CADS Completion 2024: Vorschlag für Aufgabenstellung (WIP, Stand 03.05.2024)

KI-Folgenabschätzung für ein Hausratprodukt mit soziodemographischen Merkmalen

105/180 Punkte

Beschreibung

Bei einem Komposit-Versicherer ist im Bereich Hausarat ein innovatives Produkt im Einsatz: Im Gegensatz zu Standard-Tarifen kommen hier u.a. Merkmale zum Einsatz, die im weiteren Sinne als soziodemographisch bzw. kontextuell betrachtet werden können. Das Produkt performt gut und ist beliebt, doch wirft der Artificial Intelligence Act seine Schatten voraus: Nicht nur ist unklar, wie dieser in der Rechtssystematik zum VAG stehen wird, konkret ist auch fraglich, ob die Verwendung von Merkmalen, die im Sinne des Allgemeinen Gleichbehandlungsgesetzes (AGG) als diskriminierend designiert sind, weiterhin bei signifikantem Schadeneinfluss zur Verwendung freigegeben sind.

Zu diesem Zweck soll eine Workforce gestartet werden, die eine **KI-Folgenabschätzung** für dieses Produkt vornehmen soll. Als Teil dieser Workforce sind sowohl **Data Analytics**-Themen zu behandeln, als auch **regulatorische** (AI Act) als auch **datenschutzrechtliche** (DSGVO) Aspekte zu betrachten.

Aufgabenstellung

A1 (10 Punkte): Verschaffen Sie sich einen Überblick über Bestand und Tarifgestaltung. Geben Sie eine deskriptive Übersicht über die vorhandenen Daten und formulieren Sie Hypothesen dazu, welche Problemfelder zu betrachten sein werden.

A2 (10 Punkte): Betrachten Sie die beiden Merkmale `Geschlecht` und `Alter`. Sind diese siginifikante Prädiktoren im Tarif? Trainieren Sie ein Tarifmodell, das den Schadenbedarf `Response` fittet und bewerten Sie den Impact, wenn die Merkmale entfernt werden müssten.

A3 (15 Punkte): Beschreiben Sie die Herausforderungen von indirekter Diskriminierung über Ersatzmerkmale und identifizieren Sie, ob es Proxy-Variablen für `Geschlecht` und `Alter` im Datensatz gibt. Evaluieren Sie für mindestens ein L1/L2-Regularisiertes Modell den **Lambda**-Pfad - ist diese Betrachtung hilfreich?

A4 (20 Punkte): Trainieren Sie ein Modell des Typs ***localGLMnet*** [2],[3] und vergleichen Sie es mit dem Modellen aus Aufgabe 2 und 3. Prüfen Sie mit den in [3] vorgeschlagenen Attention-Methoden und Shapley Value Contributions die funktionalen Zusammenhänge und vergleichen Sie diese mit den Modellen aus A3.

A5 (35 Punkte): Trainieren Sie mindestens zwei Modelle, die die indirekte Diskriminierung beseitigen können und bewerten Sie, ob und wie gut dies gelingt. Konzentrieren Sie sich auf die Vermeidung von indirekter Diskriminierung beim Merkmal `Age`. Vergleichen Sie dazu die Basisprämie mit der korrigierten prämie sowohl auf Einzelvertragsebene als auch auf einer geeigneten aggregierten Ebene.

Verändern Sie für Ihr `localGLMnet` die Loss-Funktion auf geeignete Weise, sodass das Modell diskriminierungsmindernd wirkt, *wenn möglich auch gegen indirekte Diskriminierung*.

Untersuchen Sie Ihre Ansätze mit geeigneten Visualisierungen und Erklärbarkeitsansätzen, und diskutieren Sie die Erklärbarkeit der Methoden hinsichtlich der Frage nach direkter und indirekter Diskriminierung.

Betrachten Sie abschließend, wie der Tarif unter den hypothetischen Rahmenbedingungen, dass keinerlei Korrelation mit geschützten Merkmalen im Tarif vorhanden sein darf, aussehen könnte und welche Herausforderungen sich im Vergleich ergeben. Betrachten Sie insbesondere Langzeitfolgen von Moral Hazard und Adverser Selektion in einem kompetitiven Marktumfeld. Beschreiben Sie einen vergleichenden Business Case, der strategische und formale Empfehlungen abgibt.

A6 (15 Punkte): Fassen Sie Ihre Erkenntnisse in einem Bericht von maximal zwei DinA4-Seiten zusammen, wovon 1/2 Seite Management Summary sein soll. Dies kann am Ende des abgegebenen Notebooks oder als separates PDF erfolgen.

Literaturhinweise:

[1] Lindholm M, Richman R, Tsanakas A, Wüthrich MV. DISCRIMINATION-FREE INSURANCE PRICING. ASTIN Bulletin. 2022;52(1):55-89. doi:10.1017/asb.2021.23
[2] Richman, R, Wüthrich, MV (2023), 'LocalGLMnet: interpretable deep learning for tabular data', Scandinavian Actuarial Journal, 2023(1), pp. 71–95. doi: 10.1080/03461238.2022.2081816.
[3] Transchel, F.: "Simpsons Paradoxon oder Die Tücken des diskriminierungsfreien Pricings" (2023), Vortrag, DAV-Herbsttagung 2023
[4] Knoop, P.: pyLocalGLMnet (2023), https://github.com/PK1706/pyLocalGLMnet

In [1]:
```
# Base tools

import os
import random
import pandas as pd
import numpy as np
import math
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Modelling

from scipy import interpolate
import scipy.stats as stats
import tensorflow as tf
from keras.optimizers import Adam
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
from sklearn.base import BaseEstimator
from sklearn.utils.validation import check_X_y, check_array, check_is_fitted
from sklearn.utils.multiclass import unique_labels
from sklearn import metrics
from sklearn.metrics import euclidean_distances

# Plotting

import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
import matplotlib.patches as patches
import plotly.express as px
import plotly.figure_factory as ff
import plotly.graph_objects as go
import seaborn as sns
from labellines import labelLines
import kaleido

# Inspection

from sklearn.inspection import partial_dependence
from sklearn.inspection import PartialDependenceDisplay
import shap

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=UserWarning)
```

Aufgabe 1

Aufgabenstellung: Verschaffen Sie sich einen Überblick über Bestand und Tarifgestaltung. Geben Sie eine deskriptive Übersicht über die vorhandenen Daten und formulieren Sie Hypothesen dazu, welche Problemfelder zu betrachten sein werden.

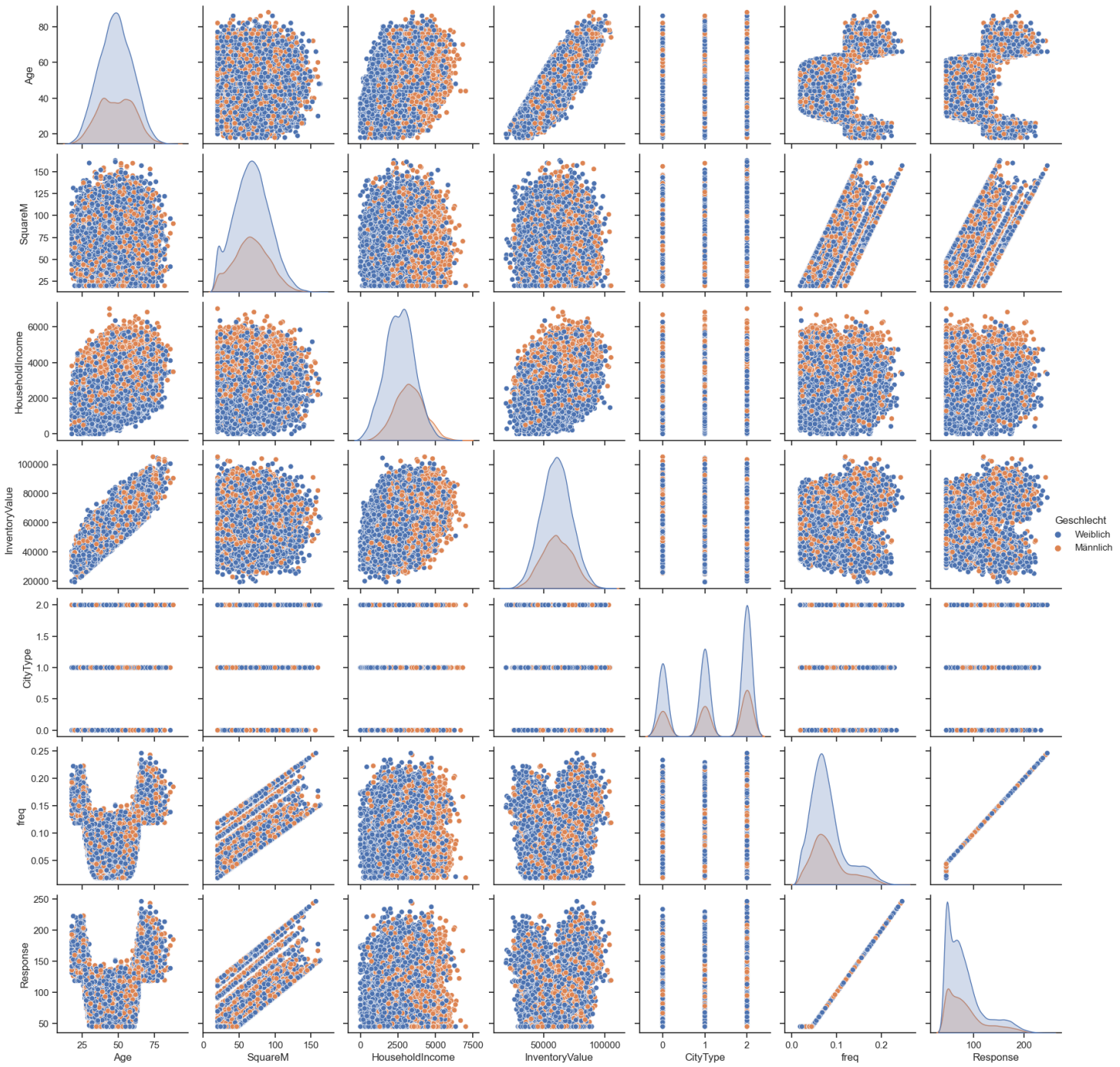
In [2]:
```
data_df = pd.read_csv("daten_teil_a.csv")
data_df = data_df.drop("Unnamed: 0",axis=1)
```

In [3]:
```
data_df.describe()
```

Out[3]:

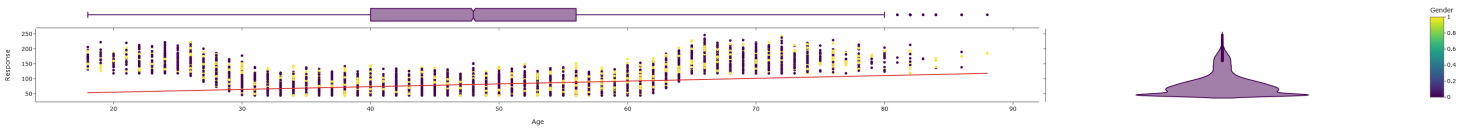
	Gender	Age	SquareM	HouseholdIncome	InventoryValue	CityType	freq	Response
count	25000.000000	25000.000000	25000.000000	25000.000000	25000.000000	25000.00000	25000.000000	25000.000000
mean	0.294480	48.308560	67.297495	2809.448877	60507.832168	1.20092	0.079821	81.803514
std	0.455818	11.481755	24.446981	1020.819194	12146.142550	0.81271	0.039671	37.368187
min	0.000000	18.000000	20.000000	0.000000	19286.747349	0.00000	0.017661	45.000000
25%	0.000000	40.000000	50.272735	2094.143615	52172.510391	1.00000	0.052946	52.945724
50%	0.000000	48.000000	66.955000	2810.773562	60408.880557	1.00000	0.071610	71.609699
75%	1.000000	56.000000	83.892950	3487.983953	68856.658097	2.00000	0.095986	95.986448
max	1.000000	88.000000	162.744554	7025.897705	105248.191167	2.00000	0.246373	246.372506

In [4]:
```
sns.set_theme(style="ticks")
p = sns.pairplot(data_df, hue="Gender")
p._legend.set_title('Geschlecht')
new_labels = ['Weiblich', 'Männlich']
for t, l in zip(p._legend.texts, new_labels):
    t.set_text(l)
```

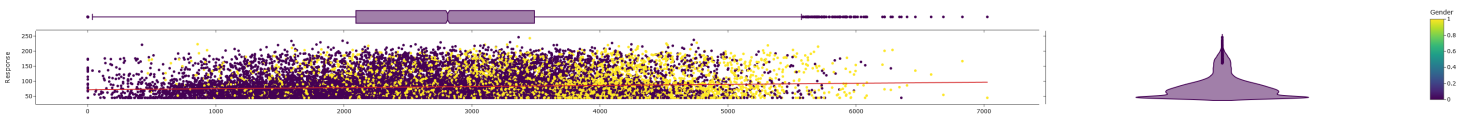


In [5]:
```
p.savefig("A1_pairplot.png")
```

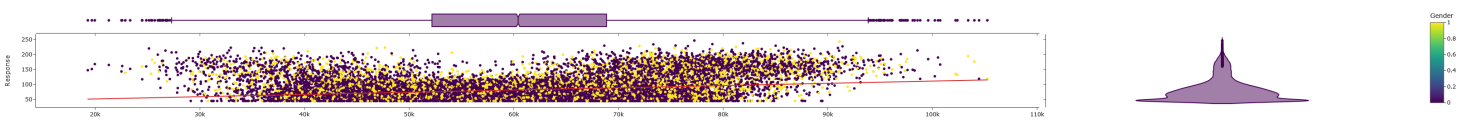
In [133…
```
fig = px.scatter(data_df, x="Age", y="Response", color="Gender", marginal_y="violin",
           marginal_x="box", trendline="ols", template="simple_white");fig.show(renderer='vscode')
```



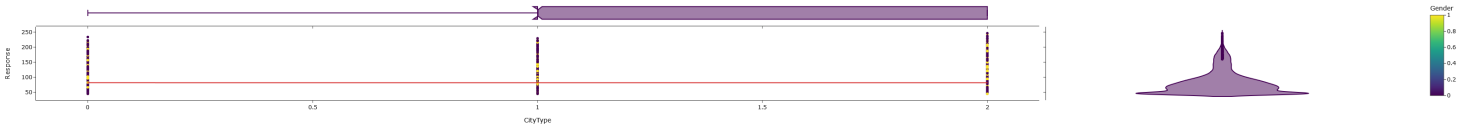
In [132…
```
fig = px.scatter(data_df, x="HouseholdIncome", y="Response", color="Gender", marginal_y="violin",
           marginal_x="box", trendline="ols", template="simple_white");fig.show(renderer='vscode')
```



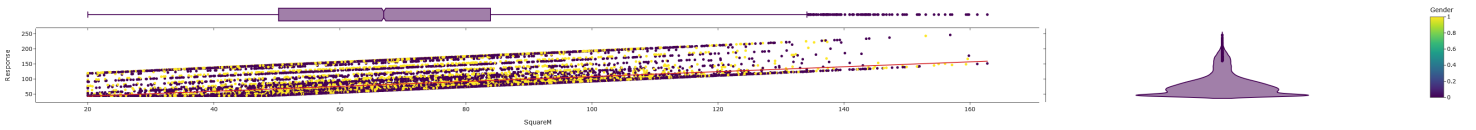
In [134…
```
fig = px.scatter(data_df, x="InventoryValue", y="Response", color="Gender", marginal_y="violin",
           marginal_x="box", trendline="ols", template="simple_white");fig.show(renderer='vscode')
```



In [135…
```
fig = px.scatter(data_df, x="CityType", y="Response", color="Gender", marginal_y="violin",
           marginal_x="box", trendline="ols", template="simple_white");fig.show(renderer='vscode')
```



In [136…
```
fig = px.scatter(data_df, x="SquareM", y="Response", color="Gender", marginal_y="violin",
           marginal_x="box", trendline="ols", template="simple_white");fig.show(renderer='vscode')
```



Antwort A1:

Zu betrachten ist ein Bestand eines Hausrattarifs, der die Merkmale `Gender`, `Age`, `SquareM`, `HouseholdIncome`, `InventoryValue` und `CityType` als Variablen aufweist. Interaktionen sind vorstellbar; der Pairplot deutet auf vielfältige Interaktionen mit Age hin. Die Spalte `freq` beschreibt die Schadenfrequenz und `Response` den mittleren(?) Schadenbedarf. Die Merkmale `Gender` und `Age` dürften per EU AI Act als diskriminierend eingestuft werden.

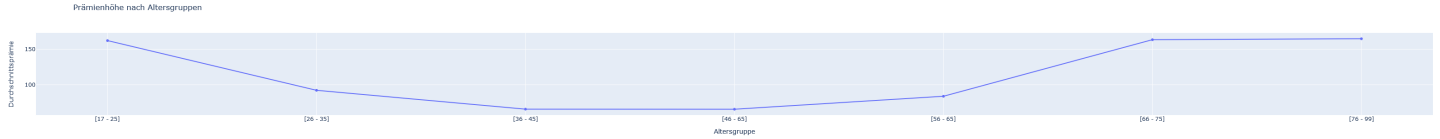
Aufgabe 2

Aufgabenstellung: Betrachten Sie die beiden Merkmale `Geschlecht` und `Alter`. Sind diese siginifikante Prädiktoren im Tarif? Trainieren Sie ein Tarifmodell und bewerten Sie den Impact, wenn die Merkmale entfernt werden müssten.

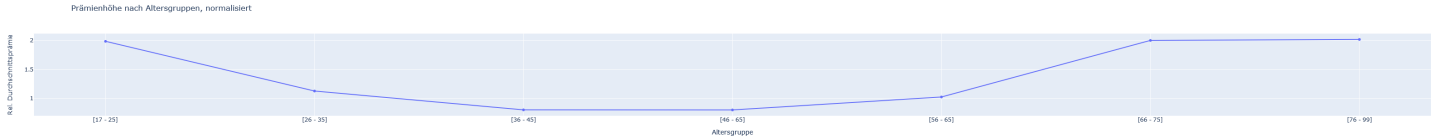
Zur Beantwortung dieser Fragestellung soll folgendes Vorgehen durchgeführt werden:

1. Prämie nach Altersgruppen plotten.
2. Prämie nach Geschlecht plotten.
3. Basismodell mit Alter + Gender bauen.
4. Basismodell ohne Alter + Gender bauen.
5. Impact auswerten.

In [137…
```
# Prämie nach Altersgruppen
# Wir wollen die durchschnittliche Prämie pro Altersgruppe ausrechnen.
avg_premium = data_df['Response'].mean()
# Einteilung in Altersgruppen
def set_age_group(age):
        if age < 25:
            return "[17 - 25]"
        if age < 35:
            return "[26 - 35]"
        if age < 45:
            return "[36 - 45]"
        if age < 55:
            return "[46 - 65]"
        if age < 65:
            return "[56 - 65]"
        if age < 75:
            return "[66 - 75]"
        else:
            return "[76 - 99]"
# Zuordnung der Altersgruppen
data_df['AgeGroup'] = data_df['Age'].apply(set_age_group)
# Ableiten der gruppierten Response-Werte
prem_data = data_df.groupby(['AgeGroup'])['Response'].mean()#/avg_premium
prem_data_ratio = data_df.groupby(['AgeGroup'])['Response'].mean()/avg_premium

# Plotten
fig = go.Figure()
fig.add_trace(go.Scatter(x=np.sort(data_df['AgeGroup'].unique()), y=prem_data, mode="lines+markers", name='Prämie nach Altersgruppen'))

fig.update_layout(title='Prämienhöhe nach Altersgruppen',
                  xaxis_title='Altersgruppe',
                  yaxis_title='Durchschnittsprämie')

fig.show(renderer='vscode')
```

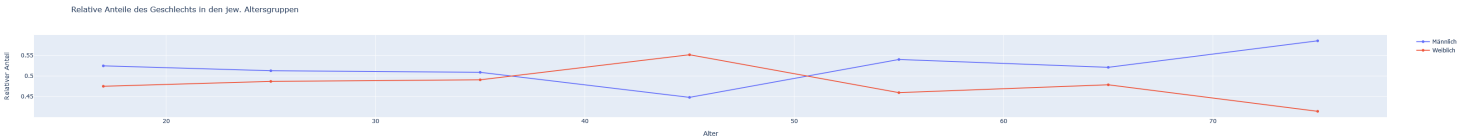


In [138…
```
# Plotten
fig = go.Figure()
fig.add_trace(go.Scatter(x=np.sort(data_df['AgeGroup'].unique()), y=prem_data_ratio, mode="lines+markers", name='Prämie nach Altersgruppen, normalisiert'))

fig.update_layout(title='Prämienhöhe nach Altersgruppen, normalisiert',
                  xaxis_title='Altersgruppe',
                  yaxis_title='Rel. Durchschnittsprämie')

fig.show(renderer='vscode')
```



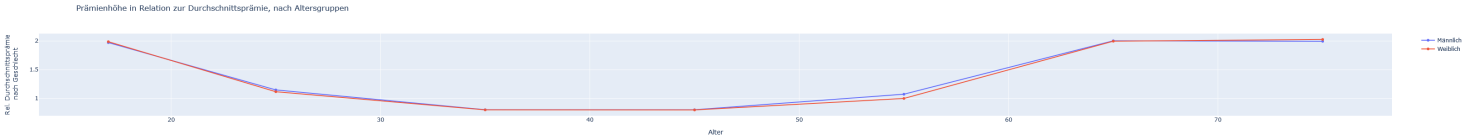
In [139…

```python
# Prämie nach Geschlecht getrennt plotten
age_groups = [17,25,35,45,55,65,75,101]

age_data_male = data_df.query('Gender > 0')['Age']
age_data_female = data_df.query('Gender < 1')['Age']

# Histogramme nach Geschlecht
hist_male, edges_male = np.histogram(age_data_male, bins=age_groups,density = True)
hist_female, edges_female = np.histogram(age_data_female, bins=age_groups,density = True)

# Normalisieren
hist_male_scaled = hist_male.astype(float) / np.sum(hist_male)
hist_female_scaled = hist_female.astype(float) / np.sum(hist_female)

rel_proportions = np.array(list(zip(hist_male_scaled,hist_female_scaled)))
rel_props = np.sum(rel_proportions,axis=1) + 0.0000001
hist_male_scaled_props = hist_male_scaled / rel_props
hist_female_scaled_props = hist_female_scaled / rel_props

# Plotten
fig = go.Figure()
fig.add_trace(go.Scatter(x=edges_male[:-1], y=hist_male_scaled_props, mode="lines+markers", name='Männlich'))
fig.add_trace(go.Scatter(x=edges_female[:-1], y=hist_female_scaled_props, mode='lines+markers', name='Weiblich'))

fig.update_layout(title='Relative Anteile des Geschlechts in den jew. Altersgruppen',
                  xaxis_title='Alter',
                  yaxis_title='Relativer Anteil')

fig.show(renderer='vscode')
```



In [140…

```python
# Wir wollen die durchscnittliche Prämie pro Altersgruppe ausrechnen.

#age_groups = [17,25,35,45,55,65,75,101]

def set_age_group(age):
        if age < 25:
            return "[17 - 25]"
        if age < 35:
            return "[26 - 35]"
        if age < 45:
            return "[36 - 45]"
        if age < 55:
            return "[46 - 65]"
        if age < 65:
            return "[56 - 65]"
        if age < 75:
            return "[66 - 75]"
        else:
            return "[76 - 99]"

data_df['AgeGroup'] = data_df['Age'].apply(set_age_group)

#print(data['AgeGroup'])

avg_premium = data_df['Response'].mean()

data_male = data_df.query('Gender > 0')
data_female = data_df.query('Gender < 1')
prem_data_male = data_male.groupby(['Gender','AgeGroup'])['Response'].mean()/avg_premium
prem_data_female = data_female.groupby(['Gender','AgeGroup'])['Response'].mean()/avg_premium

fig = go.Figure()
fig.add_trace(go.Scatter(x=age_groups, y=prem_data_male.iloc[:], mode="lines+markers", name='Männlich'))
fig.add_trace(go.Scatter(x=age_groups, y=prem_data_female.iloc[:], mode="lines+markers", name='Weiblich'))

fig.update_layout(title='Prämienhöhe in Relation zur Durchschnittsprämie, nach Altersgruppen',
                  xaxis_title='Alter',
                  yaxis_title='Rel. Durchschnittsprämie<br> nach Geschlecht')

fig.show(renderer='vscode')
```



Auf Basis der deskriptiven Analyse bis hierhin ist es sehr vernünftig anzunehmen, dass zumindest das Alter eine Rolle bei der Preisfindung spielt. Wir wollen dies über die Erstellung eines Basismodells und die Analyse mittels Permutation Importance validieren.

