0	0.0
10058	0.0	0.0
126057	0.0	0.0
98957	0.0	0.0
25278	0.0	0.0

	AMT_REQ_CREDIT_BUREAU_MON	AMT_REQ_CREDIT_BUREAU_QRT
136998	0.0	1.0
10058	0.0	0.0
126057	0.0	1.0
98957	0.0	0.0
25278	1.0	0.0

	AMT_REQ_CREDIT_BUREAU_YEAR	TARGET	TYPE	AppDK_ANNUITY_CREDIT_RATIO
136998	2.0	0	TRAIN	0.051667
10058	3.0	0	TEST	0.061580
126057	3.0	0	TRAIN	0.053428
98957	3.0	0	TRAIN	0.032593
25278	2.0	0	TEST	0.080808

	AppDK_GOODS_CREDIT_RATIO	AppDK_ANNUITY_INCOME_RATIO
136998	1.000000	0.206667
10058	0.759036	0.170370
126057	1.000000	0.313889
98957	1.000000	0.195556

```
25278                  1.000000                   0.222222

        AppDK_CREDIT_INCOME_RATIO  AppDK_GOODS_INCOME_RATIO  \
136998                   4.000000                     4.000
10058                    2.766667                     2.100
126057                   5.875000                     5.875
98957                    6.000000                     6.000
25278                    2.750000                     2.750

        AppDK_CNT_FAM_INCOME_RATIO  AppDK_EMPLOYED_BIRTH_RATIO  \
136998                       450.0                    0.269337
10058                        135.0                    0.159469
126057                       180.0                    0.284272
98957                        112.5                    0.524355
25278                         72.0                    0.026533

        AppDK_INCOME_EMPLOYED_RATIO  AppDK_INCOME_BIRTH_RATIO  \
136998                    -0.091912                 -0.024755
10058                     -0.157251                 -0.025077
126057                    -0.081930                 -0.023290
98957                     -0.028750                 -0.015075
25278                     -0.427300                 -0.011338

        AppDK_CAR_BIRTH_RATIO  AppDK_CAR_EMPLOYED_RATIO  NAME_TYPE_SUITE_C  \
136998              -0.000000                 -0.000000                  0
10058               -0.000836                 -0.005242                  0
126057              -0.000194                 -0.000683                  0
98957               -0.000000                 -0.000000                  0
25278               -0.000000                 -0.000000                  0

        NAME_TYPE_SUITE_F  NAME_TYPE_SUITE_G  NAME_TYPE_SUITE_O  \
136998                  0                  0                  0
10058                   0                  0                  0
126057                  1                  0                  0
98957                   0                  0                  0
25278                   0                  0                  0

        NAME_TYPE_SUITE_S  NAME_TYPE_SUITE_U  NAME_TYPE_SUITE_Kategorie_NaN  \
136998                  0                  1                              0
10058                   0                  1                              0
126057                  0                  0                              0
98957                   0                  1                              0
25278                   0                  1                              0

        NAME_INCOME_TYPE_B  NAME_INCOME_TYPE_C  NAME_INCOME_TYPE_M  \
136998                   0                   1                   0
10058                    0                   0                   0
126057                   0                   0                   0
98957                    0                   1                   0
25278                    0                   0                   0

        NAME_INCOME_TYPE_P  NAME_INCOME_TYPE_S  NAME_INCOME_TYPE_U  \
136998                   0                   0                   0
10058                    0                   0                   0
126057                   0                   0                   0
98957                    0                   0                   0
25278                    0                   0                   0

        NAME_INCOME_TYPE_W  NAME_EDUCATION_TYPE_A  NAME_EDUCATION_TYPE_H  \
136998                   0                      0                      0
10058                    1                      0                      1
126057                   1                      0                      1
98957                    0                      0                      0
25278                    1                      0                      0

        NAME_EDUCATION_TYPE_I  NAME_EDUCATION_TYPE_L  NAME_EDUCATION_TYPE_S  \
136998                      0                      0                      1
10058                       0                      0                      0
126057                      0                      0                      0
98957                       0                      0                      1
25278                       0                      0                      1

        NAME_FAMILY_STATUS_C  NAME_FAMILY_STATUS_M  NAME_FAMILY_STATUS_S  \
136998                     0                     0                     1
10058                      0                     1                     0
126057                     0                     1                     0
98957                      1                     0                     0
25278                      0                     0                     1

        NAME_FAMILY_STATUS_U  NAME_FAMILY_STATUS_W  NAME_HOUSING_TYPE_C  \
136998                     0                     0                    0
10058                      0                     0                    0
126057                     0                     0                    0
98957                      0                     0                    0
25278                      0                     0                    0

        NAME_HOUSING_TYPE_H  NAME_HOUSING_TYPE_M  NAME_HOUSING_TYPE_O  \
136998                    1                    0                    0
10058                     1                    0                    0
126057                    1                    0                    0
98957                     1                    0                    0
25278                     1                    0                    0

        NAME_HOUSING_TYPE_R  NAME_HOUSING_TYPE_W  OCCUPATION_TYPE_A  \
136998                    0                    0                  1
10058                     0                    0                  0
```

```
126057                   0                   0                 0
98957                    0                   0                 0
25278                    0                   0                 0

        OCCUPATION_TYPE_C  OCCUPATION_TYPE_D  OCCUPATION_TYPE_H  \
136998                  0                  0                  0
10058                   0                  0                  0
126057                  0                  0                  0
98957                   0                  0                  0
25278                   0                  0                  0

        OCCUPATION_TYPE_I  OCCUPATION_TYPE_L  OCCUPATION_TYPE_M  \
136998                  0                  0                  0
10058                   0                  0                  0
126057                  0                  0                  0
98957                   0                  1                  0
25278                   0                  0                  0

        OCCUPATION_TYPE_P  OCCUPATION_TYPE_R  OCCUPATION_TYPE_S  \
136998                  0                  0                  0
10058                   0                  0                  0
126057                  0                  0                  0
98957                   0                  0                  0
25278                   0                  0                  1

        OCCUPATION_TYPE_W  OCCUPATION_TYPE_Kategorie_NaN  \
136998                  0                             0
10058                   0                             1
126057                  0                             1
98957                   0                             0
25278                   0                             0

        REGION_RATING_CLIENT_1  REGION_RATING_CLIENT_2  \
136998                       1                       0
10058                        0                       1
126057                       0                       1
98957                        0                       1
25278                        0                       1

        REGION_RATING_CLIENT_3  REGION_RATING_CLIENT_W_CITY_1  \
136998                       0                             1
10058                        0                             0
126057                       0                             0
98957                        0                             0
25278                        0                             0

        REGION_RATING_CLIENT_W_CITY_2  REGION_RATING_CLIENT_W_CITY_3  \
136998                              0                              0
10058                               1                              0
126057                              1                              0
98957                               1                              0
25278                               1                              0

        ORGANIZATION_TYPE_A  ORGANIZATION_TYPE_B  ORGANIZATION_TYPE_C  \
136998                    0                    0                    0
10058                     0                    1                    0
126057                    0                    0                    0
98957                     0                    1                    0
25278                     0                    0                    0

        ORGANIZATION_TYPE_E  ORGANIZATION_TYPE_G  ORGANIZATION_TYPE_H  \
136998                    0                    0                    0
10058                     0                    0                    0
126057                    0                    0                    0
98957                     0                    0                    0
25278                     0                    0                    0

        ORGANIZATION_TYPE_I  ORGANIZATION_TYPE_K  ORGANIZATION_TYPE_L  \
136998                    0                    0                    0
10058                     0                    0                    0
126057                    0                    0                    0
98957                     0                    0                    0
25278                     0                    0                    0

        ORGANIZATION_TYPE_M  ORGANIZATION_TYPE_O  ORGANIZATION_TYPE_P  \
136998                    0                    0                    0
10058                     0                    0                    0
126057                    0                    1                    0
98957                     0                    0                    0
25278                     0                    0                    0

        ORGANIZATION_TYPE_R  ORGANIZATION_TYPE_S  ORGANIZATION_TYPE_T  \
136998                    0                    0                    1
10058                     0                    0                    0
126057                    0                    0                    0
98957                     0                    0                    0
25278                     0                    1                    0

        ORGANIZATION_TYPE_U  ORGANIZATION_TYPE_X  DAYS_CREDIT_sum  \
136998                    0                    0          -6839.0
10058                     0                    0          -1173.0
126057                    0                    0          -4242.0
98957                     0                    0          -8676.0
25278                     0                    0          -1065.0

        DAYS_CREDIT_mean  DAYS_CREDIT_max  DAYS_CREDIT_min  \
```

```
136998           -977.0          -437.0         -1630.0
10058            -586.5          -496.0          -677.0
126057          -1414.0          -353.0         -2789.0
98957           -1735.2          -391.0         -2899.0
25278           -1065.0         -1065.0         -1065.0

        CREDIT_DAY_OVERDUE_sum  CREDIT_DAY_OVERDUE_mean  \
136998                     0.0                      0.0
10058                      0.0                      0.0
126057                     0.0                      0.0
98957                      0.0                      0.0
25278                      0.0                      0.0

        CREDIT_DAY_OVERDUE_max  CREDIT_DAY_OVERDUE_min  \
136998                     0.0                     0.0
10058                      0.0                     0.0
126057                     0.0                     0.0
98957                      0.0                     0.0
25278                      0.0                     0.0

        DAYS_CREDIT_ENDDATE_sum  DAYS_CREDIT_ENDDATE_mean  \
136998                  58716.0               8388.000000
10058                    -502.0               -251.000000
126057                   -232.0                -77.333333
98957                   -1631.0               -326.200000
25278                    -287.0               -287.000000

        DAYS_CREDIT_ENDDATE_max  DAYS_CREDIT_ENDDATE_min  \
136998                  30928.0                 -1265.0
10058                    -131.0                  -371.0
126057                    719.0                  -963.0
98957                    1000.0                 -2119.0
25278                    -287.0                  -287.0

        DAYS_ENDDATE_FACT_sum  DAYS_ENDDATE_FACT_mean  DAYS_ENDDATE_FACT_max  \
136998                -3174.0            -453.428571                    0.0
10058                  -658.0            -329.000000                 -168.0
126057                 -961.0            -320.333333                    0.0
98957                 -4613.0            -922.600000                    0.0
25278                     0.0               0.000000                    0.0

        DAYS_ENDDATE_FACT_min  AMT_CREDIT_MAX_OVERDUE_sum  \
136998                -1267.0                       0.000
10058                  -490.0                   14562.990
126057                 -961.0                       0.000
98957                 -2147.0                      92.385
25278                     0.0                       0.000

        AMT_CREDIT_MAX_OVERDUE_mean  AMT_CREDIT_MAX_OVERDUE_max  \
136998                        0.000                       0.000
10058                      7281.495                   14562.990
126057                        0.000                       0.000
98957                        18.477                      92.385
25278                         0.000                       0.000

        AMT_CREDIT_MAX_OVERDUE_min  CNT_CREDIT_PROLONG_sum  \
136998                         0.0                     0.0
10058                          0.0                     0.0
126057                         0.0                     0.0
98957                          0.0                     0.0
25278                          0.0                     0.0

        CNT_CREDIT_PROLONG_mean  CNT_CREDIT_PROLONG_max  \
136998                      0.0                     0.0
10058                       0.0                     0.0
126057                      0.0                     0.0
98957                       0.0                     0.0
25278                       0.0                     0.0

        CNT_CREDIT_PROLONG_min  AMT_CREDIT_SUM_sum  AMT_CREDIT_SUM_mean  \
136998                     0.0              2229.0           318.428571
10058                      0.0               519.0           259.500000
126057                     0.0              4611.0          1537.000000
98957                      0.0              3651.0           730.200000
25278                      0.0                 0.0             0.000000

        AMT_CREDIT_SUM_max  AMT_CREDIT_SUM_min  AMT_CREDIT_SUM_DEBT_sum  \
136998               900.0                 5.0                      3.0
10058                351.0               168.0                      0.0
126057              3150.0               111.0                     11.0
98957               1800.0                42.0                    834.0
25278                  0.0                 0.0                      0.0

        AMT_CREDIT_SUM_DEBT_mean  AMT_CREDIT_SUM_DEBT_max  \
136998                  0.428571                      3.0
10058                   0.000000                      0.0
126057                  3.666667                     11.0
98957                 166.800000                    493.0
25278                   0.000000                      0.0

        AMT_CREDIT_SUM_DEBT_min  AMT_CREDIT_SUM_LIMIT_sum  \
136998                      0.0                       0.0
10058                       0.0                       0.0
126057                      0.0                       0.0
98957                       0.0                       0.0
25278                       0.0                       0.0
```

```
        AMT_CREDIT_SUM_LIMIT_mean  AMT_CREDIT_SUM_LIMIT_max  \
136998                        0.0                       0.0
10058                         0.0                       0.0
126057                        0.0                       0.0
98957                         0.0                       0.0
25278                         0.0                       0.0

        AMT_CREDIT_SUM_LIMIT_min  AMT_CREDIT_SUM_OVERDUE_sum  \
136998                       0.0                         0.0
10058                        0.0                         0.0
126057                       0.0                         0.0
98957                        0.0                         0.0
25278                        0.0                         0.0

        AMT_CREDIT_SUM_OVERDUE_mean  AMT_CREDIT_SUM_OVERDUE_max  \
136998                          0.0                         0.0
10058                           0.0                         0.0
126057                          0.0                         0.0
98957                           0.0                         0.0
25278                           0.0                         0.0

        AMT_CREDIT_SUM_OVERDUE_min  DAYS_CREDIT_UPDATE_sum  \
136998                         0.0                -4141.0
10058                          0.0                 -657.0
126057                         0.0                 -303.0
98957                          0.0                -3232.0
25278                          0.0                 -280.0

        DAYS_CREDIT_UPDATE_mean  DAYS_CREDIT_UPDATE_max  \
136998              -591.571429                    -8.0
10058               -328.500000                  -168.0
126057              -101.000000                   -10.0
98957               -646.400000                   -67.0
25278               -280.000000                  -280.0

        DAYS_CREDIT_UPDATE_min  AMT_ANNUITY_sum  AMT_ANNUITY_mean  \
136998                -1267.0             42.0               6.0
10058                  -489.0              0.0               0.0
126057                 -275.0              0.0               0.0
98957                 -2147.0              0.0               0.0
25278                  -280.0              0.0               0.0

        AMT_ANNUITY_max  AMT_ANNUITY_min  BurDK_CREDIT_DURATION_sum  \
136998             42.0              0.0                    65555.0
10058               0.0              0.0                      671.0
126057              0.0              0.0                     4010.0
98957               0.0              0.0                     7045.0
25278               0.0              0.0                      778.0

        BurDK_CREDIT_DURATION_mean  BurDK_CREDIT_DURATION_max  \
136998                9365.000000                    31711.0
10058                  335.500000                      365.0
126057                1336.666667                     1826.0
98957                 1409.000000                     1827.0
25278                  778.000000                      778.0

        BurDK_CREDIT_DURATION_min  BurDK_FLAG_OVERDUE_RECENT_sum  \
136998                      305.0                            0.0
10058                       306.0                            0.0
126057                      365.0                            0.0
98957                       183.0                            0.0
25278                       778.0                            0.0

        BurDK_FLAG_OVERDUE_RECENT_mean  BurDK_FLAG_OVERDUE_RECENT_max  \
136998                             0.0                            0.0
10058                              0.0                            0.0
126057                             0.0                            0.0
98957                              0.0                            0.0
25278                              0.0                            0.0

        BurDK_FLAG_OVERDUE_RECENT_min  \
136998                            0.0
10058                             0.0
126057                            0.0
98957                             0.0
25278                             0.0

        BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_sum  \
136998                                           0.0
10058                                            0.0
126057                                           0.0
98957                                            0.0
25278                                            0.0

        BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_mean  \
136998                                            0.0
10058                                             0.0
126057                                            0.0
98957                                             0.0
25278                                             0.0

        BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_max  \
136998                                           0.0
10058                                            0.0
126057                                           0.0
```

```
98957                                          0.0
25278                                          0.0

        BurDK_CURRENT_AMT_OVERDUE_DURATION_RATIO_min  \
136998                                         0.0
10058                                          0.0
126057                                         0.0
98957                                          0.0
25278                                          0.0

        BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_sum  \
136998                                        0.0
10058                                         0.0
126057                                        0.0
98957                                         0.0
25278                                         0.0

        BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_mean  \
136998                                         0.0
10058                                          0.0
126057                                         0.0
98957                                          0.0
25278                                          0.0

        BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_max  \
136998                                       -0.0
10058                                        -0.0
126057                                       -0.0
98957                                        -0.0
25278                                        -0.0

        BurDK_AMT_OVERDUE_DURATION_LEFT_RATIO_min  \
136998                                       -0.0
10058                                        -0.0
126057                                       -0.0
98957                                        -0.0
25278                                        -0.0

        BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_sum  \
136998                                      0.0
10058                                       0.0
126057                                      0.0
98957                                       0.0
25278                                       0.0

        BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_mean  \
136998                                       0.0
10058                                        0.0
126057                                       0.0
98957                                        0.0
25278                                        0.0

        BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_max  \
136998                                      0.0
10058                                       0.0
126057                                      0.0
98957                                       0.0
25278                                       0.0

        BurDK_CNT_PROLONGED_MAX_OVERDUE_MUL_min  \
136998                                      0.0
10058                                       0.0
126057                                      0.0
98957                                       0.0
25278                                       0.0

        BurDK_CNT_PROLONGED_DURATION_RATIO_sum  \
136998                                     0.0
10058                                      0.0
126057                                     0.0
98957                                      0.0
25278                                      0.0

        BurDK_CNT_PROLONGED_DURATION_RATIO_mean  \
136998                                      0.0
10058                                       0.0
126057                                      0.0
98957                                       0.0
25278                                       0.0

        BurDK_CNT_PROLONGED_DURATION_RATIO_max  \
136998                                     0.0
10058                                      0.0
126057                                     0.0
98957                                      0.0
25278                                      0.0

        BurDK_CNT_PROLONGED_DURATION_RATIO_min  \
136998                                     0.0
10058                                      0.0
126057                                     0.0
98957                                      0.0
25278                                      0.0

        BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_sum  \
136998                                0.600000
```

```
10058                                0.000000
126057                               0.099099
98957                                1.549176
25278                                0.000000

        BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_mean  \
136998                                0.085714
10058                                 0.000000
126057                                0.033033
98957                                 0.309835
25278                                      NaN

        BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_max  \
136998                               0.600000
10058                                0.000000
126057                               0.099099
98957                                0.947222
25278                                     NaN

        BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_min  \
136998                                   0.0
10058                                    0.0
126057                                   0.0
98957                                    0.0
25278                                    NaN

        BurDK_CURRENT_CREDIT_DEBT_DIFF_sum  \
136998                              2226.0
10058                                519.0
126057                              4600.0
98957                               2817.0
25278                                  0.0

        BurDK_CURRENT_CREDIT_DEBT_DIFF_mean  \
136998                           318.000000
10058                            259.500000
126057                          1533.333333
98957                            563.400000
25278                              0.000000

        BurDK_CURRENT_CREDIT_DEBT_DIFF_max  \
136998                               900.0
10058                                351.0
126057                              3150.0
98957                               1800.0
25278                                  0.0

        BurDK_CURRENT_CREDIT_DEBT_DIFF_min  \
136998                                2.0
10058                               168.0
126057                              100.0
98957                                19.0
25278                                 0.0

        BurDK_AMT_ANNUITY_CREDIT_RATIO_sum  \
136998                            0.077778
10058                             0.000000
126057                            0.000000
98957                             0.000000
25278                             0.000000

        BurDK_AMT_ANNUITY_CREDIT_RATIO_mean  \
136998                             0.011111
10058                              0.000000
126057                             0.000000
98957                              0.000000
25278                                   NaN

        BurDK_AMT_ANNUITY_CREDIT_RATIO_max  \
136998                            0.077778
10058                             0.000000
126057                            0.000000
98957                             0.000000
25278                                  NaN

        BurDK_AMT_ANNUITY_CREDIT_RATIO_min  \
136998                                0.0
10058                                 0.0
126057                                0.0
98957                                 0.0
25278                                 NaN

        BurDK_CREDIT_ENDDATE_UPDATE_DIFF_sum  \
136998                              62869.0
10058                                 155.0
126057                               1447.0
98957                                2717.0
25278                                   7.0

        BurDK_CREDIT_ENDDATE_UPDATE_DIFF_mean  \
136998                            8981.285714
10058                               77.500000
126057                             482.333333
98957                              543.400000
25278                                7.000000
```