In [15]:

```
#Erstellung Basismodelle
X = data_df.copy().drop(['AgeGroup','freq','Response'],axis=1)
y = data_df['Response'].copy()

scaler_X = StandardScaler()
print(scaler_X.fit(X))
#print(scaler_X.mean_)
X_scaled = scaler_X.transform(X)

scaler_y = StandardScaler()
print(scaler_y.fit(y.values.reshape(-1, 1)))
#print(scaler_y.mean_)
y_scaled = scaler_y.transform(y.values.reshape(-1, 1))

x_train, x_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=42)

## LM-Training

lm = LinearRegression(fit_intercept=True)
lm.fit(x_train,y_train)
print(" ")
print("Ermittelte Koeffizienten für Basis-Regression:",list(zip(X.columns,lm.coef_)))
print("Intercept:",lm.intercept_)

# Berechne mittleren Preis

lm_pred = lm.predict(X_scaled)
prediction_w_gender = lm_pred
lm_mean_price = np.mean(lm_pred)
print("Mittlere Prämie: ",lm_mean_price)

# Berechnen Preis für Männer

lm_index_gender_male = data_df[data_df['Gender'] > 0].index
lm_mean_price_male = np.mean(lm_pred[lm_index_gender_male])
print("Prämie (Männer): ",np.round(lm_mean_price_male,2))

# Berechne Preis für Frauen

lm_index_gender_female = data_df[data_df['Gender'] < 1].index
lm_mean_price_female = np.mean(lm_pred[lm_index_gender_female])
print("Prämie (Frauen): ",np.round(lm_mean_price_female,2))

### Lasso-Training

lasso = Lasso(alpha=0.1, fit_intercept=True)
lasso.fit(x_train,y_train)
print(" ")
print("Ermittelte Koeffizienten per Lasso-Regularisierung:",list(zip(X.columns,lasso.coef_)))
print("Intercept:",lasso.intercept_)

# Berechne mittleren Preis

lasso_pred = lasso.predict(X_scaled)
prediction_w_gender = lasso_pred
lasso_mean_price = np.mean(lasso_pred)
print("Mittlere Prämie: ",lasso_mean_price)

# Berechnen Preis für Männer

lasso_index_gender_male = data_df[data_df['Gender'] > 0].index
lasso_mean_price_male = np.mean(lasso_pred[lasso_index_gender_male])
print("Prämie (Männer): ",np.round(lasso_mean_price_male,2))

# Berechne Preis für Frauen

lasso_index_gender_female = data_df[data_df['Gender'] < 1].index
lasso_mean_price_female = np.mean(lasso_pred[lasso_index_gender_female])
print("Prämie (Frauen): ",np.round(lasso_mean_price_female,2))
```

```
StandardScaler()
StandardScaler()

Ermittelte Koeffizienten für Basis-Regression: [('Gender', 1.589507129529567), ('Age', 10.394535412630486), ('SquareM', 19.892283940535716), ('HouseholdIncome', -0.29065481583911507), ('InventoryValue',
0.14319017225471234), ('CityType', 0.09152088038244507)]
Intercept: 81.89876394537775
Mittlere Prämie:  81.89876394537775
Prämie (Männer):  84.39
Prämie (Frauen):  80.86

Ermittelte Koeffizienten per Lasso-Regularisierung: [('Gender', 1.4360377552073618), ('Age', 10.294218579196361), ('SquareM', 19.792888930995915), ('HouseholdIncome', -0.09119831890145014), ('InventoryVal
ue', 0.074228533577506), ('CityType', 0.0)]
Intercept: 81.89899123516776
Mittlere Prämie:  81.89899123516776
Prämie (Männer):  84.24
Prämie (Frauen):  80.92
```

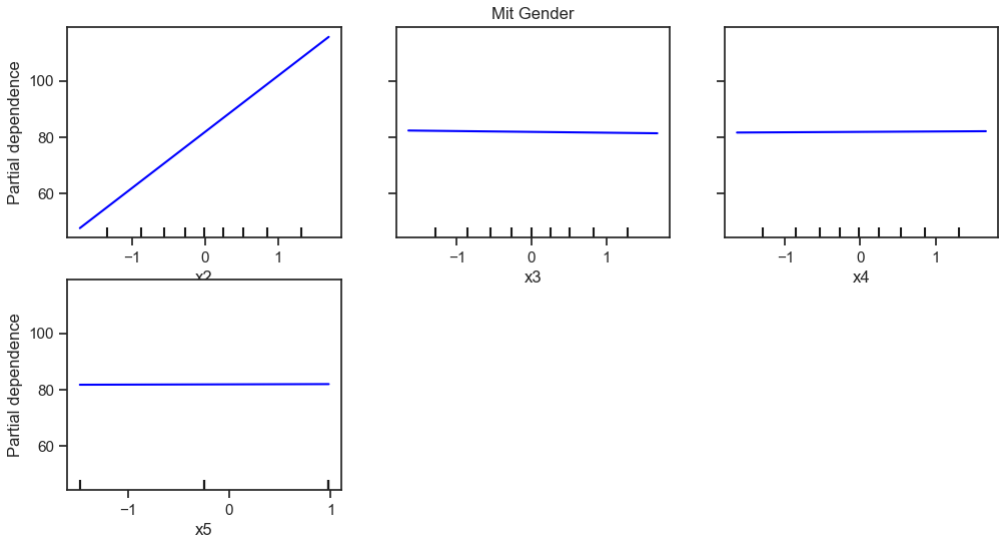
Zwischenbemerkung: Man kann bereits durch den Vergleich zum Lasso-Modell sehen, dass große Kovarianz zwischen `Alter` (größter Koeffizient im Modell) und fast allen anderen Variablen besteht. Die Vermutung, dass hier bei Entfernen des Merkmals eine indirekte Diskriminierung vorliegen könnte, drängt sich also förmlich auf.

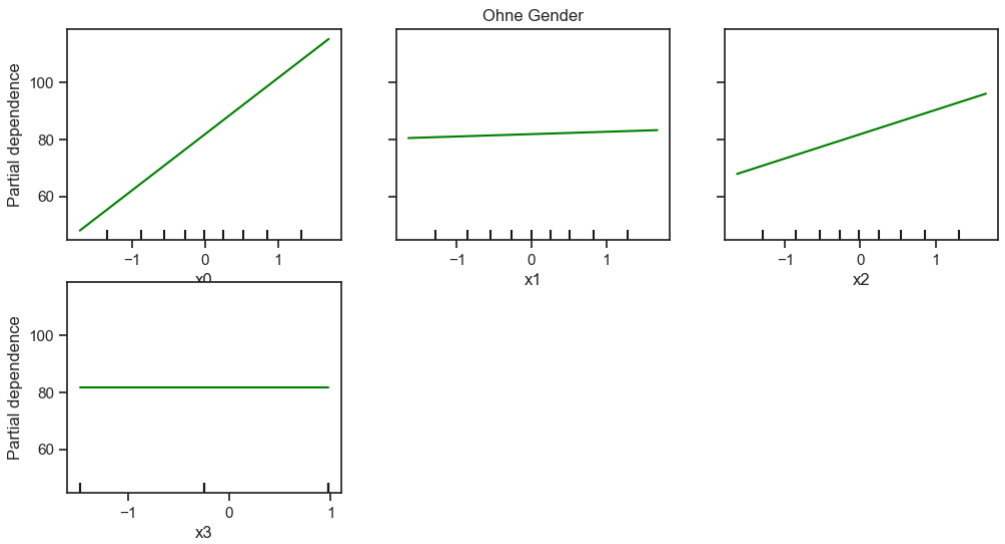
Als nächstes wird ein Modell ohne die Merkmale `Age` und `Gender` trainiert und anschließend die Prämien nach Geschlecht und Altersgruppen getrennt ausgewertet sowie Partial Dependence Plots erstellt.

In [16]:
```
#Erstelle Modell ohne Alter und Gender
#Erstellung Basismodelle
X_wo_ageandgender = x_train[:,2:]
y_wo_ageandgender = y_train
columns_wo_agegender = X.columns[2:]

lasso_wo_ageandgender = Lasso(alpha=0.1, fit_intercept=True)
lasso_wo_ageandgender.fit(X_wo_ageandgender,y_wo_ageandgender)
print("Ermittelte Koeffizienten:",list(zip(columns_wo_agegender,lasso_wo_ageandgender.coef_)))
print("Intercept:",lasso_wo_ageandgender.intercept_)

# Berechne mittleren Preis

lasso_pred_wo_ageandgender = lasso_wo_ageandgender.predict(X_wo_ageandgender)
prediction_wo_ageandgender = lasso_pred_wo_ageandgender
lasso_mean_price_wo_ageandgender = np.mean(lasso_pred_wo_ageandgender)
print("Mittlere Prämie: ",lasso_mean_price_wo_ageandgender)

# Berechnen Preis für Männer

lasso_index_gender_male_wo_ageandgender = lasso_pred_wo_ageandgender[x_train[:,0] > 0]
lasso_mean_price_male_wo_ageandgender = np.mean(lasso_index_gender_male_wo_ageandgender)
print("Prämie (Männer): ",np.round(lasso_mean_price_male_wo_ageandgender,4))

# Berechne Preis für Frauen

lasso_index_gender_female_wo_ageandgender = lasso_pred_wo_ageandgender[x_train[:,0] < 1]
lasso_mean_price_female_wo_ageandgender = np.mean(lasso_index_gender_female_wo_ageandgender)
print("Prämie (Frauen): ",np.round(lasso_mean_price_female_wo_ageandgender,4))
```

```
Ermittelte Koeffizienten: [('SquareM', 19.774015654928196), ('HouseholdIncome', 0.8309176507428081), ('InventoryValue', 8.500632697570559), ('CityType', 0.0)]
Intercept: 81.85754800601418
Mittlere Prämie:  81.89971508342909
Prämie (Männer):  82.6317
Prämie (Frauen):  81.5951
```

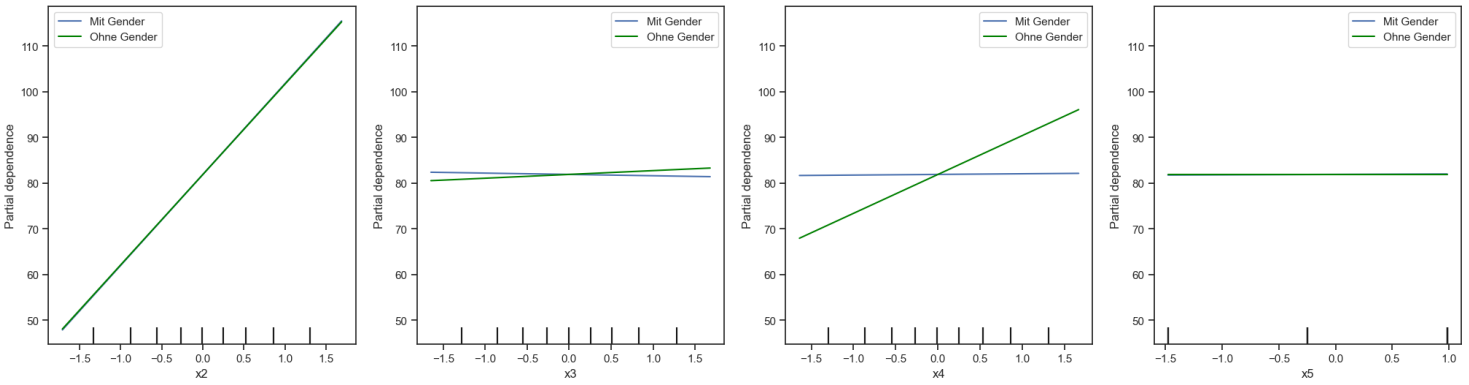
In [17]:
```
# Partial Dependence für Alter und Geschlecht
fig, ax = plt.subplots(figsize=(12, 6));
ax.set_title("Mit Gender")
disp_wi_gender = PartialDependenceDisplay.from_estimator(lm, x_train, [2,3,4,5], ax=ax, line_kw={"color": "blue"})
fig, ax = plt.subplots(figsize=(12, 6));
ax.set_title("Ohne Gender")
disp_wo_ageandgender = PartialDependenceDisplay.from_estimator(lasso_wo_ageandgender,X_wo_ageandgender, features = [0,1,2,3],ax = ax, line_kw={"color": "green"})
```





In [18]:
```
# Plots zusammensetzen
%matplotlib inline
fig, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize=(25, 6))
disp_wi_gender.plot(ax=[ax1, ax2, ax3, ax4], line_kw={"label": "Mit Gender"})
disp_wo_ageandgender.plot(ax=[ax1, ax2, ax3, ax4], line_kw={"label": "Ohne Gender", "color": "green"})
ax1.legend()
ax2.legend()
ax3.legend()
ax4.legend()
```

Out[18]:
```
<matplotlib.legend.Legend at 0x1e36a0fab90>
```



Ein wichtiger Aspekt an dieser Darstellung ist die Steigung (d.h. lineare partielle Abhängigkeit) der Merkmale x3= `HouseholdIncome` und x4= `InventoryValue` . Hier zeigt sich, dass eine Entfernung von *direkt* diskriminierenden Merkmalen eine Änderung proportional zur Kovarianz bedeutet, also im Umkehrschluss, dass x3= `HouseholdIncome` und x4= `InventoryValue` indirekt diskrimieren dürften.

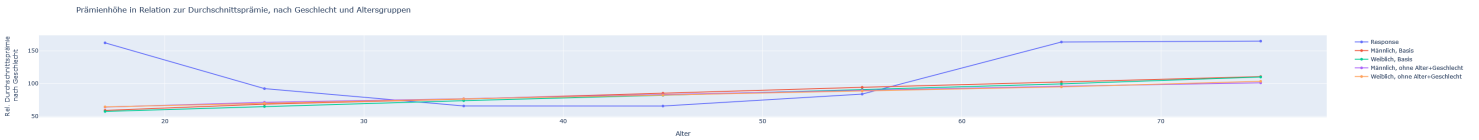
In [19]:
```
# Vorhersagen machen und als neue Spalte anfügen: Basis
data_df_pred = data_df.copy()
data_df_pred['PredBasis'] = lasso.predict(X_scaled)
data_df_pred['PredWoAgeGender'] = lasso_wo_ageandgender.predict(X_scaled[:,2:])
pd.DataFrame(data_df_pred.groupby('AgeGroup').mean())
```

Out[19]:

AgeGroup	Gender	Age	SquareM	HouseholdIncome	InventoryValue	CityType	freq	Response	PredBasis	PredWoAgeGender
[17 - 25]	0.315186	22.223496	66.525004	2103.774358	37037.708928	1.157593	0.162260	162.259860	57.870794	64.232100
[26 - 35]	0.305055	30.796748	67.086565	2295.181355	44847.646525	1.210675	0.091329	92.280939	66.010857	70.308126
[36 - 45]	0.301800	40.095667	67.338300	2591.436957	53272.623630	1.216662	0.063007	65.746978	74.566756	76.649341
[46 - 65]	0.253001	49.662063	67.151378	2827.822404	61643.758426	1.195043	0.062796	65.637732	82.868846	82.549322
[56 - 65]	0.328710	59.264409	67.351480	3144.489437	70292.625601	1.186780	0.082644	83.888002	91.903312	89.022083
[66 - 75]	0.312077	68.261851	67.885857	3362.521190	78329.847563	1.197517	0.163526	163.526404	100.380216	95.256864
[76 - 99]	0.370558	77.761421	69.269816	3557.998628	86941.790581	1.274112	0.164874	164.874087	110.237359	102.562721

In [141…

```
# Wir wollen die durchschnittliche Prämie pro Altersgruppe für die beiden erstellten Modelle ausrechnen.

avg_premium = data_df_pred['Response'].mean()
data_age_groups = data_df_pred.groupby('AgeGroup')
female_agg = data_df_pred[data_df_pred['Gender'] == 0].groupby("AgeGroup").mean()
male_agg = data_df_pred[data_df_pred['Gender'] == 1].groupby("AgeGroup").mean()
prem_data = (data_df_pred.groupby('AgeGroup').mean())['Response']
male_agg_pred_basis = male_agg['PredBasis']
male_agg_pred_woagegender = male_agg['PredWoAgeGender']
female_agg_pred_basis = female_agg['PredBasis']
female_agg_pred_woagegender = female_agg['PredWoAgeGender']

fig = go.Figure()
fig.add_trace(go.Scatter(x=age_groups, y=prem_data, mode="lines+markers", name='Response'))
fig.add_trace(go.Scatter(x=age_groups, y=male_agg_pred_basis, mode="lines+markers", name='Männlich, Basis'))
fig.add_trace(go.Scatter(x=age_groups, y=female_agg_pred_basis, mode="lines+markers", name='Weiblich, Basis'))
fig.add_trace(go.Scatter(x=age_groups, y=male_agg_pred_woagegender, mode="lines+markers", name='Männlich, ohne Alter+Geschlecht'))
fig.add_trace(go.Scatter(x=age_groups, y=female_agg_pred_woagegender, mode="lines+markers", name='Weiblich, ohne Alter+Geschlecht'))

fig.update_layout(title='Prämienhöhe in Relation zur Durchschnittsprämie, nach Geschlecht und Altersgruppen',
                  xaxis_title='Alter',
                  yaxis_title='Rel. Durchschnittsprämie<br> nach Geschlecht')

fig.show(renderer='vscode')
```



Aus den bis hierher betrachteten Auswertungen lässt sich erkennen, dass der funktionale Zusammenhang mit `Age` höchstwahrscheinlich nicht linear ist, sondern am ehesten wie Age^2 zu beschreiben wäre. Diese Interaktion soll probehalber in ein Modell aufgenommen werden. Dazu wird ein neuer Dataframe mit dieser zusätzlichen Merkmalsspalte erzeugt.

In [21]:

```
data_df_ageSQ = data_df.copy().drop(["freq","Response","AgeGroup"],axis=1)
data_df_ageSQ['ageSQ'] = np.array(list(map(lambda x:x*x,data_df['Age'])))
x_train_ageSQ, x_test_ageSQ, y_train_ageSQ, y_test_ageSQ = train_test_split(data_df_ageSQ, y, test_size=0.25, random_state=42)
lasso_ageSQ = Lasso(alpha=0.1, fit_intercept=True)
lasso_ageSQ.fit(x_train_ageSQ,y_train_ageSQ)
print("Ermittelte Koeffizienten:",list(zip(data_df_ageSQ.columns,lasso_ageSQ.coef_)))
print("Intercept:",lasso_ageSQ.intercept_)
```

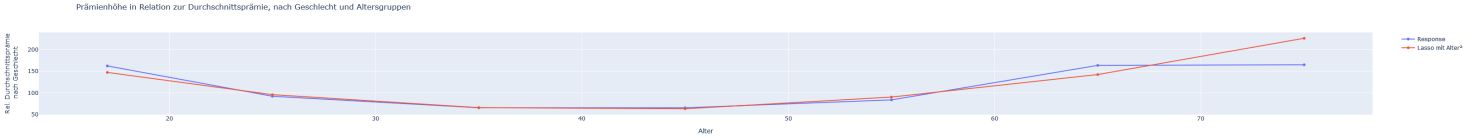
```
Ermittelte Koeffizienten: [('Gender', 0.5962308775272142), ('Age', -14.534294495675706), ('SquareM', 0.8053686852920783), ('HouseholdIncome', 4.590234511089787e-05), ('InventoryValue', 3.456196592575714
e-06), ('CityType', 0.0), ('ageSQ', 0.15902441536119205)]
Intercept: 337.1462735978661
```

In [22]:

```
data_df_pred['PredAgeSQ'] = lasso_ageSQ.predict(data_df_ageSQ)
```

In [142…

```
# Wir wollen die durchschnittliche Prämie pro Altersgruppe für die beiden erstellten Modelle ausrechnen.

fig = go.Figure()
fig.add_trace(go.Scatter(x=age_groups, y=prem_data, mode="lines+markers", name='Response'))
fig.add_trace(go.Scatter(x=age_groups, y=(data_df_pred.groupby("AgeGroup").mean())['PredAgeSQ'], mode="lines+markers", name='Lasso mit Alter²'))

fig.update_layout(title='Prämienhöhe in Relation zur Durchschnittsprämie, nach Geschlecht und Altersgruppen',
                  xaxis_title='Alter',
                  yaxis_title='Rel. Durchschnittsprämie<br> nach Geschlecht')

fig.show(renderer='vscode')
```



Antwort A2: Aus den Koeffizienten für die linearen Modelle (respektive mit und ohne die Merkmale `Gender` und `Alter`) sowie den Partial Dependence Plots ist zu erkennen, dass zum einen die Entfernung der Merkmale den Spreiz bezüglich des Geschlechts erheblich (aber nicht vollständig) reduziert und zum anderen Unterschiede, also indirekte Diskriminierung bezüglich des Merkmals `Alter` bestehen bleibt. Letzteres würde noch wesentlich deutlicher vorhanden sein, wenn der nichtlineare Anteil der Altersstruktur explizit modelliert würde.

Aufgabe 3

Aufgabenstellung: Beschreiben Sie die Herausforderungen von indirekter Diskriminierung über Ersatzmerkmale und identifizieren Sie, ob es Proxy-Variablen für `Geschlecht` und `Alter` im Datensatz gibt.

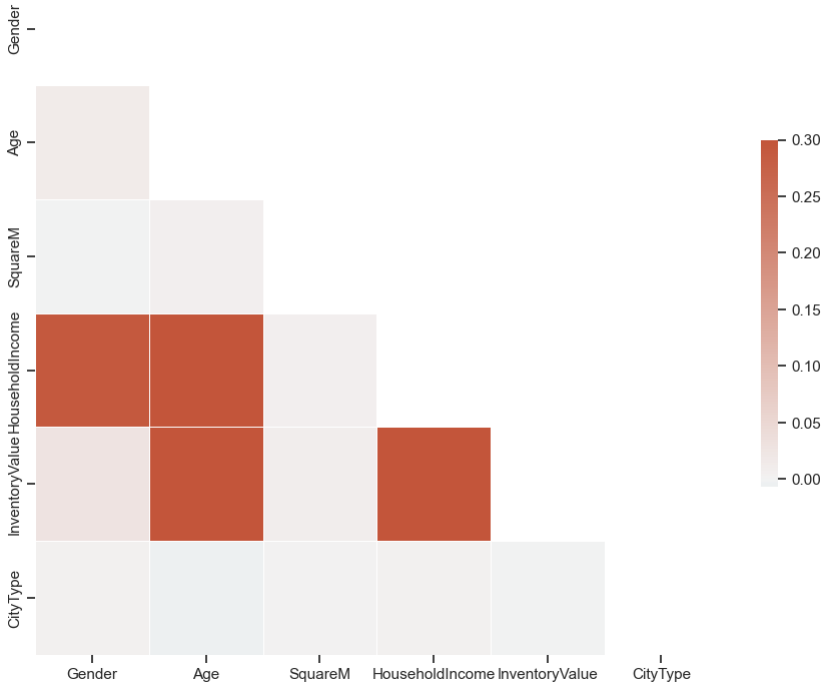
Indirekte Diskriminierung beschreibt das Charakteristikum von Modellen, dass bezüglich geschützter Merkmale (per z.B. **Allgemeines Gleichbehandlungsgesetz (AGG)** oder der **europ. Menschenrechtskonvention**) wie etwa `politische Orientierung`, `Alter`, `Geschlecht`, `Sexuelle Orientierung` etc. Ungleichbehandlungen auftreten können, auch wenn die fraglichen Merkmale nicht in das Modell aufgenommen werden.

Im vorliegenden Fall ist naheliegend zu vermuten, dass bspw. die Merkmale `HouseholdIncome` und `InventoryValue` mit dem Alter zusammenhängen.

Um diese Frage konkret zu beantworten, soll zunächst die Korrelation zwischen den Merkmalen bestimmt und analysiert werden und anschließend der *Lambda-Pfad* eines regularisierten Modells wie `Ridge` -Regression geplottet werden. Die Hypothese lautet dabei, dass korrelierte Merkmale gemeinsam gedämpft werden.