	BurDK_CREDIT_ENDDATE_UPDATE_DIFF_max	BurDK_CREDIT_ENDDATE_UPDATE_DIFF_min	CREDIT_ACTIVE_modus_A	CREDIT_ACTIVE_modus_C	CREDIT_ACTIVE_modus_S	CREDIT_ACTIVE_first_A	CREDIT_ACTIVE_first_B	CREDIT_ACTIVE_first_C	CREDIT_ACTIVE_first_S	CREDIT_ACTIVE_last_A	CREDIT_ACTIVE_last_B	CREDIT_ACTIVE_last_C	CREDIT_ACTIVE_last_S	AMT_ANNUITY_sum_prev	AMT_ANNUITY_mean_prev	AMT_ANNUITY_max_prev	AMT_ANNUITY_min_prev	AMT_APPLICATION_sum	AMT_APPLICATION_mean	AMT_APPLICATION_max	AMT_APPLICATION_min	AMT_CREDIT_sum	AMT_CREDIT_mean	AMT_CREDIT_max	AMT_CREDIT_min	AMT_DOWN_PAYMENT_sum	AMT_DOWN_PAYMENT_mean	AMT_DOWN_PAYMENT_max	AMT_DOWN_PAYMENT_min	AMT_GOODS_PRICE_sum	AMT_GOODS_PRICE_mean	AMT_GOODS_PRICE_max	AMT_GOODS_PRICE_min	DAYS_DECISION_sum	DAYS_DECISION_mean	DAYS_DECISION_max	DAYS_DECISION_min	CNT_PAYMENT_sum	CNT_PAYMENT_mean	CNT_PAYMENT_max	CNT_PAYMENT_min
136998	31710.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	13.0	6.500000	13.0	0.0	132.0	66.000000	132.0	0.0	132.0	66.000000	132.0	0.0	0.0	0.000000	0.0	0.0	132.0	66.000000	132.0	0.0	-2137.0	-1068.500000	-174.0	-1963.0	12.0	6.0	12.0	0.0
10058	118.0	37.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	46.0	15.333333	34.0	0.0	766.0	255.333333	459.0	121.0	803.0	267.666667	497.0	120.0	13.0	4.333333	13.0	0.0	766.0	255.333333	459.0	121.0	-1669.0	-556.333333	-257.0	-735.0	30.0	10.0	18.0	0.0
126057	729.0	30.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	45.0	11.250000	19.0	0.0	325.0	81.250000	157.0	0.0	337.0	84.250000	180.0	0.0	18.0	4.500000	9.0	0.0	325.0	81.250000	157.0	0.0	-2635.0	-658.750000	-142.0	-1627.0	28.0	7.0	12.0	0.0
98957	1067.0	28.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	61.0	20.333333	34.0	11.0	1070.0	356.666667	585.0	236.0	1257.0	419.000000	788.0	222.0	51.0	17.000000	27.0	0.0	1070.0	356.666667	585.0	236.0	-4211.0	-1403.666667	-666.0	-2635.0	84.0	28.0	36.0	24.0
25278	7.0	7.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	42.0	8.400000	21.0	0.0	304.0	60.800000	180.0	0.0	302.0	60.400000	180.0	0.0	13.0	2.600000	8.0	0.0	304.0	60.800000	180.0	0.0	-4627.0	-925.400000	-26.0	-2274.0	30.0	10.0	12.0	0.0

```
25278             30.0              6.0            12.0             0.0

        PreDK_MISSING_VALUES_TOTAL_PREV_sum  \
136998                                  6.0
10058                                   6.0
126057                                  7.0
98957                                   1.0
25278                                  14.0

        PreDK_MISSING_VALUES_TOTAL_PREV_mean  \
136998                              3.000000
10058                               2.000000
126057                              1.750000
98957                               0.333333
25278                               2.800000

        PreDK_MISSING_VALUES_TOTAL_PREV_max  \
136998                                  6.0
10058                                   4.0
126057                                  6.0
98957                                   1.0
25278                                   6.0

        PreDK_MISSING_VALUES_TOTAL_PREV_min  PreDK_AMT_DECLINED_sum  \
136998                                  0.0                     0.0
10058                                   0.0                   -37.0
126057                                  0.0                   -12.0
98957                                   0.0                  -187.0
25278                                   0.0                     2.0

        PreDK_AMT_DECLINED_mean  PreDK_AMT_DECLINED_max  \
136998                 0.000000                     0.0
10058                -12.333333                     1.0
126057                -3.000000                     9.0
98957                -62.333333                    27.0
25278                  0.400000                     4.0

        PreDK_AMT_DECLINED_min  PreDK_AMT_CREDIT_GOODS_RATIO_sum  \
136998                     0.0                          1.000000
10058                    -38.0                          3.074524
126057                   -23.0                          3.000663
98957                   -203.0                          3.285185
25278                     -2.0                          2.992196

        PreDK_AMT_CREDIT_GOODS_RATIO_mean  PreDK_AMT_CREDIT_GOODS_RATIO_max  \
136998                           1.000000                          1.000000
10058                            1.024841                          1.082789
126057                           1.000221                          1.146497
98957                            1.095062                          1.347009
25278                            0.997399                          1.043478

        PreDK_AMT_CREDIT_GOODS_RATIO_min  PreDK_AMT_CREDIT_GOODS_DIFF_sum  \
136998                           1.000000                              0.0
10058                            0.991736                             37.0
126057                           0.875000                             12.0
98957                            0.891566                            187.0
25278                            0.948718                             -2.0

        PreDK_AMT_CREDIT_GOODS_DIFF_mean  PreDK_AMT_CREDIT_GOODS_DIFF_max  \
136998                          0.000000                              0.0
10058                          12.333333                             38.0
126057                          3.000000                             23.0
98957                          62.333333                            203.0
25278                          -0.400000                              2.0

        PreDK_AMT_CREDIT_GOODS_DIFF_min  \
136998                              0.0
10058                              -1.0
126057                             -9.0
98957                             -27.0
25278                              -4.0

        PreDK_AMT_CREDIT_APPLICATION_RATIO_sum  \
136998                                1.000000
10058                                 2.931875
126057                                3.036356
98957                                 2.819473
25278                                 3.012387

        PreDK_AMT_CREDIT_APPLICATION_RATIO_mean  \
136998                                 1.000000
10058                                  0.977292
126057                                 1.012119
98957                                  0.939824
25278                                  1.004129

        PreDK_AMT_CREDIT_APPLICATION_RATIO_max  \
136998                                1.000000
10058                                 1.008333
126057                                1.142857
98957                                 1.121622
25278                                 1.054054

        PreDK_AMT_CREDIT_APPLICATION_RATIO_min  PreDK_AMT_INTEREST_sum  \
136998                                1.000000                    24.0
10058                                 0.923541                   -47.0
```

```
126057                              0.872222                   99.0
98957                               0.742386                  615.0
25278                               0.958333                  106.0

        PreDK_AMT_INTEREST_mean  PreDK_AMT_INTEREST_max  \
136998                12.000000                    24.0
10058                -15.666667                   115.0
126057                24.750000                    48.0
98957                205.000000                   436.0
25278                 21.200000                    72.0

        PreDK_AMT_INTEREST_min  PreDK_ANNUITY_CREDIT_RATIO_sum  \
136998                     0.0                        0.098485
10058                   -186.0                        0.168410
126057                     0.0                        0.450203
98957                     42.0                        0.157474
25278                      0.0                        0.437050

        PreDK_ANNUITY_CREDIT_RATIO_mean  PreDK_ANNUITY_CREDIT_RATIO_max  \
136998                         0.098485                        0.098485
10058                          0.056137                        0.100000
126057                         0.150068                        0.206349
98957                          0.052491                        0.064777
25278                          0.145683                        0.216216

        PreDK_ANNUITY_CREDIT_RATIO_min  NAME_CONTRACT_TYPE_modus_C  \
136998                        0.098485                         1.0
10058                         0.000000                         1.0
126057                        0.105556                         1.0
98957                         0.043147                         1.0
25278                         0.104167                         1.0

        NAME_CONTRACT_TYPE_modus_R  NAME_CONTRACT_TYPE_modus_X  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        NAME_CONTRACT_TYPE_first_C  NAME_CONTRACT_TYPE_first_R  \
136998                         1.0                         0.0
10058                          1.0                         0.0
126057                         1.0                         0.0
98957                          1.0                         0.0
25278                          1.0                         0.0

        NAME_CONTRACT_TYPE_first_X  NAME_CONTRACT_TYPE_last_C  \
136998                         0.0                        1.0
10058                          0.0                        1.0
126057                         0.0                        1.0
98957                          0.0                        1.0
25278                          0.0                        1.0

        NAME_CONTRACT_TYPE_last_R  NAME_CONTRACT_TYPE_last_X  \
136998                        0.0                        0.0
10058                         0.0                        0.0
126057                        0.0                        0.0
98957                         0.0                        0.0
25278                         0.0                        0.0

        NAME_CONTRACT_STATUS_modus_A  NAME_CONTRACT_STATUS_modus_C  \
136998                           1.0                           0.0
10058                            1.0                           0.0
126057                           1.0                           0.0
98957                            1.0                           0.0
25278                            1.0                           0.0

        NAME_CONTRACT_STATUS_modus_R  NAME_CONTRACT_STATUS_modus_U  \
136998                           0.0                           0.0
10058                            0.0                           0.0
126057                           0.0                           0.0
98957                            0.0                           0.0
25278                            0.0                           0.0

        NAME_CONTRACT_STATUS_first_A  NAME_CONTRACT_STATUS_first_C  \
136998                           1.0                           0.0
10058                            1.0                           0.0
126057                           1.0                           0.0
98957                            1.0                           0.0
25278                            1.0                           0.0

        NAME_CONTRACT_STATUS_first_R  NAME_CONTRACT_STATUS_first_U  \
136998                           0.0                           0.0
10058                            0.0                           0.0
126057                           0.0                           0.0
98957                            0.0                           0.0
25278                            0.0                           0.0

        NAME_CONTRACT_STATUS_last_A  NAME_CONTRACT_STATUS_last_C  \
136998                          0.0                          1.0
10058                           0.0                          1.0
126057                          1.0                          0.0
98957                           1.0                          0.0
25278                           1.0                          0.0

        NAME_CONTRACT_STATUS_last_R  NAME_CONTRACT_STATUS_last_U  \
```

```
136998                        0.0                        0.0
10058                         0.0                        0.0
126057                        0.0                        0.0
98957                         0.0                        0.0
25278                         0.0                        0.0

        NAME_PAYMENT_TYPE_modus_C  NAME_PAYMENT_TYPE_modus_N  \
136998                        1.0                        0.0
10058                         1.0                        0.0
126057                        1.0                        0.0
98957                         1.0                        0.0
25278                         1.0                        0.0

        NAME_PAYMENT_TYPE_modus_X  NAME_PAYMENT_TYPE_first_C  \
136998                        0.0                        1.0
10058                         0.0                        1.0
126057                        0.0                        1.0
98957                         0.0                        1.0
25278                         0.0                        1.0

        NAME_PAYMENT_TYPE_first_N  NAME_PAYMENT_TYPE_first_X  \
136998                        0.0                        0.0
10058                         0.0                        0.0
126057                        0.0                        0.0
98957                         0.0                        0.0
25278                         0.0                        0.0

        NAME_PAYMENT_TYPE_last_C  NAME_PAYMENT_TYPE_last_N  \
136998                       0.0                       0.0
10058                        1.0                       0.0
126057                       1.0                       0.0
98957                        1.0                       0.0
25278                        1.0                       0.0

        NAME_PAYMENT_TYPE_last_X  CODE_REJECT_REASON_modus_C  \
136998                        1.0                        0.0
10058                         0.0                        0.0
126057                        0.0                        0.0
98957                         0.0                        0.0
25278                         0.0                        0.0

        CODE_REJECT_REASON_modus_H  CODE_REJECT_REASON_modus_L  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        CODE_REJECT_REASON_modus_S  CODE_REJECT_REASON_modus_V  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        CODE_REJECT_REASON_modus_X  CODE_REJECT_REASON_first_C  \
136998                         1.0                         0.0
10058                          1.0                         0.0
126057                         1.0                         0.0
98957                          1.0                         0.0
25278                          1.0                         0.0

        CODE_REJECT_REASON_first_H  CODE_REJECT_REASON_first_L  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        CODE_REJECT_REASON_first_S  CODE_REJECT_REASON_first_V  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        CODE_REJECT_REASON_first_X  CODE_REJECT_REASON_last_C  \
136998                         1.0                        0.0
10058                          1.0                        0.0
126057                         1.0                        0.0
98957                          1.0                        0.0
25278                          1.0                        0.0

        CODE_REJECT_REASON_last_H  CODE_REJECT_REASON_last_L  \
136998                        0.0                        0.0
10058                         0.0                        0.0
126057                        0.0                        0.0
98957                         0.0                        0.0
25278                         0.0                        0.0

        CODE_REJECT_REASON_last_S  CODE_REJECT_REASON_last_V  \
136998                        0.0                        0.0
10058                         0.0                        0.0
126057                        0.0                        0.0
98957                         0.0                        0.0
25278                         0.0                        0.0
```

```
        CODE_REJECT_REASON_last_X  NAME_TYPE_SUITE_modus_C  \
136998                        1.0                      0.0
10058                         1.0                      0.0
126057                        1.0                      0.0
98957                         1.0                      0.0
25278                         1.0                      0.0

        NAME_TYPE_SUITE_modus_F  NAME_TYPE_SUITE_modus_G  \
136998                      1.0                      0.0
10058                       0.0                      0.0
126057                      0.0                      0.0
98957                       1.0                      0.0
25278                       0.0                      0.0

        NAME_TYPE_SUITE_modus_Kategorie_NaN  NAME_TYPE_SUITE_modus_O  \
136998                                  0.0                      0.0
10058                                   0.0                      0.0
126057                                  0.0                      0.0
98957                                   0.0                      0.0
25278                                   1.0                      0.0

        NAME_TYPE_SUITE_modus_S  NAME_TYPE_SUITE_modus_U  \
136998                      0.0                      0.0
10058                       1.0                      0.0
126057                      1.0                      0.0
98957                       0.0                      0.0
25278                       0.0                      0.0

        NAME_TYPE_SUITE_first_C  NAME_TYPE_SUITE_first_F  \
136998                      0.0                      1.0
10058                       0.0                      0.0
126057                      0.0                      0.0
98957                       0.0                      1.0
25278                       0.0                      0.0

        NAME_TYPE_SUITE_first_G  NAME_TYPE_SUITE_first_O  \
136998                      0.0                      0.0
10058                       0.0                      0.0
126057                      0.0                      0.0
98957                       0.0                      0.0
25278                       0.0                      0.0

        NAME_TYPE_SUITE_first_S  NAME_TYPE_SUITE_first_U  \
136998                      0.0                      0.0
10058                       1.0                      0.0
126057                      1.0                      0.0
98957                       0.0                      0.0
25278                       0.0                      0.0

        NAME_TYPE_SUITE_first_Kategorie_NaN  NAME_TYPE_SUITE_last_C  \
136998                                  0.0                     0.0
10058                                   0.0                     0.0
126057                                  0.0                     0.0
98957                                   0.0                     0.0
25278                                   1.0                     0.0

        NAME_TYPE_SUITE_last_F  NAME_TYPE_SUITE_last_G  \
136998                     0.0                     0.0
10058                      0.0                     0.0
126057                     0.0                     0.0
98957                      0.0                     0.0
25278                      0.0                     0.0

        NAME_TYPE_SUITE_last_O  NAME_TYPE_SUITE_last_S  \
136998                     0.0                     0.0
10058                      0.0                     0.0
126057                     0.0                     1.0
98957                      1.0                     0.0
25278                      0.0                     0.0

        NAME_TYPE_SUITE_last_U  NAME_TYPE_SUITE_last_Kategorie_NaN  \
136998                     0.0                                 1.0
10058                      1.0                                 0.0
126057                     0.0                                 0.0
98957                      0.0                                 0.0
25278                      0.0                                 1.0

        NAME_CLIENT_TYPE_modus_N  NAME_CLIENT_TYPE_modus_R  \
136998                       1.0                       0.0
10058                        0.0                       1.0
126057                       0.0                       1.0
98957                        0.0                       1.0
25278                        0.0                       1.0

        NAME_CLIENT_TYPE_modus_X  NAME_CLIENT_TYPE_first_N  \
136998                       0.0                       1.0
10058                        0.0                       1.0
126057                       0.0                       1.0
98957                        0.0                       1.0
25278                        0.0                       1.0

        NAME_CLIENT_TYPE_first_R  NAME_CLIENT_TYPE_first_X  \
136998                       0.0                       0.0
10058                        0.0                       0.0
126057                       0.0                       0.0
```

```
98957                       0.0                       0.0
25278                       0.0                       0.0

        NAME_CLIENT_TYPE_last_N  NAME_CLIENT_TYPE_last_R  \
136998                      0.0                      1.0
10058                       0.0                      1.0
126057                      0.0                      1.0
98957                       0.0                      1.0
25278                       0.0                      1.0

        NAME_CLIENT_TYPE_last_X  NAME_GOODS_CATEGORY_modus_A  \
136998                      0.0                          0.0
10058                       0.0                          0.0
126057                      0.0                          0.0
98957                       0.0                          1.0
25278                       0.0                          0.0

        NAME_GOODS_CATEGORY_modus_C  NAME_GOODS_CATEGORY_modus_D  \
136998                          1.0                          0.0
10058                           1.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_modus_E  NAME_GOODS_CATEGORY_modus_F  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_modus_G  NAME_GOODS_CATEGORY_modus_H  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_modus_I  NAME_GOODS_CATEGORY_modus_J  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_modus_M  NAME_GOODS_CATEGORY_modus_O  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          1.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_modus_P  NAME_GOODS_CATEGORY_modus_S  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_modus_T  NAME_GOODS_CATEGORY_modus_V  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_modus_W  NAME_GOODS_CATEGORY_modus_X  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          1.0

        NAME_GOODS_CATEGORY_first_A  NAME_GOODS_CATEGORY_first_C  \
136998                          0.0                          1.0
10058                           0.0                          1.0
126057                          0.0                          0.0
98957                           1.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_first_D  NAME_GOODS_CATEGORY_first_E  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_first_F  NAME_GOODS_CATEGORY_first_G  \
136998                          0.0                          0.0
10058                           0.0                          0.0
126057                          0.0                          0.0
98957                           0.0                          0.0
25278                           0.0                          0.0

        NAME_GOODS_CATEGORY_first_H  NAME_GOODS_CATEGORY_first_I  \
136998                          0.0                          0.0
```

```
10058                          0.0                          0.0
126057                         0.0                          0.0
98957                          0.0                          0.0
25278                          0.0                          0.0

        NAME_GOODS_CATEGORY_first_J  NAME_GOODS_CATEGORY_first_M  \
136998                         0.0                          0.0
10058                          0.0                          0.0
126057                         0.0                          1.0
98957                          0.0                          0.0
25278                          0.0                          1.0

        NAME_GOODS_CATEGORY_first_O  NAME_GOODS_CATEGORY_first_P  \
136998                         0.0                          0.0
10058                          0.0                          0.0
126057                         0.0                          0.0
98957                          0.0                          0.0
25278                          0.0                          0.0

        NAME_GOODS_CATEGORY_first_S  NAME_GOODS_CATEGORY_first_T  \
136998                         0.0                          0.0
10058                          0.0                          0.0
126057                         0.0                          0.0
98957                          0.0                          0.0
25278                          0.0                          0.0

        NAME_GOODS_CATEGORY_first_V  NAME_GOODS_CATEGORY_first_W  \
136998                         0.0                          0.0
10058                          0.0                          0.0
126057                         0.0                          0.0
98957                          0.0                          0.0
25278                          0.0                          0.0

        NAME_GOODS_CATEGORY_first_X  NAME_GOODS_CATEGORY_last_A  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        NAME_GOODS_CATEGORY_last_C  NAME_GOODS_CATEGORY_last_D  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         1.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        NAME_GOODS_CATEGORY_last_E  NAME_GOODS_CATEGORY_last_F  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        NAME_GOODS_CATEGORY_last_G  NAME_GOODS_CATEGORY_last_H  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        NAME_GOODS_CATEGORY_last_I  NAME_GOODS_CATEGORY_last_J  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        NAME_GOODS_CATEGORY_last_M  NAME_GOODS_CATEGORY_last_O  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          1.0                         0.0

        NAME_GOODS_CATEGORY_last_P  NAME_GOODS_CATEGORY_last_S  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        NAME_GOODS_CATEGORY_last_T  NAME_GOODS_CATEGORY_last_V  \
136998                         0.0                         0.0
10058                          0.0                         0.0
126057                         0.0                         0.0
98957                          0.0                         0.0
25278                          0.0                         0.0

        NAME_GOODS_CATEGORY_last_W  NAME_GOODS_CATEGORY_last_X  \
136998                         0.0                         1.0
10058                          0.0                         1.0
126057                         0.0                         0.0
98957                          0.0                         1.0
25278                          0.0                         0.0
```

```
        NAME_YIELD_GROUP_modus_X  NAME_YIELD_GROUP_modus_h  \
136998                       1.0                       0.0
10058                        1.0                       0.0
126057                       0.0                       1.0
98957                        0.0                       0.0
25278                        1.0                       0.0

        NAME_YIELD_GROUP_modus_l  NAME_YIELD_GROUP_modus_m  \
136998                       0.0                       0.0
10058                        0.0                       0.0
126057                       0.0                       0.0
98957                        1.0                       0.0
25278                        0.0                       0.0

        NAME_YIELD_GROUP_first_X  NAME_YIELD_GROUP_first_h  \
136998                       0.0                       0.0
10058                        0.0                       0.0
126057                       0.0                       1.0
98957                        0.0                       0.0
25278                        0.0                       1.0

        NAME_YIELD_GROUP_first_l  NAME_YIELD_GROUP_first_m  \
136998                       0.0                       1.0
10058                        0.0                       1.0
126057                       0.0                       0.0
98957                        1.0                       0.0
25278                        0.0                       0.0

        NAME_YIELD_GROUP_last_X  NAME_YIELD_GROUP_last_h  \
136998                      1.0                      0.0
10058                       0.0                      0.0
126057                      0.0                      0.0
98957                       0.0                      0.0
25278                       0.0                      0.0

        NAME_YIELD_GROUP_last_l  NAME_YIELD_GROUP_last_m  \
136998                      0.0                      0.0
10058                       1.0                      0.0
126057                      0.0                      1.0
98957                       1.0                      0.0
25278                       0.0                      1.0

        NFLAG_INSURED_ON_APPROVAL_modus_0.0  \
136998                                  1.0
10058                                   0.0
126057                                  1.0
98957                                   1.0
25278                                   1.0

        NFLAG_INSURED_ON_APPROVAL_modus_1.0  \
136998                                  0.0
10058                                   0.0
126057                                  0.0
98957                                   0.0
25278                                   0.0

        NFLAG_INSURED_ON_APPROVAL_modus_Kategorie_NaN  \
136998                                            0.0
10058                                             1.0
126057                                            0.0
98957                                             0.0
25278                                             0.0

        NFLAG_INSURED_ON_APPROVAL_first_0.0  \
136998                                  1.0
10058                                   1.0
126057                                  1.0
98957                                   1.0
25278                                   1.0

        NFLAG_INSURED_ON_APPROVAL_first_1.0  \
136998                                  0.0
10058                                   0.0
126057                                  0.0
98957                                   0.0
25278                                   0.0

        NFLAG_INSURED_ON_APPROVAL_first_Kategorie_NaN  \
136998                                            0.0
10058                                             0.0
126057                                            0.0
98957                                             0.0
25278                                             0.0

        NFLAG_INSURED_ON_APPROVAL_last_0.0  \
136998                                 0.0
10058                                  0.0
126057                                 1.0
98957                                  1.0
25278                                  1.0

        NFLAG_INSURED_ON_APPROVAL_last_1.0  \
136998                                 0.0
10058                                  0.0
126057                                 0.0
98957                                  0.0
```

```
25278                                0.0

        NFLAG_INSURED_ON_APPROVAL_last_Kategorie_NaN
136998                                           1.0
10058                                            1.0
126057                                           0.0
98957                                            0.0
25278                                            0.0
```

In [88]:
```
# auf NaN prüfen
df.isna().sum().sum()
```

Out[88]:
```
6592669
```

Es sind ca. 6,6 Millionen fehlende Werte im Datensatz enthalten

In [89]:
```
# setze den Datentyp von `TARGET` auf `int`
df = df.astype({'TARGET':'int'})

# setze den Datentyp von `TYPE` auf `int`, wobei die Ausprägung `TRAIN`
# mit `0` und die Ausprägung `TEST` mit `1` kodiert wird
df['TYPE'].replace("TRAIN", "0", inplace= True)
df['TYPE'].replace("TEST", "1", inplace= True)
df = df.astype({'TYPE':'int'})

# da nun alle Spalten numerisch sind, können alle fehlenden Werte nun durch 0 ersetzt werden
df = df.fillna(0)

# es treten nun keine fehlenden Werte mehr auf
df.isna().sum().sum()
```

Out[89]:
```
0
```

In [90]:
```
dftrain = df[df['TYPE'] == 0]
dftrain = dftrain.drop(['TYPE'], axis=1)
```

In [91]:
```
dftrain.shape
```

Out[91]:
```
(160440, 421)
```

In [92]:
```
dftest = df[df['TYPE'] == 1]
dftest = dftest.drop(['TYPE'], axis=1)
```

In [93]:
```
dftest.shape
```

Out[93]:
```
(30856, 421)
```

Qualitätssicherung: Trainings- und Testdatensatz enthalten jeweils wieder dieselbe Anzahl an Zeilen wie in Aufgabe A-1.

Aufgabe A-7: Alternative Möglichkeiten zum Umgang mit Missing Values [Lernziel 6.1.1; 5 Punkte]

In den vorherigen Aufgaben wurden fehlenden Werte in numerischen Spalten durch 0 ersetzt. Es soll diskutiert werden, ob diese Vorgehensweise bei der vorliegenden Aufgabenstellung die beste bzw. sinnvollste Möglichkeit ist. Ist eine andere numerische Ersetzung geeigneter? Wann könnte ein Ansatz mit einer nicht-numerischen Ersetzung sinnvoller sein? Was sind die Schwierigkeiten hierbei?

Lösungsvorschlag:

Bei den eingelesenen Daten handelt es sich um vorangegangene Kreditanträge. Ein simples Ersetzen der fehlenden Daten durch einen numerischen Wert, insbesondere Mittelwert / Medium erzeugt für den Kunden demnach implizit eine Kredithistorie mit über den Bestand durchschnittlichen Daten. Dies verwässert also den Informationsgehalt "kein Antrag vorhanden".

Auch eine Ersetzung durch 0 löst dieses Problem nicht vollständig. So ergibt es zwar ein stimmigeres Bild bei Werten wie der Höhe der bisher beantragten Kredite (AMT_CREDIT_SUM, AMT_CREDIT_SUM_DEBT), kann aber in anderen Kategorien ebenso in die Irre führen (z.B. DAYS_CREDIT_ENDDATE, AMT_CREDIT_MAX_OVERDUE).

Darüber hinaus ist es nicht die gleiche Information, ob ein Kunde nie einen vorherigen Kredit beantragt hat oder aber keinen ausstehenden Betrag an Schulden aufweist. Diese Information geht verloren.

Eine Möglichkeit, diese Information zu bewahren, wäre die Transformation der numerischen Variablen in den entsprechenden Tabellen in kategorische Variablen durch die Einteilung in Wertebereiche. Anschließend könnte man die Kategorie "keine Anträge vorhanden" für die NaNs ergänzen und würde damit dem Sachverhalt besser gerecht.

Dies stellt dann jedoch die Herausforderung dar, geeignete Wertebereiche zu wählen. Hierzu könnte man iterativ mögliche Grenzen für die Wertebereiche im Modell ausprobieren, um eine bestmögliche Einteilung zu finden. Dieses Verfahren ist jedoch mit einem gewissen Mehraufwand verbunden.

Aufgabe A-8: Merkmalsvorauswahl klassisch (IV) [Lernziel 6; 5 Punkte]

Im Credit Scoring wird für die Merkmalsauswahl oftmals das Konzept des „Information Value“ (kurz: IV) verwendet. Der in Anhang 2 angegebene Link enthält eine kurze Einführung in die Thematik sowie R- und Python-Code zur Berechnung dieser Werte. Der darin angegeben Code zur Bestimmung der Information Values soll für alle Merkmale des Datensatzes angewendet werden. Dabei sind für jedes Merkmal die betrachteten Klassen, Kreditausfallquoten und „Weight of Evidence“-Werte zu berechnen und die ersten fünf Zeilen der erzeugten Tabellen auszugeben. Abschließend soll für die 15 Merkmale mit den höchsten Information Values ein Balkendiagramm erstellt werden. Dabei soll der entsprechende Merkmalsname nach IV absteigend sortiert angezeigt und der jeweils zugehörige IV via Balkenlänge dargestellt werden.

Lösungsvorschlag:

In [94]:
```
df = dftrain.copy(deep = True)
```

Für jedes Merkmal des Dataframes `df` werden die beiden Größen "Information Value" und "Weight of Evidence" mittels der oben angegebenen Funktion `iv_woe` bestimmt. Dabei wird eine Aufteilung in zehn Bins verwendet.

In [95]:
```
iv, woe = iv_woe(data = df, target = 'TARGET', bins=10, show_woe = False)
```

Ausgabe der ersten Zeilen von `iv` und `woe`:

In [96]:
```
print(iv.head(5))
```

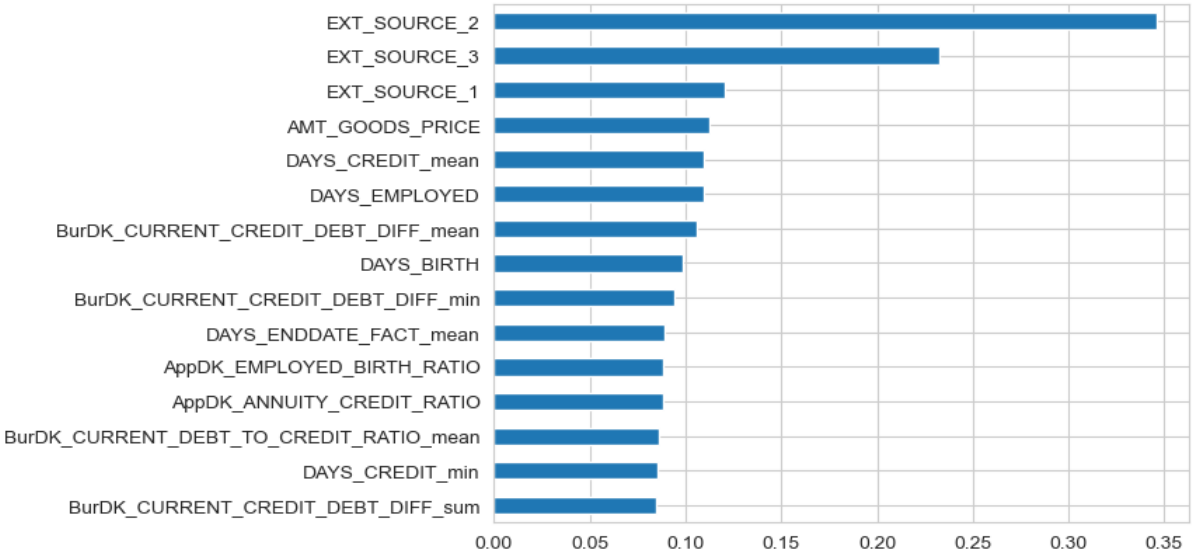
```
             Variable        IV
0          SK_ID_CURR  0.000519
0  NAME_CONTRACT_TYPE  0.015662
0         CODE_GENDER  0.044877
0        FLAG_OWN_CAR  0.010951
0     FLAG_OWN_REALTY  0.000831
```

In [97]:
```
print(woe.head(5))
```

```
     Variable                 Cutoff      N  Events  % of Events  Non-Events  \
0  SK_ID_CURR  (200000.999, 219146.9]  16044    1269     0.099976       14775
1  SK_ID_CURR    (219146.9, 238271.8]  16044    1292     0.101788       14752
2  SK_ID_CURR    (238271.8, 257433.7]  16044    1305     0.102813       14739
3  SK_ID_CURR    (257433.7, 276588.6]  16044    1271     0.100134       14773
4  SK_ID_CURR    (276588.6, 295748.5]  16044    1216     0.095801       14828

   % of Non-Events       WoE            IV
0         0.100002  0.000257  6.587928e-09
1         0.099846 -0.019263  3.741013e-05
2         0.099758 -0.030157  9.210466e-05
3         0.099988 -0.001453  2.113935e-07
4         0.100361  0.046500  2.120353e-04
```

Plot des Information Values der Merkmale des Datensatzes `df` in absteigender Reihenfolge, beschränkt auf die 15 Merkmale mit den höchsten Information Value Werten.

In [98]:
```
inf_values = pd.Series(iv.IV.to_list(), index=iv.Variable.to_list())
inf_values.nlargest(15).plot(kind='barh').invert_yaxis()
```



Aufgabe A-9: Merkmalsvorauswahl über ein Random-Forest-Modell [Lernziel 6.1.1; 4 Punkte]

Als Alternative zum Information Value (IV) ist ein Random-Forest-Modell mit 250 Bäumen und mindestens 25 Datenpunkten je Blatt zu berechnen. Die "Feature-Importance" (kurz: FI) ist für die Top 10 Merkmale grafisch darzustellen.

Im Anschluss ist der Trainingsdatensatz so zu modifizieren, dass dieser neben den Spalten `TARGET` und `SK_ID_CURR` nur noch diejenigen Spalten mit IV ≥ 0.01 und FI ≥ 0.001 enthält.

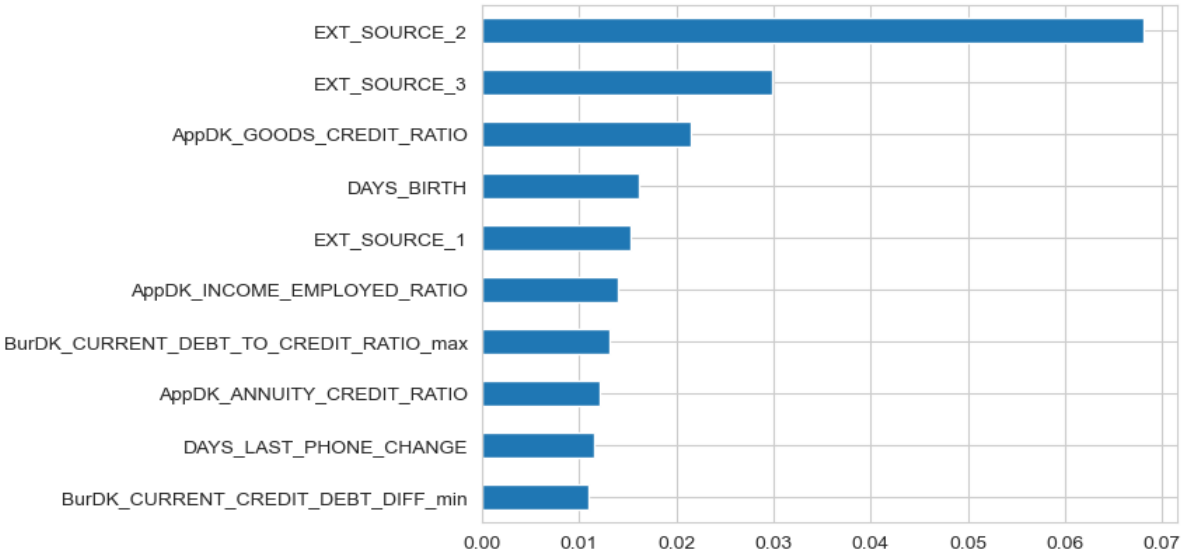
Anmerkung: Der Trainingsdatensatz sollte zum Ende dieser Aufgabe noch mindestens 100 Spalten enthalten. Wenn das nicht der Fall sein sollte, so sind die Schwellenwerte für IV und FI geeignet abzusenken, bis mindestens 100 Spalten vorliegen.

Lösungsvorschlag:

In [99]:
```
tic = time.time()
rf = RandomForestClassifier(n_estimators = 250, min_samples_leaf = 25, random_state = 42, n_jobs=-1)
X = df.loc[:, df.columns != 'TARGET']
y = df.loc[:, df.columns == 'TARGET']

rf.fit(X, y)
print("time (sec):" + "%6.0f" % (time.time() - tic))
```

```
time (sec):    26
```

In [100…
```
feat_importances = pd.Series(rf.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh').invert_yaxis()
```



In [101…
```
inf_value = iv.copy(deep=True)
inf_value = inf_value.reset_index().drop(['index'], axis=1)

feat_imp = pd.DataFrame({'Variable':feat_importances.index, 'FI': feat_importances.values})

# Gemeinsames Dataframe erstellen:
comb_iv_fi = pd.merge(inf_value, feat_imp, on = "Variable", how = "inner")
print(comb_iv_fi.head())
```

```
            Variable        IV        FI
0          SK_ID_CURR  0.000519  0.008010
1  NAME_CONTRACT_TYPE  0.015662  0.000891
2         CODE_GENDER  0.044877  0.005743
3        FLAG_OWN_CAR  0.010951  0.000800
4     FLAG_OWN_REALTY  0.000831  0.000788
```

In [102…
```
comb_iv_fi['keep_var'] = comb_iv_fi.apply(lambda row_: (row_.IV >= 0.01) & (row_.FI >= 0.001), axis=1)

keep_columns = comb_iv_fi[comb_iv_fi['keep_var'] == True].reset_index().drop(['index', 'keep_var'], axis=1)
len(keep_columns)
```

Out[102]:
```
120
```

In [103…
```
columns_to_keep = ['SK_ID_CURR', 'TARGET']
columns_to_keep.extend(keep_columns['Variable'])
columns_to_keep
```

Out[103]:
```
['SK_ID_CURR',
 'TARGET',
 'CODE_GENDER',
 'AMT_INCOME_TOTAL',
 'AMT_CREDIT',
 'AMT_ANNUITY',
 'AMT_GOODS_PRICE',
 'REGION_POPULATION_RELATIVE',
 'DAYS_BIRTH',
 'DAYS_EMPLOYED',
 'DAYS_REGISTRATION',
 'DAYS_ID_PUBLISH',
 'OWN_CAR_AGE',
 'FLAG_WORK_PHONE',
 'REG_CITY_NOT_WORK_CITY',
 'EXT_SOURCE_1',
 'EXT_SOURCE_2',
 'EXT_SOURCE_3',
 'DAYS_LAST_PHONE_CHANGE',
 'AppDK_ANNUITY_CREDIT_RATIO',
 'AppDK_GOODS_CREDIT_RATIO',
 'AppDK_CREDIT_INCOME_RATIO',
 'AppDK_GOODS_INCOME_RATIO',
 'AppDK_EMPLOYED_BIRTH_RATIO',
 'AppDK_INCOME_EMPLOYED_RATIO',
 'AppDK_INCOME_BIRTH_RATIO',
 'AppDK_CAR_BIRTH_RATIO',
 'NAME_INCOME_TYPE_P',
 'NAME_INCOME_TYPE_W',
 'NAME_EDUCATION_TYPE_H',
 'NAME_EDUCATION_TYPE_S',
 'NAME_FAMILY_STATUS_M',
 'NAME_FAMILY_STATUS_S',
 'OCCUPATION_TYPE_L',
 'REGION_RATING_CLIENT_3',
 'REGION_RATING_CLIENT_W_CITY_3',
 'ORGANIZATION_TYPE_X',
 'DAYS_CREDIT_sum',
 'DAYS_CREDIT_mean',
 'DAYS_CREDIT_max',
 'DAYS_CREDIT_min',
 'DAYS_CREDIT_ENDDATE_sum',
 'DAYS_CREDIT_ENDDATE_mean',
 'DAYS_CREDIT_ENDDATE_max',
 'DAYS_CREDIT_ENDDATE_min',
 'DAYS_ENDDATE_FACT_sum',
 'DAYS_ENDDATE_FACT_mean',
 'DAYS_ENDDATE_FACT_min',
 'AMT_CREDIT_MAX_OVERDUE_sum',
 'AMT_CREDIT_MAX_OVERDUE_mean',
 'AMT_CREDIT_MAX_OVERDUE_max',
 'AMT_CREDIT_SUM_sum',
 'AMT_CREDIT_SUM_mean',
 'AMT_CREDIT_SUM_max',
 'AMT_CREDIT_SUM_min',
 'AMT_CREDIT_SUM_DEBT_min',
 'DAYS_CREDIT_UPDATE_sum',
 'DAYS_CREDIT_UPDATE_mean',
 'DAYS_CREDIT_UPDATE_max',
 'DAYS_CREDIT_UPDATE_min',
 'BurDK_CREDIT_DURATION_sum',
 'BurDK_CREDIT_DURATION_mean',
 'BurDK_CREDIT_DURATION_max',
 'BurDK_CREDIT_DURATION_min',
 'BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_sum',
 'BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_mean',
 'BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_max',
 'BurDK_CURRENT_DEBT_TO_CREDIT_RATIO_min',
 'BurDK_CURRENT_CREDIT_DEBT_DIFF_sum',
 'BurDK_CURRENT_CREDIT_DEBT_DIFF_mean',
 'BurDK_CURRENT_CREDIT_DEBT_DIFF_max',
 'BurDK_CURRENT_CREDIT_DEBT_DIFF_min',
 'BurDK_CREDIT_ENDDATE_UPDATE_DIFF_sum',
 'BurDK_CREDIT_ENDDATE_UPDATE_DIFF_mean',
 'BurDK_CREDIT_ENDDATE_UPDATE_DIFF_max',
 'CREDIT_ACTIVE_modus_C',
 'AMT_ANNUITY_sum_prev',
 'AMT_ANNUITY_mean_prev',
 'AMT_ANNUITY_max_prev',
 'AMT_APPLICATION_sum',
 'AMT_APPLICATION_mean',
 'AMT_APPLICATION_max',
 'AMT_CREDIT_sum',
 'AMT_CREDIT_mean',
 'AMT_CREDIT_max',
 'AMT_DOWN_PAYMENT_sum',
 'AMT_DOWN_PAYMENT_mean',
 'AMT_DOWN_PAYMENT_max',
 'AMT_GOODS_PRICE_sum',
 'AMT_GOODS_PRICE_mean',
 'AMT_GOODS_PRICE_max',
 'DAYS_DECISION_sum',
 'DAYS_DECISION_mean',
 'DAYS_DECISION_min',
 'PreDK_MISSING_VALUES_TOTAL_PREV_mean',
 'PreDK_MISSING_VALUES_TOTAL_PREV_min',
```

```
'PreDK_AMT_DECLINED_sum',
'PreDK_AMT_DECLINED_mean',
'PreDK_AMT_DECLINED_max',
'PreDK_AMT_DECLINED_min',
'PreDK_AMT_CREDIT_GOODS_RATIO_sum',
'PreDK_AMT_CREDIT_GOODS_RATIO_mean',
'PreDK_AMT_CREDIT_GOODS_RATIO_max',
'PreDK_AMT_CREDIT_GOODS_RATIO_min',
'PreDK_AMT_CREDIT_GOODS_DIFF_sum',
'PreDK_AMT_CREDIT_GOODS_DIFF_mean',
'PreDK_AMT_CREDIT_GOODS_DIFF_max',
'PreDK_AMT_CREDIT_GOODS_DIFF_min',
'PreDK_AMT_CREDIT_APPLICATION_RATIO_sum',
'PreDK_AMT_CREDIT_APPLICATION_RATIO_mean',
'PreDK_AMT_CREDIT_APPLICATION_RATIO_max',
'PreDK_AMT_CREDIT_APPLICATION_RATIO_min',
'PreDK_AMT_INTEREST_min',
'PreDK_ANNUITY_CREDIT_RATIO_sum',
'PreDK_ANNUITY_CREDIT_RATIO_mean',
'PreDK_ANNUITY_CREDIT_RATIO_max',
'PreDK_ANNUITY_CREDIT_RATIO_min',
'NAME_CONTRACT_STATUS_modus_R',
'NAME_CONTRACT_STATUS_last_R',
'NAME_YIELD_GROUP_first_h',
'NAME_YIELD_GROUP_last_h',
'NFLAG_INSURED_ON_APPROVAL_modus_Kategorie_NaN']
```

In [104…
```
train_featsel = df[columns_to_keep]
```

Aufgabe A-10: Speicherung der Datensätze für die Weiterverarbeitung [Lernziel 6.1.1; 2 Punkte]

Für die Weiterverarbeitung in Teil B sollen folgende Datensätze im CSV-Format abgespeichert werden:

- **Der Trainingsdatensatz mit allen verfügbaren Spalten. Name:** `train_komplett.csv`
- **Der Trainingsdatensatz mit eingeschränkten Spalten (nach den Aufgaben A-8 und A-9). Name:** `train_featsel.csv`
- **Der Testdatensatz mit aufbereiteten Merkmalen (nach der Aufgabe A-6). Name:** `test_komplett.csv`

Lösungsvorschlag:

In [105…
```
dftrain.to_csv('../../1_Datensatz/train_komplett.csv.zip', sep=';', decimal=',', float_format='%.3f', compression='zip', index=False)
```

In [106…
```
train_featsel.to_csv('../../1_Datensatz/train_featsel.csv.zip', sep=';', decimal=',', float_format='%.3f', compression='zip', index=False)
```

In [107…
```
dftest.to_csv('../../1_Datensatz/test_komplett.csv.zip', sep=';', decimal=',', float_format='%.3f',compression='zip', index=False)
```

Lösungsvorschlag zur Prüfungsaufgabe Immersion 2023

Block B: Modellierung von Kreditrisiken [105 Punkte]

Hinweise:

- **Für diesen Block ist ein Notebook (R-Markdown oder Jupyter Notebook) mit R oder Python zu erstellen.**
- **Die Modelle sollen anhand der Metrik „AUC“ (also der Fläche unter der ROC-Kurve zwischen der vorhergesagten Wahrscheinlichkeit und dem beobachteten Wert) bewertet und verglichen werden.**
- **Für die binäre Klassifikation der Prognosewahrscheinlichkeiten soll die im Trainingsdatensatz beobachtete Ereignisrate (Anteil Kreditausfälle) als Schwellenwert verwendet werden.**.
- **Für die Darstellung der Ergebnisse sind die Klarnamen der Variablen zu verwenden.**
- **Zur Orientierung: Die gesamte Laufzeit des zum Aufgabenteil B entwickelten Lösungsvorschlages beträgt bei einem schnellen Rechner (16 Rechenkerne) in etwa eine Viertelstunde, bei einem Standardrechner (4 Rechenkerne) rund eine Stunde. In diesem Laufzeitbudget ist eine AUC (Prognosegüte) von mindestens 0,75 erreichbar. Das Notebook ist einschließlich Output rund 1 MB groß.**
- **Benötigte Materialien: Erzeugte Ergebnisdateien aus Block A**

Variante: GPU-Verwendung bei LightGBM, XGBoost und Keras

Die Ausführung des Notebooks auf GPU führt insbesondere bei XGBoost zu stark verkürzten Laufzeiten. Gegenüber der CPU-Ausführung ist XGBoost nun sogar deutlich schneller als lightGBM.

Empfehlung: Auch bei XGBoost grundsätzlich die schnelle, von lightGBM bekannte Histogramm-Methode verwenden (tree_method = 'hist' bzw. 'gpu_hist').

Aufgabe B-1: Modellierung vorbereiten [Lernziel 3.3/3.4, 2.2; 10 Punkte]

---

a) "Programmbibliotheken und Reproduzierbarkeit": Binden Sie alle für die weiteren Aufgaben benötigten Bibliotheken ein und sorgen Sie für die Reproduzierbarkeit Ihres Notebooks.

b) "Daten einlesen und aufbereiten": Lesen Sie die im Aufgabenteil A erzeugte Datei "train_featsel.csv" in einen "DataFrame" ein, zeigen Sie die Anzahl der Zeilen und Spalten sowie die ersten beiden Datensätze an. Geben Sie eine Häufigkeitstabelle des Merkmals TARGET aus und prüfen Sie die Übereinstimmung mit Aufgabenteil A. Entfernen Sie alle Variablen aus dem DataFrame, die nicht für die Modellierung bestimmt sind (z.B. IDs) und trennen Sie den DataFrame in die üblichen Modellierungsmatrizen X und y (TARGET) auf.

c) "Daten aufteilen und skalieren": Teilen Sie die Daten reproduzierbar in Trainings- und Validierungsdaten auf. Wählen Sie dazu ein geeignetes Aufteilungsverhältnis aus und begründen Sie ihre Wahl. Zeigen Sie die Häufigkeitsverteilung der Zielgröße in der Validierungsstichprobe (y_val) an. Wählen Sie ein geeignetes Skalierungsverfahren für die Features aus, beschreiben Sie kurz das Verfahren, nennen Sie die Aufgabenteile für die es benötigt wird und begründen Sie Ihre Wahl. Führen Sie die Skalierung auf Basis der Werteverteilungen der Trainingsdaten durch. Falls nicht anders vermerkt, sind die in diesem Aufgabenteil erzeugten Trainings- und Validierungsdaten bei den folgenden Aufgaben B-3 bis B-7 zu verwenden.

d) "Prozessorgeschwindigkeit und Parallelisierung": Während sich die Taktfrequenz der Mikroprozessoren zwischen den Jahren 1970 und 2000 etwa vertausendfacht hat, stagniert sie seit Anfang der 2000er Jahre auf Gigahertz-Niveau. Suchen Sie im Internet nach einem glaubhaften Schaubild, das diesen Zusammenhang zeigt, prüfen Sie die Quelle und fügen Sie die Grafik in das Notebook ein. Beschreiben Sie die Ursache dieser Stagnation und erläutern Sie den Zusammenhang zu Mehrkernprozessoren und zur Parallelisierung. Geben Sie die Taktfrequenz der CPU, auf dem Ihr Notebook ausgeführt wird, sowie die Anzahl der zur Verfügung stehenden Kerne aus.

Stellen Sie sicher, dass Sie beim "fitten" der rechenintensiven Modelle ab Aufgabe B-3 alle zur Verfügung stehenden Prozessorkerne (bzw. Threads) verwenden.

Lösungsvorschlag:

a) Programmbibliotheken und Reproduzierbarkeit

In [1]:
```
# Import nötiger Pakete

import pandas as pd
import numpy as np
from scipy.stats import uniform, loguniform
```

```
# System und Konfiguration
import sys
print(sys.version)
import time
import psutil
import os
import gc
from sklearn._config import get_config, set_config
from IPython.display import HTML

# Unterdrücke unnötige Warnungen
import warnings
warnings.filterwarnings('ignore')

# Plotting Bibliotheken
import matplotlib.pyplot as plt
#plt.rcParams.update({'font.size': 12})
#from matplotlib.ticker import MaxNLocator
#import pylab as p
import seaborn as sns
from scikitplot.metrics import cumulative_gain_curve
import graphviz

# Datenaufbereitung
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve, confusion_matrix

# Modellierungsbilbiotheken
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
```

```
3.10.10 | packaged by conda-forge | (main, Mar 24 2023, 20:08:06) [GCC 11.3.0]
```

In [2]:
```
# Prevent TensorFlow From Fully Allocating GPU Memory

import tensorflow as tf
# Ref: https://www.tensorflow.org/guide/gpu#limiting_gpu_memory_growth
gpus = tf.config.list_physical_devices('GPU')
if gpus:
  try:
    # Currently, memory growth needs to be the same across GPUs
    for gpu in gpus:
      tf.config.experimental.set_memory_growth(gpu, True)
    logical_gpus = tf.config.list_logical_devices('GPU')
    print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
  except RuntimeError as e:
    # Memory growth must be set before GPUs have been initialized
    print(e)
```

```
1 Physical GPUs, 1 Logical GPUs
```

In [3]:
```
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization
from tensorflow.keras.regularizers import l2
from keras.wrappers.scikit_learn import KerasClassifier
```

In [4]:
```
# Reproduzierbarkeit
seed = 42

# Startzeitpunkt (für Laufzeitmessung)
StartNotebook = time.time()
```

b) Daten einlesen und aufbereiten

In [5]:
```
# Einlesen der Trainingsdaten
df = pd.read_csv('train_featsel.csv', sep=';', decimal=',')

# Zeige alle Spalten. Verberge keine
pd.set_option('display.max_columns', None)

print("\nAnzahl Zeilen und Spalten: ", df.shape)

# Zeige die ersten 2 Zeilen der Daten um einen Überblick zu gewinnen
df.head(2)
```

```
Anzahl Zeilen und Spalten:  (160440, 122)
```

Out[5]:

	SK_ID_CURR	TARGET	CODE_GENDER	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	DAYS
0	200001	0	0	450	1413	73.0	1413.0	0.046	
1	200002	1	0	450	450	51.0	450.0	0.046	

In [6]:
```
# Häufigkeitstabelle des Merkmals TARGET
df.TARGET.value_counts()
```

Out[6]:
```
0    147747
1     12693
Name: TARGET, dtype: int64
```

In [7]:
```
print("Anzahl fehlende Werte: ", df.isna().sum().sum())

# ML-Matrizen anlegen (u.a.ID entfernen)
X = df.drop(['SK_ID_CURR','TARGET'], axis=1)
y = df['TARGET']

# Speichere Anzahl der Features (für NN-Dimensionierung)
n_features = X.shape[1]
print("Anzahl der Features in X: ",n_features)

# Erzeuge Liste mit den "Feature Names"
feature_names=list(X.columns)
```

```
Anzahl fehlende Werte:  0
Anzahl der Features in X:  120
```

c) Daten aufteilen und skalieren

Antwort zur Aufteilung: Die Daten werden in den Anteilen 75% für Training und 25% für Validierung aufgeteilt. Dies entspricht auch dem Standardwert der verwendeten Prozedur. Dadurch soll sicher gestellt werden, dass insbesondere für das Modelltraining, aber auch für die Validierung genügend Datensätze vorhanden sind.

In [8]:
```
# Train-Test-Split und Häufigkeitsverteilung
X_train_raw, X_val_raw, y_train, y_val = train_test_split(X, y, test_size=0.25, random_state=seed)
print("Häufigkeitsverteilung der Zielgröße in y_val:\n",y_val.value_counts())
```

```
Häufigkeitsverteilung der Zielgröße in y_val:
 0    37008
1     3102
Name: TARGET, dtype: int64
```

Antwort zur Skalierung: Es wird das in der Python-Bibliothek "sklearn.preprocessing" implementierte Skalierungsverfahren "StandardScaler", das für jedes Feature eine lineare Transformation (Normalisierung) mit Mittelwert 0 und Standardabweichung 1 vornimmt, ausgewählt. Die skalierten Features werden in den Aufgabenteilen, in denen Hauptkomponentenanalyse, Logistische Regression und Neuronale Netze angewendet werden sollen, benötigt.

In [9]:
```
# Feature-Skalierung mit Standard-Scaler auf Basis der Werteverteilungen der Trainingsdaten
scaler = StandardScaler()
scaler.fit(X_train_raw)
X_train = pd.DataFrame(scaler.transform(X_train_raw), columns = feature_names)
X_val = pd.DataFrame(scaler.transform(X_val_raw), columns = feature_names)
```

Skalierungseffekt für die ersten fünf Variablen anzeigen:

In [10]:
```
# vor Skalierung:
X_train_raw.iloc[:, : 5].describe().loc[['mean', 'std','min','max']]
```

Out[10]:

	CODE_GENDER	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
mean	0.359968	321.316995	1189.069742	54.533982	1066.286462
std	0.479993	218.329219	812.237063	29.487195	744.451523
min	0.000000	53.000000	90.000000	0.000000	0.000000
max	1.000000	36001.000000	8100.000000	517.000000	8100.000000

In [11]:
```
# Nach Skalierung:
X_train.iloc[:, : 5].describe().loc[['mean', 'std','min','max']]
```

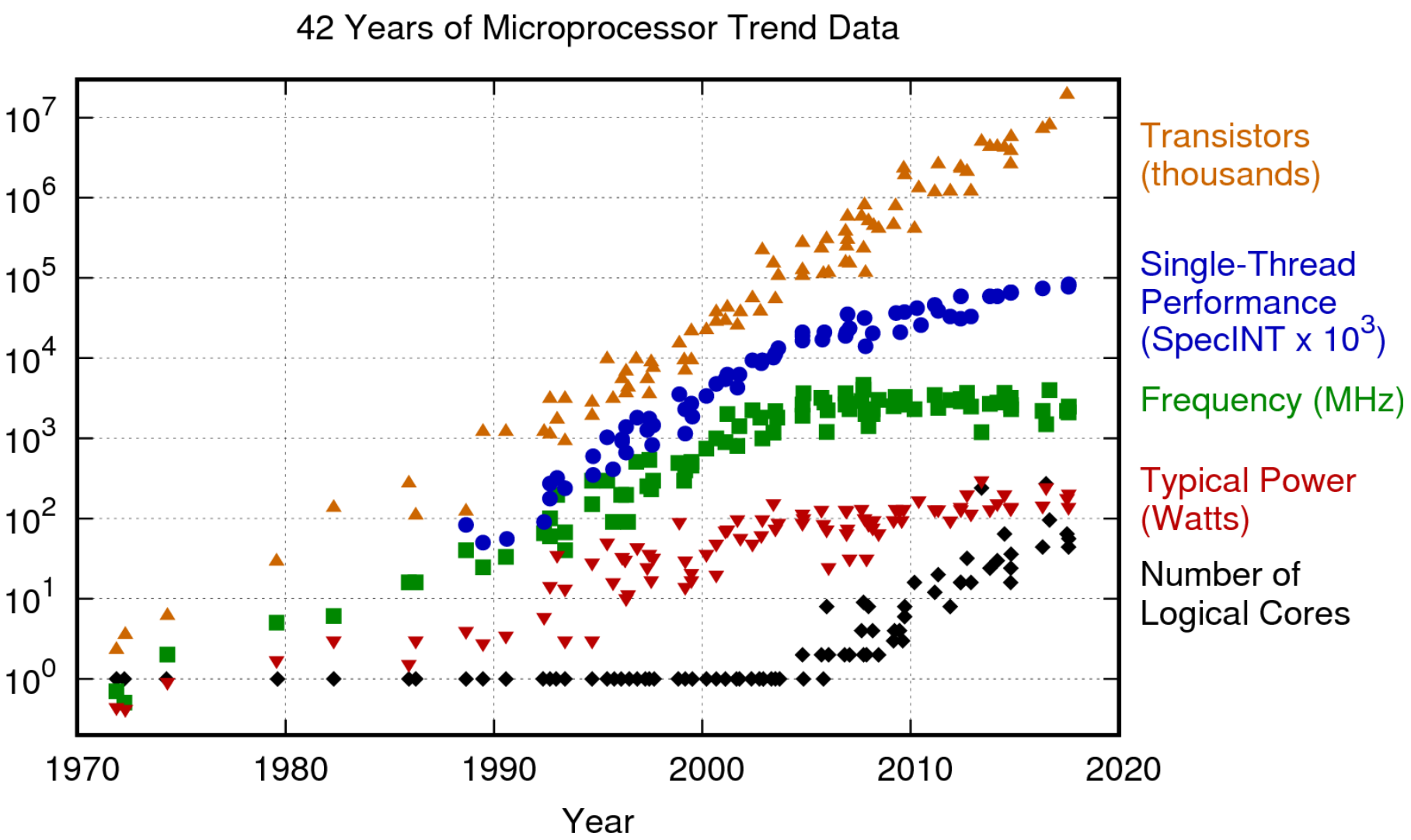
Out[11]:

	CODE_GENDER	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
mean	1.925014e-17	-4.599957e-17	6.672594e-17	5.786852e-17	1.352234e-16
std	1.000004e+00	1.000004e+00	1.000004e+00	1.000004e+00	1.000004e+00
min	-7.499486e-01	-1.228961e+00	-1.353145e+00	-1.849420e+00	-1.432317e+00
max	1.333425e+00	1.634222e+02	8.508549e+00	1.568369e+01	9.448221e+00

d) Prozessorgeschwindigkeit und Parallelisierung

Das folgende Schaubild zur Entwicklung (u.a.) der Prozessorgeschwindigkeit gemäß https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/, zeigt den in der Aufgabenstellung beschriebenen Zusammenhang, siehe "Frequency (MHz)" (grüne Quadrate). Die

Darstellung ist glaubhaft, die Zusammenstellung der zugrunde liegenden Daten ab 2010 ist einsehbar: https://github.com/karlrupp/microprocessor-trend-data.



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Ursache dieser Stagnation und Zusammenhang mit Mehrkernprozessoren/Parallelisierung:

Wenn die Taktfrequenz einer CPU erhöht wird, muss auch mehr Energie eingesetzt werden, um die zusätzliche Rechenleistung bereitzustellen. Dieser Energieeinsatz erwärmt die CPU, die dadurch sehr heiß und beschädigt werden kann. Die physikalischen Grenzen, diese Wärme von der CPU effektiv abzuführen, wurden bereits Anfang der 2000er Jahre erreicht und mit der Entwicklung von Mehrkernprozessoren und neuen Architekturen begonnen. Diese Änderungen ermöglichen es CPUs, mehrere Aufgaben "parallelisiert", also gleichzeitig auszuführen, was zu einer besseren Leistung führt, ohne die Taktfrequenz erhöhen zu müssen.

Taktfrequenz (MHz) und Anzahl Rechenkerne der CPU anzeigen:

In [12]:
```
# CPU-Informationen besorgen und ausgeben
cpu_info = psutil.cpu_freq()
print("Taktfrequenz der CPU (MHz): ", round(cpu_info.current))

num_cores = os.cpu_count()
print("Anzahl Rechenkerne der CPU: ", num_cores)
```

```
Taktfrequenz der CPU (MHz):  2000
Anzahl Rechenkerne der CPU:  2
```

Aufgabe B-2: Metriken und "Null-Modelle" [Lernziele 3.3/3.4, 6.1.2; 7 Punkte]

---

a) „Evaluationsmetriken“: Die vorgegebene Evaluationsmetrik ist „AUC“. Mögliche Alternativen wären die „Accuracy“ (Anteil der korrekten Klassifizierungen) und der sogenannte F1-Score. Erläutern Sie mit Bezug auf den vorliegenden Fall, welche Problematik die Accuracy aufweist, welche Vorteile der F1-Score gegenüber der AUC hat und was der Grund für die Beliebtheit der Metrik AUC sein könnte. Welche weiteren Bewertungskriterien sollten berücksichtigt werden, wenn auf Basis der Kreditausfallprognose konkrete Geschäftsentscheidungen gefällt werden sollen?

b) „Bewertungsfunktion schreiben“: Schreiben Sie eine Funktion, die auf Basis der Eingabeparameter y_obs (Beobachtete Werte), y_hat (Prognostizierte Wahrscheinlichkeit) und threshold (Schwellenwert für 0/1-Klassifikation) folgende Werte berechnet und in kompakter Form samt Bezeichnung ausgibt:

- **Beschreibende Statistik y_hat: count, mean, std, min, median, max**
- **Accuracy**
- **Confusion Matrix**
- **AUC**

c) "Null-Modell 1, Kreditausfälle nicht zu erwarten": Erstellen Sie ein einfaches "Bewertungsmodell", das für jeden Kreditantrag die Kreditausfallwahrscheinlichkeit mit 0 beziffert und wenden Sie die Bewertungsfunktion an.

d) "Null-Modell 2, Kreditausfälle sind rein zufällig": Erstellen Sie ein weiteres einfaches Bewertungsmodell, das auf Basis reproduzierbarer Zufallszahlen einem Teil der Kreditanträge die Kreditausfallwahrscheinlichkeit von exakt 1 zuordnet und den restlichen Kreditanträgen exakt 0. Wenden Sie die Bewertungsfunktion an. Vergleichen und kommentieren Sie anschließend das Ergebnis mit dem aus Aufgabenteil c).

e) "Datenstrukturen und Funktionen für Modellvergleiche anlegen": Erstellen Sie Datenstrukturen, in die Sie bei den folgenden Aufgaben die AUC-Werte, Prognosewerte und Modellnamen abspeichern und für Vergleiche verwenden können. Erstellen Sie Funktionen, mit denen Sie diese Angaben für einen Modellvergleich grafisch darstellen können.