Betrachtung von Korrelationsmatrizen:

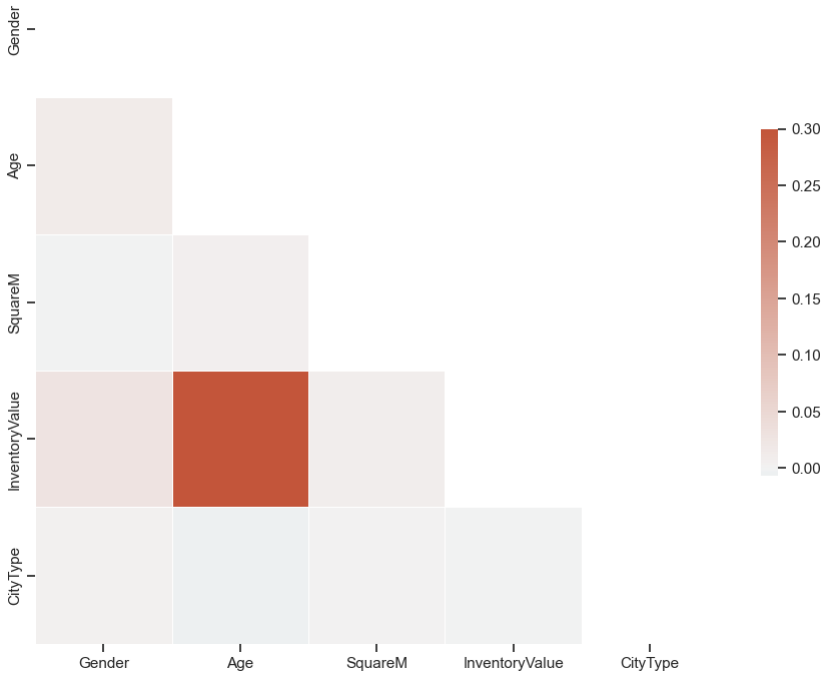
In [24]:
```
# Plotten der Korrelationsmatrix der Merkmale
# Korrelationsmatrix berechnen
corr = data_df.drop(["AgeGroup","freq","Response"],axis=1).corr()
# Maske für oberes (oder unteres) Dreieck setzen
mask = np.triu(np.ones_like(corr, dtype=bool))
# Größe des Plots einstellen
f, ax = plt.subplots(figsize=(11, 9))
# Colormap erzeugen
cmap = sns.diverging_palette(230, 20, as_cmap=True)
#Heatmap plotten
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,square=True, linewidths=.5, cbar_kws={"shrink": .5})
f.savefig("A3_corr_matrix.png")
```



Wir erkennen hier zunächst, dass `HouseholdIncome` stark mit `Gender` korreliert ist. Wenngleich dies möglicherweise horizontal mit dem "GenderPayGap" in Verbindung gebracht werden kann, ist dies deskriptiv für uns nur insofern relevant als wir später sicherstellen müssen, dass dieser Effekt das in Bezug auf `Gender` bereits recht gleichverteilte Prämiengeschehen nicht wieder verkompliziert. Wir entfernen `Gender` aus dem betrachteten Datensatz, um den Kontrast der restlichen Merkmale zu erhöhen:

In [25]:
```
# Plotten der Korrelationsmatrix der Merkmale
# Korrelationsmatrix berechnen
corr = data_df.drop(["AgeGroup","freq","Response","HouseholdIncome"],axis=1).corr()
# Maske für oberes (oder unteres) Dreieck setzen
mask = np.triu(np.ones_like(corr, dtype=bool))
# Größe des Plots einstellen
f, ax = plt.subplots(figsize=(11, 9))
# Colormap erzeugen
cmap = sns.diverging_palette(230, 20, as_cmap=True)
# Heatmap plotten
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

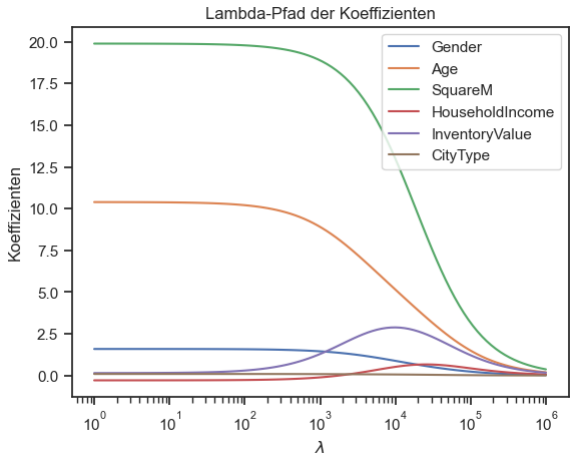
Out[25]:
```
<Axes: >
```



Erstellung eines *lambda*-Pfads:

Bei der Verwendung von regularisierten linearen Modellen wie *Ridge* oder *Lasso* und ihren Mischformen ist es durch Variation des Dämpfungsparameters λ möglich eine Hyperparameteroptimierung bezüglich der Response durchzuführen. Dies erlaubt durch die Betrachtung des Verlaufs $\lambda \mapsto (0, \infty)$ das Verständnis darüberm, welche Parameter gemeinsam gedämpft werden (-> Kovarianz), bzw. wie nach Elimination eines Koeffizienten die weiteren "stärkeren" Zusammenhänge sich darstellen.

In [26]:
```
n_alphas = 100
alphas = np.logspace(0,6, n_alphas)

coefs = []
for a in alphas:
    ridge = Ridge(alpha=a, fit_intercept=True)
    ridge.fit(x_train, y_train)
    coefs.append(ridge.coef_)

ax = plt.gca()
ax.plot(alphas, np.array(coefs).reshape(100,6))
ax.set_xscale("log")
# Auskommentieren, falls Achse getauscht werden soll - diese Konvention gibt es.
#ax.set_xlim(ax.get_xlim()[::-1])
plt.xlabel("$\lambda$")
plt.ylabel("Koeffizienten")
plt.title("Lambda-Pfad der Koeffizienten")
plt.axis("tight")
plt.legend(labels=X.columns)
plt.show()
```



Antwort A3: Es ist zu erkennen, dass das Merkmal `Geschlecht` mit `HouseholdIncome` korreliert ist und das Merkmale `Alter` mit `InventoryValue` . Insofern sind diese Merkmale (potenzielle) Träger von indirekter Diskriminierung. Dies wird in den weiteren Aufgaben adressiert werden (müssen).

Aufgabe 4

Aufgabenstellung: Trainieren Sie ein Modell des Typs ***localGLMnet***[3],[4] und vergleichen Sie es mit dem Modellen aus Aufgabe 2 und 3. Prüfen Sie mit den in [3] vorgeschlagenen Attention-Methoden in Sahpley Value Contributions die funktionalen Zusammenhänge und vergleichen Sie diese mit den Modellen aus A3.

In [27]:
```
# Code in Anlehnung an https://github.com/PK1706/pyLocalGLMnet
reg = linear_model.LinearRegression()
reg.fit(x_train, y_train)
```

Out[27]:
```
▾ LinearRegression
LinearRegression()
```

In [28]:
```
# LocalGLMnet strukturieren

input = tf.keras.Input(shape=(6), dtype="float32")

attention = input
attention = tf.keras.layers.Dense(units=20, activation="tanh")(attention)
attention = tf.keras.layers.Dense(units=15, activation="tanh")(attention)
attention = tf.keras.layers.Dense(units=10, activation="tanh")(attention)
attention = tf.keras.layers.Dense(units=6, activation="linear", name="Attention")(
    attention
)

# Skip-Connection
response = tf.keras.layers.Dot(axes=1)([input, attention])

# Response Schicht = lokales GLM
response = tf.keras.layers.Dense(units=1, activation="linear", name="Response")(
    response
)
```

In [74]:
```
# Modell kompilieren
local_glm_net = tf.keras.Model(inputs=input, outputs=response)
optimizer = Adam(learning_rate=0.005)
local_glm_net.compile(loss="mse", optimizer=optimizer)
local_glm_net.summary()
```

```
Model: "model_12"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 6)]          0           []

 dense (Dense)                  (None, 20)           140         ['input_1[0][0]']

 dense_1 (Dense)                (None, 15)           315         ['dense[0][0]']

 dense_2 (Dense)                (None, 10)           160         ['dense_1[0][0]']

 Attention (Dense)              (None, 6)            66          ['dense_2[0][0]']

 dot (Dot)                      (None, 1)            0           ['input_1[0][0]',
                                                                  'Attention[0][0]']

 Response (Dense)               (None, 1)            2           ['dot[0][0]']

==================================================================================================
Total params: 683
Trainable params: 683
Non-trainable params: 0
__________________________________________________________________________________________________
```

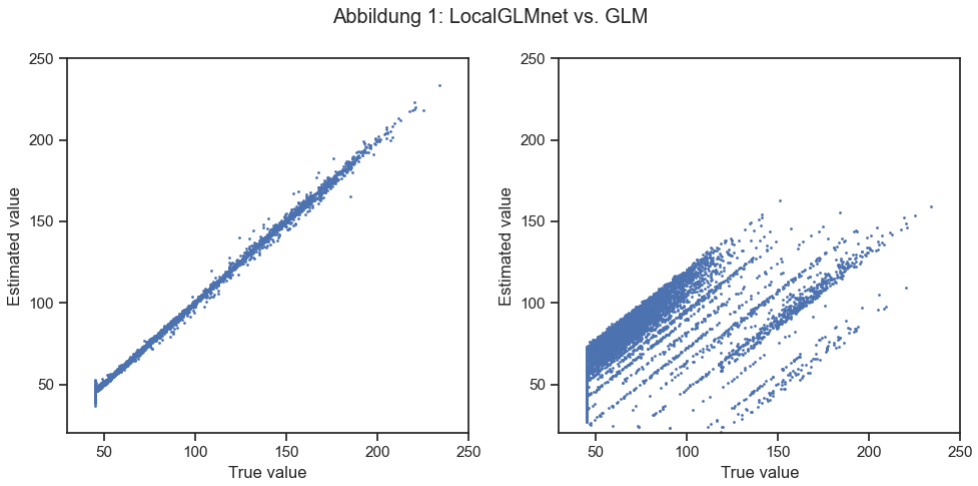
In [75]:

```python
# Modell trainieren
history = local_glm_net.fit(
    x_train, y_train, batch_size=16, epochs=50, validation_split=0.2, verbose=1
)
```

```
Epoch 1/50
938/938 [==============================] - 2s 2ms/step - loss: 5.1412 - val_loss: 4.1517
Epoch 2/50
938/938 [==============================] - 1s 1ms/step - loss: 5.0449 - val_loss: 3.9623
Epoch 3/50
938/938 [==============================] - 1s 1ms/step - loss: 4.3198 - val_loss: 3.5952
Epoch 4/50
938/938 [==============================] - 1s 1ms/step - loss: 3.7992 - val_loss: 2.6015
Epoch 5/50
938/938 [==============================] - 1s 1ms/step - loss: 5.2672 - val_loss: 4.9577
Epoch 6/50
938/938 [==============================] - 1s 1ms/step - loss: 3.6536 - val_loss: 3.3332
Epoch 7/50
938/938 [==============================] - 1s 1ms/step - loss: 5.6722 - val_loss: 1.9511
Epoch 8/50
938/938 [==============================] - 1s 1ms/step - loss: 4.3057 - val_loss: 3.6887
Epoch 9/50
938/938 [==============================] - 1s 1ms/step - loss: 3.8170 - val_loss: 2.3378
Epoch 10/50
938/938 [==============================] - 1s 1ms/step - loss: 3.8075 - val_loss: 12.9835
Epoch 11/50
938/938 [==============================] - 1s 1ms/step - loss: 3.8957 - val_loss: 2.8141
Epoch 12/50
938/938 [==============================] - 1s 1ms/step - loss: 4.7911 - val_loss: 4.5397
Epoch 13/50
938/938 [==============================] - 1s 1ms/step - loss: 3.5128 - val_loss: 1.7887
Epoch 14/50
938/938 [==============================] - 1s 1ms/step - loss: 4.2569 - val_loss: 2.4499
Epoch 15/50
938/938 [==============================] - 1s 1ms/step - loss: 3.4800 - val_loss: 1.6309
Epoch 16/50
938/938 [==============================] - 1s 1ms/step - loss: 3.3211 - val_loss: 1.5164
Epoch 17/50
938/938 [==============================] - 1s 1ms/step - loss: 3.5054 - val_loss: 1.7973
Epoch 18/50
938/938 [==============================] - 1s 1ms/step - loss: 3.7140 - val_loss: 3.8932
Epoch 19/50
938/938 [==============================] - 1s 1ms/step - loss: 3.1368 - val_loss: 2.3811
Epoch 20/50
938/938 [==============================] - 1s 1ms/step - loss: 3.9461 - val_loss: 2.5443
Epoch 21/50
938/938 [==============================] - 1s 1ms/step - loss: 3.6834 - val_loss: 1.7213
Epoch 22/50
938/938 [==============================] - 1s 1ms/step - loss: 4.0261 - val_loss: 6.3347
Epoch 23/50
938/938 [==============================] - 1s 1ms/step - loss: 2.7965 - val_loss: 1.8661
Epoch 24/50
938/938 [==============================] - 1s 1ms/step - loss: 3.5076 - val_loss: 4.1627
Epoch 25/50
938/938 [==============================] - 1s 1ms/step - loss: 3.1386 - val_loss: 2.4812
Epoch 26/50
938/938 [==============================] - 1s 1ms/step - loss: 3.0461 - val_loss: 1.5771
Epoch 27/50
938/938 [==============================] - 1s 1ms/step - loss: 2.5056 - val_loss: 5.2596
Epoch 28/50
938/938 [==============================] - 1s 1ms/step - loss: 4.4199 - val_loss: 2.1690
Epoch 29/50
938/938 [==============================] - 1s 1ms/step - loss: 2.8970 - val_loss: 2.9351
Epoch 30/50
938/938 [==============================] - 1s 1ms/step - loss: 3.2692 - val_loss: 3.5738
Epoch 31/50
938/938 [==============================] - 1s 1ms/step - loss: 3.2688 - val_loss: 3.2072
Epoch 32/50
938/938 [==============================] - 1s 1ms/step - loss: 3.2047 - val_loss: 4.2992
Epoch 33/50
938/938 [==============================] - 1s 1ms/step - loss: 3.1060 - val_loss: 2.1039
Epoch 34/50
938/938 [==============================] - 1s 1ms/step - loss: 3.5638 - val_loss: 1.9459
Epoch 35/50
938/938 [==============================] - 1s 1ms/step - loss: 3.3946 - val_loss: 2.3697
Epoch 36/50
938/938 [==============================] - 1s 1ms/step - loss: 3.2982 - val_loss: 2.0437
Epoch 37/50
938/938 [==============================] - 1s 1ms/step - loss: 2.4281 - val_loss: 4.3541
Epoch 38/50
938/938 [==============================] - 1s 1ms/step - loss: 4.4374 - val_loss: 4.2855
Epoch 39/50
938/938 [==============================] - 1s 1ms/step - loss: 3.0386 - val_loss: 1.6433
Epoch 40/50
938/938 [==============================] - 1s 1ms/step - loss: 3.0383 - val_loss: 2.1124
Epoch 41/50
938/938 [==============================] - 1s 1ms/step - loss: 2.5537 - val_loss: 3.9169
Epoch 42/50
938/938 [==============================] - 1s 1ms/step - loss: 3.5007 - val_loss: 2.3749
Epoch 43/50
938/938 [==============================] - 1s 1ms/step - loss: 3.3330 - val_loss: 3.1925
Epoch 44/50
938/938 [==============================] - 1s 1ms/step - loss: 2.8401 - val_loss: 2.0015
Epoch 45/50
938/938 [==============================] - 1s 1ms/step - loss: 2.7478 - val_loss: 3.4646
Epoch 46/50
938/938 [==============================] - 1s 1ms/step - loss: 2.9489 - val_loss: 6.6453
Epoch 47/50
938/938 [==============================] - 1s 1ms/step - loss: 2.9068 - val_loss: 2.3110
Epoch 48/50
938/938 [==============================] - 1s 1ms/step - loss: 3.8596 - val_loss: 1.4282
Epoch 49/50
938/938 [==============================] - 1s 1ms/step - loss: 2.6396 - val_loss: 1.6950
Epoch 50/50
938/938 [==============================] - 1s 1ms/step - loss: 2.5075 - val_loss: 1.7617
```

In [76]:
```
# Vorhersage mit localGLMnet und GLM
pred_local = local_glm_net.predict(x_test)
pred_reg = reg.predict(x_test)

fig_performance = plt.figure(tight_layout=True, figsize=(10, 5))

spec = GridSpec(ncols=2, nrows=1, figure=fig_performance)
axs_perf = [
    fig_performance.add_subplot(spec[0, 0]),
    fig_performance.add_subplot(spec[0, 1]),
]

axs_perf[0].scatter(y_test, pred_local, s=1)
axs_perf[1].scatter(y_test, pred_reg, s=1)

# Layout
for ax in axs_perf:
    ax.set_xlabel("True value")
    ax.set_ylabel("Estimated value")
    ax.set_xlim((30, 250))
    ax.set_ylim((20, 250))

fig_performance.suptitle("Abbildung 1: LocalGLMnet vs. GLM")
plt.show()

print("MSE LocalGLMnet: " + str(metrics.mean_squared_error(y_test, pred_local)))
print("MSE GLM: " + str(metrics.mean_squared_error(y_test, pred_reg)))
```

```
196/196 [==============================] - 0s 831us/step
```




```
MSE LocalGLMnet: 1.7671011999103137
MSE GLM: 859.6596171638254
```

In [77]:
```
# Über die Methode get_weights() erhält man die Kantengewichte, sowie den Bias für jeder Schicht
# --> man erhält also eine Liste mit numpy Arrays die in der Länge der Anzahl der Ebenen * 2 entspricht
weights = local_glm_net.get_weights()
for i in weights:
    print(i.shape, end=" | ")
```

```
(6, 20) | (20,) | (20, 15) | (15,) | (15, 10) | (10,) | (10, 6) | (6,) | (1, 1) | (1,) |
```

In [78]:
```
# Neues Model ohne Response-Schicht --> ermöglicht auslesen der Attention Gewichte
weights_local_glm = tf.keras.Model(
    inputs=local_glm_net.input, outputs=local_glm_net.get_layer(name="Attention").output
)

# Gewichte bestimmen
beta_x = weights_local_glm.predict(x_test)

# Skalierung der Attention-Gewichte mithilfe des Gewichts der Response Schicht ( = Intercept beta_0)
beta_x_scaled = beta_x * weights[8]
```

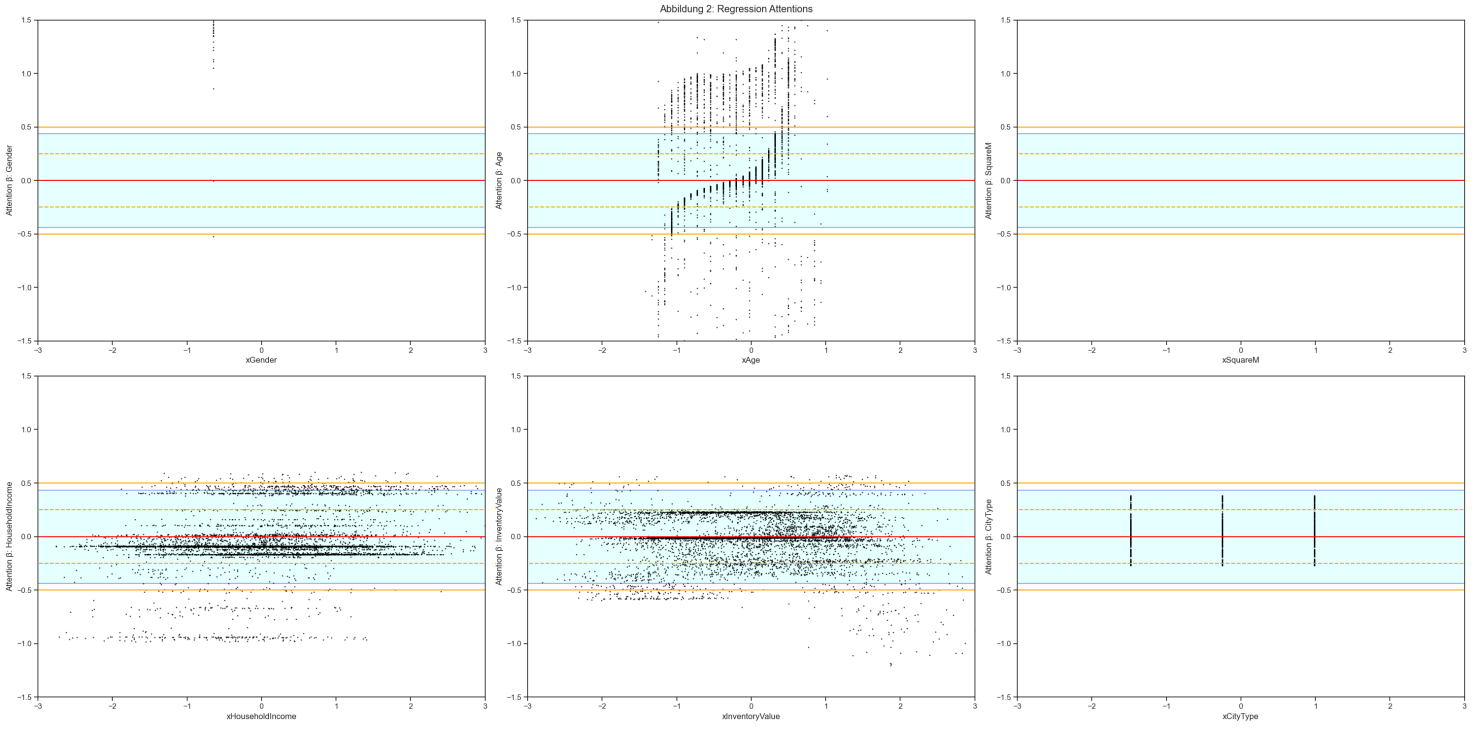
```
196/196 [==============================] - 0s 708us/step
```

In [79]:
```
# Merkmal 7 ist von der wahren Regressionsfunktion unabhängig
# --> Einsatz zur Berechnung des Konfidenzintervals

print("Mittelwert β6: " + str(beta_x_scaled[:, 5].mean()))
print("Standardabweichung β6: " + str(beta_x_scaled[:, 5].std()))

# Intervalgrenzen bestimmen
alpha = 0.001
bound = stats.norm.ppf(alpha / 2) * beta_x_scaled[:, 5].std()

print("Quantil " + str(1 - alpha / 2) + ": " + str(stats.norm.ppf(alpha / 2)))
print("Grenzen: ± " + str(abs(bound)))
```

```
Mittelwert β6: 0.03902574
Standardabweichung β6: 0.1324885
Quantil 0.9995: -3.2905267314918945
Grenzen: ± 0.4359569641990118
```

In [80]:

```
# Attention Plot
fig_attention = plt.figure(tight_layout=True, figsize=(30, 15))

# Gliederung der Subplots
spec = GridSpec(ncols=3, nrows=2, figure=fig_attention)
ax1_att = fig_attention.add_subplot(spec[0, 0])
ax2_att = fig_attention.add_subplot(spec[0, 1])
ax3_att = fig_attention.add_subplot(spec[0, 2])
ax4_att = fig_attention.add_subplot(spec[1, 0])
ax5_att = fig_attention.add_subplot(spec[1, 1])
ax6_att = fig_attention.add_subplot(spec[1, 2])
axs_att = [ax1_att, ax2_att, ax3_att, ax4_att, ax5_att, ax6_att]

# Ein Subplot pro Input Feature erstellen
for i in range(len(axs_att)):

    # Linien zur Verdeutlichung der Höhe der Attention Gewichte
    axs_att[i].hlines(y=0.5, xmin=-4, xmax=4, colors="orange")
    axs_att[i].hlines(y=-0.5, xmin=-4, xmax=4, colors="orange")

    axs_att[i].hlines(y=0.25, xmin=-4, xmax=4, colors="orange", linestyles="dashed")
    axs_att[i].hlines(y=-0.25, xmin=-4, xmax=4, colors="orange", linestyles="dashed")

    axs_att[i].hlines(y=0, xmin=-4, xmax=4, colors="red")

    # Intervalgrenzen
    interval = patches.Rectangle(
        xy=(-4, bound),
        height=2 * abs(bound),
        width=8,
        edgecolor="royalblue",
        facecolor="lightcyan",
        alpha=0.8,
        zorder=1,
    )
    axs_att[i].add_patch(interval)

    # Scatter Plot --> x: Werte der Inputfeatures, y: Attention Gewichte
    axs_att[i].scatter(x_test[:, i], beta_x_scaled[:, i], s=0.5, c="black")