Lösungsvorschlag:

a) Evaluationsmetriken

Im vorliegenden Fall ist der Datensatz bezüglich der Zielgröße „TARGET“ mit einer Kreditausfallquote von ca. 8% stark unbalanciert. Bereits die unzutreffende Behauptung, es gäbe keine Kreditausfälle, hätte eine Genauigkeit von 92%, siehe auch Null-Modell 1 im Aufgabenteil c). Der F1-Score gibt als harmonisches Mittel aus Genauigkeit und Trefferquote ein ausgewogenes Maß auch für unbalancierte Daten, allerdings muss eine Wahrscheinlichkeitsgrenze zwischen den Klassen festgelegt werden. Letzteres ist bei AUC nicht der Fall, die von den Modellen vorhergesagte Wahrscheinlichkeit kann direkt verwendet werden. Diese einfache, quasi „gedankenlose“ Anwendbarkeit der AUC-Metrik könnte ein Grund für die Beliebtheit sein. Im vorliegenden Fall können damit auch verschiedene „subsampling“-Stichproben einfach verglichen werden, siehe Aufgabe B-5. Bei einer konkreten Anwendung der prognostizierten Wahrscheinlichkeiten müssen die (sehr) unterschiedlichen finanziellen Wirkungen der „falsch positiven“ und „falsch negativen“ Ereignisse (Kreditausfall vs. Kreditablehnung) in Form einer Nutzen-Maximierung berücksichtigt werden.

b) Bewertungsfunktion schreiben

Die Funktion 'assess' gibt in kompakter Form die beschreibende Statistik des Prognosewerts sowie Accuracy, Confusion Matrix und AUC aus.
Eingabeparameter: Beobachtete Werte, Prognostizierte Wahrscheinlichkeit, Schwellenwert

In [13]:
```python
def assess(y_obs, y_hat, threshold):
    """
    Die Funktion 'assess' gibt in kompakter Form die beschreibende Statistik des Prognosewerts
    sowie Accuracy, Confusion Matrix und AUC aus.
    Eingabeparameter: Beobachtete Werte, Prognostizierte Wahrscheinlichkeit, Schwellenwert
    """
    global result

    print("\nSummary statistics of predicted 'y':\n",
          pd.DataFrame(y_hat).describe(percentiles = []).T)

    # Die Vorhersagen in binäre Klassen umwandeln (positiv/negativ)
    y_hat_binary = np.where(y_hat > threshold, 1, 0)

    # Genauigkeit berechnen und ausgeben
    print("\n Accuracy:\n", round(accuracy_score(y_obs, y_hat_binary),4))

    # Confusion Matrix ausgeben
    cm = confusion_matrix(y_obs, y_hat_binary)
    print("\n Confusion Matrix:\n Pred. 0    1\n", cm)

    # AUC an Validierungsdaten berechnen
    result = roc_auc_score(y_obs, y_hat)
    print("\nAUC: " + "%6.4f" % result)
    print("===========")

  # return result
```

In [14]:
```python
# Schwellenwert in Höhe der Ereignisrate (Anteil Kreditausfälle) des Trainingsdatensatz festlegen:
cm_threshold = len(y_train[y_train == 1])/len(y_train)
print("Schwellenwert für Binärklassifikation (cm_threshold): ", round(cm_threshold,6))
```

```
Schwellenwert für Binärklassifikation (cm_threshold):  0.079706
```

c) Null-Modell 1, Kreditausfälle nicht zu erwarten

In [15]:
```python
# Null-Modell 1: numpy array mit dem wert 0
probs_n1 = np.zeros(len(y_val))

# Bewertungsfunktion anwenden:
assess(y_val, probs_n1, cm_threshold)
```

```
Summary statistics of predicted 'y':
      count  mean  std  min  50%  max
0  40110.0   0.0  0.0  0.0  0.0  0.0

 Accuracy:
 0.9227

 Confusion Matrix:
 Pred. 0    1
 [[37008     0]
 [ 3102     0]]

AUC: 0.5000
===========
```

d) Null-Modell 2, Kreditausfälle sind rein zufällig

In [16]:
```
# Null-Modell 2: numpy array mit Wert 1 in einem Teil der Fälle (cm_threshold), sonst 0. Zufallszuordnung.
np.random.seed(1)
probs_n2 = np.random.choice([0, 1], size=(len(y_val),), p=[1-cm_threshold, cm_threshold])

# Bewertungsfunktion anwenden:
assess(y_val, probs_n2, cm_threshold)
```

```
Summary statistics of predicted 'y':
      count      mean       std  min  50%  max
0  40110.0  0.079257  0.270143  0.0  0.0  1.0

 Accuracy:
 0.8556

 Confusion Matrix:
 Pred. 0    1
 [[34074  2934]
 [ 2857   245]]

AUC: 0.4999
===========
```

Fazit: Das Null-Modell 1 erzielt eine sehr hohe Accuracy und vergleichsweise wenige falsch positive und falsch negative Vorhersagen. Beide Null-Modelle haben jedoch eine sehr geringe AUC von rund 0.5 und damit die Vorhersagekraft einer Zufallszuordnung (was im Null-Modell 2 tatsächlich der Fall ist).

e) Datenstrukturen und Funktionen für Modellvergleiche anlegen

In [17]:
```
# Liste für Modellnamen
mname = []

# Liste für AUC der Modelle
mauc = []

# Dictionary für AUC und Modellnamen anlegen
dict = {'AUC': mauc, 'Modellname': mname}

# Liste für Prongosewerte
mprobs = []
```

In [18]:
```
def plot_auc_vergleich(d,x1,x2,t):
    """
    Die Funktion 'plot_auc_vergleich' stellt die Prognosegüte der gefitteten Modelle
    grafisch dar.
    Eingabeparameter: Dictionary mit Modellnamen und AUC, Text
    """
    df_eval = pd.DataFrame(d)
    sns.set_style('darkgrid')
    plt.title(" Vergleich der Modellgüte (AUC): " + t)
    sns.barplot(data = df_eval, x = "AUC", y = "Modellname", color = 'magenta')
    plt.xlim(x1, x2)
    plt.show()
```

In [19]:
```
def plot_prob_dist(n,v,ln):
    """
    Die Funktion 'plot_prob_dist' stellt die Prognosewahrscheinlichkeiten der
    gefitteten Modelle grafisch dar
    Eingabeparameter: Liste mit Modellnamen und Prognosewahrscheinlichkeiten
    """
    df = pd.DataFrame(np.transpose(v), columns=n)
    df_melted = pd.melt(df,value_vars=n, var_name="Modell", value_name="Prognosewert")
    df_selected = df_melted[df_melted["Modell"].isin(ln)]
    plt.title('Prognostizierte Wahrscheinlichkeiten (samt Kerndichteschätzer)')
    sns.histplot(data = df_selected, x = "Prognosewert", hue = "Modell",  kde = True)
    plt.show()
```

Aufgabe B-3: Baumbasierte Basismodelle (ohne HP-Tuning) [Lernziele 2.2, 4.1, 6.1.2; 14 Punkte]

---

a) "Kleiner Entscheidungsbaum": Wählen Sie ein geeignetes Entscheidungsbaumverfahren aus. Begrenzen Sie die Baumtiefe auf 3 (d.h. maximal acht Endknoten). Zeigen Sie alle verwendeten (Standard-)Parameter an und "fitten" Sie das Prognosemodell auf den Trainingsdaten. Erstellen Sie anschließend Prognosewahrscheinlichkeiten für die Validierungsdaten und wenden Sie die Bewertungsfunktion an. Zeigen Sie die resultierenden Regeln textlich oder grafisch mit Verwendung der Merkmalsnamen an. Wie stark differenziert der Entscheidungsbaum hinsichtlich der Kreditausfallrate? Sind die Entscheidungen mit Blick auf die in Block A durchgeführte explorative Datenanalyse nachvollziehbar? Durch welche Maßnahmen kann die Prognosekraft eines Entscheidungsbaums verbessert werden?

b) "Entscheidungsbaumbasierte Ensembles und Parallelisierung": Beschreiben und vergleichen Sie, wie die entscheidungsbaumbasierten Klassifikationsverfahren "Random Forest" und "Gradient Tree Boosting" funktionieren und wie sie parallelisiert werden können.

c) "Basismodell Random Forest (mit Standardeinstellungen)": Wählen Sie eine geeignete und parallelisierte Implementation des Random-Forest-Verfahrens aus. Zeigen Sie alle verwendeten (Standard-)Parameter an und "fitten" Sie mit Standardparametern ein Prognosemodell. Beobachten Sie dabei die CPU-Auslastung. Falls diese nicht voll ausgelastet ist: Setzen Sie den Parallelisierungsparameter manuell auf "alle Kerne" und wiederholen Sie die Anpassung (ggf. den ganzen Aufgabenteil). Erstellen Sie anschließend Prognosewahrscheinlichkeiten für die Validierungsdaten und wenden Sie die Bewertungsfunktion an.

d) "Basismodell Gradient Tree Boosting (mit Standardeinstellungen)": Wählen Sie eine geeignete (parallelisierte) CatBoost-Implementierung aus und führen Sie die in Aufgabenteil c) beschriebenen weiteren Schritte durch.

e) "Ergebnisvergleich":

- **Wie viele Entscheidungsbäume bzw. Iterationen wurden bei den einzelnen Modellen berechnet, welche Auswirkungen hatte das auf die Laufzeiten und welches Modell erzielt den höchsten AUC-Wert? Sind die Laufzeit- und Güteunterschiede als wesentlich anzusehen?**
- **Zeigen Sie die ROC-Curve des (bzgl. AUC) besten Modells grafisch an und erläutern Sie den Zusammenhang mit dem AUC-Wert. Fügen Sie die ROC-Kurven der Null-Modelle aus Aufgabe B-2 hinzu und erläutern Sie die Grafik.**
- **Erstellen und erläutern Sie eine Grafik, welche die Verteilung der prognostizierten Wahrscheinlichkeiten aus b) und d) samt Kerndichteschätzer anzeigt.**

Lösungsvorschlag:

a) Kleiner Entscheidungsbaum

In [20]:
```
##### DecisionTreeClassifier mit Default-Werten anwenden
tic = time.time()

DT = DecisionTreeClassifier(max_depth=3)
DT.fit(X_train, y_train)

print("time (sec):" + "%6.0f" % (time.time() - tic))

set_config(print_changed_only=False)
print("\nStandardparameter: ", DT)
```

```
time (sec):     3

Standardparameter:  DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       random_state=None, splitter='best')
```

In [21]:
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = DT.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], cm_threshold)

# Die Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("DecisionTree")
mprobs.append(probs[:,1])
```

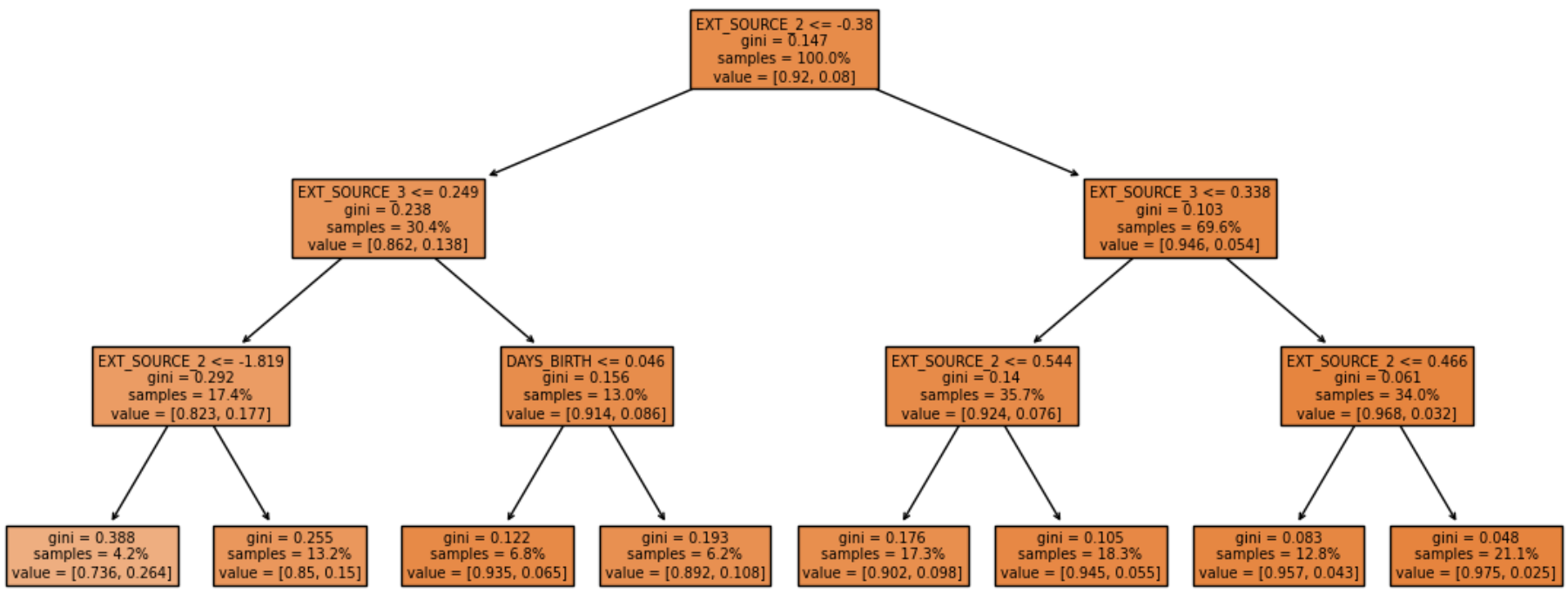
```
Summary statistics of predicted 'y':
      count      mean       std       min       50%       max
0  40110.0  0.079526  0.055372  0.024541  0.055422  0.263505

 Accuracy:
 0.6175

 Confusion Matrix:
 Pred. 0    1
 [[22728 14280]
 [ 1061  2041]]

AUC: 0.6784
===========
```

Resultierendes Regelwerk textlich mit Klarnamen und zuätzlich grafisch (wg. Kreditausfallquoten) anzeigen:

In [22]:
```
from sklearn import tree
text_representation = tree.export_text(DT,feature_names=feature_names,max_depth=3)
print(text_representation)
```

```
|--- EXT_SOURCE_2 <= -0.38
|   |--- EXT_SOURCE_3 <= 0.25
|   |   |--- EXT_SOURCE_2 <= -1.82
|   |   |   |--- class: 0
|   |   |--- EXT_SOURCE_2 >  -1.82
|   |   |   |--- class: 0
|   |--- EXT_SOURCE_3 >  0.25
|   |   |--- DAYS_BIRTH <= 0.05
|   |   |   |--- class: 0
|   |   |--- DAYS_BIRTH >  0.05
|   |   |   |--- class: 0
|--- EXT_SOURCE_2 >  -0.38
|   |--- EXT_SOURCE_3 <= 0.34
|   |   |--- EXT_SOURCE_2 <= 0.54
|   |   |   |--- class: 0
|   |   |--- EXT_SOURCE_2 >  0.54
|   |   |   |--- class: 0
|   |--- EXT_SOURCE_3 >  0.34
|   |   |--- EXT_SOURCE_2 <= 0.47
|   |   |   |--- class: 0
|   |   |--- EXT_SOURCE_2 >  0.47
|   |   |   |--- class: 0
```

In [23]:
```
fig, axes = plt.subplots(figsize=(14,6))
tree.plot_tree(DT, feature_names=X_train.columns, filled = True, proportion=True)
plt.show()
```



Der Entscheidungsbaum differenziert hinsichtlich der Kreditausfallrate zwischen 2,5% (21,1% der Anträge, siehe rechtes Blatt) und 26,4% (4,2% der Anträge). Die Entscheidungen sehen mit Blick auf bereits im Aufgabenteil A erkennbaren Bedeutung der Features EXT_SOURCE* plausibel aus. Die Prognosekraft eines Entscheidungsbaumes kann durch Parameteroptimierung (hier z.B. eine größere Tiefe zulassen) und durch Anwendung des "Pruning"-Verfahrens ("Zurückschneiden des Baumes") verbessert werden.

b) Entscheidungsbaumbasierte Ensembles und Parallelisierung

Funktionsweise: Die Klassifikationsverfahren "Random Forest" (RF) und "Gradient Tree Boosting" (GB) berechnen und berücksichtigen eine große Anzahl Entscheidungsbäume. Beim RF werden aus dem Trainingsdaten mittels des Bootstrap-Verfahrens eine große Anzahl neuer Stichproben erstellt, darauf jeweils ein Entscheidungsbaum berechnet und das Ergebnis gemittelt. Dadurch, dass in jede Split-Entscheidung nur ein kleiner, zufällig ausgewählter Teil der Features einbezogen wird, "wachsen" die Bäume recht unterschiedlich. Diese Diversität führt zu einer Verringerung der Varianz. Beim GB-Verfahren wird hingegen zunächst ein i.d.R. recht kleiner Baum berechnet und mit einer Vielzahl von Iterationen versucht, sukzessive die Residuen zu verkleinern. Das Ergebnis wird ebenfalls gemittelt, wobei ein neues Interationensergebnis mit einem geringen Gewicht in Höhe der sogenannten "Lernrate" einfließt und das Verfahren daher langsam lernt und gewissenhaft "getunt" werden muss.

Parallelisierung: Während die Entscheidungsbäume beim RF unabhängig voneinander sind und vergleichsweise einfach auf verschiedene Prozessorkerne verteilt werden können, ist das bei den aufeinander aufbauenden Bäumen des GB schwieriger. Hier werden beispielsweise die für die Split-Entscheidungen benötigten Sortierungen und Berechnungen eines Baumes parallelisiert sowie einige weitere "Kniffe" angewendet, siehe Wen et al. (2021), "Challenges and Opportunities of Building Fast GBDT Systems", https://www.ijcai.org/proceedings/2021/632.

c) Basismodell Random Forest (mit Standardeinstellungen): RandomForestClassifier

In [24]:
```
# RandomForestClassifier mit Default-Werten anwenden
tic = time.time()

# Modellanpassung mit Default Parametern
RF = RandomForestClassifier()
```

```
cpu_auslastung = psutil.cpu_percent() # "Nullsetzung"
RF.fit(X_train, y_train)
cpu_auslastung = psutil.cpu_percent() # Auslastung in der Zeit zum letzten Funktionsaufruf
print(f"Die CPU-Auslastung beträgt: {cpu_auslastung}%")

print("Laufzeit (sec):" + "%6.0f" % (time.time() - tic))

set_config(print_changed_only=False)
print("\nStandardparameter: ", RF)
```

```
Die CPU-Auslastung beträgt: 53.0%
Laufzeit (sec):   113

Standardparameter:  RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=100, n_jobs=None, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)
```

Beobachtung: CPU ist nicht ausgelastet.

Wiederholung: Ausführung auf allen zur Verfügung stehenden Rechenkernen (n_jobs=-1)

In [25]:
```
# RandomForestClassifier parallelisiert und ansonsten mit Default-Werten anwenden
tic = time.time()

# Modellanpassung mit Default Parametern
RF = RandomForestClassifier(n_jobs=-1)
cpu_auslastung = psutil.cpu_percent() # "Nullsetzung"
RF.fit(X_train, y_train)
cpu_auslastung = psutil.cpu_percent() # Auslastung in der Zeit zum letzten Funktionsaufruf
print(f"Die CPU-Auslastung beträgt: {cpu_auslastung}%")

print("time (sec):" + "%6.0f" % (time.time() - tic))

set_config(print_changed_only=False)
print("\nStandardparameter: ", RF)
```

```
Die CPU-Auslastung beträgt: 99.7%
time (sec):    77

Standardparameter:  RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=100, n_jobs=-1, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)
```

In [26]:
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = RF.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], cm_threshold)

# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("RandomForest")
mprobs.append(probs[:,1])
```

```
Summary statistics of predicted 'y':
      count      mean       std  min   50%   max
0  40110.0  0.092913  0.073968  0.0  0.07  0.58

 Accuracy:
 0.5417

 Confusion Matrix:
 Pred. 0    1
 [[19400 17608]
 [  773  2329]]

AUC: 0.7061
===========
```

d) Basismodell Gradient Boosting: CatBoostClassifier mit Standardeinstellungen

In [27]:
```
# CatBoostClassifier: Zwecks Unterdrückung des 1000-zeiligen Logs von Default-Parametern abgewichen
tic = time.time()
CBC = CatBoostClassifier(logging_level='Silent')
CBC.fit(X_train, y_train)
print("time (sec):" + "%6.0f" % (time.time() - tic))

print("\nStandardparameter: ", CBC.get_all_params())
```

```
time (sec):    97

Standardparameter:  {'nan_mode': 'Min', 'eval_metric': 'Logloss', 'iterations': 1000, 'sampling_frequency': 'PerTree', 'leaf_est
imation_method': 'Newton', 'grow_policy': 'SymmetricTree', 'penalties_coefficient': 1, 'boosting_type': 'Plain', 'model_shrink_m
ode': 'Constant', 'feature_border_type': 'GreedyLogSum', 'bayesian_matrix_reg': 0.10000000149011612, 'eval_fraction': 0, 'force_
unit_auto_pair_weights': False, 'l2_leaf_reg': 3, 'random_strength': 1, 'rsm': 1, 'boost_from_average': False, 'model_size_reg':
0.5, 'pool_metainfo_options': {'tags': {}}, 'subsample': 0.800000011920929, 'use_best_model': False, 'class_names': [0, 1], 'ran
dom_seed': 0, 'depth': 6, 'posterior_sampling': False, 'border_count': 254, 'classes_count': 0, 'auto_class_weights': 'None', 's
parse_features_conflict_fraction': 0, 'leaf_estimation_backtracking': 'AnyImprovement', 'best_model_min_trees': 1, 'model_shrink
_rate': 0, 'min_data_in_leaf': 1, 'loss_function': 'Logloss', 'learning_rate': 0.07966200262308121, 'score_function': 'Cosine',
'task_type': 'CPU', 'leaf_estimation_iterations': 10, 'bootstrap_type': 'MVS', 'max_leaves': 64}
```

In [28]:
```python
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = CBC.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], cm_threshold)

# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("CatBoost")
mprobs.append(probs[:,1])
```

```
Summary statistics of predicted 'y':
      count      mean       std       min       50%       max
0  40110.0  0.077658  0.087956  0.000321  0.046939  0.891666

 Accuracy:
 0.7163

 Confusion Matrix:
 Pred. 0    1
 [[26645 10363]
 [ 1015  2087]]

AUC: 0.7620
===========
```

In [29]:
```python
# Die Vorhersagen in binäre Klassen umwandeln (positiv/negativ) zum Vergleich: Mit 50%-Threshold
y_pred_binary = np.where(probs[:,1] > 0.5, 1, 0)

# Genauigkeit berechnen und ausgeben
print("\n Accuracy:", round(accuracy_score(y_val, y_pred_binary),4))

print("\nConfusion Matrix:");
cm = confusion_matrix(y_val, y_pred_binary)
print(cm)
```

```
 Accuracy: 0.9229

Confusion Matrix:
[[36898   110]
 [ 2984   118]]
```

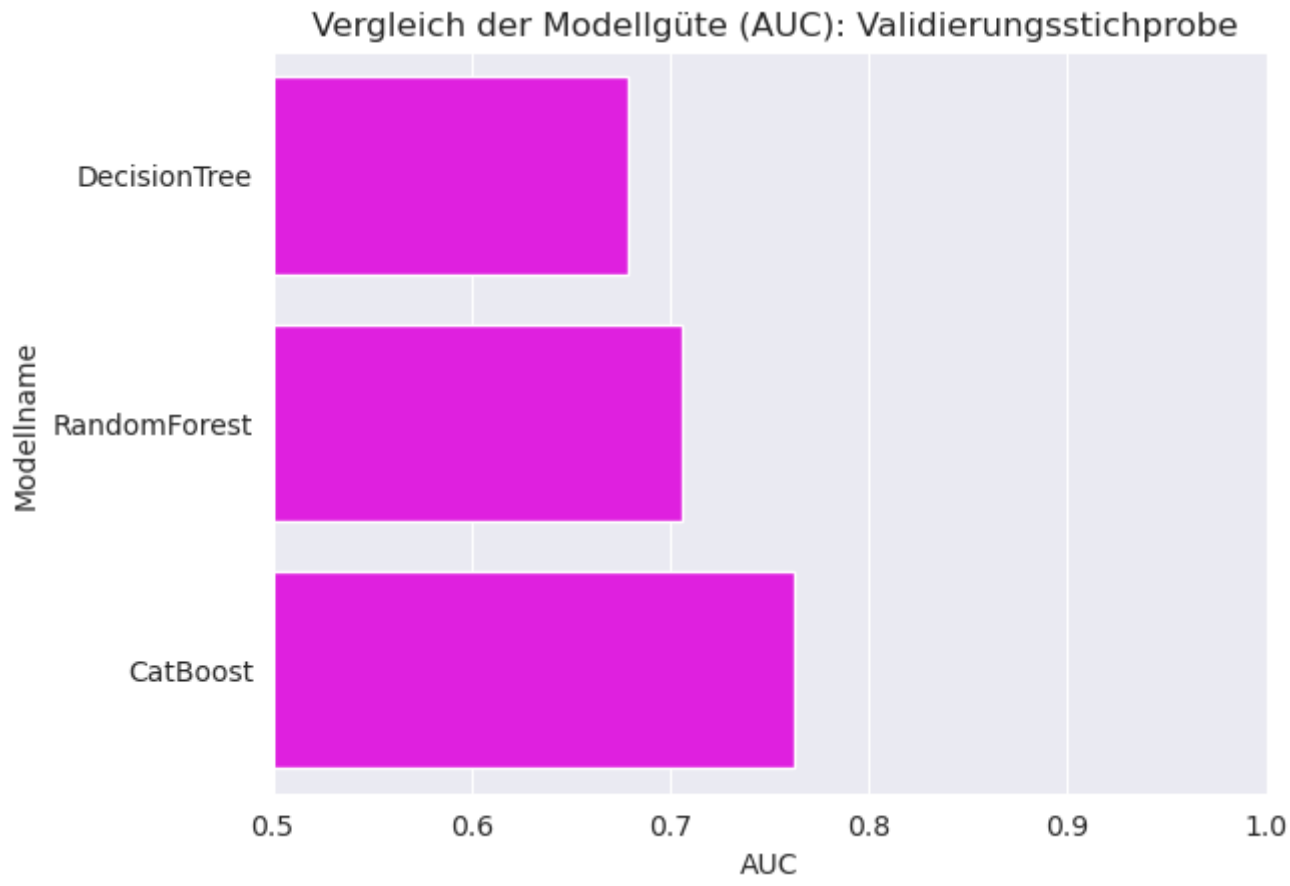
Zum Vergleich: Bei einem Schwellenwert von 50% gibt es sehr wenige korrekt vorhergesagte Ereignisse, aber auch sehr wenige falsch positive Ereignisse. Dadurch ist die Modellgenauigkeit (Accuracy) wie beim Null-Modell 1 sehr hoch. Auf den AUC-Wert hat der Schwellenwert keinen Einfluss.

e) Ergebnisvergleich

Bewertung:

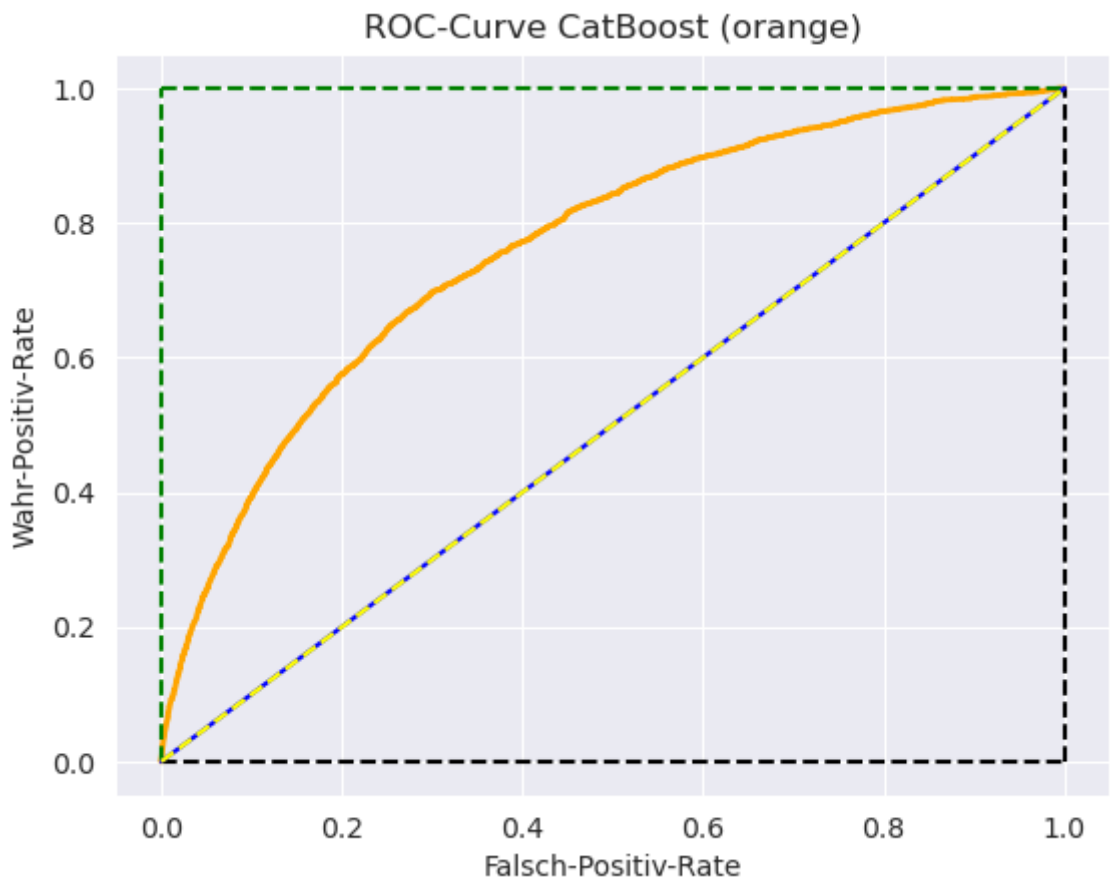
Das CatBoost-Modell benötigte mit 1000 Interationen eine **unwesentlich** längere Laufzeit als der RandomForestClassifier mit 100 Bäumen und der DecisionTreeClassifier (Ein Baum), erzielte aber eine **wesentlich** bessere Prognosegüte (deutlich höhere AUC) als die beiden anderen Modelle:

In [30]:
```python
plot_auc_vergleich(dict,0.5,1.0, "Validierungsstichprobe")
```



ROC-Kurve des besten Modells:

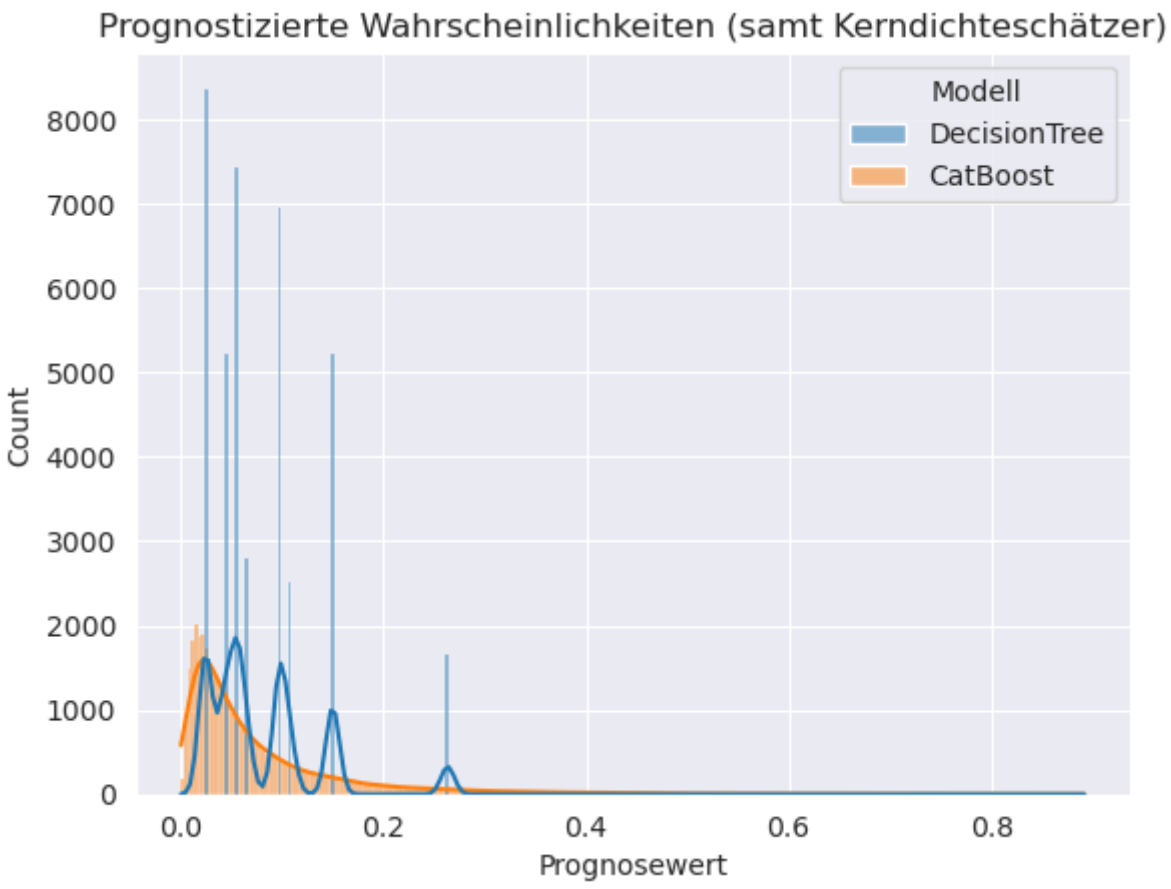
In [31]:
```
# Plot der ROC-Kurve für das beste Modell:
fpr, tpr, _ = roc_curve(y_val, probs[:,1])
plt.title('ROC-Curve CatBoost (orange)')
plt.plot(fpr,tpr, color='orange', linewidth=2)
plt.xlabel("Falsch-Positiv-Rate")
plt.ylabel("Wahr-Positiv-Rate")
plt.plot([1,1], [0,1], color='black', linestyle='--')
plt.plot([0,1], [0,0], color='black', linestyle='--')
plt.plot([0,0], [0,1], color='green', linestyle='--')
plt.plot([0,1], [1,1], color='green', linestyle='--')
# Null-Modell 1 hinzufügen
fpr1, tpr1, _ = roc_curve(y_val, probs_n1)
plt.plot(fpr1,tpr1, color='blue', linestyle='-');
# Null-Modell 2 hinzufügen
fpr2, tpr2, _ = roc_curve(y_val, probs_n2)
plt.plot(fpr2,tpr2, color='yellow', linestyle='--');
```



Die AUC (Area Under Curve) für CatBoost ist die Fläche zwischen der orangen Linie und den schwarz gestrichelten Linien. Im besten Falle, wenn es keine falsch positiv vorhergesagten Ereignisse gäbe, verliefe die ROC-Kurve wie grün gestrichelt eingezeichnet und die AUC wäre gleich 1 . Die ROC-Kurven der Null-Modelle (blau und gelb gestrichelt) bilden die Diagonale der Grafik und die Fläche unter der Kurve (AUC) beträgt 0,5. Ein AUC-Wert von unter 0,5 bedeutet, dass die vom Modell vorhergesagten Werte schlechter als ein Zufallswert sind.

Wahrscheinlichkeitsverteilungen (Entscheidungsbaum vs. CatBoost):

In [32]:
```
# Plots der  prognostizierten Wahrscheinlichkeiten samt Kerndichteschätzer für den DecisionTreeClassifier und CatBoost
liste = ["DecisionTree","CatBoost"]
plot_prob_dist(mname, mprobs, liste)
```



Während die Prognosewerte des Gradient-Boosting-Tools "CatBoost" eine recht gleichmäßige, rechtsschiefe Verteilung aufweisen, kann man bei den Prognosewerten des Entscheidungsbaumes "DecisionTree" die Einzelwerte der acht Endknoten erkennen.

Aufgabe B-4: Gradient Boosting (HP-Tuning) [Lernziele 4.1, 6.1.2; 14 Punkte]

---

a) "Gradient Boosting und HP-Tuning": Worin unterscheiden sich die für ihre sehr gute Performance bekannten Gradient-Boosting-Tools CatBoost, lightGBM und XGBoost und wie können diese verwendet werden? Wie funktioniert Kreuzvalidierung und was ist der Unterschied zwischen einer "Grid Search" und einer "Randomized Search"?

b) "lightGBM": Führen Sie eine Hyperparameteroptimierung über eine mehrdimensionale "Randomized Search" mit vierfacher Kreuzvalidierung für lightGBM parallelisiert durch und optimieren Sie dabei die Parameter 'learning_rate', 'num_leaves', 'subsample' und 'colsample_bytree'. Verwenden Sie die gleiche Anzahl an Iterationen wie CatBoost in Aufgabe B-3. "Fitten" Sie anschließend das Modell mit den besten Parametern auf allen Folds. Erstellen Sie anschließend Prognosewahrscheinlichkeiten für die Validierungsdaten und wenden Sie die Bewertungsfunktion an.

c) "XGBoost": Führen Sie die in Aufgabenteil b) beschriebenen Schritte für XGBoost (anstelle von lightGBM) aus, wobei der Parameter 'num_leaves' durch 'max_depth' zu ersetzen ist.

d) "Vergleichen und bewerten": Vergleichen Sie die Laufzeiten und Prognoseergebnisse der beiden Verfahren miteinander und mit denen aus Aufgabe B-3 und interpretieren Sie das Ergebnis.

Lösungsvorschlag:

a) Gradient Boosting und HP-Tuning

Die drei hier zu betrachtenden Gradient-Boosting-Tools CatBoost, lightGBM und XGBoost sind aufgrund der guten Parallelisierung viel schneller als frühere Implementierungen und unterscheiden sich (u.a.) in ihrer Vorgehensweise beim schrittweisen Aufbau der Bäume, die CatBoost stufeneinheitlich, XGBoost stufenweise und lightGBM blattweise aufbaut. LightGBM verwendet standardgemäß einen sehr effizienten histogrammbasierten Entscheidungsprozess, der auch bei XGBoost aktiviert werden kann und sollte. CatBoost ist besonders für die effiziente Verarbeitung kategorieller Merkmale geeignet und liefert oft bereits mit Standardeinstellungen schnelle und sehr gute Ergebnisse.

Bei der Kreuzvaldierung werden die Daten zufällig in mehrere "Folds" eingeteilt und reihum eine dieser Folds für die Validierung und die restlichen Folds für das Modelltraining verwendet. Dadurch werden mehrere Modelle gefittet, mit unterschiedlichen Daten validiert, die Prognosen gemittelt und das Ergebnis repräsentativer.

Bei der "Grid Search" werden für i.d.R. mehrere Hyperparameter verschiedene Werte vorgeben und alle Parameterkombinationen gefittet, während das bei der "Randomized Search" nur für eine vorgegebene Anzahl zufällig ausgewählter Kombinationen durchgeführt wird. Die Randomized Search eignet sich daher besonders für höherdimensionale Suchräume.

b) lightGBM

In [33]:
```
# Parametersuchbereich für lightGBM
param_grid_LGB ={'learning_rate': loguniform(0.006, 0.06),
                 'num_leaves':[8,12,16,20,25,32,40,50,64],
                 'subsample': uniform(0,1),
                 'colsample_bytree': uniform(0,1) }
```

```
# Variablenliste für die Anzeige des CV-Ergebnisses anlegen
sel_params_LGB = ['param_learning_rate','param_num_leaves','param_subsample',
                  'param_colsample_bytree','mean_test_score','rank_test_score']
```

In [34]:
```
# LGBMClassifier: Hyperparameteroptimierung mit RandomizedSearchCV
tic = time.time()

LGB_rs = RandomizedSearchCV(LGBMClassifier(device='gpu', n_estimators=1000),
                            param_grid_LGB, cv=4, n_iter=20, scoring='roc_auc',
                            n_jobs=-1, random_state=seed)
LGB_rs.fit(X_train,y_train)

# Laufzeit und beste Parameter (sowie die Nächstbesten) ausgeben
print("time (sec):" + "%6.0f" % (time.time() - tic))
print("Beste Parameter:",LGB_rs.best_params_)
pd.DataFrame(LGB_rs.cv_results_)[sel_params_LGB].sort_values("rank_test_score").head()
```

```
time (sec):  1481
Beste Parameter: {'colsample_bytree': 0.15601864044243652, 'learning_rate': 0.008593018973466909, 'num_leaves': 50, 'subsample':
0.33370861113902184}
```

Out[34]:

	param_learning_rate	param_num_leaves	param_subsample	param_colsample_bytree	mean_test_score	rank_test_score
1	0.008593	50	0.333709	0.156019	0.771819	1
2	0.026856	25	0.96991	0.142867	0.771688	2
17	0.014895	50	0.005522	0.423401	0.771645	3
8	0.014574	25	0.230894	0.563288	0.771105	4
11	0.009685	20	0.184854	0.425156	0.770636	5

In [35]:
```
# LGBMClassifier: Erstelle neues Modell auf allen Folds mit den besten Parametern
tic = time.time()
LGB2 = LGBMClassifier(**LGB_rs.best_params_, device='gpu', n_estimators=1000)
LGB2.fit(X_train, y_train)
print("time (sec):" + "%6.0f" % (time.time() - tic))
```

```
time (sec):    19
```

In [36]:
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = LGB2.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], cm_threshold)

# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("lightGBM")
mprobs.append(probs[:,1])
```

```
Summary statistics of predicted 'y':
      count      mean       std       min       50%       max
0  40110.0  0.078826  0.073555  0.004595  0.054329  0.713276

 Accuracy:
 0.6995

 Confusion Matrix:
 Pred. 0    1
 [[25899 11109]
 [  945  2157]]
 AUC: 0.7625
 ===========
```

c) XGBoost

In [37]:
```
# Parametersuchbereich für XGBoost
param_grid_XGB ={'learning_rate': loguniform(0.006, 0.06),
                 'max_depth':[3,4,5,6,7,8,10],
                 'subsample': uniform(0,1),
                 'colsample_bytree': uniform(0,1) }

# Variablenliste für die Anzeige des CV-Ergebnisses anlegen
sel_params_XGB = ['param_learning_rate','param_max_depth','param_subsample',
                  'param_colsample_bytree','mean_test_score','rank_test_score']
```

In [38]:
```
# XGBClassifier: Hyperparameteroptimierung mit RandomizedSearchCV
tic = time.time()

XGB_rs = RandomizedSearchCV(XGBClassifier(n_estimators=1000, tree_method='gpu_hist'),
                            param_grid_XGB, cv=4, n_iter=20, scoring='roc_auc',
                            n_jobs=-1, random_state=seed)
XGB_rs.fit(X_train,y_train)

# Laufzeit und beste Parameter (sowie die Nächstbesten) ausgeben
print("time (sec):" + "%6.0f" % (time.time() - tic))
print("Beste Parameter:",XGB_rs.best_params_)
pd.DataFrame(XGB_rs.cv_results_)[sel_params_XGB].sort_values("rank_test_score").head()
```

```
time (sec):   635
Beste Parameter: {'colsample_bytree': 0.9922115592912175, 'learning_rate': 0.024867536268078135, 'max_depth': 4, 'subsample': 0.
5247564316322378}
```

Out[38]:

	param_learning_rate	param_max_depth	param_subsample	param_colsample_bytree	mean_test_score	rank_test_score
4	0.024868	4	0.524756	0.992212	0.773056	1
13	0.019871	4	0.5677	0.311711	0.773006	2
17	0.040447	4	0.965255	0.271349	0.772631	3
18	0.011328	7	0.802197	0.607034	0.772559	4
12	0.014764	4	0.662522	0.173365	0.771757	5

In [39]:
```
# XGBClassifier: Erstelle neues Modell auf allen Folds mit den besten Parametern
tic = time.time()
XGB2 = XGBClassifier(**XGB_rs.best_params_,
                     n_estimators=1000, tree_method='gpu_hist', eval_metric='auc')
XGB2.fit(X_train, y_train)
print("time (sec):" + "%6.0f" % (time.time() - tic))
```

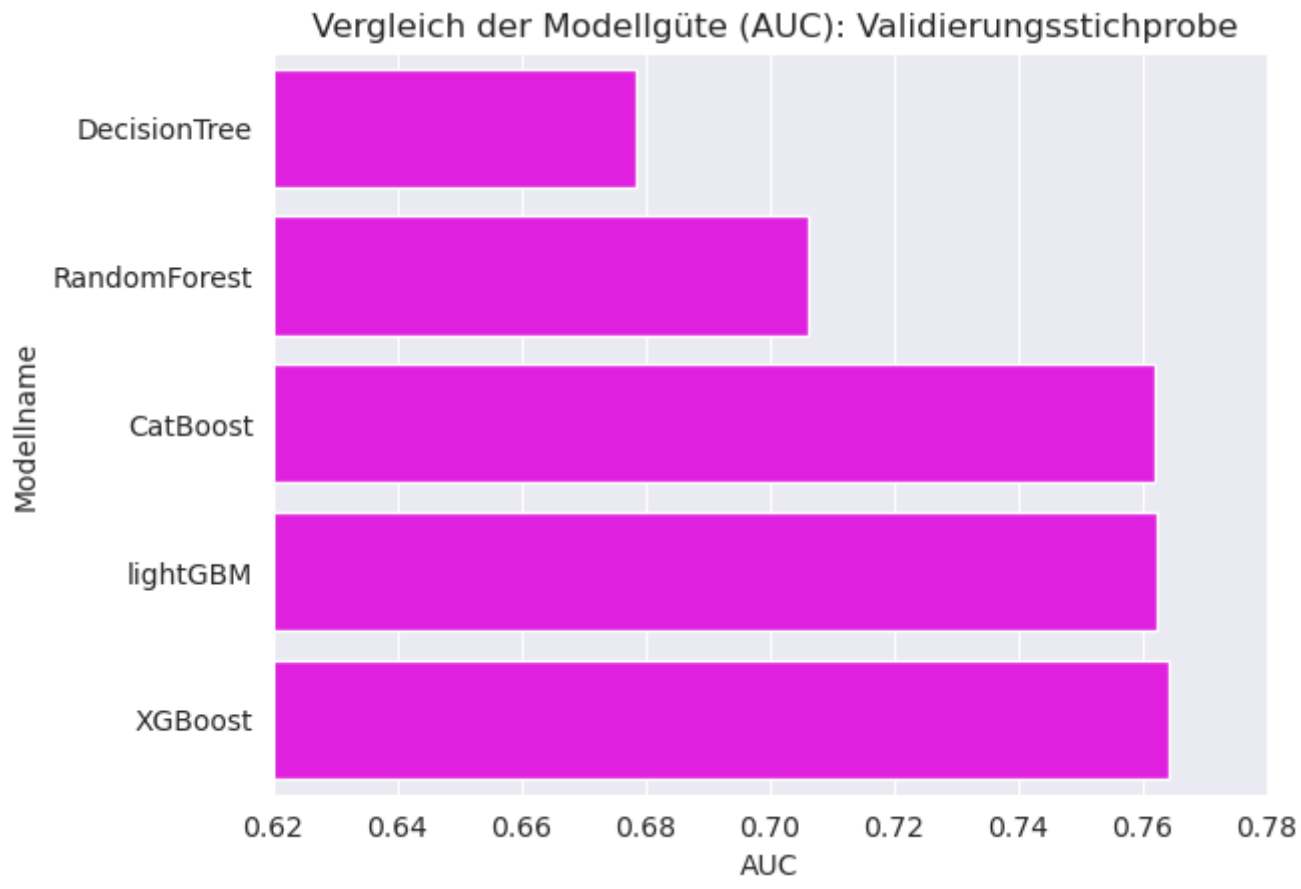
```
time (sec):     5
```

In [40]:
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = XGB2.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], cm_threshold)

# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("XGBoost")
mprobs.append(probs[:,1])
```

```
Summary statistics of predicted 'y':
      count      mean       std       min      50%       max
0  40110.0  0.078981  0.085264  0.000768  0.04908  0.770549

 Accuracy:
 0.7075

 Confusion Matrix:
 Pred. 0    1
 [[26248 10760]
 [  974  2128]]

AUC: 0.7643
===========
```

d) Vergleichen und bewerten

In [41]:
```
plot_auc_vergleich(dict, 0.62, 0.78, "Validierungsstichprobe")
```



Die Prognosegüte der durch HP-Tuning optimierten GB-Modelle lightGBM und XGBoost ist nur unwesentlich besser als die von CatBoost. Die Laufzeiten und der Programmieraufwand hingegen sind wesentlich länger bzw. aufwändigerals bei CatBoost. In diesem Vergleich (GPU-Verwendung) ist lightGBM deutlich am langsamsten und XGBoost knapp am besten. Insgesamt können sich die drei GB-Modelle bei der Modellgüte deutlich von den anderen (nicht optimierten) Modellen aus Aufgabe B-3 absetzen. Innerhalb der entscheidungsbaumbasierten (Ensemble)-Verfahren führt Gradient Boosting zu den genauesten Prognosemodellen und kann wie im Falle von CatBoost (hier ohne GPU-Nutzung und nur auf zwei CPU-Cores) bereits mit Standardeinstellungen sehr gut und vergleichsweise schnell sein.

Aufgabe B-5: Boosting mit balancierten Daten [Lernziele 4.1, 6.1.2; 12 Punkte]

Im vorliegenden Datensatz sind die Kreditausfälle mit einer Häufigkeit von rund 8% ein eher seltenes Ereignis. Der weitaus größte Teil der Daten besteht aus "Nicht-Ereignissen", die bei der Modellierung mit entsprechend hohem Gewicht einfließen und zudem zu erheblichem Rechenaufwand und längeren Laufzeiten führen. In dieser Aufgabe soll ein balancierter Trainingsdatensatz mit gleich vielen Ereignissen und Nicht-Ereignissen erzeugt und verwendet werden. Für die Validierung soll weiterhin die unbalancierte Validierungsdatei X_val verwendet werden.

a) "Balancierte Trainingsdaten erzeugen": Erzeugen Sie aus y_train und X_train neue Trainingsdaten mit gleich vielen Ereignissen und Nicht-Ereignissen. Stellen Sie sicher, dass dabei sowohl aus y_train als auch aus X_train die gleichen Fälle ausgewählt werden. Zeigen Sie die Häufigkeitsverteilung der balancierten Zielgröße TARGET an und prüfen Sie das Ergebnis. Wie stark ist das Datenvolumen zurückgegangen?

b) "lightGBM (balanciert)": Wiederholen Sie die im Aufgabenteil B-4 b) beschriebenen Schritte für den balancierten Trainingsdatensatz.

c) "XGBoost (balanciert)": Wiederholen Sie die im Aufgabenteil B-4 c) beschriebenen Schritte für den balancierten Trainingsdatensatz.

d) "Vergleichen und bewerten (balanciert)": Vergleichen Sie die Laufzeiten und Prognoseergebnisse mit den entsprechenden Werten aus Aufgabe B-4. Erstellen Sie eine Grafik, die die Verteilung der Prognosewahrscheinlichkeiten der vier Modelle aus den Ausgaben B-4 und B-5 zeigt und interpretieren Sie das Ergebnis.

Lösungsvorschlag:

a) Balancierte Trainingsdaten erzeugen

In [42]:
```
# subsampling: Gleich viele Target-Werte 0 wie 1 erzeugen
Xs = pd.concat([X_train.reset_index(drop=True), y_train.reset_index(drop=True)], axis=1)
Xs = pd.concat([ Xs[Xs['TARGET']==1], Xs[Xs['TARGET']==0].sample(
           n=len(Xs[Xs['TARGET']==1]), random_state=seed)],axis=0).sort_index(ascending=True)
Xs_train =Xs.drop(columns=["TARGET"])
ys_train = Xs["TARGET"]

# Anzahl fehlender Werte ermitteln
print("Anzahl fehlende Werte in Xs: ", Xs.isna().sum().sum())

print("\nHäufigkeitsverteilung TARGET:\n",Xs.TARGET.value_counts())
print("\nÄnderung des Datenvolumens in Prozent: ", round((len(ys_train)/len(y_train)-1)*100))
```

```
Anzahl fehlende Werte in Xs:  0

Häufigkeitsverteilung TARGET:
 1    9591
0    9591
Name: TARGET, dtype: int64

Änderung des Datenvolumens in Prozent:  -84
```

b) lightGBM (balanciert)

In [43]:
```
# LGBMClassifier: Hyperparameteroptimierung mit RandomizedSearchCV
tic = time.time()

LGBs_rs = RandomizedSearchCV(LGBMClassifier(device='gpu', n_estimators=1000),
                             param_grid_LGB, cv=4, n_iter=20, scoring='roc_auc',
                             n_jobs=-1, random_state=seed)
LGBs_rs.fit(Xs_train,ys_train)

# Laufzeit und beste Parameter (sowie die Nächstbesten) ausgeben
print("time (sec):" + "%6.0f" % (time.time() - tic))
print("Beste Parameter:",LGBs_rs.best_params_)
pd.DataFrame(LGBs_rs.cv_results_)[sel_params_LGB].sort_values("rank_test_score").head()
```

```
time (sec):   830
Beste Parameter: {'colsample_bytree': 0.15601864044243652, 'learning_rate': 0.008593018973466909, 'num_leaves': 50, 'subsample':
0.33370861113902184}
```

Out[43]:

	param_learning_rate	param_num_leaves	param_subsample	param_colsample_bytree	mean_test_score	rank_test_score
1	0.008593	50	0.333709	0.156019	0.765604	1
11	0.009685	20	0.184854	0.425156	0.765477	2
4	0.006098	64	0.291229	0.611653	0.764393	3
5	0.008273	16	0.382462	0.611853	0.764253	4
2	0.026856	25	0.96991	0.142867	0.764208	5

In [44]:
```
# LGBMClassifier: Erstelle neues Modell auf allen Folds mit den besten Parametern
tic = time.time()
LGB2s = LGBMClassifier(**LGBs_rs.best_params_, device='gpu', n_estimators=1000)
```

```
LGB2s.fit(Xs_train, ys_train)
print("time (sec):" + "%6.0f" % (time.time() - tic))
```

```
time (sec):    11
```

In [45]:
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = LGB2s.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], 0.5)

# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("lightGBM (balanciert)")
mprobs.append(probs[:,1])
```

```
Summary statistics of predicted 'y':
      count      mean       std       min       50%       max
0  40110.0  0.417747  0.193548  0.019784  0.396745  0.932279

 Accuracy:
 0.6972

 Confusion Matrix:
 Pred. 0    1
 [[25830 11178]
 [  966  2136]]

AUC: 0.7581
===========
```

c) XGBoost (balanciert)

In [46]:
```
# XGBClassifier: Hyperparameteroptimierung mit RandomizedSearchCV
tic = time.time()

XGBs_rs = RandomizedSearchCV(XGBClassifier(n_estimators=1000, tree_method='gpu_hist'),
                             param_grid_XGB, cv=4, n_iter=20, scoring='roc_auc',
                             n_jobs=-1, random_state=seed)
XGBs_rs.fit(Xs_train,ys_train)


# Laufzeit und beste Parameter (sowie die Nächstbesten) ausgeben
print("time (sec):" + "%6.0f" % (time.time() - tic))
print("Beste Parameter:",XGBs_rs.best_params_)
pd.DataFrame(XGBs_rs.cv_results_)[sel_params_XGB].sort_values("rank_test_score").head()
```