    # Layout
    axs_att[i].set_xlim((-3, 3))
    axs_att[i].set_ylim((-1.5, 1.5))
    axs_att[i].set_xlabel("x" + X.columns[i])
    axs_att[i].set_ylabel("Attention β: " + X.columns[i])

fig_attention.suptitle("Abbildung 2: Regression Attentions")
plt.show()
```



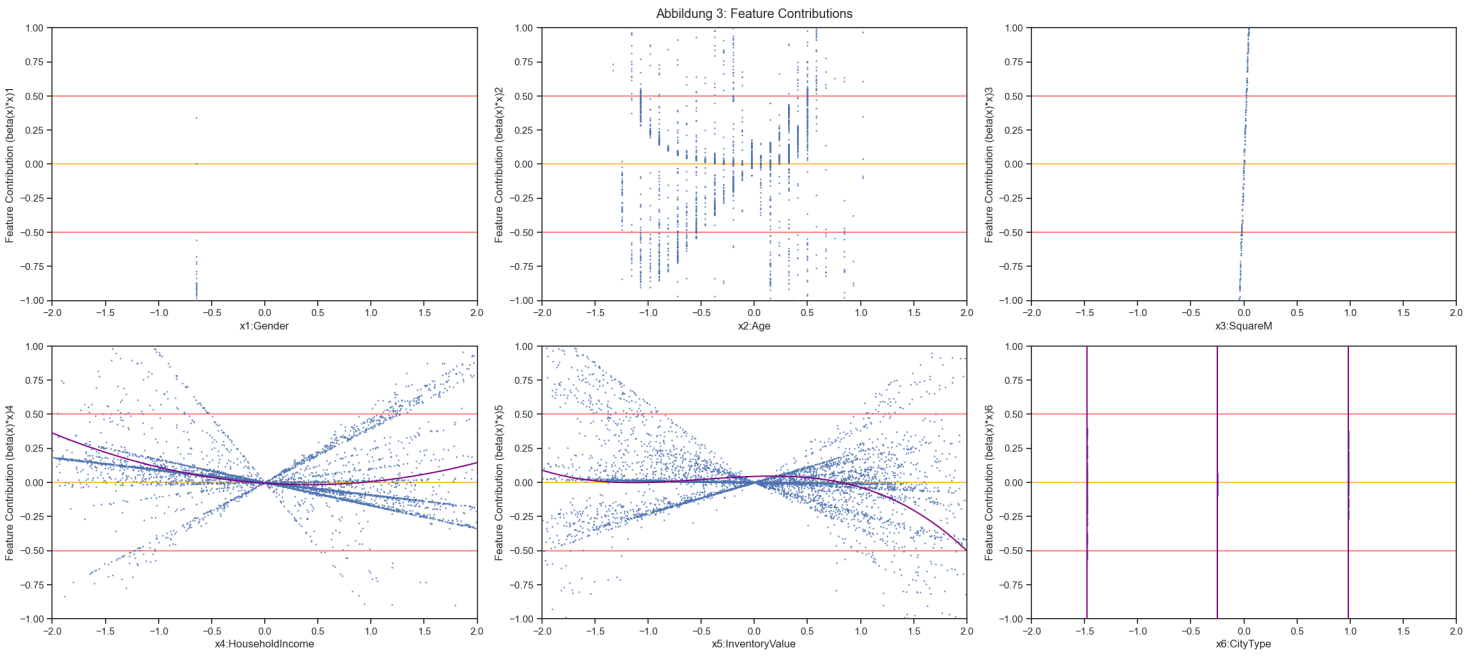
In [81]:

```
for i in range(6):
    if i != 6:
        size = beta_x_scaled.shape[0]
        coverage = np.count_nonzero(
            beta_x_scaled[:, i] < abs(bound)
        ) - np.count_nonzero(beta_x_scaled[:, i] < -abs(bound))
        coverage_ratio = coverage / size
        print("Coverage Ratio β" + str(i + 1) + ": " + str(coverage_ratio))
```

```
Coverage Ratio β1: 0.00016
Coverage Ratio β2: 0.30352
Coverage Ratio β3: 0.0
Coverage Ratio β4: 0.89728
Coverage Ratio β5: 0.93056
Coverage Ratio β6: 1.0
```

In [82]:

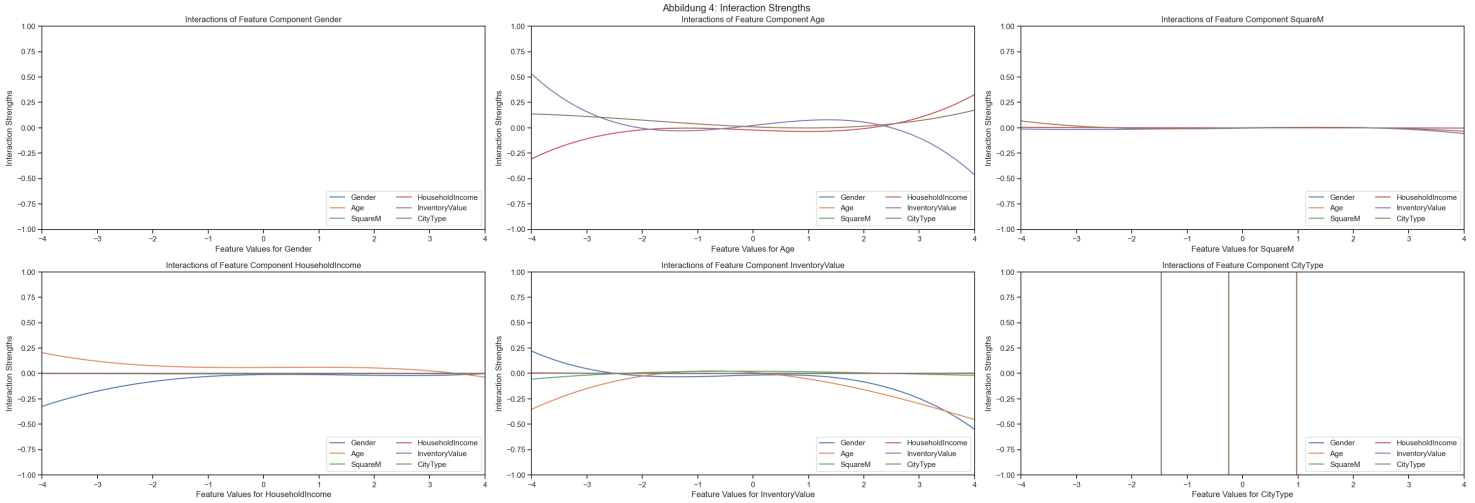
```python
# Feature Contribution Plot
fig_contribution = plt.figure(tight_layout=True, figsize=(30, 15))

spec = GridSpec(ncols=8, nrows=3, figure=fig_contribution)
ax1_con = fig_contribution.add_subplot(spec[0, 1:3])
ax2_con = fig_contribution.add_subplot(spec[0, 3:5])
ax3_con = fig_contribution.add_subplot(spec[0, 5:7])
ax4_con = fig_contribution.add_subplot(spec[1, 1:3])
ax5_con = fig_contribution.add_subplot(spec[1, 3:5])
ax6_con = fig_contribution.add_subplot(spec[1, 5:7])

axs_con = [ax1_con, ax2_con, ax3_con, ax4_con, ax5_con, ax6_con]

xs = np.linspace(-4, 4, 100)

for i in range(len(axs_con)):

    # Feature Contribution Splines berechnen
    # Feature Contribution = beta(xi)*xi
    contribution = np.column_stack([x_test[:, i], beta_x_scaled[:, i] * x_test[:, i]])
    con_ind = np.lexsort((contribution[:, 1], contribution[:, 0]))
    contribution_sorted = contribution[con_ind]
    con_spline = interpolate.UnivariateSpline(
        contribution_sorted[:, 0], contribution_sorted[:, 1]
    )

    # Hinzufügen von horizontalen Linien um die Stärke der Feature Contribution zu visualisieren
    axs_con[i].hlines(y=0, xmin=-4, xmax=4, colors="orange", alpha=0.7, zorder=1)

    axs_con[i].hlines(y=0.5, xmin=-4, xmax=4, colors="red", alpha=0.5, zorder=1)
    axs_con[i].hlines(y=-0.5, xmin=-4, xmax=4, colors="red", alpha=0.5, zorder=1)

    axs_con[i].hlines(y=1, xmin=-4, xmax=4, colors="lightcyan", alpha=0.7, zorder=1)
    axs_con[i].hlines(y=-1, xmin=-4, xmax=4, colors="lightcyan", alpha=0.7, zorder=1)

    axs_con[i].hlines(y=1.5, xmin=-4, xmax=4, colors="royalblue", alpha=0.7, zorder=1)
    axs_con[i].hlines(y=-1.5, xmin=-4, xmax=4, colors="royalblue", alpha=0.7, zorder=1)

    # Scatter Plot --> x: Werte der Inputfeatures, y:Feature Contribution (θ(x)*x)
    axs_con[i].scatter(contribution[:, 0], contribution[:, 1], s=0.5, zorder=10)

    # Feature Contribution Spline plotten
    axs_con[i].plot(xs, con_spline(xs), color="purple", zorder=20)

    # Layout
    axs_con[i].set_xlim((-2, 2))
    axs_con[i].set_ylim((-1, 1))
    axs_con[i].set_xlabel("x" + str(i + 1) + ":" + X.columns[i])
    axs_con[i].set_ylabel("Feature Contribution (beta(x)*x)" + str(i + 1))

fig_contribution.suptitle("Abbildung 3: Feature Contributions")
plt.show()
```



In [83]:
```
# Gradienten bestimmen
gradients = []
x = tf.constant(x_train)

# Für jede Inputvariable wird ein Modell gefittet, um anschließend die partiellen Ableitungen auslesen zu können
for i in range(input.shape[-1]):

    # Lambda Layer als Output Schicht, um beta_i als Output zu erhalten (partielle Ableitungen ∂θ_j(x)/∂x_j')
    beta = attention
    beta = tf.keras.layers.Lambda(lambda x: x[:, i])(beta)
    grad_model = tf.keras.Model(inputs=input, outputs=beta)

    # GradientTape ermöglicht das auslesen der Gradienten
    with tf.GradientTape() as g:
        g.watch(x)
        pred_attention = grad_model.call(x)

    grad = g.gradient(pred_attention, x)

    # Array das sowohl den Wert von x, als auch den entsprechenden Wert von θk(x) enthält
    grad_wrt_x = np.column_stack((x[:, i].numpy(), grad.numpy()))

    # Um später die Splines zu modellieren muss die x-Komponente monoton steigend sein --> sortieren des Arrays
    ind = np.lexsort((grad_wrt_x[:, 2], grad_wrt_x[:, 0]))
    grad_wrt_x_sorted = grad_wrt_x[ind]

    # Gradienten in Liste speichern
    gradients.append(grad_wrt_x_sorted)
```

In [84]:
```
# Univariate Splines modellieren, um die Interaktion zwischen Features darzustellen
splines = []

# Für alle Attention Gewichte θ
for i in range(input.shape[-1]):
    splines.append([])

    # Für alle Inputvariablen x
    for j in range(input.shape[-1]):
        splines[i].append(
            interpolate.UnivariateSpline(gradients[i][:, 0], gradients[i][:, j + 1])
        )
```

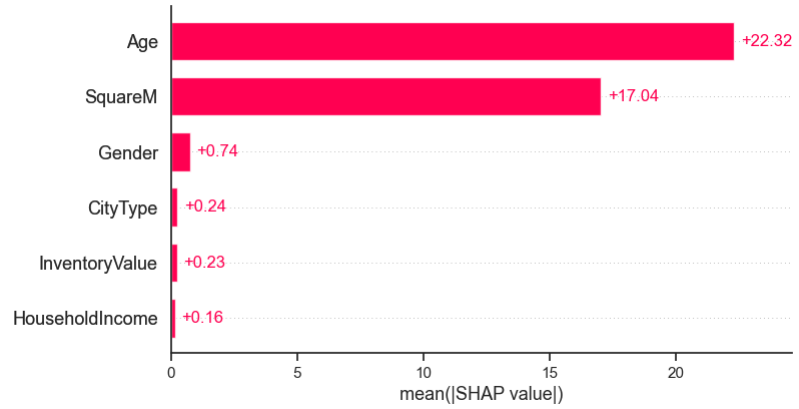
In [85]:
```
# Spline Interaction Plot

fig_spline = plt.figure(tight_layout=True, figsize=(30, 15))
spec = GridSpec(ncols=3, nrows=3, figure=fig_spline)
ax1_sp = fig_spline.add_subplot(spec[0, 0])
ax2_sp = fig_spline.add_subplot(spec[0, 1])
ax3_sp = fig_spline.add_subplot(spec[0, 2])
ax4_sp = fig_spline.add_subplot(spec[1, 0])
ax5_sp = fig_spline.add_subplot(spec[1, 1])
ax6_sp = fig_spline.add_subplot(spec[1, 2])

axs_sp = [ax1_sp, ax2_sp, ax3_sp, ax4_sp, ax5_sp, ax6_sp]

xs = np.linspace(-4, 4, 50)

for i in range(input.shape[-1]):

    # Splines für jedes Merkmal plotten
    for j in range(input.shape[-1]):
        axs_sp[i].plot(xs, splines[i][j](xs), label=X.columns[j])

    # Legende hinzufügen
    axs_sp[i].legend(loc="lower right", ncol=2)

    # Layout
    axs_sp[i].set_xlim((-4, 4))
    axs_sp[i].set_ylim((-1, 1))
    axs_sp[i].set_xlabel("Feature Values for " + X.columns[i])
    axs_sp[i].set_ylabel("Interaction Strengths")
    axs_sp[i].set_title("Interactions of Feature Component " + X.columns[i])
fig_spline.suptitle("Abbildung 4: Interaction Strengths")

plt.show()
```



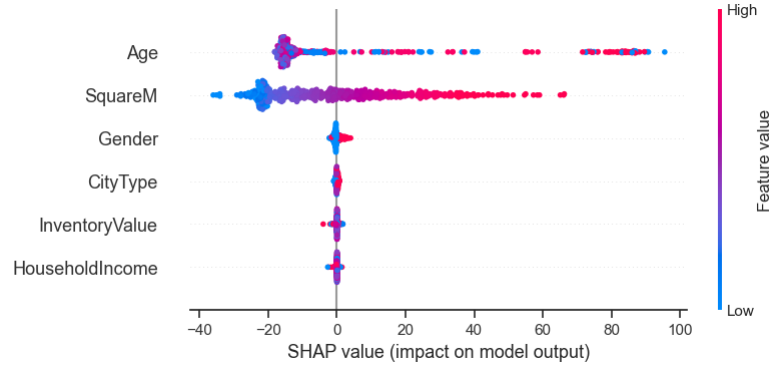
In [41]:
```
background = pd.DataFrame(x_train[:500],columns = X.columns)
explainer = shap.explainers.SamplingExplainer(lambda x:local_glm_net.predict(x,verbose=False), background)
shap_values_glm_net = explainer(pd.DataFrame(x_train[:500],columns = X.columns))
```

```
0%|          | 0/500 [00:00<?, ?it/s]
```

In [42]:
```
shap.plots.bar(shap_values_glm_net)
```



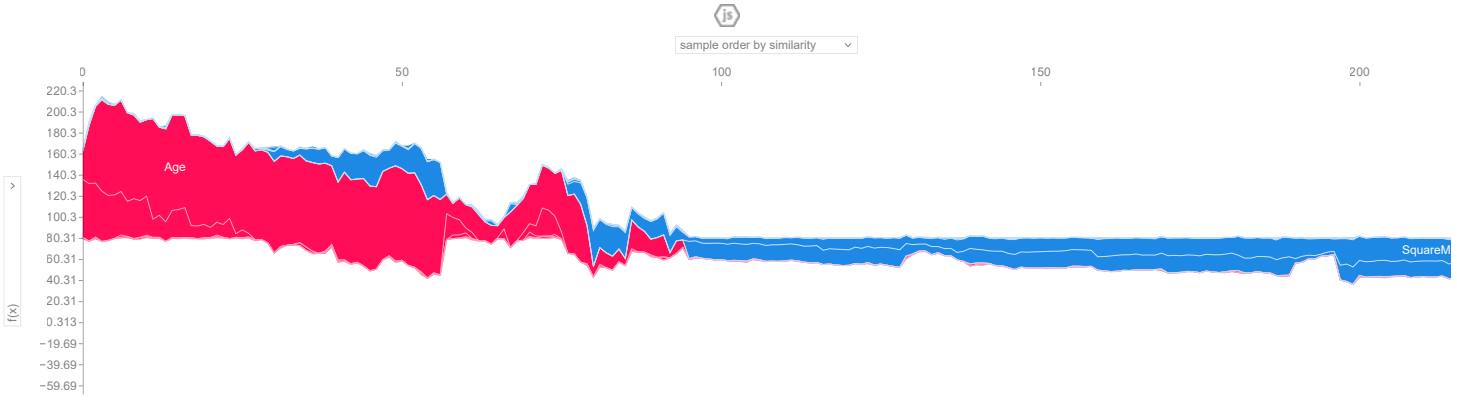
In [43]:
```
shap.plots.beeswarm(shap_values_glm_net)
```



Aus dem *beeswarm*-Plot lässt sich die Spreizung der Merkmale in ihrer Wirkung ablesen. Erwartungsgemäß liegt diese für `Age` am weitesten auseinander, wohingegen bspw. `CityType` und `HouseholdIncome`, die in dem Modell wenig Vorhersagekraft haben, auch keinen Einfluss in den Shapley-Values aufweisen.

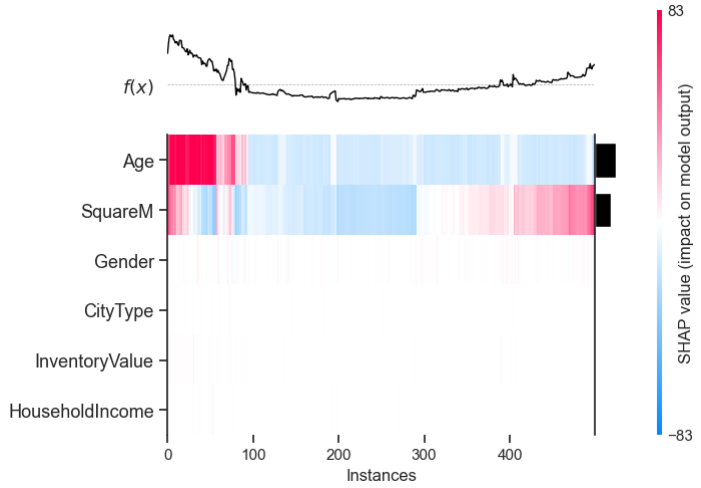
In [44]:
```
shap.initjs()
shap.plots.force(shap_values_glm_net)
```

Out[44]:

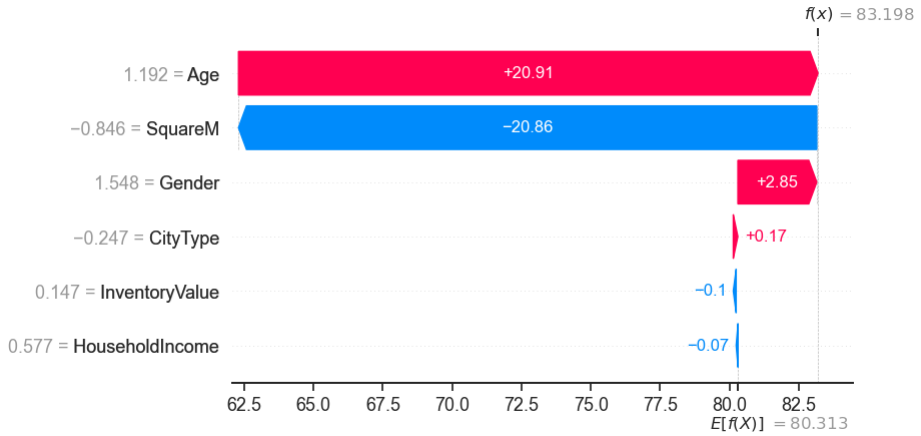


Der *force*-Plot kann verwendet werden, um die Marginalverteilungen abzulesen, und so eine Art Partial Dependence zu illustrieren (weswegen letztere für diese Aufgabe auch nicht mehr geplottet wird). Für bspw. `Alter` zeigt sich, dass die Werte am Rand aufgrund der nichtlinearen Verteilung großen Einfluss haben - andererseits muss man anhand der Modellvorgabe aber auch konstatieren, dass hier das Modell unterangepasst ist.

In [45]:
```
shap.plots.heatmap(shap_values_glm_net)
```



In [46]:
```
shap.plots.waterfall(shap_values_glm_net[0])
#shap.plots.waterfall(shap_values_glm_net[:,42])
#shap_values_glm_net
```



Antwort A4: Insgesamt lässt sich festhalten, dass die Modellklasse *localGLMnet* - unter der Annahme, dass diese hier richtig und zielführend angewendet wurde - gute Anpassungseigenschaften erlaubt und für korrekt Modellierte und unabhängige Merkmale auch zugleich näherungsweise die Linearkoeffizienten herausgeben kann. Dies ist überall dort nützlich, wo die genauen Modellannahmen entweder unbekannt oder zumindest unsicher sind. In diesem Falle kann man hier wieder gut ablesen, dass der Zusammenhang mit `Age` eigentlich quadratisch ist und ein solches Merkmal bzw. Interaktion aufzunehmen wäre. Insgesamt kann man aber festhalten, dass die Modellklasse für die Problemstellung per se ersteinmal für geeignet gehalten werden kann.

Aufgabe 5

Aufgabenstellung: Trainieren Sie mindestens zwei Modelle, die die indirekte Diskriminierung beseitigen können und bewerten Sie, ob und wie gut dies gelingt. Konzentrieren Sie sich auf die Vermeidung von indirekter Diskriminierung beim Merkmal `Age` . Vergleichen Sie dazu die Basisprämie mit der korrigierten prämie sowohl auf Einzelvertragsebene als auch auf einer geeigneten aggregierten Ebene.

Verändern Sie für Ihr `localGLMnet` die Loss-Funktion auf geeignete Weise, sodass das Modell diskriminierungsmindernd wirkt, *wenn möglich auch gegen indirekte Diskriminierung*.

Untersuchen Sie Ihre Ansätze mit geeigneten Visualisierungen und Erklärbarkeitsansätzen, und diskutieren Sie die Erklärbarkeit der Methoden hinsichtlich der Frage nach direkter und indiorekter Diskriminierung.

Betrachten Sie abschließend, wie der Tarif unter den hypothetischen Rahmenbedingungen, dass keinerlei Korrelation mit geschützten Merkmalen im Tarif vorhanden sein darf, aussehen könnte und welche Herausforderungen sich im Vergleich ergeben. Betrachten Sie insbesondere Langzeitfolgen von Moral Hazard und Adverser Selektion in einem kompetitiven Marktumfeld. Beschreiben Sie einen vergleichenden Business Case, der strategische und formale Empfehlungen abgibt.

Hinweis 1: Ein geeigneter Ansatz findet sich in [1], auch [3] kann Hinweise zur Durchführung geben.

Zur Lösung dieser Aufgabe soll der Ansatz verfolgt werden, Diskriminierung nach Alter durch gruppenweise Angleichung zu vermeiden. Dies hat, wie wir sehen werden, natürlich eine schwächere Anpassungsleistung zur Folge.

In [47]:
```
print("Ermittelte Koeffizienten für Basismodell:",list(zip(X.columns,lasso.coef_)))
print("Intercept:",lasso.intercept_)

# Berechne mittleren Preis

lasso_pred = lasso.predict(X_scaled)
lasso_mean_price = np.mean(lasso_pred)
print("Mittlere Prämie: ",lasso_mean_price)
```

```
Ermittelte Koeffizienten für Basismodell: [('Gender', 1.4360377552073618), ('Age', 10.294218579196361), ('SquareM', 19.792888930995915), ('HouseholdIncome', -0.09119831890145014), ('InventoryValue', 0.074228533577506), ('CityType', 0.0)]
Intercept: 81.89899123516776
Mittlere Prämie:  81.89899123516776
```

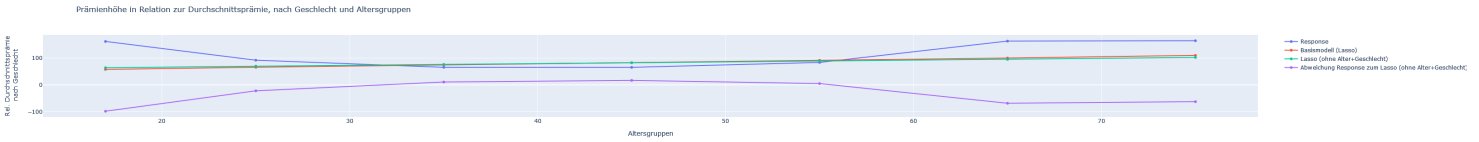
Wir haben in Aufgabe 2 bereits die Differenzierung nach `Gender` durchgeführt und festgestellt, dass die Unterschiede ziemlich klein sind, sodass hier eher keine weitere Angleichungsmodellierung notwendig ist.

Das Ziel unserer Vorgehensweise ist es nun, den Koeffizienten für `Age` gruppenbasiert so anzupassen, dass für jede Altersgruppe der gleiche (mittlere) Preis entsteht.

In [48]:
```
# Pseudocode:
    # Berechne die mittlere Prämie für jede Gruppe
    # Bastle einen Prädiktor, der erlaubt, auch die partial Dependence zu plotten.
    # Berechnen auf Basis der Response die Residuen
```

In [49]:
```
avg_response_age_groups = (data_df_pred.groupby('AgeGroup').mean())['Response']
age_group_index_dict = list(data_df_pred.groupby('AgeGroup').indices.items())
age_group_indices_list = []
for item in age_group_index_dict:
    age_group_indices_list.append(item[1])
age_group_predictions_lm = []
age_group_predictions_lasso = []
age_group_predictions_lm_wo_agegender = []
age_group_predictions_lasso_wo_agegender = []
for age_group in range(len(age_group_indices_list)):
    age_group_predictions_lm.append(lm.predict(X_scaled[age_group_indices_list[age_group]]).mean())
    age_group_predictions_lasso.append(lasso.predict(X_scaled[age_group_indices_list[age_group]]).mean())
    age_group_predictions_lasso_wo_agegender.append(lasso_wo_ageandgender.predict(X_scaled[:,2:][age_group_indices_list[age_group]]).mean())
# Berechne Abweichung pro Gruppe
diff_age_groups_lm_wo_agegender = age_group_predictions_lasso_wo_agegender - avg_response_age_groups
```

In [143…
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=age_groups, y=avg_response_age_groups, mode="lines+markers", name='Response'))
#fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lm, mode="lines+markers", name='Basismodell (LM)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lasso, mode="lines+markers", name='Basismodell (Lasso)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lasso_wo_agegender, mode="lines+markers", name='Lasso (ohne Alter+Geschlecht)'))
fig.add_trace(go.Scatter(x=age_groups, y=diff_age_groups_lm_wo_agegender, mode="lines+markers", name='Abweichung Response zum Lasso (ohne Alter+Geschlecht)'))

fig.update_layout(title='Prämienhöhe in Relation zur Durchschnittsprämie, nach Geschlecht und Altersgruppen',
                  xaxis_title='Altersgruppen',
                  yaxis_title='Rel. Durchschnittsprämie<br> nach Geschlecht')

fig.show(renderer='vscode')
```



In [51]:
```
# Altersgrenzen bestimmen
age_groups_upper = []
age_groups_upper.append(scaler_X.transform(np.array([[0,26,0,0,0,0]]))[:,1][0])
age_groups_upper.append(scaler_X.transform(np.array([[0,36,0,0,0,0]]))[:,1][0])
age_groups_upper.append(scaler_X.transform(np.array([[0,46,0,0,0,0]]))[:,1][0])
age_groups_upper.append(scaler_X.transform(np.array([[0,56,0,0,0,0]]))[:,1][0])
age_groups_upper.append(scaler_X.transform(np.array([[0,66,0,0,0,0]]))[:,1][0])
age_groups_upper.append(scaler_X.transform(np.array([[0,76,0,0,0,0]]))[:,1][0])
age_groups_lower = []
age_groups_lower.append(scaler_X.transform(np.array([[0,25,0,0,0,0]]))[:,1][0])
age_groups_lower.append(scaler_X.transform(np.array([[0,35,0,0,0,0]]))[:,1][0])
age_groups_lower.append(scaler_X.transform(np.array([[0,45,0,0,0,0]]))[:,1][0])
age_groups_lower.append(scaler_X.transform(np.array([[0,55,0,0,0,0]]))[:,1][0])
age_groups_lower.append(scaler_X.transform(np.array([[0,65,0,0,0,0]]))[:,1][0])
age_groups_lower.append(scaler_X.transform(np.array([[0,75,0,0,0,0]]))[:,1][0])
#age_groups_lower
```

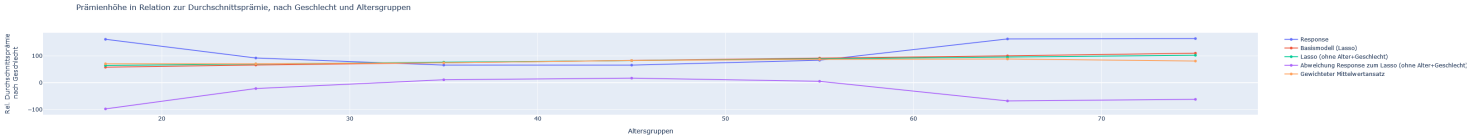
In [52]:
```
# Erstelle Modell, das diese Abweichung von der Basisprediction abzieht

class UnbiasedWeightedLassoEstimator(BaseEstimator):
    def __init__(self, column_name, correction_bias, group_ratios,beta_zero):
        # expected format for bias_dict: {column_name:bias_amount}
        self.beta_zero = beta_zero
        self.biased_column = column_name
        self.group_ratios = group_ratios
        self.correction_bias = correction_bias
        self.subclf = Lasso(alpha=0.1, fit_intercept=True)
        self.lastpred = 0
        self._estimator_type = "regressor"
    def __bias_corrector(self,biased_instance):
        bias_correction = 0
        correction_ratio = 1.0
        if biased_instance < age_groups_upper[0]:
            bias_correction = self.correction_bias[0]
            correction_ratio = self.group_ratios[0]
        if biased_instance > age_groups_lower[0] and biased_instance < age_groups_upper[1]:
            bias_correction = self.correction_bias[1]
            correction_ratio = self.group_ratios[1]
        if biased_instance > age_groups_lower[1] and biased_instance < age_groups_upper[2]:
            bias_correction = self.correction_bias[2]
            correction_ratio = self.group_ratios[2]
        if biased_instance > age_groups_lower[2] and biased_instance < age_groups_upper[3]:
            bias_correction = self.correction_bias[3]
            correction_ratio = self.group_ratios[3]
        if biased_instance > age_groups_lower[3] and biased_instance < age_groups_upper[4]:
            bias_correction = self.correction_bias[4]
            correction_ratio = self.group_ratios[4]
        if biased_instance > age_groups_lower[4] and biased_instance < age_groups_upper[5]:
            bias_correction = self.correction_bias[5]
            correction_ratio = self.group_ratios[5]
        if biased_instance > age_groups_lower[5]:
            bias_correction = self.correction_bias[6]
            correction_ratio = self.group_ratios[6]
        #print("| Alter: "+ str(biased_instance), " | Korrektur: " + str(bias_correction - bias_correction * correction_ratio) + "|")
        return bias_correction# + bias_correction * correction_ratio
    def __sklearn_is_fitted__(self):
        return True
    def fit(self, X, y):
        # Check that X and y have correct shape
        X, y = check_X_y(X, y)
        # Store the classes seen during fit
        self.X_ = X
        self.y_ = y
        self.subclf.fit(self.X_[:,2:],self.y_)
        # Return the classifier
        return self
    def predict(self, X):
        # Check if fit has been called
        check_is_fitted(self)
        # Input validation
        X = check_array(X)
        biased_instance = X[:,2]
        bias_correction = 0
        bias_correction = np.array(list(map(self.__bias_corrector,biased_instance))) + np.array(list(map(lambda x:self.beta_zero * x,biased_instance))) # Korrektur des Mittelwert +
lineare Korrektur auf Basis des Koeffizienten im Basismodell
        return self.subclf.predict(X[:,2:]) - bias_correction
```

In [53]:
```
# Benötige relativen Anteil der Gruppengröße
age_group_ratios =  (data_df_pred.groupby('AgeGroup').count() / len(X))['Age']
# Gesamtmittel
expectation = y.mean()
# Modell "trainieren"
lasso_wo_gender_unbiased_weighted =
UnbiasedWeightedLassoEstimator("Age",diff_age_groups_lm_wo_agegender.values,age_group_ratios,lasso.coef_[1]*expectation*2*np.mean(age_group_ratios))
lasso_wo_gender_unbiased_weighted.fit(X_scaled,y)
age_group_predictions_unbiased_weighted = []
# Modell auswerten
for age_group in range(len(age_group_indices_list)):
    age_group_predictions_unbiased_weighted.append(lasso_wo_gender_unbiased_weighted.predict(X_scaled[age_group_indices_list[age_group]]).mean())
print("Erwarteter Mittelwert: ",str(expectation))
print("Erreichter Mittelwert: ", str(np.mean(age_group_predictions_unbiased_weighted)))
print("Gruppenweise Mittelwerte Response:")
print(avg_response_age_groups.values)
print("Gruppenweise Mittelwerte Korrekturansatz:")
print(age_group_predictions_unbiased_weighted)
```

```
Erwarteter Mittelwert:  81.80351377733712
Erreichter Mittelwert:  79.28222620249097
Gruppenweise Mittelwerte Response:
[162.25985952  92.28093926  65.746978    65.63773191  83.88800243
 163.52640446 164.87408731]
Gruppenweise Mittelwerte Korrekturansatz:
[70.5892357058092, 70.54372935260129, 74.5478040111598, 82.84418324280534, 87.02903675491798, 88.55815893539162, 80.86343541475159]
```

In [144…

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=age_groups, y=avg_response_age_groups, mode="lines+markers", name='Response'))
#fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lm, mode="lines+markers", name='Basismodell (LM)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lasso, mode="lines+markers", name='Basismodell (Lasso)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lasso_wo_agegender, mode="lines+markers", name='Lasso (ohne Alter+Geschlecht)'))
fig.add_trace(go.Scatter(x=age_groups, y=diff_age_groups_lm_wo_agegender, mode="lines+markers", name='Abweichung Response zum Lasso (ohne Alter+Geschlecht)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_unbiased_weighted, mode="lines+markers", name='Gewichteter Mittelwertansatz'))

fig.update_layout(title='Prämienhöhe in Relation zur Durchschnittsprämie, nach Geschlecht und Altersgruppen',
                  xaxis_title='Altersgruppen',
                  yaxis_title='Rel. Durchschnittsprämie<br> nach Geschlecht')

fig.show(renderer='vscode')
```



Man sieht, dass der Ansatz mit dem gewichteten Mittelwert eine fast gerade Verteilung der Altersmarginale erzeugt - nur fast, denn es gibt irreduzible Anteile von Varianz, die auf Kovarianz mit den anderen Merkmalen zurückzuführen sind - und damit von indirekter Diskriminierung.

localGLMnet anpassen: Spezifische Loss-Funktion

In [86]:

```
def custom_loss_with_l2(y_true, y_pred, l2_loss_all, l2_loss_special):
    mse_loss = tf.reduce_mean(tf.square(y_true - y_pred))  # Mean squared error loss

    total_loss = mse_loss + l2_loss_all + l2_loss_special
    return total_loss

input_new = tf.keras.Input(shape=(6,), dtype="float32")

attention_new = input_new
attention_new = tf.keras.layers.Dense(units=20, activation="tanh")(attention_new)
attention_new = tf.keras.layers.Dense(units=15, activation="tanh")(attention_new)
attention_new = tf.keras.layers.Dense(units=10, activation="tanh")(attention_new)
attention_new = tf.keras.layers.Dense(units=6, activation="linear", name="Attention")(attention_new)

# Skip-Connection
response_new = tf.keras.layers.Dot(axes=1)([input_new, attention_new])

# Response Schicht = lokales GLM
response_new = tf.keras.layers.Dense(units=1, activation="linear", name="Response")(response_new)

local_glm_net_new = tf.keras.Model(inputs=input_new, outputs=response_new)

# Compute L2 regularization terms
l2_lambda = 0.01
l2_lambda_special = 0.005

l2_loss_all = 0.5 * l2_lambda * tf.reduce_sum([tf.reduce_sum(tf.square(w)) for w in local_glm_net_new.trainable_weights])

attention_weights = local_glm_net_new.get_layer("Attention").get_weights()[0]
l2_loss_special = 0.5 * l2_lambda_special * tf.reduce_sum(tf.square(attention_weights))

optimizer = Adam(learning_rate=0.01)
local_glm_net_new.compile(loss=lambda y_true, y_pred: custom_loss_with_l2(y_true, y_pred, l2_loss_all, l2_loss_special), optimizer=optimizer)
local_glm_net_new.summary()
```

```
Model: "model_20"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_4 (InputLayer)        [(None, 6)]               0         []

 dense_9 (Dense)             (None, 20)                140       ['input_4[0][0]']

 dense_10 (Dense)            (None, 15)                315       ['dense_9[0][0]']

 dense_11 (Dense)            (None, 10)                160       ['dense_10[0][0]']

 Attention (Dense)           (None, 6)                 66        ['dense_11[0][0]']

 dot_3 (Dot)                 (None, 1)                 0         ['input_4[0][0]',
                                                                  'Attention[0][0]']

 Response (Dense)            (None, 1)                 2         ['dot_3[0][0]']

==================================================================================================
Total params: 683
Trainable params: 683
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [87]:

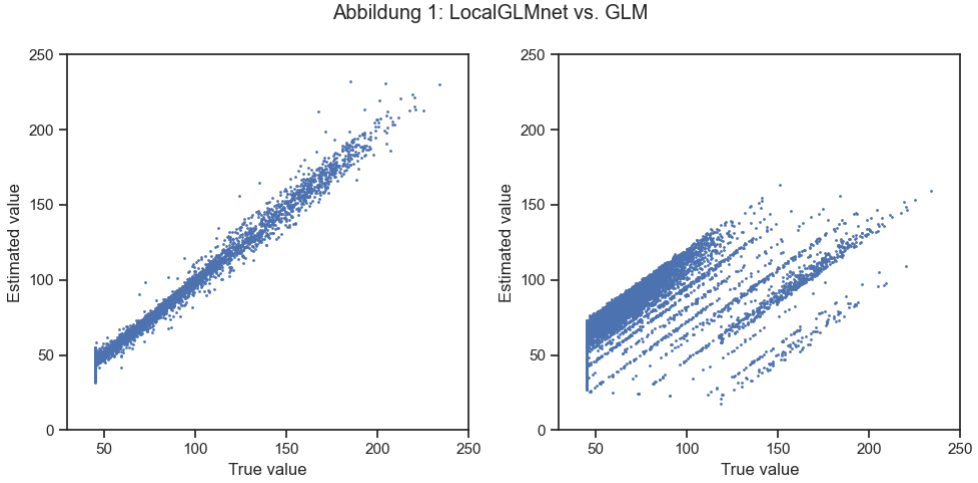
```
# Modell trainieren
history = local_glm_net_new.fit(
    x_train, y_train, batch_size=16, epochs=25, validation_split=0.2, verbose=1
)
```

```
Epoch 1/25
938/938 [==============================] - 2s 2ms/step - loss: 861.4366 - val_loss: 176.0651
Epoch 2/25
938/938 [==============================] - 1s 1ms/step - loss: 133.9854 - val_loss: 87.3687
Epoch 3/25
938/938 [==============================] - 1s 1ms/step - loss: 79.5560 - val_loss: 77.4334
Epoch 4/25
938/938 [==============================] - 1s 1ms/step - loss: 62.3833 - val_loss: 38.4805
Epoch 5/25
938/938 [==============================] - 1s 1ms/step - loss: 44.1624 - val_loss: 36.3241
Epoch 6/25
938/938 [==============================] - 1s 1ms/step - loss: 42.2114 - val_loss: 25.8590
Epoch 7/25
938/938 [==============================] - 1s 1ms/step - loss: 36.6468 - val_loss: 49.7175
Epoch 8/25
938/938 [==============================] - 1s 1ms/step - loss: 34.3831 - val_loss: 30.4106
Epoch 9/25
938/938 [==============================] - 1s 1ms/step - loss: 39.0712 - val_loss: 31.0605
Epoch 10/25
938/938 [==============================] - 1s 1ms/step - loss: 33.8081 - val_loss: 34.8442
Epoch 11/25
938/938 [==============================] - 1s 1ms/step - loss: 30.2122 - val_loss: 21.9268
Epoch 12/25
938/938 [==============================] - 1s 1ms/step - loss: 31.1102 - val_loss: 28.1271
Epoch 13/25
938/938 [==============================] - 1s 1ms/step - loss: 24.9692 - val_loss: 25.3884
Epoch 14/25
938/938 [==============================] - 1s 1ms/step - loss: 24.9718 - val_loss: 19.6067
Epoch 15/25
938/938 [==============================] - 1s 1ms/step - loss: 20.5172 - val_loss: 30.6501
Epoch 16/25
938/938 [==============================] - 1s 1ms/step - loss: 23.0583 - val_loss: 23.8025
Epoch 17/25
938/938 [==============================] - 1s 1ms/step - loss: 19.5332 - val_loss: 18.7282
Epoch 18/25
938/938 [==============================] - 1s 1ms/step - loss: 29.3288 - val_loss: 28.2917
Epoch 19/25
938/938 [==============================] - 1s 1ms/step - loss: 20.8458 - val_loss: 16.3759
Epoch 20/25
938/938 [==============================] - 1s 1ms/step - loss: 14.6206 - val_loss: 19.0058
Epoch 21/25
938/938 [==============================] - 1s 1ms/step - loss: 18.2446 - val_loss: 11.0289
Epoch 22/25
938/938 [==============================] - 1s 1ms/step - loss: 18.7595 - val_loss: 30.3513
Epoch 23/25
938/938 [==============================] - 1s 1ms/step - loss: 14.8523 - val_loss: 15.6584
Epoch 24/25
938/938 [==============================] - 1s 1ms/step - loss: 19.6707 - val_loss: 20.5858
Epoch 25/25
938/938 [==============================] - 1s 1ms/step - loss: 20.1556 - val_loss: 14.2447
```

In [88]:

```python
# Vorhersage mit localGLMnet und GLM
pred_local = local_glm_net_new.predict(x_test)
pred_reg = reg.predict(x_test)

fig_performance = plt.figure(tight_layout=True, figsize=(10, 5))

spec = GridSpec(ncols=2, nrows=1, figure=fig_performance)
axs_perf = [
    fig_performance.add_subplot(spec[0, 0]),
    fig_performance.add_subplot(spec[0, 1]),
]

axs_perf[0].scatter(y_test, pred_local, s=1)
axs_perf[1].scatter(y_test, pred_reg, s=1)

# Layout
for ax in axs_perf:
    ax.set_xlabel("True value")
    ax.set_ylabel("Estimated value")
    ax.set_xlim((30, 250))
    ax.set_ylim((0, 250))

fig_performance.suptitle("Abbildung 1: LocalGLMnet vs. GLM")
plt.show()

print("MSE LocalGLMnet: " + str(metrics.mean_squared_error(y_test, pred_local)))
print("MSE GLM: " + str(metrics.mean_squared_error(y_test, pred_reg)))
```

```
196/196 [==============================] - 0s 908us/step
```




```
MSE LocalGLMnet: 13.960251037487769
MSE GLM: 859.6596171638254
```

In [89]:
```
# Über die Methode get_weights() erhält man die Kantengewichte, sowie den Bias für jeder Schicht
# --> man erhält also eine Liste mit numpy Arrays die in der Länge der Anzahl der Ebenen * 2 entspricht
weights = local_glm_net_new.get_weights()
for i in weights:
    print(i.shape, end=" | ")
weights[7]
```

```
(6, 20) | (20,) | (20, 15) | (15,) | (15, 10) | (10,) | (10, 6) | (6,) | (1, 1) | (1,) |
```

Out[89]:
```
array([ 3.818381  , -2.0353596 , -6.827118  , -0.05665383, -0.08153754,
        0.20572531], dtype=float32)
```

In [90]:
```
# Neues Model ohne Response-Schicht --> ermöglicht auslesen der Attention Gewichte
weights_local_glm_new = tf.keras.Model(
    inputs=local_glm_net_new.input, outputs=local_glm_net_new.get_layer(name="Attention").output
)

# Gewichte bestimmen
beta_x = local_glm_net_new.predict(x_test)

# Skalierung der Attention-Gewichte mithilfe des Gewichts der Response Schicht ( = Intercept beta_0)
beta_x_scaled = beta_x * weights[7]
```

```
196/196 [==============================] - 0s 841us/step
```

In [91]:
```
print("Mittelwert β1: " + str(beta_x_scaled[:, 0].mean()))
print("Standardabweichung β1: " + str(beta_x_scaled[:, 0].std()))

# Intervalgrenzen bestimmen
alpha = 0.001
bound = stats.norm.ppf(alpha / 2) * beta_x_scaled[:, 0].std()

print("Quantil " + str(1 - alpha / 2) + ": " + str(stats.norm.ppf(alpha / 2)))
print("Grenzen: ± " + str(abs(bound)))
```

```
Mittelwert β1: 308.53595
Standardabweichung β1: 140.34357
Quantil 0.9995: -3.2905267314918945
Grenzen: ± 461.80425845937594
```

In [97]:

```
# Attention Plot
fig_attention = plt.figure(tight_layout=True, figsize=(30, 15))

# Gliederung der Subplots
spec = GridSpec(ncols=3, nrows=2, figure=fig_attention)
ax1_att = fig_attention.add_subplot(spec[0, 0])
ax2_att = fig_attention.add_subplot(spec[0, 1])
ax3_att = fig_attention.add_subplot(spec[0, 2])
ax4_att = fig_attention.add_subplot(spec[1, 0])
ax5_att = fig_attention.add_subplot(spec[1, 1])
ax6_att = fig_attention.add_subplot(spec[1, 2])
axs_att = [ax1_att, ax2_att, ax3_att, ax4_att, ax5_att, ax6_att]

# Ein Subplot pro Input Feature erstellen
for i in range(len(axs_att)):



    axs_att[i].hlines(y=0, xmin=-4, xmax=4, colors="red")

    # Intervalgrenzen
    interval = patches.Rectangle(
        xy=(-4, bound),
        height=2 * abs(bound),
        width=8,
        edgecolor="royalblue",
        facecolor="lightcyan",
        alpha=0.8,
        zorder=1,
    )
    axs_att[i].add_patch(interval)

    # Scatter Plot --> x: Werte der Inputfeatures, y: Attention Gewichte
    axs_att[i].scatter(x_test[:, i], beta_x_scaled[:, i], s=0.5, c="black")

    # Layout
    axs_att[i].set_xlim((-10, 10))
    axs_att[i].set_ylim((-200, 250))
    axs_att[i].set_xlabel(X.columns[i])
    axs_att[i].set_ylabel("Attention β: " + X.columns[i])

fig_attention.suptitle("Abbildung 2: Regression Attentions")
# Manuelle Achseneinstellung
# Gender
axs_att[0].set_xlim((-2, 2))
axs_att[0].set_ylim((0, 1000))
# Age
axs_att[1].set_xlim((-2.5, 2.5))
axs_att[1].set_ylim((-400, 0))
# SquareM
axs_att[2].set_xlim((-2.5, 2.5))
axs_att[2].set_ylim((-750, -250))
# HouseholdIncome
axs_att[3].set_xlim((-3, 3))
axs_att[3].set_ylim((-10, 5))
# InventoryValue
axs_att[4].set_xlim((-3, 3))
axs_att[4].set_ylim((-50, 25))
# Citytype
axs_att[5].set_xlim((-2, 2))
axs_att[5].set_ylim((0, 100))
plt.show()
```

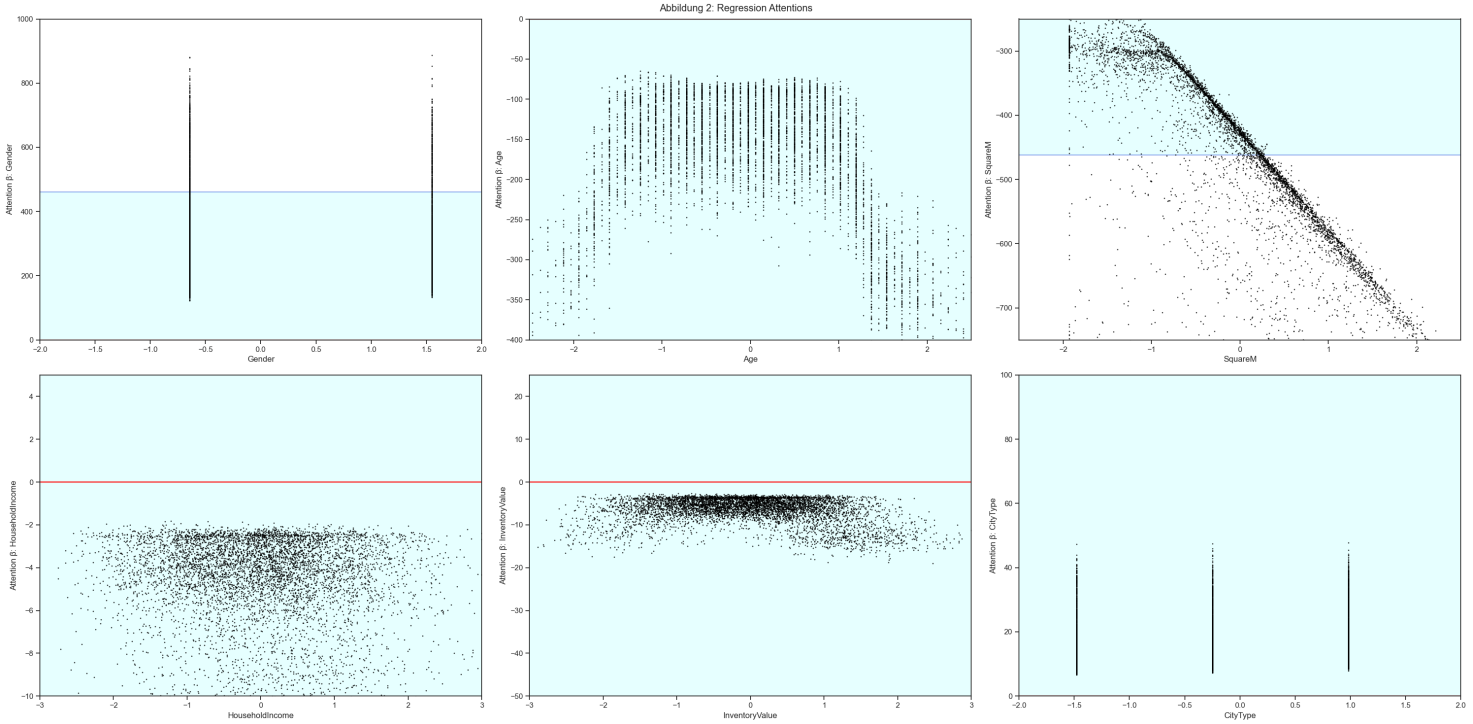


Abbildung 2: Regression Attentions

In den Attention Plots sticht neben der Beobachtung, dass hier erneut im Wesentlichen die Scatterplots " `$Merkmal` vs `Response` " zu erkennen sind, die Kombination von `Age` und `InventoryValue` in's Auge: Es ist gut zu erkennen, dass an den Rändern der Verteilung der Einfluss von `InventoryValue` besonders groß ist - weil die Nichtlinearität des Merkmals `Age` hier von dem Merkmal `InventoryValue` ersetzt wird - in der Tat also ein Hinweis auf noch immer bestehende indirekte Diskriminierung!