```
time (sec):   348
Beste Parameter: {'colsample_bytree': 0.31171107608941095, 'learning_rate': 0.019870979328434287, 'max_depth': 4, 'subsample':
0.5677003278199915}
```

Out[46]:

	param_learning_rate	param_max_depth	param_subsample	param_colsample_bytree	mean_test_score	rank_test_score
13	0.019871	4	0.5677	0.311711	0.767473	1
12	0.014764	4	0.662522	0.173365	0.767235	2
11	0.007947	7	0.833195	0.440152	0.766853	3
5	0.011732	5	0.399861	0.431945	0.766663	4
4	0.024868	4	0.524756	0.992212	0.766094	5

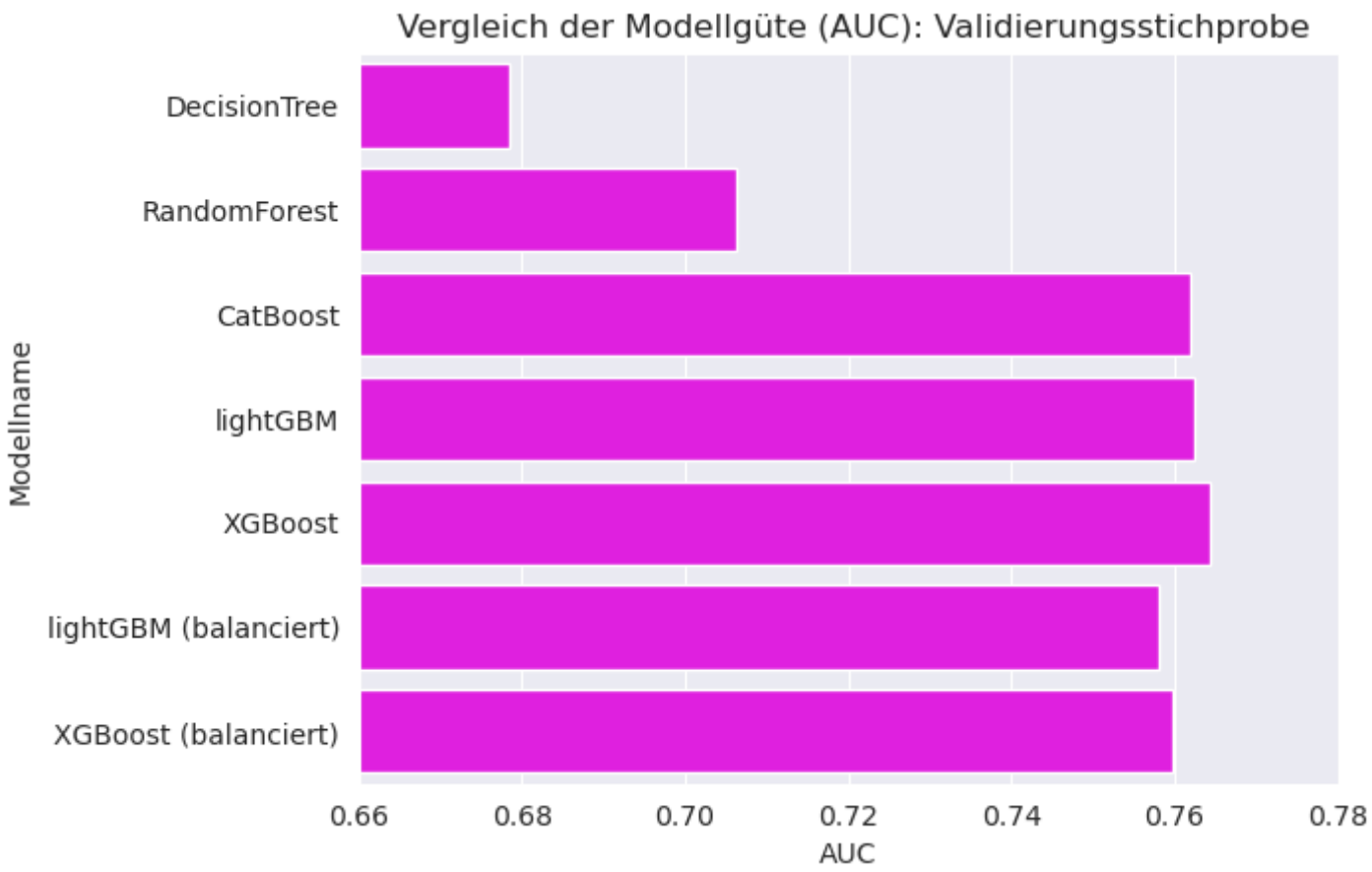
In [47]:
```
# XGBClassifier: Erstelle neues Modell auf allen Folds mit den besten Parametern
tic = time.time()
XGB2s = XGBClassifier(**XGBs_rs.best_params_,
                      n_estimators=1000, tree_method='gpu_hist', eval_metric='auc')
XGB2s.fit(Xs_train, ys_train)
print("time (sec):" + "%6.0f" % (time.time() - tic))
```

```
time (sec):     2
```

In [48]:
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = XGB2s.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], 0.5)

# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("XGBoost (balanciert)")
mprobs.append(probs[:,1])
```

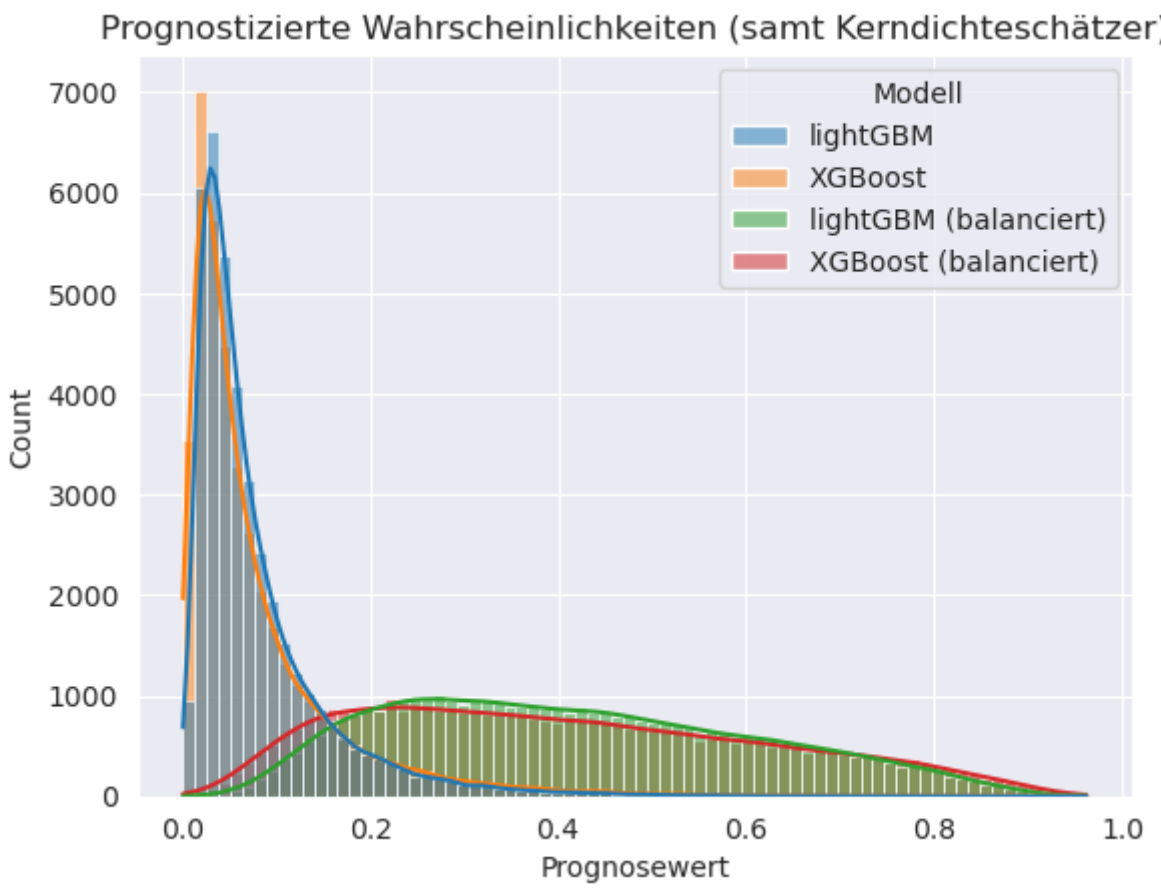
```
Summary statistics of predicted 'y':
      count      mean      std       min       50%       max
0  40110.0  0.407904  0.21355  0.006397  0.381842  0.961945

 Accuracy:
 0.6966

 Confusion Matrix:
 Pred. 0    1
 [[25810 11198]
 [  971  2131]]

AUC: 0.7596
===========
```

In [49]:
```
plot_auc_vergleich(dict, 0.66, 0.78, "Validierungsstichprobe")
```



Vergleich: Auch hier ist XGBoost über doppelt so schnell wie lightGBM (beide auf GPU trainiert). Gegenüber den Modellen aus Aufgabe B-4 (unbalanciert) benötigen die Modelle "LightGBM (balanciert)" und "XGBoost (balanciert)" aufgrund des geringeren Datenvolumens deutlich weniger Laufzeit. Allerdings fällt auch die Prognosegüte jeweils etwas geringer aus, wobei XGBoost seinen kleinen Vorsprung vor lightGBM knapp halten kann. Ein Kompromiss aus Laufzeit und Prognosegüte könnte ein etwas weniger extremes Subsampling darstellen (z.B. auf drei bis fünf Nicht-Ereignisse je Ereignis).

In [50]:
```
# Plots der  prognostizierten Wahrscheinlichkeiten für die vier Modell der Aufgaben B-4 und B-5
liste = [ "lightGBM", "XGBoost", "lightGBM (balanciert)", "XGBoost (balanciert)"]
plot_prob_dist(mname, mprobs, liste)
```



Bewertung der Prognoseverteilung: Wie bereits bei den oben ausgegeben Summenstatistiken der Bewertungsfunktion erkennbar, unterscheiden sich die prognostizierten Wahrscheinlichkeiten zwischen den auf unbalancierten sowie balancierten Trainingdaten gefitteten Modelle ganz erheblich!

Bei der Anwendung von auf Stichproben gefitteten Prognosemodellen und der Nennung erzielbarer Prognosewahrscheinlichkeiten sollte daher die ursprüngliche Verteilung mit berücksichtigt werden, andernfalls könnten zu hohe Erwartungen geweckt werden.

Aufgabe B-6: Logistische Regression und Regularisierung [Lernziele 4.1, 6.1.2; 12 Punkte]

---

a) "Logistische Regression und Regularisierung": Erläutern Sie, weshalb die logistische Regression für eine Binärklassifikation geeignet ist, welche Rolle im vorliegenden Fall eine Regularisierung (Penalisierung) spielt und gehen sie dabei auf die Unterschiede zwischen "Ridge"-, "LASSO"- und "Elastic Net"-Regularisierung ein.

b) "Logistische Regression mit L1-Regularisierung": "Fitten" Sie auf Basis der Trainingsdaten das Verfahren „regularisierte logistische Regression“ mit L1-Regularisierung ("LASSO") und wählen Sie den Regularisierungsparameter so, dass höchstens 30 Features relevant sind (d.h. Koeffizienten <> 0). Erstellen Sie damit Prognosewahrscheinlichkeiten für die Validierungsdaten und wenden Sie die Bewertungsfunktion an.

c) "Logistische Regression mit L1- und L2-Regularisierung": Führen Sie auf Basis der Trainingsdaten das Verfahren „regularisierte logistische Regression“ mit L1- und L2-Regularisierung ("Elastic Net") durch und bestimmen sie mittels vierfacher Kreuzvalidierung und "Randomized Search" den optimalen Bestrafungsparameter und das optimale L1/L2-Verhältnis (Hyperparameteroptimierung). "Fitten" Sie anschließend das Modell mit den besten Parametern auf allen Folds. Erstellen Sie damit Prognosewahrscheinlichkeiten für die Validierungsdaten und wenden Sie die Bewertungsfunktion an.

d) "Vergleichen und bewerten": Vergleichen und interpretieren Sie die Ergebnisse aus b) und c) hinsichtlich Aufwand, Laufzeit und erzielter Modellgröße. Gehen Sie dabei auch auf die Anzahl verwendeter Features und die Modellinterpretierbarkeit ein.

Hinweis: Für diese Aufgabe kann bei Bedarf der in Aufgabe B-5 a) erzeugte balancierte Trainingsdatensatz verwendet werden.

Lösungsvorschlag:

a) Logistische Regression und Regularisierung

Die logistische Regression modelliert eine Wahrscheinlichkeitsfunktion, die die Wahrscheinlichkeit einer Kategorie (z.B. ein Kreditausfall) vorhersagt. Sie tut dies, indem sie eine lineare Funktion der Eingabefunktionen (Merkmale) anpasst, gefolgt von einer Sigmoid-Funktion, die die Ausgabe auf einen Bereich zwischen 0 und 1 beschränkt. Die Ausgabe der Sigmoid-Funktion wird dann als Wahrscheinlichkeit interpretiert, dass eine gegebene Beobachtung in der positiven Klasse ist.

Die logistische Regression kann im vorliegenen Fall (Ermittlung von Kreditausfallwahrscheinlichkeiten) eine Überanpassung vermeiden, indem sie eine Regularisierung verwendet, die als Penalisierung bezeichnet wird. Durch die Penalisierung werden die Gewichte, die der Algorithmus für jedes Merkmal lernt, gedämpft, um eine Überanpassung an die Trainingsdaten zu vermeiden. Es gibt zwei Hauptformen der Regularisierung, die in der logistischen Regression verwendet werden: Gemäß Ridge-Regression (L2-Regularisierung) und LASSO (L1-Regularisierung). Die L2-Regularisierung fügt der Kostenfunktion eine Penalisierung der Quadratsumme der Gewichte hinzu, während bei der L1-Regularisierung hierfür die Summe der absoluten Gewichte verwendet wird. Das Verfahren "Elastic Net" verwendet eine Kombination aus der L1- und L2-Regularisierungen und hat das Potential für eine höhere Modellgüte, allerdings zu Lasten eines weiteren zu optimierenden Hyperparameters (Anteil L1/L2).

Die LASSO- und "Elastic Net"-Regularisierung (L1-Komponente) kann bei der logistischen Regression dazu führen, dass für zahlreiche Features die Koeffizienten exakt den Wert 0 annehmen und damit abgewählt und für das Scoring und die Erklärbarkeit der Modellvorhersagen nicht mehr relevant sind. Die damit erzielbare "Parametersparsamkeit" kann als ein ggf. entscheidender Vorteil der LASSO-Methode angesehen werden.

b) Logistische Regression mit L1-Regularisierung

In [51]:
```
# Suche den zu max. 30 Features <> 0 passenden Komplexitätsparameter C
tic = time.time()

c_list = np.array([0.001, 0.002, 0.0034, 0.005, 0.1])

for c in c_list:
    algo = "LogReg (L1), C:"
    model = LogisticRegression(penalty='l1', solver = 'saga', C=c, random_state=seed)
    model.fit(Xs_train,ys_train)
    probs = model.predict_proba(X_val)
    result = roc_auc_score(y_val, probs[:,1])
    print(algo + " %8s" % c + ", AUC:" + "%8.3f" % result + ", Coefficients <> 0:" + " %s" % len(model.coef_[model.coef_ != 0]))

print("\ntime (sec):" + "%6.0f" % (time.time() - tic))
```

```
LogReg (L1), C:    0.001, AUC:   0.681, Coefficients <> 0: 4
LogReg (L1), C:    0.002, AUC:   0.710, Coefficients <> 0: 15
LogReg (L1), C:   0.0034, AUC:   0.723, Coefficients <> 0: 30
LogReg (L1), C:    0.005, AUC:   0.729, Coefficients <> 0: 38
LogReg (L1), C:      0.1, AUC:   0.739, Coefficients <> 0: 111

time (sec):    23
```

In [52]:
```
# Den passenden Wert für C anwenden
LRLs = LogisticRegression(penalty='l1', solver = 'saga', C=0.0034, random_state=seed)
```

```
LRLs.fit(Xs_train,ys_train)
```

Out[52]:

```
LogisticRegression(C=0.0034, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l1',
                   random_state=42, solver='saga', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [53]:

```
print("Anzahl Koeffizienten <> 0: ",len(LRLs.coef_[LRLs.coef_ != 0]), " von ", n_features)
```

```
Anzahl Koeffizienten <> 0:  30  von  120
```

In [54]:

```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = LRLs.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], 0.5)

# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("LogReg L1-30 (bal.)")
mprobs.append(probs[:,1])
```

```
Summary statistics of predicted 'y':
      count      mean       std       min       50%  max
0  40110.0  0.443734  0.153868  0.051724  0.427839  1.0

 Accuracy:
 0.6787

 Confusion Matrix:
 Pred. 0    1
 [[25251 11757]
 [ 1132  1970]]

AUC: 0.7226
===========
```

c) Logistische Regression mit L1- und L2-Regularisierung

In [55]:

```
tic = time.time()

# Erstellen eines Elastic Net-Logistikregressionsmodells
model = LogisticRegression(penalty='elasticnet', solver = 'saga', tol = 0.01, random_state=seed)

# Bestimmung des optimalen Bestrafungsparameters und L1/L2-Verhältnisses durch Hyperparameteroptimierung

# Erstellen eines Parameters-Grids
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'l1_ratio': [0,0.05,0.1,0.25,0.5,0.75,0.9,0.95,1]}

# Erstellen des RandomizedSearchCV-Objekts
LREs = RandomizedSearchCV(model, param_grid, cv=4, n_iter=20, scoring='roc_auc',
                          n_jobs=-1, random_state=seed)

# Training des Modells mittels GridSearchCV
LREs.fit(Xs_train, ys_train)

# Ausgabe des besten Parameters und L1/L2-Verhältnisses
print("Best Penalty:", LREs.best_params_)


print("time (sec):" + "%6.0f" % (time.time() - tic))
```

```
Best Penalty: {'l1_ratio': 0.5, 'C': 10}
time (sec):    58
```

In [56]:

```
# LogisticRegression (Elastic Net): Erstelle neues Modell auf allen Folds mit den besten Parametern
LRE2s = LogisticRegression(**LREs.best_params_)
LRE2s.fit(Xs_train, ys_train)
```

Out[56]:

```
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=0.5, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [57]:

```
print("Anzahl Koeffizienten <> 0: ",len(LRE2s.coef_[LRE2s.coef_ != 0]), " von ", n_features)
```

```
Anzahl Koeffizienten <> 0:  120  von  120
```

In [58]:

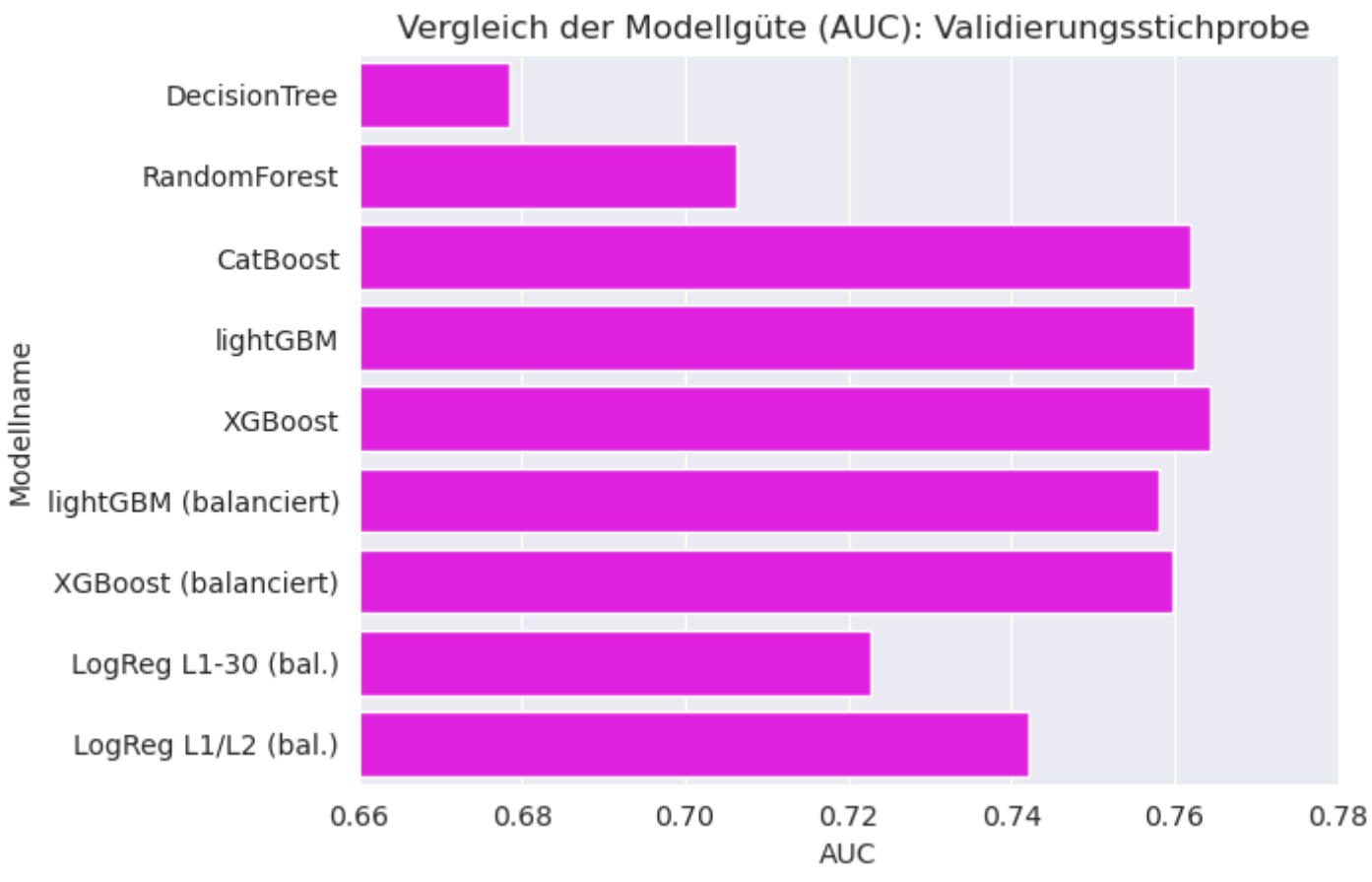
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = LRE2s.predict_proba(X_val)

# Bewertungsfunktion anwenden:
assess(y_val, probs[:,1], 0.5)

# Modellliste um dieses Ergebnis ergänzen
```

```
mauc.append(result)
mname.append("LogReg L1/L2 (bal.)")
mprobs.append(probs[:,1])
```

```
Summary statistics of predicted 'y':
      count      mean       std       min      50%  max
0  40110.0  0.419602  0.205434  0.000004  0.39881  1.0

 Accuracy:
 0.6851

 Confusion Matrix:
 Pred. 0    1
 [[25376 11632]
 [  999  2103]]

AUC: 0.7422
===========
```

d) Vergleichen und bewerten

In [59]:
```
plot_auc_vergleich(dict, 0.66, 0.78, "Validierungsstichprobe")
```



Fazit: Die Prognosegüte der logistischen Regression mit L1-Regularisierung und Beschränkung auf 30 Koeffizienten liegt im Mittelfeld, während die der logistischen Regression mit optimiertem L1- und L2-Regularisierungsparametern etwas besser ausfällt. Beide Modelle fallen in dieser Hinsicht deutlich hinter die Prognosegüte der GB-Modelle zurück. Sie weisen aber eine sehr kurze Laufzeit von wenigen Sekunden auf und sind transparente und nachvollziehbare Modelle. Während das Modell "LogReg L1-30 (bal.)" bereits als parameterarm definiert und ausgewählt wird, besteht diese Einschränkung bei "LogReg L1/L2 (bal.)" nicht. Im optimierten Ergebnis werden bei letzterem Modell keine Koeffizienten auf 0 gesetzt und damit keine Features abgewählt.

Aufgabe B-7: Künstliche Neuronale Netze anwenden [Lernziele 4.3, 6.1.2; 10 Punkte]

---

a) "Neuronales Netz aufbauen und fitten": Erstellen Sie mit Keras ein vorwärts gerichtetes voll verknüpftes Neuronales Netz mit drei verborgenen Schichten und verwenden Sie Kernel-Regularisierung L2. Geben Sie einen Überblick über die Modellparameter aus ("summary"). Trainieren Sie das Neuronale Netz max. 100 Epochen und visualisieren sie die Konvergenz. Verwenden Sie dabei eine „Batch Size“ zwischen 100 und 10000 und einen beliebigen „Optimizer“. Erstellen Sie Prognosewahrscheinlichkeiten für die Validierungsdaten und wenden Sie die Bewertungsfunktion an. Falls die Prognosegüte dieses hochparametrigen Modells schlechter als die der Logistischen Regression (Aufgabe B-6) ist sind die Ursachen zu suchen und zu beheben.

b) "Vergleichen und bewerten": Vergleichen und interpretieren Sie das Ergebnisse mit den in Aufgabe B-6 gefitteten Modellen. Erstellen Sie eine Grafik, die die Verteilung der Prognosewahrscheinlichkeiten der letzten vier gefitteten Modelle (Aufgaben B-5c, B-6, B-7) zeigt und interpretieren Sie das Ergebnis.

c) "Diskussion der Parameter": Begründen Sie bezüglich „Batch Size“ und „Optimizer“ Ihre Wahl. Geben Sie dabei an, wie viele Schritte des Gradientenabstiegsverfahrens bei der von Ihnen gewählten "Batch Size" je Epoche für den Trainingsdatensatz erforderlich sind, und erläutern Sie, wie sich das auf die Laufzeit und typischerweise die Prognosegüte auswirkt (siehe auch SAV-Tutorial 2 „Insights from Inside Neural Nets“).

Hinweis: Für diese Aufgabe kann bei Bedarf der in Aufgabe B-5 a) erzeugte balancierte Trainingsdatensatz verwendet werden.

a) Neuronales Netz aufbauen und fitten

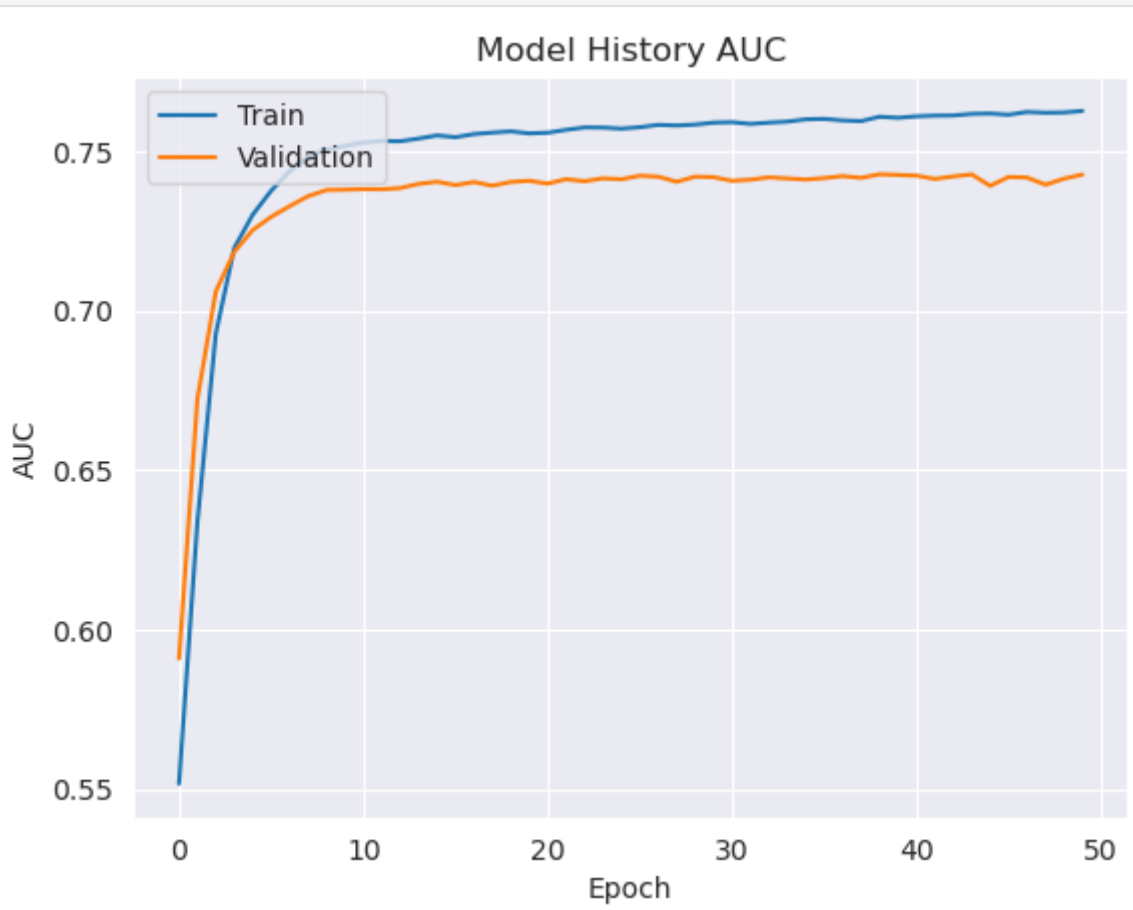
In [60]:
```
# Bestimmen des input_shape von Xs_train
input_shape = Xs_train.shape