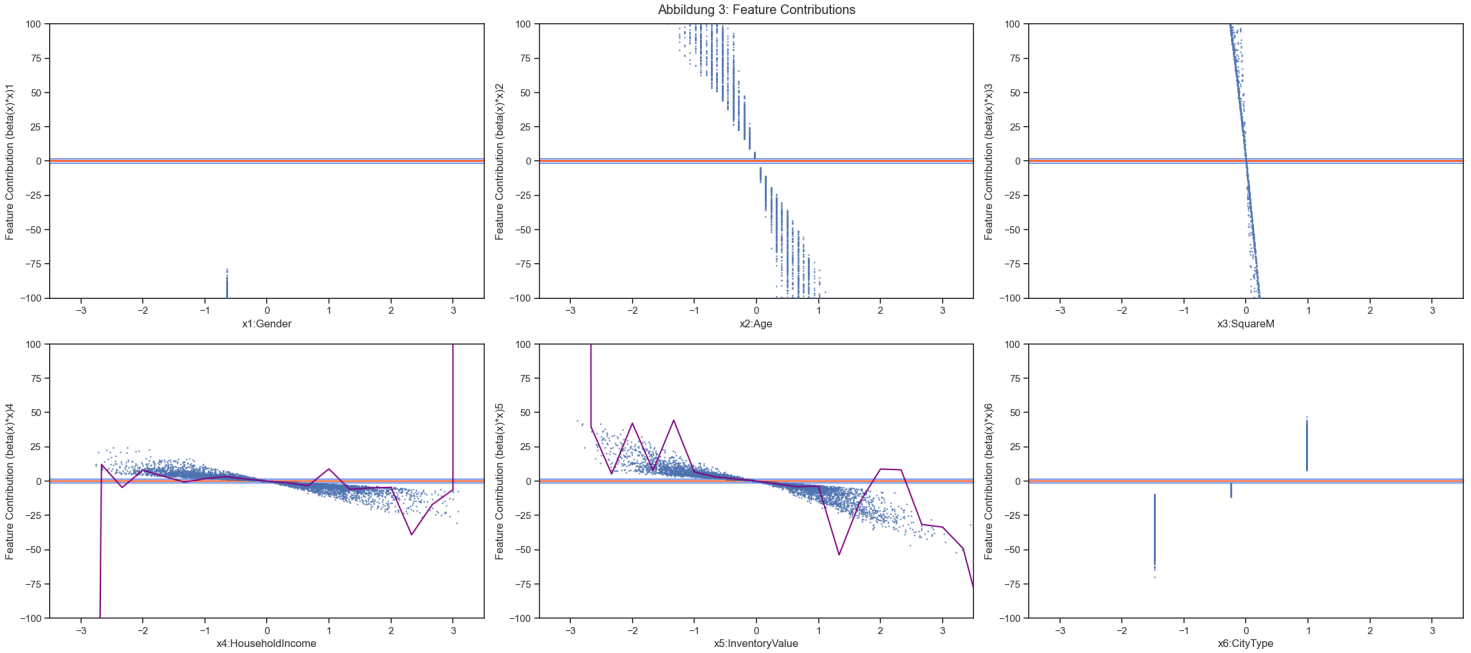
In [98]:
```
for i in range(6):
    if i != 6:
        size = beta_x_scaled.shape[0]
        coverage = np.count_nonzero(
            beta_x_scaled[:, i] < abs(bound)
        ) - np.count_nonzero(beta_x_scaled[:, i] < -abs(bound))
        coverage_ratio = coverage / size
        print("Coverage Ratio β" + str(i + 1) + ": " + str(coverage_ratio))
```

```
Coverage Ratio β1: 0.85776
Coverage Ratio β2: 0.99952
Coverage Ratio β3: 0.452
Coverage Ratio β4: 1.0
Coverage Ratio β5: 1.0
Coverage Ratio β6: 1.0
```

In [102…

```
# Feature Contribution Plot
fig_contribution = plt.figure(tight_layout=True, figsize=(30, 15))

spec = GridSpec(ncols=8, nrows=3, figure=fig_contribution)
ax1_con = fig_contribution.add_subplot(spec[0, 1:3])
ax2_con = fig_contribution.add_subplot(spec[0, 3:5])
ax3_con = fig_contribution.add_subplot(spec[0, 5:7])
ax4_con = fig_contribution.add_subplot(spec[1, 1:3])
ax5_con = fig_contribution.add_subplot(spec[1, 3:5])
ax6_con = fig_contribution.add_subplot(spec[1, 5:7])

axs_con = [ax1_con, ax2_con, ax3_con, ax4_con, ax5_con, ax6_con]

xs = np.linspace(-4, 4, 25)

for i in range(len(axs_con)):

    # Feature Contribution Splines berechnen
    # Feature Contribution = beta(xi)*xi
    contribution = np.column_stack([x_test[:, i], beta_x_scaled[:, i] * x_test[:, i]])
    con_ind = np.lexsort((contribution[:, 1], contribution[:, 0]))
    contribution_sorted = contribution[con_ind]
    con_spline = interpolate.UnivariateSpline(
        contribution_sorted[:, 0], contribution_sorted[:, 1]
    )

    # Hinzufügen von horizontalen Linien um die Stärke der Feature Contribution zu visualisieren
    axs_con[i].hlines(y=0, xmin=-4, xmax=4, colors="orange", alpha=0.7, zorder=1)

    axs_con[i].hlines(y=0.5, xmin=-4, xmax=4, colors="red", alpha=0.5, zorder=1)
    axs_con[i].hlines(y=-0.5, xmin=-4, xmax=4, colors="red", alpha=0.5, zorder=1)

    axs_con[i].hlines(y=1, xmin=-4, xmax=4, colors="lightcyan", alpha=0.7, zorder=1)
    axs_con[i].hlines(y=-1, xmin=-4, xmax=4, colors="lightcyan", alpha=0.7, zorder=1)

    axs_con[i].hlines(y=1.5, xmin=-4, xmax=4, colors="royalblue", alpha=0.7, zorder=1)
    axs_con[i].hlines(y=-1.5, xmin=-4, xmax=4, colors="royalblue", alpha=0.7, zorder=1)

    # Scatter Plot --> x: Werte der Inputfeatures, y:Feature Contribution (θ(x)*x)
    axs_con[i].scatter(contribution[:, 0], contribution[:, 1], s=0.5, zorder=10)

    # Feature Contribution Spline plotten
    axs_con[i].plot(xs, con_spline(xs), color="purple", zorder=20)

    # Layout
    axs_con[i].set_xlim((-3.5, 3.5))
    axs_con[i].set_ylim((-100, 100))
    axs_con[i].set_xlabel("x" + str(i + 1) + ":" + X.columns[i])
    axs_con[i].set_ylabel("Feature Contribution (beta(x)*x)" + str(i + 1))

# Manuelle Achseneinstellung
# Gender
#axs_con[0].set_xlim((-2, 2))
#axs_con[0].set_ylim((-1250, 650))
#axs_con[1].set_xlim((0, 3))
#axs_con[1].set_ylim((0, 1000))
#axs_con[2].set_xlim((0, 2.5))
#axs_con[2].set_ylim((0, 2000))
#axs_con[3].set_xlim((-3, 3))
#axs_con[3].set_ylim((-20, 20))
#axs_con[4].set_xlim((0, 3))
#axs_con[4].set_ylim((0,250))
#axs_con[5].set_xlim((-2, 2))
#axs_con[5].set_ylim((-150, 100))

fig_contribution.suptitle("Abbildung 3: Feature Contributions")
plt.show()
```

Abbildung 3: Feature Contributions



localGLMnet ohne Age und Gender

Wir erstellen nun ein weiteres Modell in dem wir die Merkmale `Age` und `Gender` wieder weglassen. Das Ziel ist hier erneut, die Anpassung zu prüfen, um dadurch ein Maß für die indirekte Diskriminierung zu erhalten. Die Erwartung ist, dass das mit dem localGLMnet besser gelingt, da eine explizite Modellierung der Interaktionen nicht erforderlich ist.

In [103…

```
input_wo_agegender = tf.keras.Input(shape=(4,), dtype="float32")

attention_wo_agegender = input_wo_agegender
attention_wo_agegender = tf.keras.layers.Dense(units=16, activation="tanh")(attention_wo_agegender)
attention_wo_agegender = tf.keras.layers.Dense(units=12, activation="tanh")(attention_wo_agegender)
attention_wo_agegender = tf.keras.layers.Dense(units=8, activation="tanh")(attention_wo_agegender)
attention_wo_agegender = tf.keras.layers.Dense(units=4, activation="linear", name="Attention")(attention_wo_agegender)

# Skip-Connection
response_wo_agegender = tf.keras.layers.Dot(axes=1)([input_wo_agegender, attention_wo_agegender])

# Response Schicht = lokales GLM
response_wo_agegender = tf.keras.layers.Dense(units=1, activation="linear", name="Response")(response_wo_agegender)

local_glm_net_wo_agegender = tf.keras.Model(inputs=input_wo_agegender, outputs=response_wo_agegender)

attention_weights_wo_agegender = local_glm_net_wo_agegender.get_layer("Attention").get_weights()[0]

optimizer = Adam(learning_rate=0.01)
local_glm_net_wo_agegender.compile(loss="mse", optimizer=optimizer)
local_glm_net_wo_agegender.summary()
```

```
Model: "model_22"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================================
 input_5 (InputLayer)        [(None, 4)]               0           []

 dense_12 (Dense)            (None, 16)                80          ['input_5[0][0]']

 dense_13 (Dense)            (None, 12)                204         ['dense_12[0][0]']

 dense_14 (Dense)            (None, 8)                 104         ['dense_13[0][0]']

 Attention (Dense)           (None, 4)                 36          ['dense_14[0][0]']

 dot_4 (Dot)                 (None, 1)                 0           ['input_5[0][0]',
                                                                   'Attention[0][0]']

 Response (Dense)            (None, 1)                 2           ['dot_4[0][0]']

=================================================================================================
Total params: 426
Trainable params: 426
Non-trainable params: 0
_________________________________________________________________________________________________
```

In [109…

```
# Modell trainieren
history = local_glm_net_wo_agegender.fit(
    x_train[:,2:], y_train, batch_size=32, epochs=25, validation_split=0.2, verbose=1
)
```

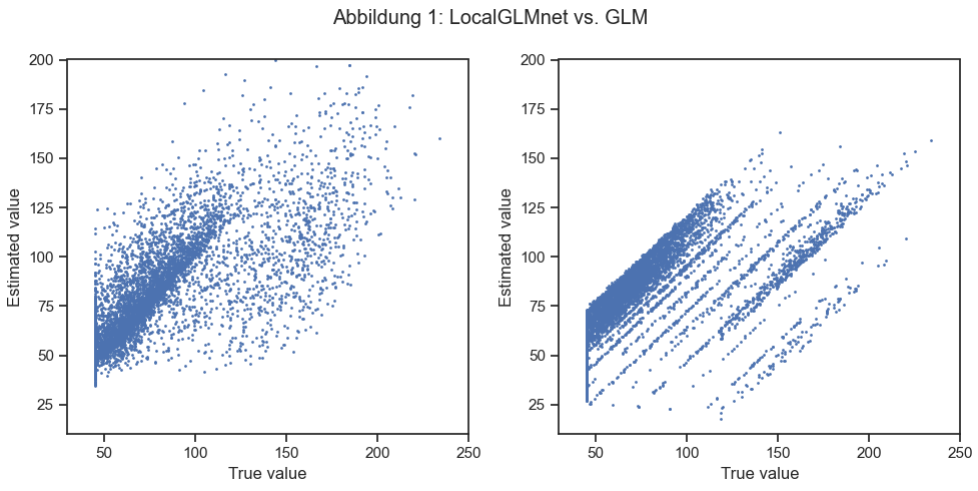
```
Epoch 1/25
469/469 [==============================] - 1s 1ms/step - loss: 590.6455 - val_loss: 612.3661
Epoch 2/25
469/469 [==============================] - 1s 1ms/step - loss: 592.8835 - val_loss: 615.4020
Epoch 3/25
469/469 [==============================] - 1s 1ms/step - loss: 592.3769 - val_loss: 611.2054
Epoch 4/25
469/469 [==============================] - 1s 1ms/step - loss: 590.8967 - val_loss: 613.5118
Epoch 5/25
469/469 [==============================] - 1s 1ms/step - loss: 589.2015 - val_loss: 614.6292
Epoch 6/25
469/469 [==============================] - 1s 1ms/step - loss: 590.3167 - val_loss: 612.0834
Epoch 7/25
469/469 [==============================] - 1s 1ms/step - loss: 589.3732 - val_loss: 609.5981
Epoch 8/25
469/469 [==============================] - 1s 1ms/step - loss: 589.0803 - val_loss: 611.1074
Epoch 9/25
469/469 [==============================] - 1s 1ms/step - loss: 589.1190 - val_loss: 612.8824
Epoch 10/25
469/469 [==============================] - 1s 1ms/step - loss: 592.6641 - val_loss: 609.2852
Epoch 11/25
469/469 [==============================] - 1s 1ms/step - loss: 591.8113 - val_loss: 614.1494
Epoch 12/25
469/469 [==============================] - 1s 1ms/step - loss: 590.2422 - val_loss: 613.5529
Epoch 13/25
469/469 [==============================] - 1s 1ms/step - loss: 591.8368 - val_loss: 614.8286
Epoch 14/25
469/469 [==============================] - 1s 1ms/step - loss: 591.6135 - val_loss: 613.4890
Epoch 15/25
469/469 [==============================] - 1s 1ms/step - loss: 591.6019 - val_loss: 610.7794
Epoch 16/25
469/469 [==============================] - 1s 1ms/step - loss: 591.4257 - val_loss: 624.8231
Epoch 17/25
469/469 [==============================] - 1s 1ms/step - loss: 594.4444 - val_loss: 609.0838
Epoch 18/25
469/469 [==============================] - 1s 1ms/step - loss: 591.9807 - val_loss: 612.0372
Epoch 19/25
469/469 [==============================] - 1s 1ms/step - loss: 588.4193 - val_loss: 616.5834
Epoch 20/25
469/469 [==============================] - 1s 1ms/step - loss: 590.4019 - val_loss: 613.7917
Epoch 21/25
469/469 [==============================] - 1s 1ms/step - loss: 591.4847 - val_loss: 617.8448
Epoch 22/25
469/469 [==============================] - 1s 1ms/step - loss: 591.6526 - val_loss: 613.0602
Epoch 23/25
469/469 [==============================] - 1s 1ms/step - loss: 590.3654 - val_loss: 610.1358
Epoch 24/25
469/469 [==============================] - 1s 1ms/step - loss: 588.7709 - val_loss: 610.2938
Epoch 25/25
469/469 [==============================] - 1s 1ms/step - loss: 589.5527 - val_loss: 610.9503
```

Anhand der Loss-Historie ist dies ein schönes Beispiel für ein Modell, das strukturell funktioniert (das wissen wir aus der vorherigen Anwendung), aber ohne die geeigneten Informationen - `Age` fehlt hier nun einmal, einfach nicht besser anpassen kann. Eine Vergrößerung der Anzahl Parameter o.ä. nützt nichts, wenn die erforderlichen Informationen nicht vorhanden sind.

In [110…

```python
# Vorhersage mit localGLMnet und GLM
pred_local = local_glm_net_wo_agegender.predict(x_test[:,2:])
pred_reg = reg.predict(x_test)

fig_performance = plt.figure(tight_layout=True, figsize=(10, 5))

spec = GridSpec(ncols=2, nrows=1, figure=fig_performance)
axs_perf = [
    fig_performance.add_subplot(spec[0, 0]),
    fig_performance.add_subplot(spec[0, 1]),
]

axs_perf[0].scatter(y_test, pred_local, s=1)
axs_perf[1].scatter(y_test, pred_reg, s=1)

# Layout
for ax in axs_perf:
    ax.set_xlabel("True value")
    ax.set_ylabel("Estimated value")
    ax.set_xlim((30, 250))
    ax.set_ylim((10, 200))

fig_performance.suptitle("Abbildung 1: LocalGLMnet vs. GLM")
plt.show()

print("MSE LocalGLMnet: " + str(metrics.mean_squared_error(y_test, pred_local)))
print("MSE GLM: " + str(metrics.mean_squared_error(y_test, pred_reg)))
```

```
196/196 [==============================] - 0s 836us/step
```



```
MSE LocalGLMnet: 602.7446191944493
MSE GLM: 859.6596171638254
```

In [67]:
```python
# Über die Methode get_weights() erhält man die Kantengewichte, sowie den Bias für jeder Schicht
# --> man erhält also eine Liste mit numpy Arrays die in der Länge der Anzahl der Ebenen * 2 entspricht
weights = local_glm_net_wo_agegender.get_weights()
for i in weights:
    print(i.shape, end=" | ")
weights[7]
```

```
(4, 16) | (16,) | (16, 12) | (12,) | (12, 8) | (8,) | (8, 4) | (4,) | (1, 1) | (1,) |
```

Out[67]:
```
array([5.751421  , 0.41202518, 2.6190398 , 0.24519682], dtype=float32)
```

In [68]:
```python
# Neues Model ohne Response-Schicht --> ermöglicht auslesen der Attention Gewichte
weights_local_glm_new = tf.keras.Model(
    inputs=local_glm_net_wo_agegender.input, outputs=local_glm_net_wo_agegender.get_layer(name="Attention").output
)

# Gewichte bestimmen
beta_x = local_glm_net_wo_agegender.predict(x_test[:,2:])

# Skalierung der Attention-Gewichte mithilfe des Gewichts der Response Schicht ( = Intercept beta_0)
beta_x_scaled = beta_x * weights[7]
```

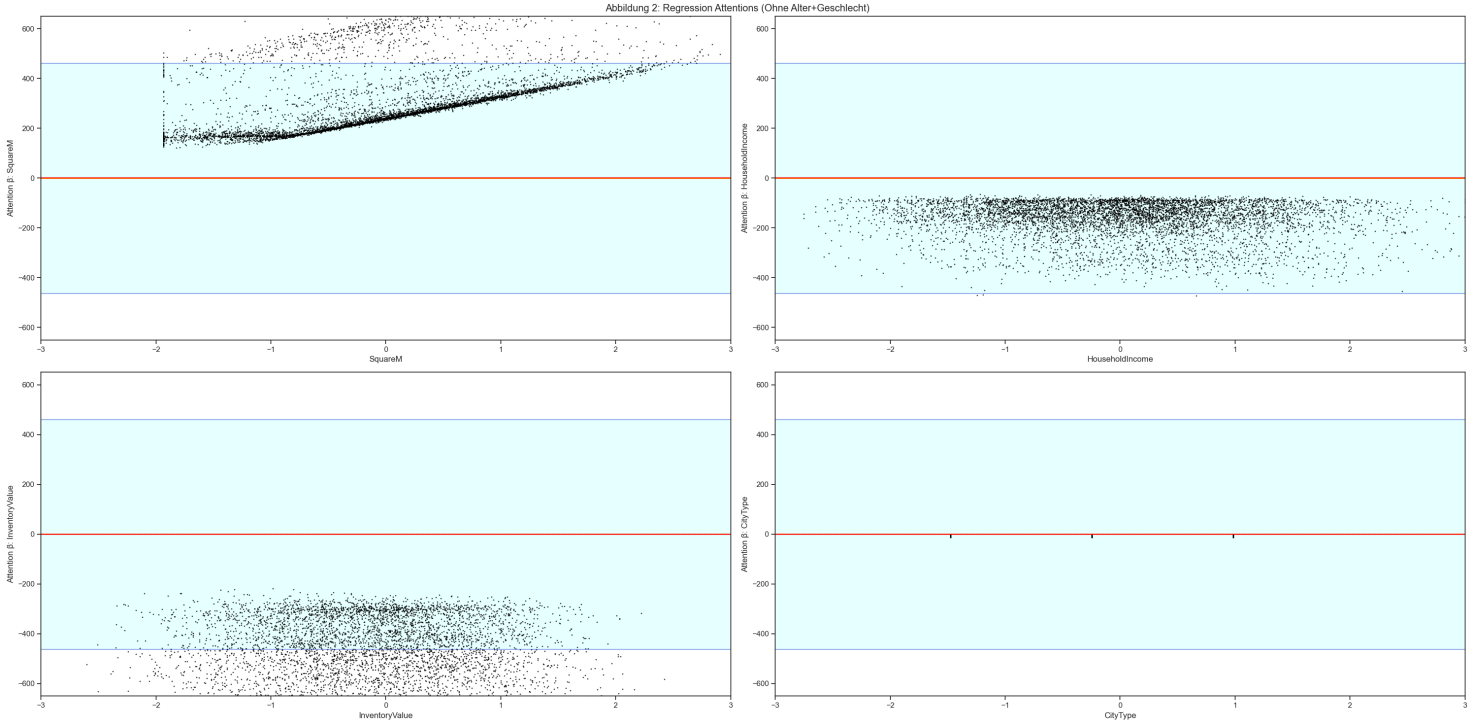
```
196/196 [==============================] - 0s 821us/step
```

In [69]:
```python
print("Mittelwert β1: " + str(beta_x_scaled[:, 0].mean()))
print("Standardabweichung β1: " + str(beta_x_scaled[:, 0].std()))

# Intervalgrenzen bestimmen
alpha = 0.001
bound = stats.norm.ppf(alpha / 2) * beta_x_scaled[:, 0].std()

print("Quantil " + str(1 - alpha / 2) + ": " + str(stats.norm.ppf(alpha / 2)))
print("Grenzen: ± " + str(abs(bound)))
```

```
Mittelwert β1: 464.97125
Standardabweichung β1: 164.94872
Quantil 0.9995: -3.2905267314918945
Grenzen: ± 542.7681567236201
```

In [113…
```python
# Attention Plot
fig_attention = plt.figure(tight_layout=True, figsize=(30, 15))

# Gliederung der Subplots
spec = GridSpec(ncols=2, nrows=2, figure=fig_attention)
ax1_att = fig_attention.add_subplot(spec[0, 0])
ax2_att = fig_attention.add_subplot(spec[0, 1])
ax3_att = fig_attention.add_subplot(spec[1, 0])
ax4_att = fig_attention.add_subplot(spec[1, 1])
axs_att = [ax1_att, ax2_att, ax3_att, ax4_att]

# Ein Subplot pro Input Feature erstellen
for i in range(len(axs_att)):

    # Linien zur Verdeutlichung der Höhe der Attention Gewichte
    axs_att[i].hlines(y=0.5, xmin=-4, xmax=4, colors="orange")
    axs_att[i].hlines(y=-0.5, xmin=-4, xmax=4, colors="orange")

    axs_att[i].hlines(y=0.25, xmin=-4, xmax=4, colors="orange", linestyles="dashed")
    axs_att[i].hlines(y=-0.25, xmin=-4, xmax=4, colors="orange", linestyles="dashed")

    axs_att[i].hlines(y=0, xmin=-4, xmax=4, colors="red")

    # Intervalgrenzen
    interval = patches.Rectangle(
        xy=(-4, bound),
        height=2 * abs(bound),
        width=8,
        edgecolor="royalblue",
        facecolor="lightcyan",
        alpha=0.8,
        zorder=1,
    )
    axs_att[i].add_patch(interval)

    # Scatter Plot --> x: Werte der Inputfeatures, y: Attention Gewichte
    axs_att[i].scatter(x_test[:, i+2], beta_x_scaled[:, i], s=0.5, c="black")

    # Layout
    axs_att[i].set_xlim((-3, 3))
    axs_att[i].set_ylim((-650, 650))
    axs_att[i].set_xlabel(X.columns[i+2])
    axs_att[i].set_ylabel("Attention β: " + X.columns[i+2])

fig_attention.suptitle("Abbildung 2: Regression Attentions (Ohne Alter+Geschlecht)")
plt.show()
```

Abbildung 2: Regression Attentions (Ohne Alter+Geschlecht)



In [71]:

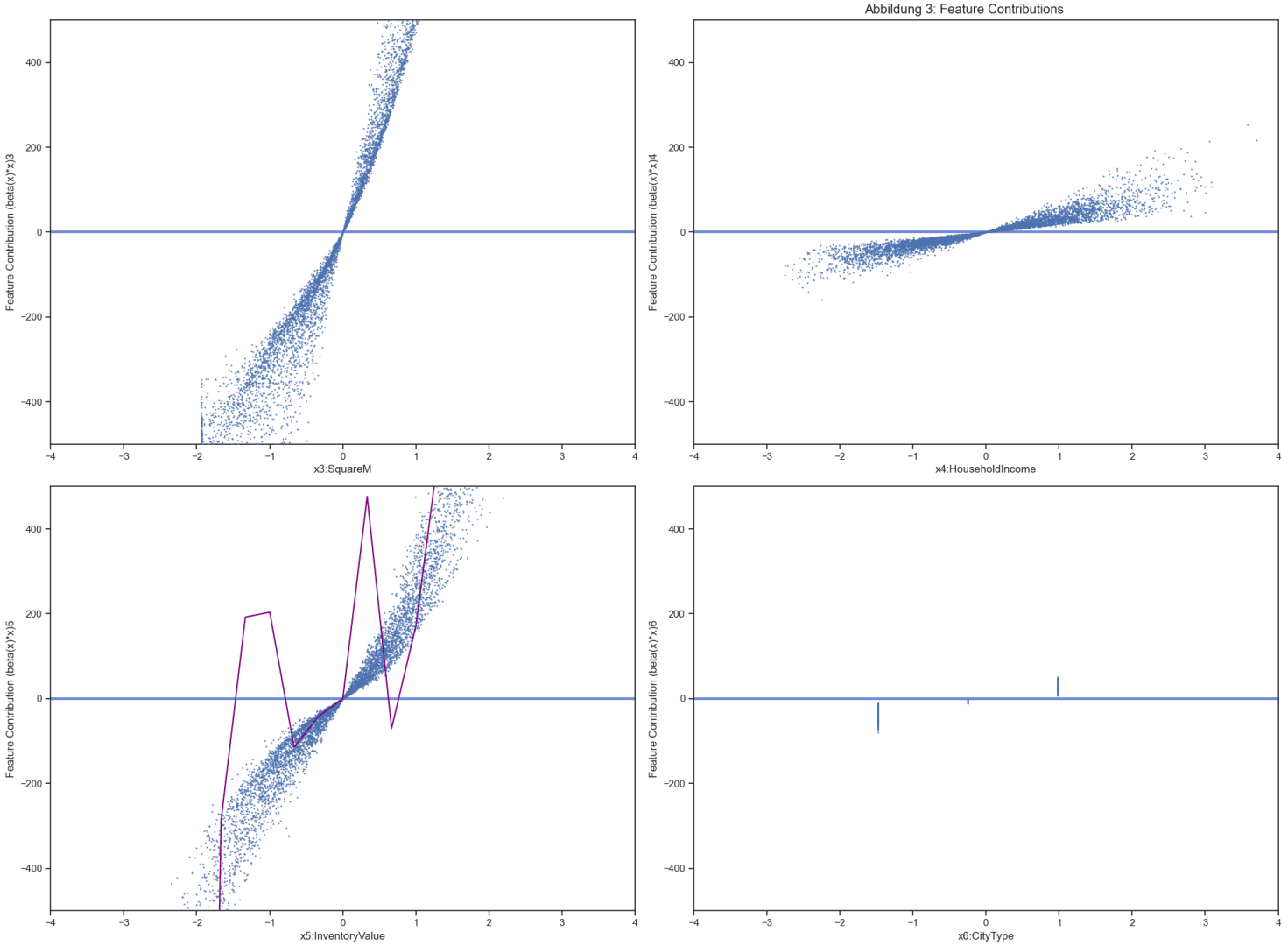
```
for i in range(4):
    if i != 6:
        coverage = np.count_nonzero(
            beta_x_scaled[:, i] < abs(bound)
        ) - np.count_nonzero(beta_x_scaled[:, i] < -abs(bound))
        coverage_ratio = coverage / size
        print("Coverage Ratio β" + str(i + 1) + ": " + str(coverage_ratio))
```

```
Coverage Ratio β1: 0.71792
Coverage Ratio β2: 1.0
Coverage Ratio β3: 0.99968
Coverage Ratio β4: 1.0
```

In [72]:

```
# Feature Contribution Plot
fig_contribution = plt.figure(tight_layout=True, figsize=(30, 15))

spec = GridSpec(ncols=6, nrows=2, figure=fig_contribution)
ax1_con = fig_contribution.add_subplot(spec[0, 0:2])
ax2_con = fig_contribution.add_subplot(spec[0, 2:4])
ax3_con = fig_contribution.add_subplot(spec[1, 0:2])
ax4_con = fig_contribution.add_subplot(spec[1, 2:4])

axs_con = [ax1_con, ax2_con, ax3_con, ax4_con]

xs = np.linspace(-4, 4, 25)

for i in range(len(axs_con)):

    # Feature Contribution Splines berechnen
    # Feature Contribution = beta(xi)*xi
    contribution = np.column_stack([x_test[:, i+2], beta_x_scaled[:, i] * x_test[:, i+2]])
    con_ind = np.lexsort((contribution[:, 1], contribution[:, 0]))
    contribution_sorted = contribution[con_ind]
    con_spline = interpolate.UnivariateSpline(
        contribution_sorted[:, 0], contribution_sorted[:, 1]
    )

    # Hinzufügen von horizontalen Linien um die Stärke der Feature Contribution zu visualisieren
    axs_con[i].hlines(y=0, xmin=-4, xmax=4, colors="orange", alpha=0.7, zorder=1)

    axs_con[i].hlines(y=0.5, xmin=-4, xmax=4, colors="red", alpha=0.5, zorder=1)
    axs_con[i].hlines(y=-0.5, xmin=-4, xmax=4, colors="red", alpha=0.5, zorder=1)

    axs_con[i].hlines(y=1, xmin=-4, xmax=4, colors="lightcyan", alpha=0.7, zorder=1)
    axs_con[i].hlines(y=-1, xmin=-4, xmax=4, colors="lightcyan", alpha=0.7, zorder=1)

    axs_con[i].hlines(y=1.5, xmin=-4, xmax=4, colors="royalblue", alpha=0.7, zorder=1)
    axs_con[i].hlines(y=-1.5, xmin=-4, xmax=4, colors="royalblue", alpha=0.7, zorder=1)

    # Scatter Plot --> x: Werte der Inputfeatures, y:Feature Contribution (θ(x)*x)
    axs_con[i].scatter(contribution[:, 0], contribution[:, 1], s=0.5, zorder=10)

    # Feature Contribution Spline plotten
    axs_con[i].plot(xs, con_spline(xs), color="purple", zorder=20)

    # Layout
    axs_con[i].set_xlim((-4, 4))
    axs_con[i].set_ylim((-500, 500))
    axs_con[i].set_xlabel("x" + str(i + 3) + ":" + X.columns[i+2])
    axs_con[i].set_ylabel("Feature Contribution (beta(x)*x)" + str(i + 3))

fig_contribution.suptitle("Abbildung 3: Feature Contributions")
plt.show()
```

Abbildung 3: Feature Contributions



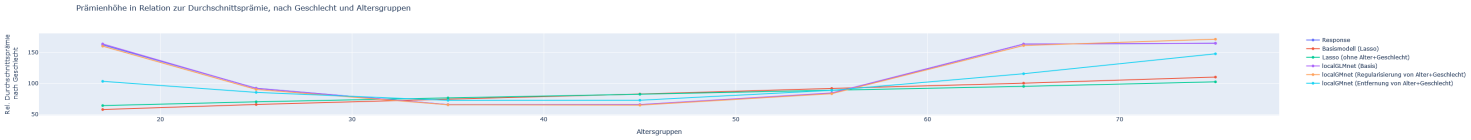
Ist das nun ein gutes Modell? Auf jeden Fall stellt man fest, dass eine Anpassung des Netz-Anteils (d.h. jenseits der Skip-Connection) auf die Residuen erfolgt und daher davon auszugehen ist, dass ein gewisser Teil der Information aus der Korrelation mit Alter in dem Modell mit abtgebildet sein wird.

Wir müssen selbstverständlich wieder die Vorhersagen nach Altersgruppen überprüfen.

In [145…

```
# avg_response_age_groups = (data_df_pred.groupby('AgeGroup').mean())['Response']
# age_group_index_dict = list(data_df_pred.groupby('AgeGroup').indices.items())
# age_group_indices_list = [] # Schon fertig
age_group_predictions_LGN = []
age_group_predictions_LGN_regularized = []
age_group_predictions_LGN_woagegender = []
for age_group in range(len(age_group_indices_list)):
    age_group_predictions_LGN.append((local_glm_net.predict(np.array(X_scaled[age_group_indices_list[age_group]]).reshape(-1,6))).mean())
    age_group_predictions_LGN_regularized.append((local_glm_net_new.predict(np.array(X_scaled[age_group_indices_list[age_group]]).reshape(-1,6))).mean())
    age_group_predictions_LGN_woagegender.append((local_glm_net_wo_agegender.predict(np.array(X_scaled[age_group_indices_list[age_group]][:,2:]).reshape(-1,4))).mean())

fig = go.Figure()
fig.add_trace(go.Scatter(x=age_groups, y=avg_response_age_groups, mode="lines+markers", name='Response'))
#fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lm, mode="lines+markers", name='Basismodell (LM)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lasso, mode="lines+markers", name='Basismodell (Lasso)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_lasso_wo_agegender, mode="lines+markers", name='Lasso (ohne Alter+Geschlecht)'))
#fig.add_trace(go.Scatter(x=age_groups, y=diff_age_groups_lm_wo_agegender, mode="lines+markers", name='Abweichung Response zum Lasso (ohne Alter+Geschlecht)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_LGN, mode="lines+markers", name='localGLMnet (Basis)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_LGN_regularized, mode="lines+markers", name='localGMnet (Regularisierung von Alter+Geschlecht)'))
fig.add_trace(go.Scatter(x=age_groups, y=age_group_predictions_LGN_woagegender, mode="lines+markers", name='localGMnet (Entfernung von Alter+Geschlecht)'))

fig.update_layout(title='Prämienhöhe in Relation zur Durchschnittsprämie, nach Geschlecht und Altersgruppen',
                  xaxis_title='Altersgruppen',
                  yaxis_title='Rel. Durchschnittsprämie<br> nach Geschlecht')

fig.show(renderer='vscode')
#fig.write_image("A5_all_models.png")
```

```
11/11 [==============================] - 0s 900us/step
11/11 [==============================] - 0s 1ms/step
11/11 [==============================] - 0s 1ms/step
89/89 [==============================] - 0s 841us/step
89/89 [==============================] - 0s 943us/step
89/89 [==============================] - 0s 886us/step
205/205 [==============================] - 0s 873us/step
205/205 [==============================] - 0s 966us/step
205/205 [==============================] - 0s 858us/step
243/243 [==============================] - 0s 884us/step
243/243 [==============================] - 0s 905us/step
243/243 [==============================] - 0s 876us/step
174/174 [==============================] - 0s 884us/step
174/174 [==============================] - 0s 913us/step
174/174 [==============================] - 0s 861us/step
56/56 [==============================] - 0s 927us/step
56/56 [==============================] - 0s 945us/step
56/56 [==============================] - 0s 927us/step
7/7 [==============================] - 0s 1ms/step
7/7 [==============================] - 0s 1ms/step
7/7 [==============================] - 0s 1ms/step
```



Antwort A5:

Wir erkennen hier also, dass auch mit starker L2-Regularisierung für die beiden fraglichen Parameter `Age` und `Gender` das *localGLMnet* die Altersverteilung fittet, da die restlichen Merkmale diese annähernd ausreichend erklären können. Diese prinzipiell beeindruckende Fähigkeit des neuronalen Netz zeigt hier beispielhaft auf, dass die Verwendung einer "punktweisen" (bezogen auf die Beobachtungen lokal wirkenden) Verlustfunktion der Loss nicht auf die Gruppe bezogen wird, und somit in der verwendeten Form nicht geeignet ist, die indirekte Diskriminierung zu vermeiden. Auch ganz ohne die beiden fraglichen Parameter wird die Altersverteilung annähernd gelernt, sodass die Entfernung *per se* die indirekte Diskriminierung sogar noch verstärkt. Im Vergleich mit den anderen Modellen könnte dies aber dennoch einen annehmbaren Kompromiss darstellen.

Um das Ziel dennoch zu erreichen wären verscheidene Ansätze wie oben diskutiert zu prüfen:

- **Gruppenbasierte Quantilsregression**: Dieses Vorgehen würde darauf setzen, dass die Prämie nach Alter je Gruppe identisch ist und aber *innerhalb* der Gruppe einen maximalen Spreiz erzeugen. Dies erzeugt Diskontinuitäten an den Gruppenrändern und deshalb ist dessen "Fairness" nur schwer zu begründen.
- **Korrektur der mittlere Abweichung an den Ränden**: Eine weitere Möglichkeit den Spreiz zu reduzieren, ist die Änderung des Regressionsparameters *ex post*. Dies hat natürlich zur Folge, dass insgesamt eine Unteranpassung eintritt und somit Teilkollektive entweder zu hoch oder zu tief eingeordnet sind. Daraus folgt entweder eine adverse Selektion von (zu günstig tarifierten) Risiken und/oder eine Vermeidung von Abschlüssen von zu teuer tarifierten "günstigen" Risiken, da diese am Markt eine bessere Option finden dürften.
- **Entfernung *aller* Merkmale mit indirekter Diskriminierung**: Selbstverständlich ist es möglich, auch alle indirekt diskriminierenden Merkmale aus dem Modell zu entfernen. Hierbei stellt sich dann die Frage, inwiefern eine ausreichende Risikodifferenzierung möglich ist. Die Hypothese nach praktischen Erwägungen lautet, dass dies den dynamischen Bereich der Differenzierung stark verengen würde und eine ähnliche Wirkung wie die Marginalkorrektur entfalten würde.
- **Verhaltensbasierte Versicherung (Usage Based Insurance)**: Gewissermaßen im Komplement des Lösungsraumes der drei bereits diskutierten Ansätze liegt das Themengebiet *Usage-based Insurance* (UBI): Anstatt auf diskriminierende Merkmale zu verzichten werden diese *direkt* auf Basis des (vermuteten) Generators der Beobachtungen betrachtet: Dem Risikoverhalten der Versicherten. Dies ist in Hausrat sicherlich schwerlich umzusetzen, da eine umfassende Erfassung innerhalb des geschützten Raumes der persönlichen Wohnung erforderlich wäre und hier große ethische-moralische und datenschutzrechtliche Hürden zu betrachten wären. Dort, wo sich die Risiken aber auf einige wenige Verhaltensweisen aggregieren lassen wie etwa in der *Kfz-Versicherung* (konkretes Verhalten bei der Führung des Fahrzeugs) sowie in der *Lebens-* und auch *Krankenversicherung* (Ernährung, Betrachtung riskanter und/oder rekreationeller Tätigkeiten) ist vorstellbar, dass dem Individuum konkrete Verantwortlichkeit für das Risikoverhalten unterstellt wird. Ob die Betrachtung dann "fair" im Sinne der Nicht-Diskriminierungszielsetzungen ist kann hier nicht abschließend bewertet werden, doch dafür hat die DAV (und andere Gremien wie GDV, EIOPA etc.) in den jeweiligen Sparten Stellungnahmen verfasst.

Zusammengefasst stellt keiner der skizzierten Ansätze eine zufriedenstellende Lösung dar, da die bestehenden Ungleichheiten sich im Tarifgefüge manifestieren und nicht nur "virtuell" zu beobachten sind. Eine Vermeidung dieser Situation ist daher an beiden Rändern des Entscheidungsspektrums nur durch Einheitstarife oder das konzeptionelle Verwerfen von indirekter Diskriminierung zu lösen - zufriedenstellend ist dies nach ethischen Maßstäben aber auch nicht unbedingt, denn stets wird es Grenz- und Härtefälle geben, deren Chrakteristika von der vorliegenden Preisfindung nicht oder sogar advers berücksichtigt werden. Deshalb scheint es umso wichtiger zu betonen, dass die bestehenden Ansätze ***im Kollektiv*** wirken und gerade ***nicht*** auf individueller Ebene.

Aufgabe 6 (15 Punkte)

Aufgabenstellung: Fassen Sie Ihre Erkenntnisse in einem Bericht von maximal zwei DinA4-Seiten zusammen, wovon 1/2 Seite Management Summary sein soll. Dies kann am Ende des abgegebenen Notebooks oder als separates PDF erfolgen.

Zusammenfassung

Management Summary

Zielsetzung:

- In dieser Arbeit wurde untersucht, inwieweit sich der vorliegende Hausrattarif diskriminierungsfrei umsetzen lässt.

Einleitende Erläuterungen:

- Die Analyse wurde vor dem Hintergrund der Verabschiedung des EU Artificial Intelligence Acts und etwaiger Rechtsunsicherheiten im Hinblick auf Auflagen hinsichtlich von sog. "Protected Attributes" durchgeführt. Die Ergebnisse sind hypothetischer Natur dahingehend, dass aktuell nicht anzunehmen ist, dass der entsprechende Paragraph im *VAG* vom AI Act außer Kraft gesetzt würden, auch wenn Versicherungstarifierung als (nicht-hochrisiko-)KI-Anwendung betrachtet würde, doch scheint eine Untersuchung zumindest aus Sicht des Risikomenagements relevant.

Ergebnis:

- Ein Hausrattarif kann ohne die direkt als geschützt geführten Attribute `Gender` und `Age` umgesetzt werden, würde aber ohne Merkmale, deren Wirkung als (pontenziell) indirekt diskriminierend zu bewerten ist, nur minimale Risikodifferenzierung erreichen.
- Der Effekt wäre mithin, dass eine kompetitive Teilnahme am Hausrat-Markt nicht länger vorstellbar ist.
- Es ergäben sich große Unsicherheit hinsichtlich einer Tendenz zum "Einheitstarif".
- In "teuren" Segmenten (d.h. mit hoher Risikoprämie) besteht das Risiko einer Negativselektion, also dem Zeichnen von unterbewerteten Risiken, wohingegen in Segmenten mit unterdurchschnittlichem Preis kein wettbewerbsfähiger Preis erreicht werden wird, aus dem im Umkehrschluss dann keine oder nur geringfügige Abschlusszahlen resultieren.
- Im Ergebnis dürfte sich die demografische Struktur des Portfolios sukzessive verschlechtern, sofern die nicht-diskriminierende Tarifierung beibehalten würde.

Ergänzende Informationen

Vorüberlegungen:

Laut EU Artificial Intelligence Act müssen Anwendungen der künstlichen Intelligenz nach *Annex 3* in *freiwilliger Selbstverpflichtung* **nicht-diskriminierend** gestaltet sein. Da davon auszugehen ist, dass die Versicherungswirtschaft per Branchenverband oder unternehmensindividuell dieser Selbstverpflichtung anschließend wird - und zugleich de jure mangels Präzedenz ungeklärt ist, ob (auch klassische) Tarifierungsmethoden *"Anwendungen der künstlichen Intelligenz"* sind - erscheint eine Untersuchung möglicher Auswirkungen präventiv ratsam, um nicht in kurzfristigen Zugzwang zu geraten. (*Anmerkung*: Hierbei stellt sich die Herausforderung, dass nicht nur die geschützten Attribute selbst, sondern auch sog. "indirekte Diskriminieung" zu evaluieren und ggf. vermeiden ist.)

Bestandsaufnahme:

Im Bestand gibt es sechs Vertragsmerkmale, die nicht ohne Korrelation sind. Es liegen zwei mögliche Zielgrößen, `freq` (Schadenfrequenz) und `Response` (Schadenbedarf) vor. Die Herkunft der Zielgrößen spielt hier keine Rolle, bzw. ist unbekannt.

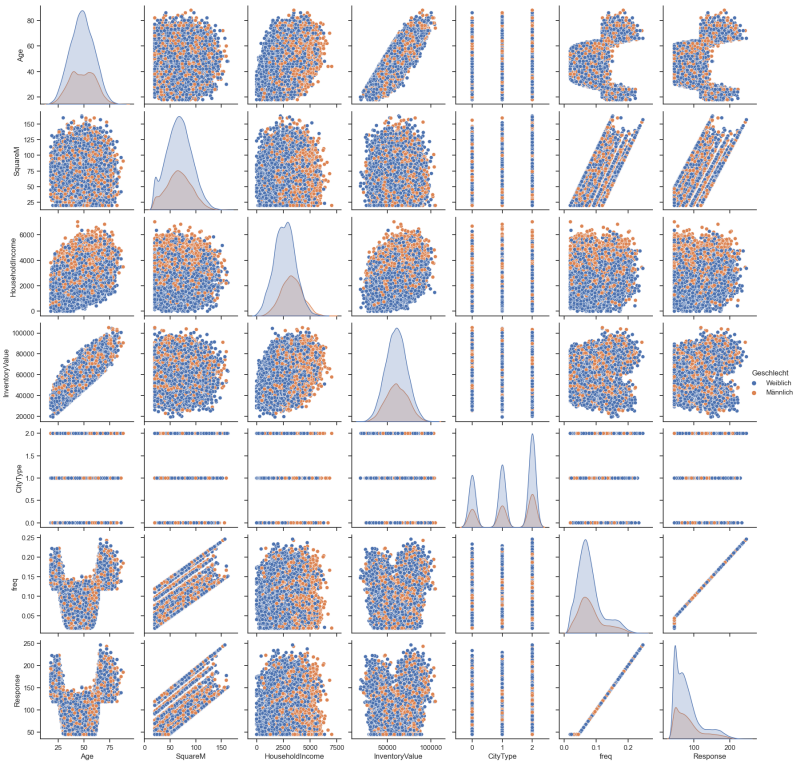


Abbildung 1. Pairplot der Merkmale und ihrer paarweisen Ausprägung

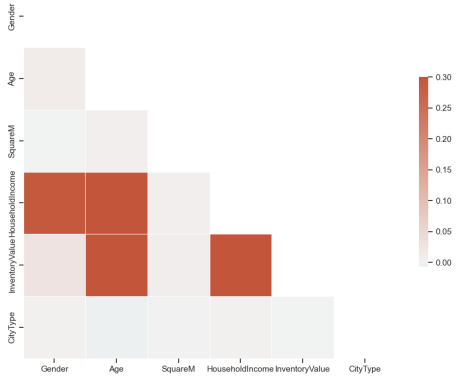


Abbildung 2. Korrelations-Heatmap zur Beschreibung der Abhängigkeiten erster Ordnung zwischen den Merkmalen

Methodik:

Es wurden die tarifrelevanten Merkmale `Gender`, `Age`, `SquareM`, `HouseholdIncome`, `InventoryValue` und `CityType` untersucht und Regressionsmodelle mittels der Ansätze `LinearRegression`, `Lasso` und `localGLMnet` trainiert und der Unterschied zwischen Einbezug von `Gender` und `Age` betrachtet. Zur Illustration bzw. Erklärbarkeit wurden *Partial Dependence Plots* und *Shapley-Values* verwendet.

Wie die Abbildung 3. zeigt, passen sich die Modelle entweder unter indirekter Diskriminierung an die Trainingsdaten an oder bleiben flach, haben also geringe Vorhersagekraft. Besonders bemerkenswert ist dies bei der Modellinstanz des `localGLMnet`, welches ohne die Merkmale `Age` und `Gender` trainiert die beste Anpassung erreicht und zugleich die größe indirekte Diskriminierung enthält.

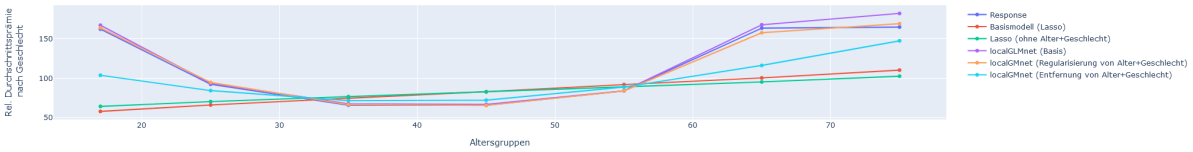


Abbildung 3. Darstellung der mittleren Prämie nach Altersgruppen für die verschiedenen geprüften Modelle

Ergebnisse:

- Ein Hausrattarif kann ohne die direkt als geschützt geführten Attribute `Gender` und `Age` umgesetzt werden, würde aber ohne Merkmale, deren Wirkung als (pontenziell) indirekt diskriminierend zu bewerten ist, nur minimale Risikodifferenzierung erreichen.
- Der Effekt wäre mithin, dass eine kompetitive Teilnahme am Hausrat-Markt nicht länger vorstellbar ist.
- Es ergäben sich große Unsicherheit hinsichtlich einer Tendenz zum "Einheitstarif".
- In "teuren" Segmenten (d.h. mit hoher Risikoprämie) besteht das Risiko einer Negativselektion, also dem Zeichnen von unterbewerteten Risiken, wohingegen in Segmenten mit unterdurchschnittlichem Preis kein wettbewerbsfähiger Preis erreicht werden wird, aus dem im Umkehrschluss dann keine oder nur geringfügige Abschlusszahlen resultieren.
- Im Ergebnis dürfte sich die demografische Struktur des Portfolios sukzessive verschlechtern, sofern die nicht-diskriminierende Tarifierung beibehalten würde.

Fazit:

Eine rechtliche Verpflichtung auf nicht-Diskriminierung im engsten vorstellbaren Wortsinne würde den kompetitiven Hausratmarkt sehr wahrscheinlich in Richtung zurück zu einem Einheitstarif verengen, da praktisch alle ökonomisch getriebenen Merkmale mit dem Merkmal `Alter` korrelieren.