## Erstellen des Modells
def create_model(l2_param=0.0):
    model = Sequential()
    model.add(Dense(16, input_dim=input_shape[1], activation='relu', kernel_regularizer=l2(l2_param)))
    model.add(Dense(8, activation='relu', kernel_regularizer=l2(l2_param)))
    model.add(Dense(4, activation='relu', kernel_regularizer=l2(l2_param)))
    model.add(Dense(1,kernel_initializer='normal', activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['AUC'])
    return model

# Erstellen eines Modells mit L2-Regularisierung
NNs = create_model(l2_param=0.005)

# Ausgabe eines Überblicks über die Modellparameter
NNs.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                1936

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 4)                 36

 dense_3 (Dense)             (None, 1)                 5

=================================================================
Total params: 2,113
Trainable params: 2,113
Non-trainable params: 0
_________________________________________________________________
```

In [61]:
```
tic = time.time()

# Training des Modells
history = NNs.fit(Xs_train, ys_train, batch_size=500, epochs=50,
                  validation_data=(X_val, y_val),validation_split = 0.25, verbose=False)

print("time (sec):" + "%6.0f" % (time.time() - tic))
```

```
time (sec):    42
```

In [62]:
```
# Anzeige von loss und auc über die max. 10 Epochen
#pd.DataFrame(history.history).head(10)

# Visualisieren der Konvergenz
plt.plot(history.history['auc'])
plt.plot(history.history['val_auc'])
plt.title('Model History AUC')
plt.ylabel('AUC')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



In [63]:
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = NNs.predict(X_val)
```

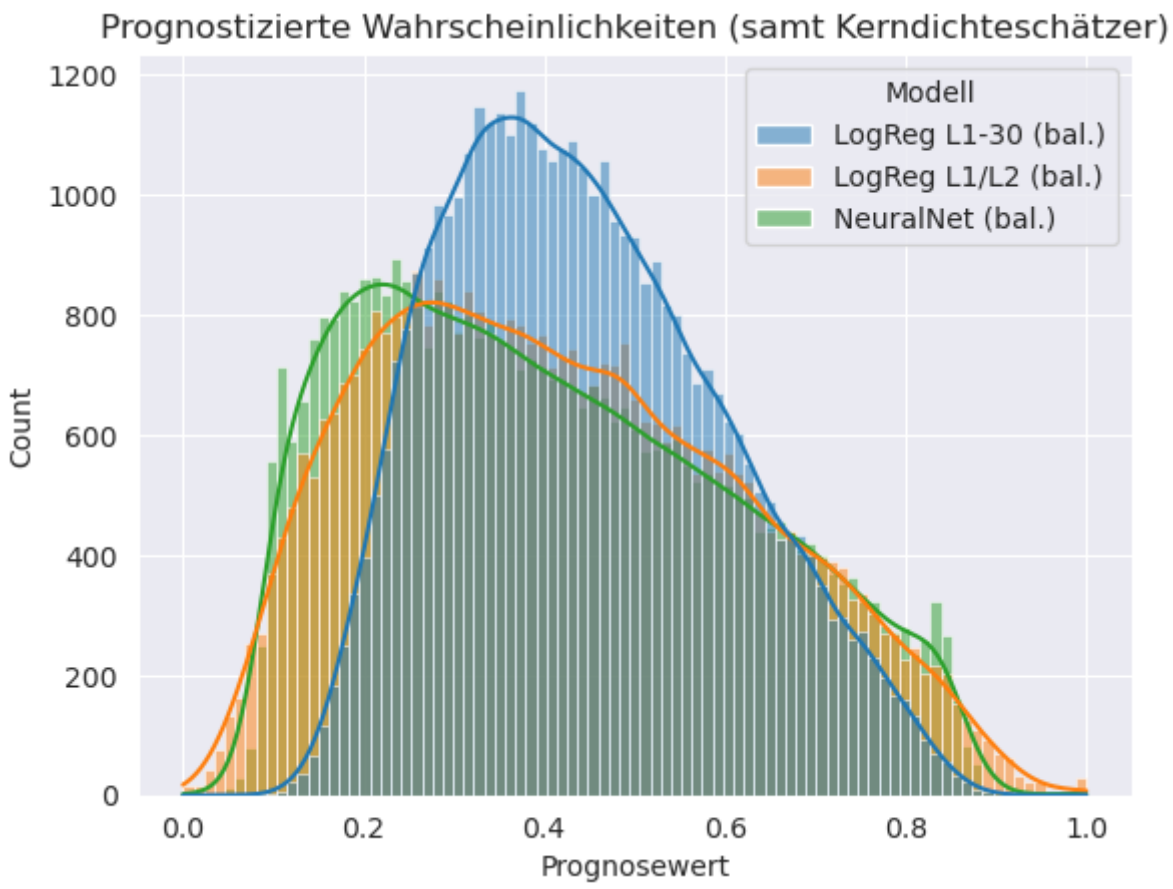
```
# Bewertungsfunktion anwenden:
assess(y_val, probs, 0.5)
```

```
1254/1254 [==============================] - 2s 2ms/step

Summary statistics of predicted 'y':
      count      mean       std       min       50%  max
0  40110.0  0.409665  0.204927  0.001034  0.383382  1.0

 Accuracy:
 0.6942

 Confusion Matrix:
 Pred. 0    1
 [[25792 11216]
 [ 1051  2051]]

AUC: 0.7427
===========
```

In [64]:
```
# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("NeuralNet (bal.)")
mprobs.append(probs[:,0])
```

b) Vergleichen und bewerten

In [65]:
```
# Plots der  prognostizierten Wahrscheinlichkeiten für die drei letzten gefitteten Modelle
liste = [ "LogReg L1-30 (bal.)", "LogReg L1/L2 (bal.)", "NeuralNet (bal.)"]
plot_prob_dist(mname, mprobs, liste)
```



Modellgüte und Laufzeit: Die Modellgüte des Neuronalen Netzes ist geringfügig besser als die der logistischen Regressionsmodelle aus Aufgabe B-6, die Laufzeit ebenfalls recht kurz (unter einer Minute). Im Vergleich zur logistischen Regression konnte sich das oben gefittete neuronale Netz nicht als vorteilhaft erweisen.

Interpretation der Wahrscheinlichkeitsverteilung: Die drei zu vergleichenden Modelle wurden alle auf der balancierten Stichprobe gefittet und sollten eine ähnliche Verteilung der Prognosewahrscheinlichkeiten aufweisen. Bei den beiden Modellen "LogReg L1/L2 (bal.) und "NeuralNet (bal.)", die annähernd die gleiche Prognosegüte (AUC) aufweisen, ist das überwiegend auch der Fall. Die Verteilung des Modells mit der im Vergleich schlechtesten Prognosegüte "LogReg L1-30 (bal.)" ist vergleichsweise schmal und konzentriert sich auf mittlere Wahrscheinlichkeiten.

c) Diskussion der Parameter

Das stochastische Gradientenabstiegsverfahren wurde mit der "adam"-Optimierung durchgeführt, die auf der adaptiven Schätzung von Momenten erster und zweiter Ordnung beruht und sich hier als geeignet erweist.

Die "Batch Size" (BS) gibt die Größe der Teilstichproben an, die in jedem Schritt der Gradientenabstiegsmethode (GDM) berücksichtigt werden. Je größer die BS ist, umso weniger GDM-Schritte werden je Epoche benötigt und umso kürzer ist die Laufzeit je Epoche. Allerdings kann eine zu große BS dazu führen, dass das Modell in ein lokales Minimum fällt und die Konvergenz und Modellgüte beeinträchtigt wird. Der gewählte BS-Parameter (500) führt zu einer schnellen Konvergenz und zur Erfüllung der oben genannten Anforderungen hinsichtlich der Modellgüte. Da die balancierte Trainingsstichprobe Xs_train nur 19.182 Datensätze enthält, von denen 75% verwendet werden, sind je Epoche lediglich 29 GDM-Schritte notwendig (29*500=14.500).

Aufgabe B-8: Hauptkomponentenanalyse (PCA) und XGBoost [Lernziele 4.1, 5.1, 6.1.2; 11 Punkte]

Als Alternative zur im Block A mittels IV und FI durchgeführten Merkmalsvorauswahl "Auswahl A" soll in dieser Aufgabe die Merkmalsvorauswahl mittels einer Hauptkomponentenanalyse (PCA) vorgenommen werden und anschließend XGBoost angewendet werden.

a) "PCA erläutern und einordnen": Beschreiben Sie kurz die Hauptkomponentenanalyse (PCA) und wie diese zur Merkmalsvorauswahl eingesetzt werden kann. Worin liegen die wesentlichen Unterschiede im Vergleich zu den im Aufgabenteil A verwendeten Verfahren und welche Vorteile und Nachteile sind damit verbunden?

b) "Daten aufbereiten und PCA durchführen": Bereiten Sie die im Aufgabenteil A erzeugte Datei "train_komplett.csv" für eine sachgerechte Durchführung der Hauptkomponentenanalyse (PCA) auf, führen Sie diese durch und prüfen Sie das Ergebnis. Zeigen Sie die erklärende Varianz in Abhängigkeit von der Anzahl der Komponenten geeignet grafisch an und erläutern und interpretieren Sie das Ergebnis. Wählen Sie eine mit der "Auswahl A" vergleichbar große Anzahl an "Features" (Komponenten) aus und schließen Sie die Datenaufbereitung ab.

c) "XGBoost nach PCA": Wiederholen Sie die im Aufgabenteil B-4 c) beschriebenen Schritte mit den hier aufbereiten Daten. Vergleichen Sie die Laufzeiten und Prognoseergebnisse mit den entsprechenden Werten aus Aufgabe B-4 bzw. B-5 und interpretieren Sie das Ergebnis. Legen Sie fest, inwieweit dieses Modell beim abschließenden Scoring betrachtet werden soll und begründen Sie Ihre Wahl.

Lösungsvorschlag:

a) PCA erläutern und einordnen

Die Hauptkomponentenanalyse (PCA) ist ein Verfahren des „unüberwachten Lernen“ zur Dimensionenreduktion, das die Merkmale identifiziert, die die meiste Varianz innerhalb der Daten aufweisen. Dies geschieht durch eine lineare Transformation der Daten in ein neues Koordinatensystem, in dem (der größte Teil) der Variation in den Daten mit weniger Dimensionen beschrieben werden kann als in den Ausgangsdaten. Diese Eigenschaft soll im Folgenden als Alternative zur Merkmalsvorauswahl analog Aufgabenteil A verwendet werden. Ein wesentlicher Vorteil ist die Unkorreliertheit der Hauptkomponenten, die sich positiv auf die Stabilität der Modelle auswirken könnte. Wesentliche Nachteile bestehen darin, dass keine echte Merkmalsauswahl stattfindet (die Hauptkomponenten verwenden alle "Features" des zugrundeliegenden Datensatzes), dass die "Prognosekraft" der Merkmale keine Rolle spielt und dass das resultierende Prognoseergebnis selbst bei Verwendung der logistischen Regression schwer interpretierbar ist.

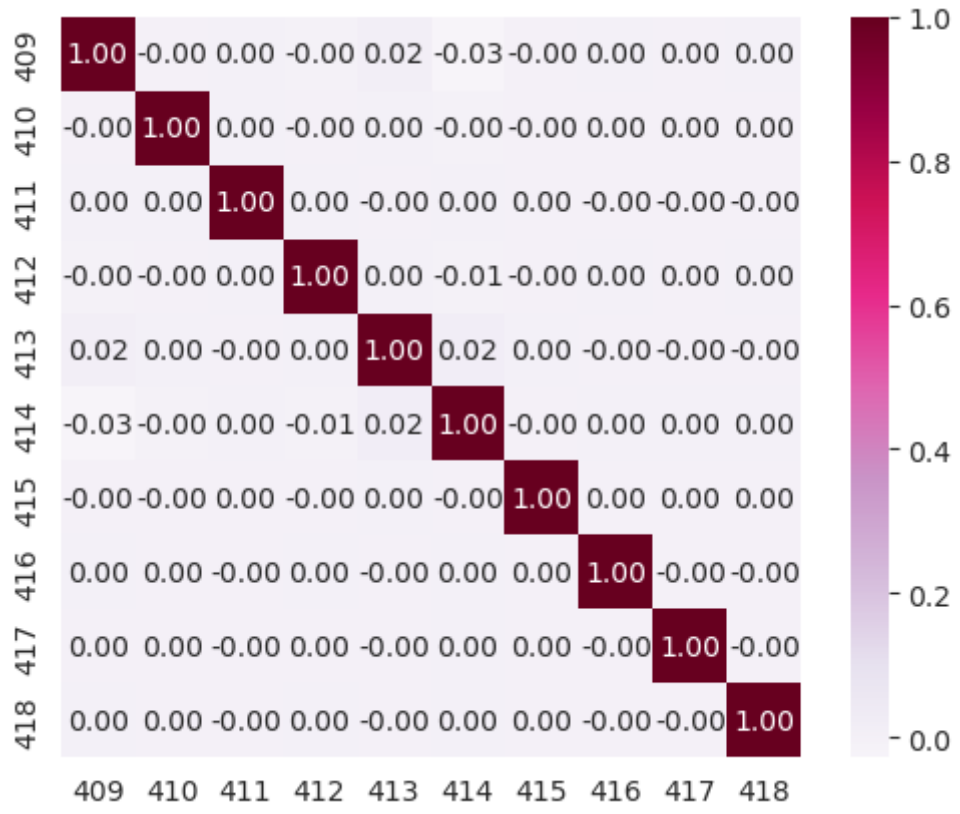
b) Daten aufbereiten und PCA durchführen

In [66]:
```
# Einlesen und Check der Trainingsdaten mit allen Features
df_full = pd.read_csv("train_komplett.csv", sep=';', decimal=',')

print("\nAnzahl Zeilen und Spalten: ", df_full.shape)
print("Anzahl fehlende Werte: ", df_full.isna().sum().sum())
# ML-Matrizen anlegen (u.a.ID entfernen)
Xf = df_full.drop(['SK_ID_CURR','TARGET'], axis=1)
yf = df_full['TARGET']
```

```
Anzahl Zeilen und Spalten:  (160440, 421)
Anzahl fehlende Werte:  0
```

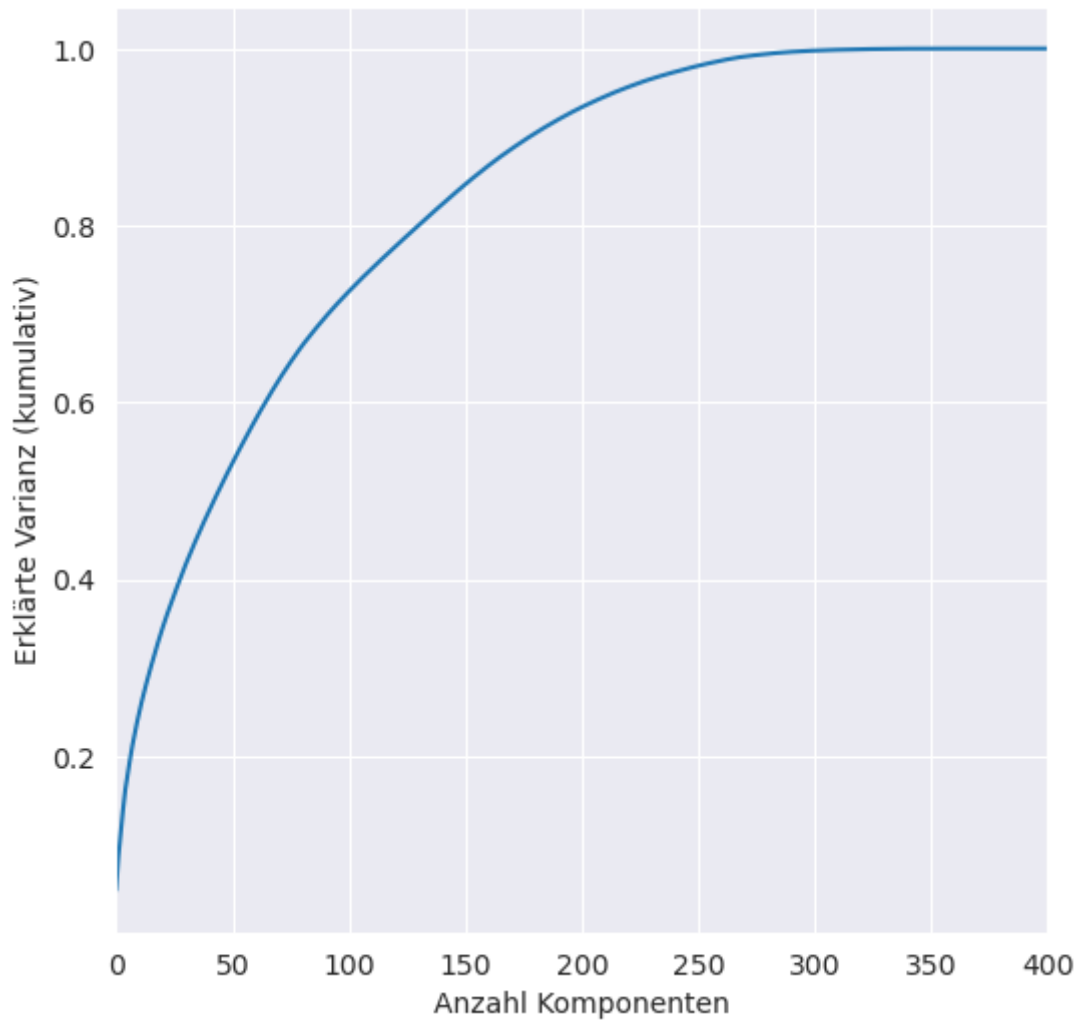
In [67]:
```
# Gesamte Feature Matrix skalieren (analog Teil A Trainings- und Validierungsdaten verwenden):
scaler_full = StandardScaler()
Xf_scaled = scaler_full.fit_transform(Xf)

# PCA durchführen
pca = PCA()
Xp = pd.DataFrame(pca.fit_transform(Xf_scaled))
```

In [68]:
```
# Korrelation der letzten zehn Hauptkomponenten prüfen und visualisieren:
corrmat = Xp.iloc[:,-10:].corr()
sns.heatmap(corrmat, annot=True, square=True, cmap='PuRd', fmt='.2f');
```



Fazit der obigen Plausibilitätsprüfungen: Die Daten wurden korrekt verarbeitet.

In [69]:
```
# Erklärte Varianz anzeigen
plt.figure(figsize=(6,6))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlim(0,400,1)
plt.xlabel('Anzahl Komponenten')
plt.ylabel('Erklärte Varianz (kumulativ)')
plt.show()
```



Die erklärte Varianz der ersten Hauptkomponenten ist überdurchschnittlich hoch, daher steigt die kumulative Betrachtung zuächst steil an und flacht dann ab. Die letzten 100 Komponenten leisten nur noch einen sehr geringen Beitrag hierzu von unter einem Prozent.

Für die Auswahl einer mit der Merkmalsauswahl im Aufgabenteil A vergleichbar großen Anzahl an "Features" wird der Schwellenwert mit Bezug auf die obige Grafik auf 75% festgelegt:

In [70]:
```
# Erklärte Varianz kumulieren und Schwellenwert festlegen
cumsum = np.cumsum(pca.explained_variance_ratio_)
dim = np.argmax(cumsum >= 0.75) + 1
print('Die Anzahl der Dimensionen, die erforderlich ist, um 75 % der Varianz zu erhalten, beträgt ',dim,'.', sep='')
```

```
Die Anzahl der Dimensionen, die erforderlich ist, um 75 % der Varianz zu erhalten, beträgt 111.
```

Die Entfernung der nicht weiter berücksichtigten Komponenten und die anschließende Aufteilung der Daten in die Modellierungstichproben analog Aufgabe B-1 schließen die Datenaufbereitung ab:

In [71]:
```
# Aus Xp nicht zu berücksichtigende Features entfernen
Xp = Xp.iloc[:,0:dim]

# Modellierungstichprobe für die ausgewählte Anzahl an Hauptkomponenten "dim" erzeugen
```

```
Xp_train, Xp_val, yp_train, yp_val = train_test_split(Xp, y, test_size=0.25, random_state=seed)
Xp_train.shape # Check Dimensions
```

Out[71]:
```
(120330, 111)
```

In [72]:
```
Xp_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 120330 entries, 142133 to 121958
Columns: 111 entries, 0 to 110
dtypes: float64(111)
memory usage: 102.8 MB
```

c) XGBoost nach PCA

In [73]:
```
# XGBClassifier: Hyperparameteroptimierung mit RandomizedSearchCV
tic = time.time()

XGBp_rs = RandomizedSearchCV(XGBClassifier(n_estimators=1000, tree_method='gpu_hist'),
                             param_grid_XGB, cv=4, n_iter=20, scoring='roc_auc',
                             n_jobs=-1, random_state=seed)
XGBp_rs.fit(Xp_train,yp_train)

# Laufzeit und beste Parameter (sowie die Nächstbesten) ausgeben
print("time (sec):" + "%6.0f" % (time.time() - tic))
print("Beste Parameter:",XGBp_rs.best_params_)
pd.DataFrame(XGBp_rs.cv_results_)[sel_params_XGB].sort_values("rank_test_score").head()
```

```
time (sec):   665
Beste Parameter: {'colsample_bytree': 0.31171107608941095, 'learning_rate': 0.019870979328434287, 'max_depth': 4, 'subsample':
0.5677003278199915}
```

Out[73]:

	param_learning_rate	param_max_depth	param_subsample	param_colsample_bytree	mean_test_score	rank_test_score
13	0.019871	4	0.5677	0.311711	0.729948	1
4	0.024868	4	0.524756	0.992212	0.729858	2
18	0.011328	7	0.802197	0.607034	0.728950	3
5	0.011732	5	0.399861	0.431945	0.728414	4
17	0.040447	4	0.965255	0.271349	0.728249	5

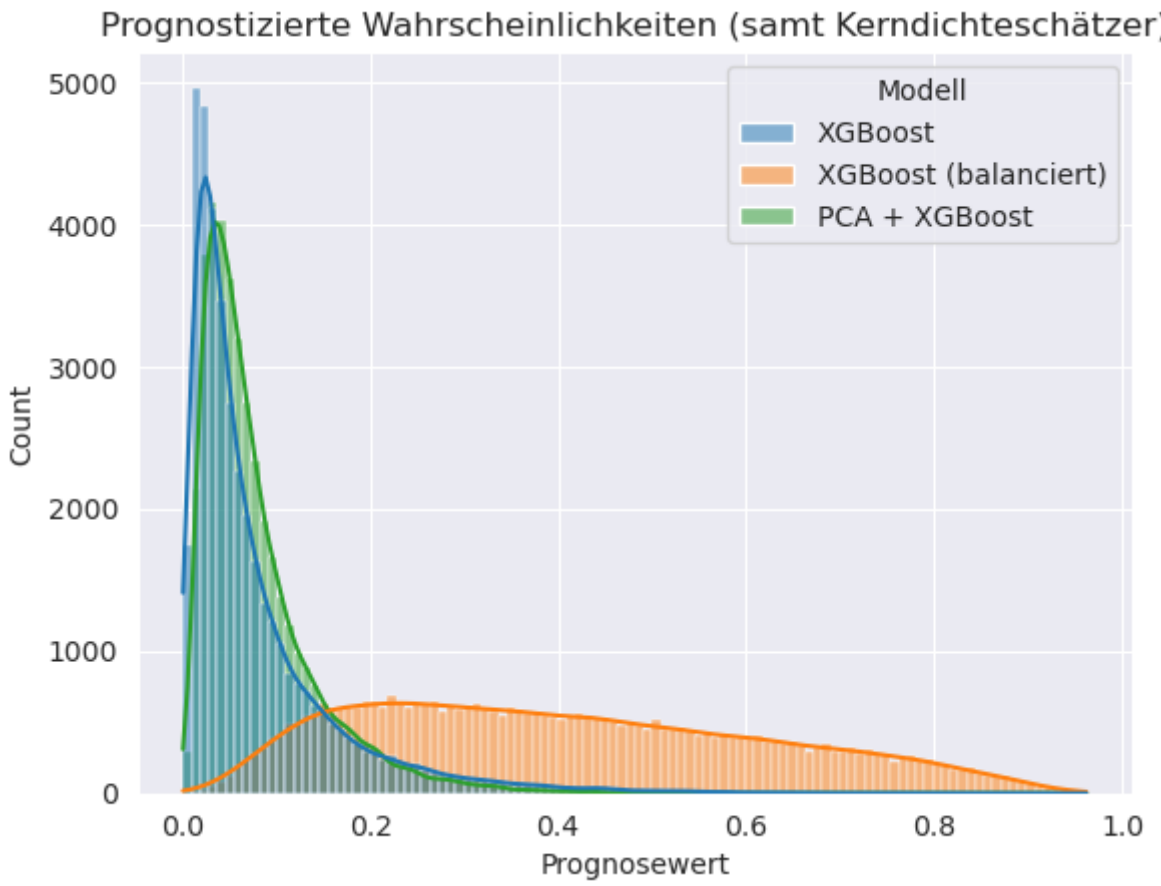
In [74]:
```
# XGBClassifier: Erstelle neues Modell auf allen Folds mit den besten Parametern
tic = time.time()
XGB2p = XGBClassifier(**XGBp_rs.best_params_,
                      n_estimators=1000, tree_method='gpu_hist', eval_metric='auc')
XGB2p.fit(Xp_train, yp_train)
print("time (sec):" + "%6.0f" % (time.time() - tic))
```

```
time (sec):     5
```

In [75]:
```
# Prognosewahrscheinlichkeiten für die Validierungsdaten ermitteln
probs = XGB2p.predict_proba(Xp_val)

# Bewertungsfunktion anwenden:
assess(yp_val, probs[:,1], cm_threshold)

# Modellliste um dieses Ergebnis ergänzen
mauc.append(result)
mname.append("PCA + XGBoost")
mprobs.append(probs[:,1])
```

```
Summary statistics of predicted 'y':
      count     mean       std       min       50%       max
0  40110.0  0.07944  0.062878  0.003489  0.060805  0.540555

 Accuracy:
 0.6645

 Confusion Matrix:
 Pred. 0    1
 [[24593 12415]
 [ 1041  2061]]

AUC: 0.7242
===========
```

In [76]:
```
# Plots der  prognostizierten Wahrscheinlichkeiten für die drei XGBoost-Modelle
liste = [ "XGBoost", "XGBoost (balanciert)", "PCA + XGBoost"]
plot_prob_dist(mname, mprobs, liste)
```



Vergleich der Modellgüte

In [77]:
```
dfp = pd.DataFrame(dict)
dfp = dfp[["Modellname","AUC"]]
HTML(dfp[dfp["Modellname"].isin(liste)].to_html(index=False))
```

Out[77]:

Modellname	AUC
XGBoost	0.764276
XGBoost (balanciert)	0.759628
PCA + XGBoost	0.724161

Fazit: Das "fitten" des XGBoost-Modells mit den PCA-Daten dauert im Vergleich mit dem Modell aus Aufgabe B-4 nicht wesentlich länger. Die an den Validierungsdaten gemessene Modellgüte hingegen fällt deutlich hinter die der zuvor gefitteten XBoost-Modelle zurück.

Festlegung: Die in das Modell eingeflossenen Daten wurden über einen alternativen Ansatz aus dem Bereich des "unüberwachten Lernens" erstellt. Das hat sich bei der vorliegenden Modellierung jedoch nicht als vorteilhaft erwiesen. Die im Aufgabenteil a) bereits genannten Nachteile kommen noch hinzu. Daher soll dieses Model beim abschließenden Scoring nicht betrachtet werden.

Aufgabe B-9: Modellbewertung und Anwendung [Lernziel 6; 15 Punkte]

---

a) "Modellvergleich und Gesamtbewertung":

- **Vergleichen Sie auf Basis der vorherigen Aufgaben abschließend alle Modelle und nehmen Sie eine Gesamtbewertung vor. Erstellen Sie eine grafische Darstellung der Modellgüte (AUC) der elf Modelle aus den Aufgaben B-3 bis B-8 an der Validierungsstichprobe. Welche Modelle haben die beste Prognosegüte? Gibt es im Hinblick auf Prognosegüte, Optimierungsaufwand und Laufzeit ein besonders geeignetes und empfehlenswertes Modell?**
- **Welche Auswirkungen hat das "Subsampling" auf die prognostizierten Wahrscheinlichkeiten, die Laufzeiten und die Modellgüte?**
- **Wie schneiden die transparenten (parameterarmen) Modelle hinsichtlich der Prognosegüte ab?**

b) "Testdaten aufbereiten, Vorhersagen erstellen und Modellgüte vergleichen": Lesen Sie den im Aufgabenteil A erzeugten Testdatensatz "test_komplett.csv" ein, bereiten Sie die Daten sachgerecht auf und erstellen Sie für die Testdaten Vorhersagen mit den fünf besten Prognosemodellen sowie der logistische Regression aus Aufgabe B-6 a). Stellen Sie die Prognosegüte grafisch dar, vergleichen Sie die Ergebnisse mit a) und erläutern und begründen Sie ggf. beobachtete Unterschiede.

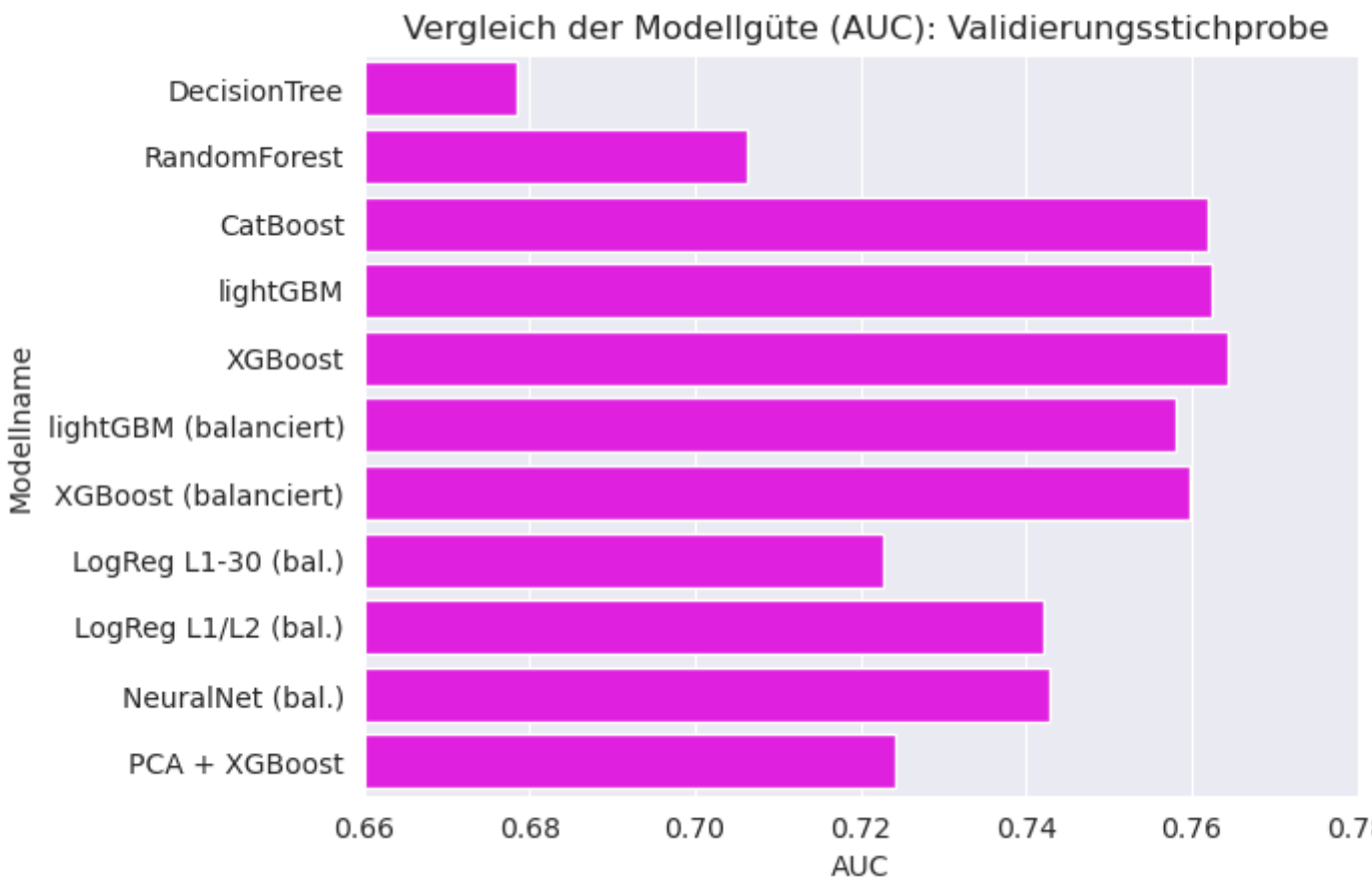
c) "Gain- und Lift-Chart für das Top-Dezil": Während bisher die Modelle im gesamten Prognosebereich bewertet wurden, soll nun im Hinblick auf konkrete Anwendungen das Dezil mit den höchsten vorhergesagten Kreditausfallwahrscheinlichkeiten näher untersucht werden. Erstellen Sie dazu auf Basis der in b) aufbereiten Daten und Vorhersagen einen gemeinsamen kumulativen "Gain-Chart" und einen gemeinsamen Lift-Chart für die in b) genannten Prognosemodelle, schränken Sie die Grafik auf das genannte Dezil ein und interpetieren Sie das Ergebnis hinsichtlich der beobachteten Unterschiede zwischen den Modellen.

d) "Rechtlicher Rahmen": Die Datenschutz-Grundverordnung (DSGVO) stellt hohe Anforderungen an Scoring-Modelle, insbesondere im Hinblick auf Datenschutz, Transparenz und Nachvollziehbarkeit. Erläutern Sie, welche Auswirkungen das auf die bisher betrachten Modelle hat und welche modellbezogenen Aspekte gegebenenfalls noch zu untersuchen und prüfen sind.

Lösungsvorschlag:

a) Modellvergleich und Gesamtbewertung

In [78]:
```
plot_auc_vergleich(dict, 0.66, 0.78, "Validierungsstichprobe")
```



Beste Modelle: Die drei Gradient-Tree-Boosting-Tools CatBoost, LightGBM und XGBoost erzielen bei der Validierungsstichprobe die mit Abstand besten Prognosen. Die Ergebnisse von lightGBM und XGBoost werden mittels eines aufwändigen Hyperparametertunings und längeren Laufzeiten erzielt. Das XGBoost-Modell mit unbalancierten Daten (Aufgabe B-4) weist die höchste Prognosegüte auf. CatBoost erzielt bereits mit Standardeinstellungen in kurzer Zeit eine ähnliche Prognosegüte und verdient hier eine Empfehlung.

Subsampling: Generell führt Subsampling dazu, dass die prognostizierten Wahrscheinlichkeiten (viel) zu hoch sind und auf die ursprünglichen Gegebenheiten umgerechnet werden sollten. Durch das Subsampling der Trainingsdaten auf eine balancierte Stichprobe mit gleich vielen Ereignissen (Kreditausfälle) und Nicht-Ereignissen werden die Laufzeiten der Modellanpassungen von LightGBM und XGBoost stark reduziert. Die Modellgüte geht dabei nur leicht zurück. Bei einem weniger radikalen "Subsampling" kann der Verlust an Prognosegüte vernachlässigbar sein und in bestimmten Fällen (hier z.B. bei 1:4) ist sogar eine Prognoseverbesserung möglich.

Transparente und nachvollziehbare Modelle: Während der recht kleine und nicht optimierte Entscheidungsbaum hinsichtlich der Prognosegüte weit abfällt, liegt die Prognosegüte der logistischen Regression mit L1-Regularisierung und Beschränkung auf höchstens 30 Koeffizienten im Mittelfeld und ist das beste parameterarme, transparente und nachvollziehbare Modell.

b) Testdaten aufbereiten, Vorhersagen erstellen und Modellgüte vergleichen

In [79]:
```
# Einlesen und aufbereiten der Testdaten (inkl. Skalierung)
df_test = pd.read_csv("test_komplett.csv", sep=';', decimal=',')

X_test = df_test[feature_names]
X_test = pd.DataFrame(scaler.transform(X_test), columns = feature_names)
y_test = df_test['TARGET']
event_rate_test = len(y_test[y_test == 1])/len(y_test)

print("\nAnzahl Zeilen und Spalten von X_test:", X_test.shape)
print("Anzahl fehlende Werte: ", X_test.isna().sum().sum())
print("\nEreignisrate in y_test:",round(event_rate_test,6))
print(" Abweichung zu y_train:", round((event_rate_test/cm_threshold-1)*100,2),"%")
```

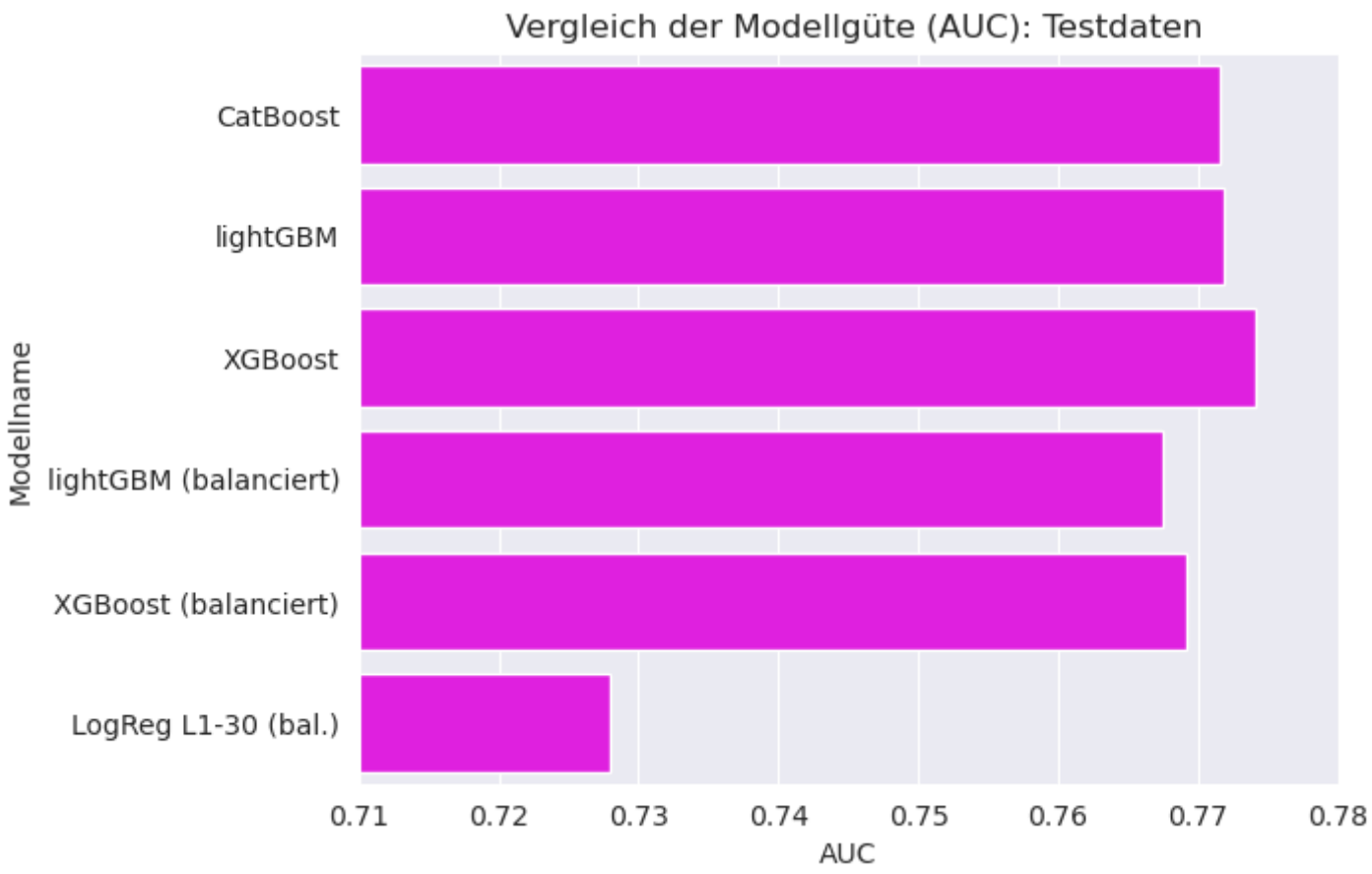
```
Anzahl Zeilen und Spalten von X_test: (30856, 120)
Anzahl fehlende Werte:  0

Ereignisrate in y_test: 0.081313
 Abweichung zu y_train: 2.02 %
```

In [80]:
```
# Prognosemodelle an Testdaten anwenden (wg. plot_lift_chart beide Klassen behalten)
y_test_CBC = CBC.predict_proba(X_test)
y_test_LGB2 = LGB2.predict_proba(X_test)
y_test_XGB2 = XGB2.predict_proba(X_test)
y_test_LGB2s = LGB2s.predict_proba(X_test)
y_test_XGB2s = XGB2s.predict_proba(X_test)
y_test_LRLs = LRLs.predict_proba(X_test)
```

In [81]:
```
# Liste für die Speicherung der Testparameter vorbereiten (siehe Aufgabe B-2)
mname_test = []
mauc_test = []
dict_test = {'AUC': mauc_test, 'Modellname': mname_test}
```

In [82]:
```
# AUC-Werte für Testdaten ermitteln und Grafik ausgeben
mname_test.append("CatBoost")
mauc_test.append(roc_auc_score(y_test, y_test_CBC[:,1]))
```

```
mname_test.append("lightGBM")
mauc_test.append(roc_auc_score(y_test, y_test_LGB2[:,1]))
mname_test.append("XGBoost")
mauc_test.append(roc_auc_score(y_test, y_test_XGB2[:,1]))
mname_test.append("lightGBM (balanciert)")
mauc_test.append(roc_auc_score(y_test, y_test_LGB2s[:,1]))
mname_test.append("XGBoost (balanciert)")
mauc_test.append(roc_auc_score(y_test, y_test_XGB2s[:,1]))
mname_test.append("LogReg L1-30 (bal.)")
mauc_test.append(roc_auc_score(y_test, y_test_LRLs[:,1]))

plot_auc_vergleich(dict_test, 0.71, 0.78, "Testdaten")
```



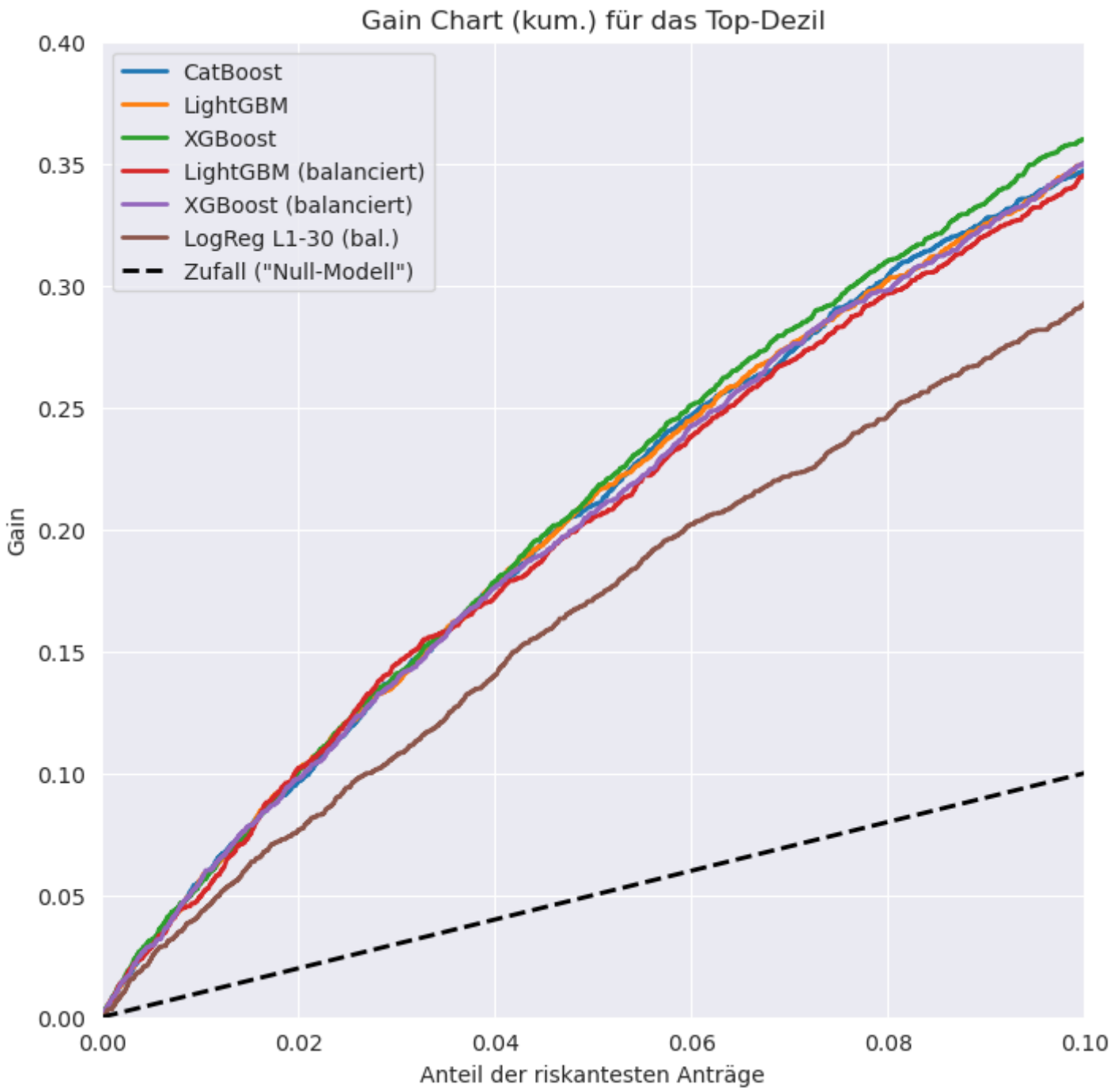
Im Vergleich zu den Validierungsdaten werden mit den Testdaten etwas höhere Prognosegüten ermittelt (bei drei Modellen ist AUC>0,770). Eine Ursache könnte eine andere Zusammensetzung der Testdaten sein, worauf auch deren etwas höhere Ereignisrate hindeutet. Kleinere Prognoseunterschiede zwischen Validierung und Test sind bei echten Testdaten (z.B. aktuellere Daten) eher typisch als untypisch.

Insgesamt werden die bisherigen AUC-Ergebnisse an den Testdaten bestätigt. Die Reihenfolge der drei besten Verfahren hat sich nicht verändert und die Prognosegüte der logistische Regression ist weiterhin rund fünf Prozentpunkte schlechter.

c) Gain- und Lift-Chart für das Top-Dezil

In [83]:
```
# Kumulative Gain Kurve vorbereiten
# scikitplot.metrics. mit Begrenzung auf die Ereignisklasse 1 nachbauen, siehe
# https://github.com/reiinakano/scikit-plot/blob/26007fbf9f05e915bd0f6acb86850b01b00944cf/scikitplot/metrics.py
y_true = np.array(np.ravel(y_test))
classes = np.unique(y_true)
percentages, gains1 = cumulative_gain_curve(y_true, np.array(y_test_CBC)[:,1], classes[1])
percentages, gains2 = cumulative_gain_curve(y_true, np.array(y_test_LGB2)[:,1], classes[1])
percentages, gains3 = cumulative_gain_curve(y_true, np.array(y_test_XGB2)[:,1], classes[1])
percentages, gains4 = cumulative_gain_curve(y_true, np.array(y_test_LGB2s)[:,1], classes[1])
percentages, gains5 = cumulative_gain_curve(y_true, np.array(y_test_XGB2s)[:,1], classes[1])
percentages, gains6 = cumulative_gain_curve(y_true, np.array(y_test_LRLs)[:,1], classes[1])
```

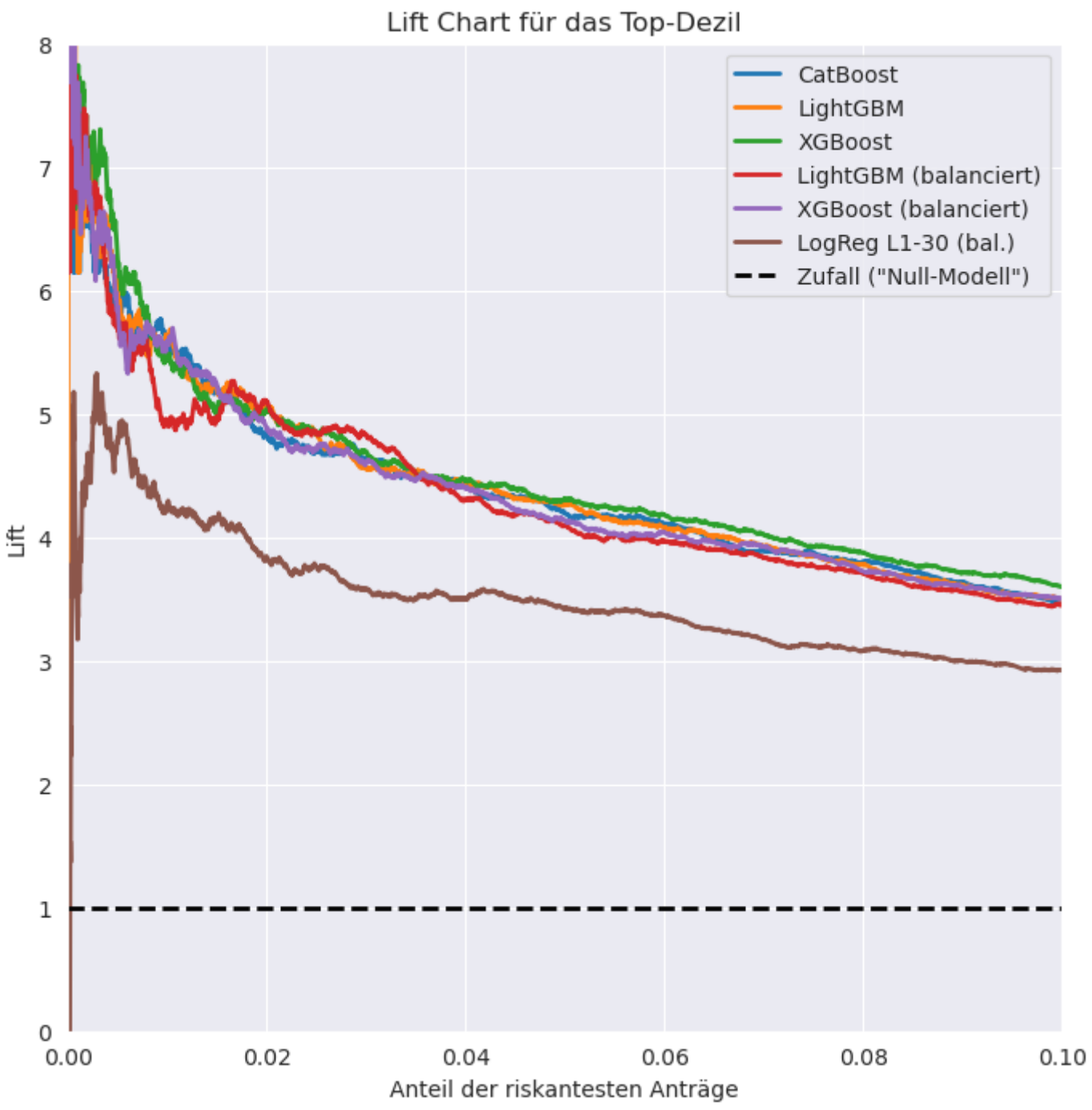
In [84]:
```
# Gain Chart erstellen
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
ax.set_title("Gain Chart (kum.) für das Top-Dezil")
ax.plot(percentages, gains1, lw=2, label='CatBoost')
ax.plot(percentages, gains2, lw=2, label='LightGBM')
ax.plot(percentages, gains3, lw=2, label='XGBoost')
ax.plot(percentages, gains4, lw=2, label='LightGBM (balanciert)')
ax.plot(percentages, gains5, lw=2, label='XGBoost (balanciert)')
ax.plot(percentages, gains6, lw=2, label='LogReg L1-30 (bal.)')
ax.plot([0, 1], [0, 1], 'k--', lw=2, label='Zufall ("Null-Modell")')
ax.set_xlabel('Anteil der riskantesten Anträge')
ax.set_ylabel('Gain')
plt.xlim(0, 0.10)
plt.ylim(0, 0.40)
ax.legend(loc='upper left')
plt.show()
```



Gain: Der kumulative Gain Chart zeigt die Klassifikationsleistung der Modelle untereinander und gegenüber einer zufälligen Klassifikation an. In der Grafik erkennt man, dass die logistische Regression klar abfällt und XGBoost bei den 5% bis 10% riskantesten Anträgen (d.h. höchste Kreditausfallwahrscheinlichkeit) eine leicht höhere Trefferquote aufweist. Die betrachteten GB-Modelle weißen bei den riskantesten 2% der Anträge eine sehr ähnliche Trefferquote von 10% auf und sind damit rund fünf Mal so gut wie eine zufällige Auswahl. Dieser sogenannte Modell-Lift kann in einem Lift-Chart dargestellt werden:

In [85]:
```
# Daten für das Lfit Chart vorbereiten (auf Basis der Kumulative Gain Kurve, s.o.)
percentages = percentages[1:]
gains1 = gains1[1:] / percentages
gains2 = gains2[1:] / percentages
gains3 = gains3[1:] / percentages
gains4 = gains4[1:] / percentages
gains5 = gains5[1:] / percentages
gains6 = gains6[1:] / percentages
```

In [86]:
```
# Lift Chart erstellen
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
ax.set_title("Lift Chart für das Top-Dezil")
ax.plot(percentages, gains1, lw=2, label='CatBoost')
ax.plot(percentages, gains2, lw=2, label='LightGBM')
ax.plot(percentages, gains3, lw=2, label='XGBoost')
ax.plot(percentages, gains4, lw=2, label='LightGBM (balanciert)')
ax.plot(percentages, gains5, lw=2, label='XGBoost (balanciert)')
ax.plot(percentages, gains6, lw=2, label='LogReg L1-30 (bal.)')
ax.plot([0, 1], [1, 1], 'k--', lw=2, label='Zufall ("Null-Modell")')
ax.set_xlabel('Anteil der riskantesten Anträge')
ax.set_ylabel('Lift')
plt.xlim(0, 0.1)
plt.ylim(0, 8.0)
ax.legend(loc='upper right')
plt.show()
```



Lift: Ein Lift-Chart zeigt, wie viel besser die Modelle im Vergleich zu einer zufälligen Klassifikation sind. Es wird dabei die Anzahl der korrekt klassifizierten Fälle im Verhältnis zur Anzahl der Fälle darstellt, die von einer zufälligen Klassifikation korrekt klassifiziert werden würden.

Im Lift Chart kann man die oben bei "Gain" getroffene Aussage, dass die betrachteten GB-Modelle bei den riskantesten 2% der Anträge eine sehr ähnliche Trefferquote von 10% aufweisen und damit rund fünf Mal so gut wie eine zufällige Auswahl sind, direkt ablesen. Zudem sind die Modellunterschiede im interessanten linken Bereich der Grafik aufgrund der Linienspreizung etwas deutlicher erkennbar.

Der Lift des parameterarmen, transparenten und nachvollziehbaren logistischen Regression mit L1-Regularisierung und 30 Koeffizienten ist deutlich schlechter als der der fünf besten Modelle. Dieser Unterschied könnte aber durch die Hinzunahme weiterer Merkmale und zielgerichtetes Feature-Engineering (z.B. Merkmalsinteraktionen) verringert werden.

d) Rechtlicher Rahmen

Die nachvollziehbare und transparente klassische logistische Regression war und ist ein Standardverfahren bei der Erstellung von Scoring-Modellen. Dies trifft auch auf regularisierte Varianten und damit auch auf das vorliegende Modell "LogReg L1-30" zu.

Die in b) und c) betrachteten fünf besten Modelle gehören hingegen alle zur Klasse der hochparametrigen Gradient-Tree-Boosting-Verfahren und zeichnen sich nicht durch hohe Nachvollziehbarkeit und Transparenz aus. Für ML-Methoden wurden spezielle Interpretierbarkeitstools entwickelt, siehe Lernziele des Vertiefungswissen Actuarial Data Science Completion.

In [87]:
```
print("LaufZeit Notebook (sec):" + "%6.0f" % (time.time() - StartNotebook))
```

```
LaufZeit Notebook (sec):  4506
```