Anhang: Literatur

[1] Lindholm M, Richman R, Tsanakas A, Wüthrich MV. DISCRIMINATION-FREE INSURANCE PRICING. ASTIN Bulletin. 2022;52(1):55-89. doi:10.1017/asb.2021.23
[2] Richman, R, Wüthrich, MV (2023), 'LocalGLMnet: interpretable deep learning for tabular data', Scandinavian Actuarial Journal, 2023(1), pp. 71–95. doi: 10.1080/03461238.2022.2081816.
[3] Transchel, F.: "Simpsons Paradoxon oder Die Tücken des diskriminierungsfreien Pricings" (2023), Vortrag, DAV-Herbsttagung 2023
[4] Knoop, P.: pyLocalGLMnet (2023), https://github.com/PK1706/pyLocalGLMnet

Block B (Krankenversicherung)

PK ADS Completion

2024-05-15

- Für diesen Block ist ein Notebook (R-Markdown oder Jupyter Notebook) mit R oder Python zu erstellen.
- Benötigte Materialien: Diagnosedaten.csv, Neue_Kunden.csv

Aufgabe 1. [*Business Case Krankenversicherung - Datenanalyse*] [10 Punkte]

Sie sind im Aktuariat der Krankenversicherung „AKK“, einem der innovativsten Unternehmen am Markt, tätig. Gemeinsam mit ihrem Team versuchen sie immer wieder neue Services für Ihre Kunden zu entwickeln. Im Zuge der Gesetzesänderung zur Digitalisierung im Gesundheitswesen, siehe https://www.bundesgesundheitsministerium.de/presse/pressemitteilungen/bundeskabinett-beschliesst-digitalgesetze-fuer-bessere-versorgung-und-forschung-im-gesundheitswesen erwarten auch Sie in Ihrem Unternehmen Veränderungen. Konkret dürfen künftig Versicherungsnehmer:innen kontaktiert werden, wenn ein erhöhtes Risiko für gewisse Krankheiten vermutet wird. Sie möchten diese nun klargestellte Möglichkeit nutzen und ihre Kunden darüber informieren und im Rahmen eines Case Managements versuchen den Ausbruch der Krankheit zu verhindern oder zumindest zu verzögern. Hierfür müssen Sie zunächst ein gutes Modell zur Erkennung potentieller, künftiger Krankheiten entwickeln. Die Idee ist, anhand verschiedener Merkmale wie Alter, Geschlecht, oder BMI, sowie der medizinischen Historie (bisherige Behandlungen) eine Vorhersage für künftige Krankheiten abzuleiten. Als Datengrundlage dient der Datensatz Diagnosedaten.csv. Dieser umfasst, neben Informationen zur versicherten Person eine Historie vergangener, durch die AKK erfasste Diagnosen.

```
library(tensorflow)
library(keras)
library(data.table)
library(dplyr)
library(wordcloud)
library(tm)
library(stringr)
library(magrittr)
library(pROC)
library(ggplot2)
library(ggrepel)
library(caret)
library(forcats)
```

Lesen Sie die Datei Diagnosedaten.csv ein. Die Spalten “D1”, “D2”, . . . , “D10” beschreiben für eine Historie von 10 Jahren die bisherigen Behandlungen - diese werden in Spalte “Historie” aggregiert. Bei den Werten handelt es sich um die ICD10-Kodierungen, welche jeweils für eine gewisse Krankheit stehen. Sind in einem Jahr mehrere Krankheiten erfasst worden, wird nur die schwerwiegendste gespeichert. Sind in einem Jahr keine Krankheiten erfasst worden, wird “ka” für “keine Angabe” gespeichert. Die gespeicherten Krankheiten sind auf grober Diagnoseebene erfasst (keine 4. und 5. Stellen oder Zusatzinformationen). Wird in einem Jahr beispielsweise „F10“ erfasst, war eine Behandlung aufgrund „Psychische und Verhaltensstörungen durch Alkohol“ in diesem Jahr am schwerwiegendsten. Führen Sie eine explorative Datenanalyse durch. Visualisieren Sie die zehn häufigsten Krankheiten im ersten Diagnosejahr. Was sind Auffälligkeiten von Rauchern im Vergleich zu Nichtrauchern? Achten Sie insbesondere auf akute Bronchitis (J20) und Lungenkrebs (C34).

Wie wahrscheinlich ist es, dass ein junger Kunde (<= 30 Jahre) eine Historie hat, in der mindestens einmal "ka" auftritt? Vergleichen Sie mit dem Altersbereich (>30 und <=50 Jahre) bzw. dem Altersbereich (>50). Erstellen Sie eine Übersicht der historischen Diagnosen, um eine bessere Übersicht zu bekommen.

```
data = read.csv("Diagnosedaten.csv",stringsAsFactors = T)

#Summary
summary(data)
```

```
##        ID        Geschlecht        BMI            Raucherstatus      Alter
##  Min.   :    1   Frau:4925   extrem  :2385   Nichtraucher:7929   Min.   :18
##  1st Qu.: 2504   Mann:5070   ok      :4463   Raucher     :2066   1st Qu.:26
##  Median : 5003               schlecht:3147                       Median :39
##  Mean   : 5003                                                   Mean   :39
##  3rd Qu.: 7502                                                   3rd Qu.:51
##  Max.   :10000                                                   Max.   :64
##
##        D1              D2              D3              D4              D5
##  kA     :2674   kA     :2735   kA     :2629   kA     :2714   kA     :2610
##  J20    : 962   J20    : 836   J20    : 813   J20    : 767   J20    : 769
##  F10    : 763   F10    : 700   F10    : 690   F10    : 673   F10    : 738
##  I10    : 520   I10    : 467   I10    : 429   C34    : 455   I10    : 433
##  F32    : 411   F32    : 361   C34    : 410   I10    : 414   C34    : 409
##  R07    : 391   R07    : 328   F32    : 365   C18    : 348   F32    : 353
##  (Other):4274   (Other):4568   (Other):4659   (Other):4624   (Other):4683
##        D6              D7              D8              D9             D10
##  kA     :2698   kA     :2700   kA     :2615   kA     :2699   kA     :2694
##  J20    : 810   J20    : 812   J20    : 763   J20    : 892   J20    : 931
##  F10    : 687   F10    : 696   F10    : 721   F10    : 765   F10    : 790
##  I10    : 451   C34    : 434   I10    : 449   I10    : 482   I10    : 512
##  C34    : 417   I10    : 432   C34    : 415   F32    : 407   F32    : 388
##  F32    : 372   F32    : 352   F32    : 377   R07    : 318   R07    : 357
##  (Other):4560   (Other):4569   (Other):4655   (Other):4432   (Other):4323
##                                      Historie
##  f10 ka ka f10 ka ka ka ka ka ka        :   2
##  ka ka ka ka ka ka c18 ka ka ka         :   2
##  a08 r10 ka o47 ka s06 ka ka a09 o99    :   1
##  a09 a09 j20 f60 ka ka s06 o06 ka r42   :   1
##  a09 b99 a09 s61 f10 r55 f19 k92 s32 j20:   1
##  a09 f10 c18 f14 ka f10 c18 g40 e10 f10 :   1
##  (Other)                                :9987
```

```
#Zeige die 10 häufigsten Krankheiten für D1
D1 = data %>%
  group_by(D1) %>%
  summarise(Anzahl = length(D1)) %>%
  arrange(desc(Anzahl))

ggplot(head(D1, n=10), aes(y=Anzahl, x=D1)) +
  geom_bar(stat="identity",color ="#A5A5A5",fill = "#70AD47") + theme_minimal()
```

```
#Summary nach Raucherstatus
R = data[which(data$Raucherstatus == "Raucher"),]
NR = data[which(data$Raucherstatus == "Nichtraucher"),]
summary(R)
```

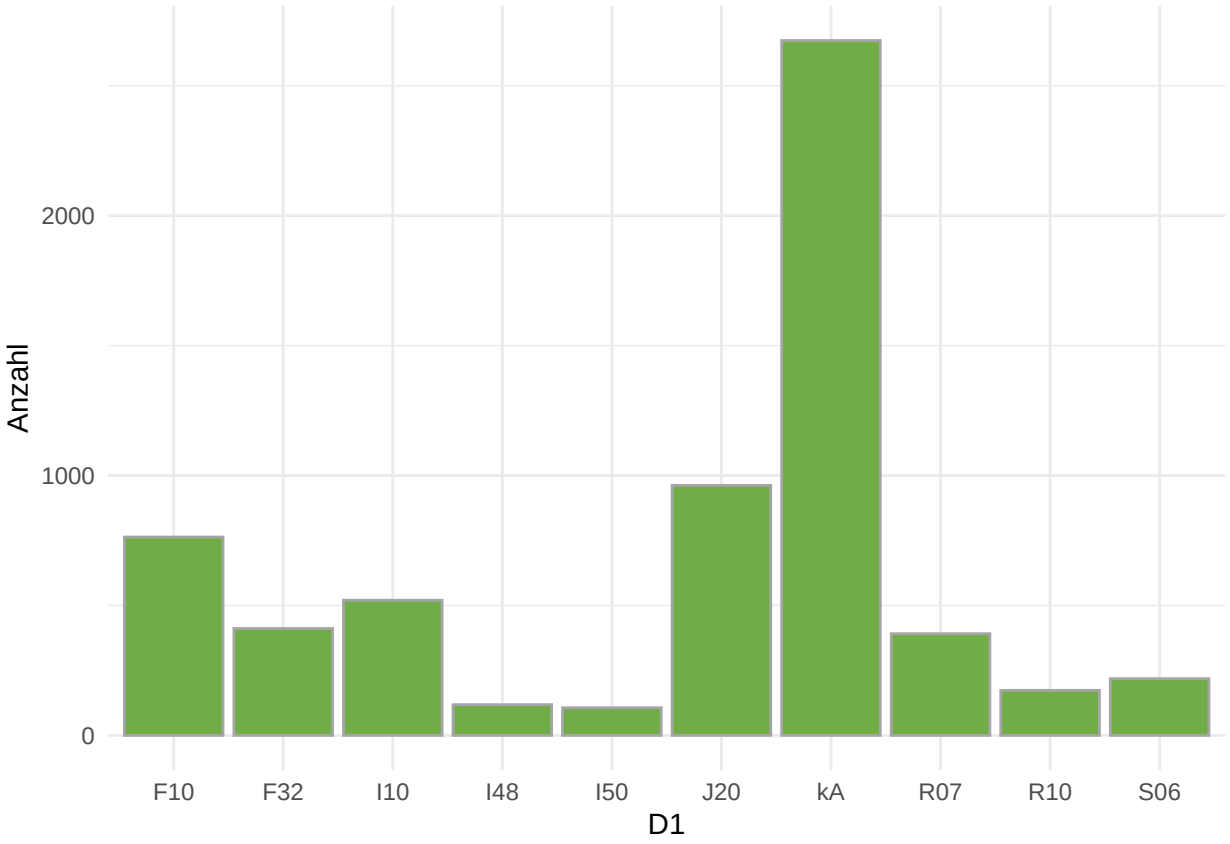


Figure 1: Verteilung Diagnosen im ersten Jahr.

```
##        ID       Geschlecht        BMI           Raucherstatus      Alter
##  Min.   :   1   Frau: 869   extrem  :528   Nichtraucher:   0   Min.   :18.00
##  1st Qu.:2428   Mann:1197   ok      :906   Raucher     :2066   1st Qu.:26.00
##  Median :4967               schlecht:632                       Median :37.00
##  Mean   :4954                                                  Mean   :38.13
##  3rd Qu.:7405                                                  3rd Qu.:49.00
##  Max.   :9999                                                  Max.   :64.00
##
##        D1            D2            D3            D4            D5
##  J20    :440   J20    :377   J20    :365   J20    :360   J20    :354
##  kA     :221   kA     :243   kA     :236   kA     :246   kA     :221
##  F10    :178   F10    :188   C34    :187   C34    :189   F10    :170
##  F32    :126   C34    :135   F10    :159   F10    :173   C34    :158
##  I50    : 95   I50    :102   F32    :105   I50    :104   I50    :124
##  R07    : 71   F32    : 90   I50    :100   F32    : 92   F32    :104
##  (Other):935   (Other):931   (Other):914   (Other):902   (Other):935
##        D6            D7            D8            D9           D10
##  J20    :358   J20    :355   J20    :329   J20    :428   J20    :443
##  kA     :207   kA     :237   kA     :240   kA     :219   kA     :235
##  C34    :187   C34    :185   C34    :176   F10    :190   F10    :174
##  F10    :162   F10    :159   F10    :148   F32    :119   F32    :112
##  F32    :115   I50    :107   F32    :118   I50    : 95   I50    : 94
##  I50    : 95   F32    :106   I50    : 98   I10    : 81   I10    : 71
##  (Other):942   (Other):917   (Other):957   (Other):934   (Other):937
##                                    Historie
##  a09 a09 j20 f60 ka ka s06 o06 ka r42   :   1
##  a09 b99 a09 s61 f10 r55 f19 k92 s32 j20:   1
##  a09 f10 ka r40 j20 a41 r07 j20 ka j20  :   1
##  a09 f15 ka ka j20 c34 o47 f10 f33 ka   :   1
##  a09 f20 r10 f15 i10 i50 s06 ka n20 s06 :   1
##  a09 i10 i50 o80 j20 ka ka j20 o80 r51  :   1
##  (Other)                               :2060
```

```
summary(NR)
```

```
##        ID        Geschlecht        BMI           Raucherstatus
##  Min.   :    2   Frau:4056   extrem  :1857   Nichtraucher:7929
##  1st Qu.: 2534   Mann:3873   ok      :3557   Raucher     :   0
##  Median : 5011               schlecht:2515
##  Mean   : 5016
##  3rd Qu.: 7522
##  Max.   :10000
##
##      Alter            D1            D2            D3            D4
##  Min.   :18.00   kA     :2453   kA     :2492   kA     :2393   kA     :2468
##  1st Qu.:26.00   F10    : 585   F10    : 512   F10    : 531   F10    : 500
##  Median :39.00   J20    : 522   J20    : 459   J20    : 448   J20    : 407
##  Mean   :39.23   I10    : 450   I10    : 392   I10    : 376   I10    : 348
##  3rd Qu.:51.00   R07    : 320   F32    : 271   R07    : 262   C18    : 269
##  Max.   :64.00   F32    : 285   R07    : 265   F32    : 260   C34    : 266
##                  (Other):3314   (Other):3538   (Other):3659   (Other):3671
##        D5            D6            D7            D8            D9
##  kA     :2389   kA     :2491   kA     :2463   kA     :2375   kA     :2480
##  F10    : 568   F10    : 525   F10    : 537   F10    : 573   F10    : 575
##  J20    : 415   J20    : 452   J20    : 457   J20    : 434   J20    : 464
```

```
##  I10    : 369   I10    : 389   I10    : 376   I10    : 373   I10    : 401
##  R07    : 269   R07    : 292   C18    : 266   C18    : 266   F32    : 288
##  C18    : 264   F32    : 257   R07    : 253   F32    : 259   R07    : 253
##  (Other):3655   (Other):3523   (Other):3577   (Other):3649   (Other):3468
##       D10                                          Historie
##  kA     :2459   f10 ka ka f10 ka ka ka ka ka ka       :   2
##  F10    : 616   ka ka ka ka ka ka c18 ka ka ka        :   2
##  J20    : 488   a08 r10 ka o47 ka s06 ka ka a09 o99   :   1
##  I10    : 441   a09 f10 c18 f14 ka f10 c18 g40 e10 f10:   1
##  R07    : 286   a09 f10 e10 c18 g40 ka j20 ka ka e10  :   1
##  F32    : 276   a09 f10 ka f15 e86 ka f10 s06 j20 f10 :   1
##  (Other):3363   (Other)                               :7921
```

```
sum(grepl("ka",R$Historie))/dim(R)[1]
```

```
## [1] 0.6040658
```

```
sum(grepl("ka",NR$Historie))/dim(NR)[1]
```

```
## [1] 0.9134822
```

```
sum(grepl("j20",R$Historie))/dim(R)[1]
```

```
## [1] 0.8247822
```

```
sum(grepl("j20",NR$Historie))/dim(NR)[1]
```

```
## [1] 0.4342288
```

```
sum(grepl("c34",R$Historie))/dim(R)[1]
```

```
## [1] 0.4927396
```

```
sum(grepl("c34",NR$Historie))/dim(NR)[1]
```

```
## [1] 0.2057006
```

```
#Summary nach Alter
J = data[which(data$Alter <= 30),]
M = data[which(data$Alter > 30 & data$Alter <= 50),]
A = data[which(data$Alter > 50),]

sum(grepl("ka",J$Historie))/dim(J)[1]
```

```
## [1] 0.9717598
```

```
sum(grepl("ka",M$Historie))/dim(M)[1]
```

```
## [1] 0.8851436
```

```
sum(grepl("ka",A$Historie))/dim(A)[1]
```

```
## [1] 0.6386139
```

```
#Weitere Plots
data %<>% mutate(BMI = fct_relevel(BMI,"ok","schlecht"))
ggplot(data,aes(BMI)) + geom_bar(color ="#A5A5A5",
                                 fill = "#70AD47") + theme_minimal()
```

```
#Weitere Plots
ggplot(data,aes(Alter)) + geom_histogram(color ="#A5A5A5",
```

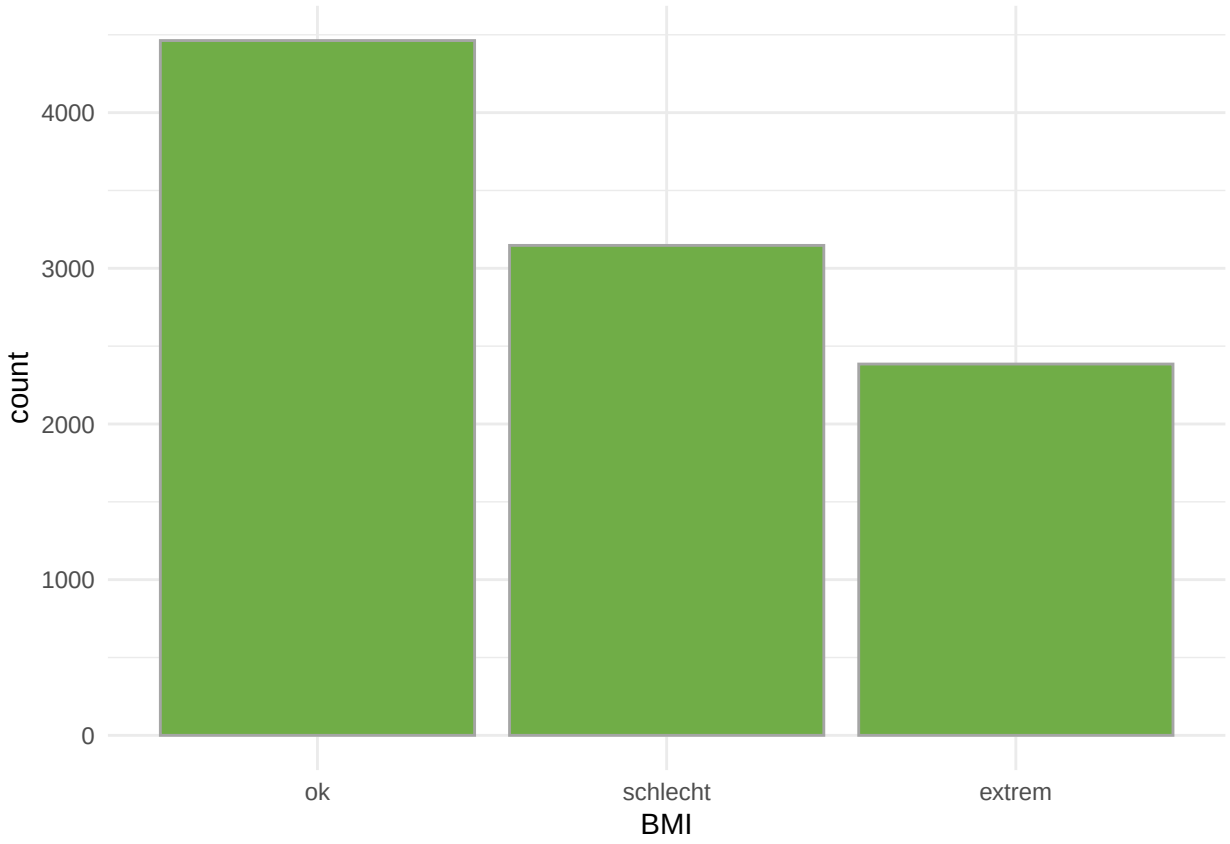
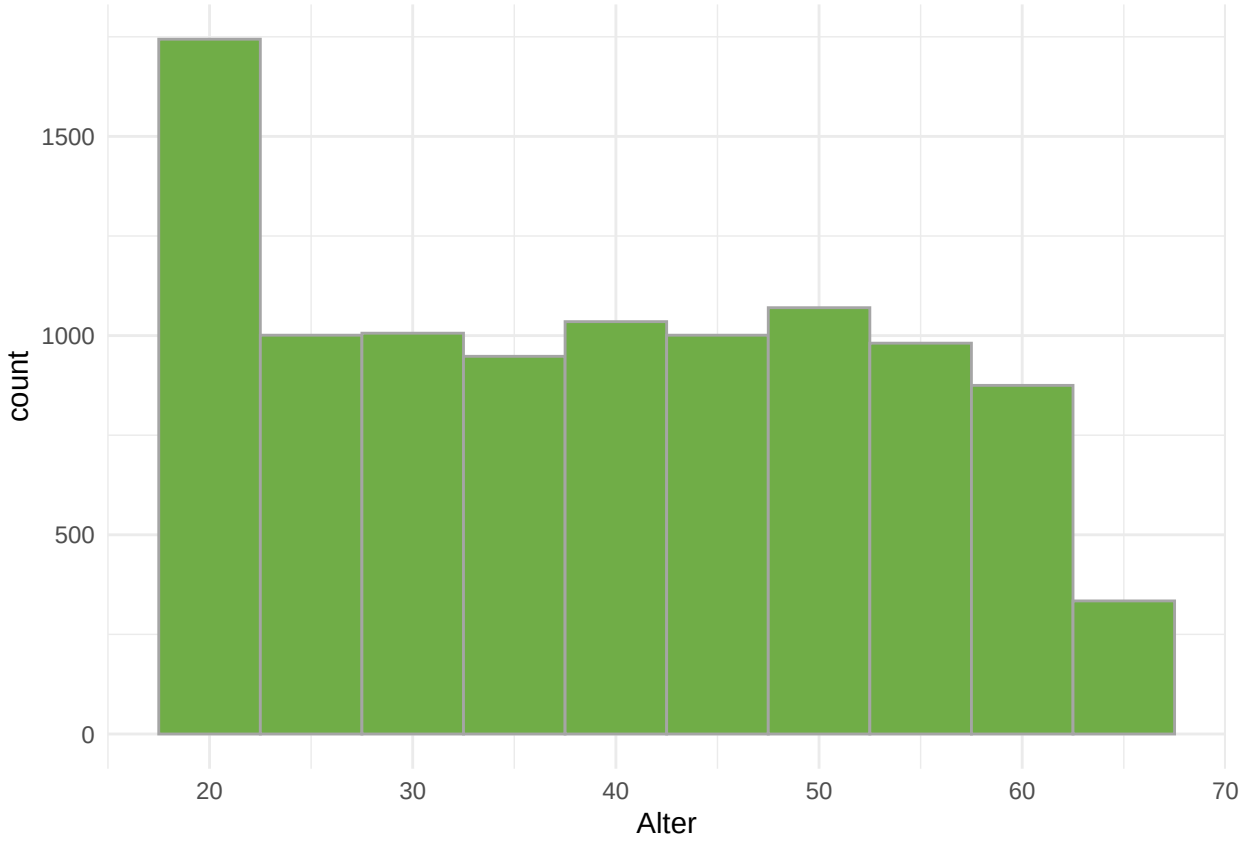


Figure 2: Verteilung BMI

```
                                    fill = "#70AD47",binwidth = 5) + theme_minimal()
```



Antwort [Raucher / Nichtraucher]

- Raucher werden häufig aufgrund J20 (Bronchitis) behandelt.
- Raucher werden häufig aufgrund C34 (Lungenkrebs) behandelt.
- Raucher werden generell häufiger behandelt.
- weitere Auffälligkeiten in Bezug auf Merkmale nicht beobachtbar.

Antwort [Alter]

Die Wahrscheinlichkeiten mindestens in einem Jahr ohne Krankheit zu bleiben ist: - Für junge Kunden: 97%

- Für mittelalte Kunden: 89%
- Für alte Kunden: 64%

```
#Wordcloud
wordcloud = data %>%
  mutate(text = str_c(Historie,sep = " "))

words <- Corpus(VectorSource(wordcloud$text))
dtm <- TermDocumentMatrix(words,control=list(wordLengths = c(2,3)))
matrix <- as.matrix(dtm)
words <- sort(rowSums(matrix),decreasing=TRUE)
cloud <- data.frame(word = names(words),freq=words)

wordcloud(words = cloud$word, freq = cloud$freq,
```

```
          min.freq = 1,max.words=200,
          random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

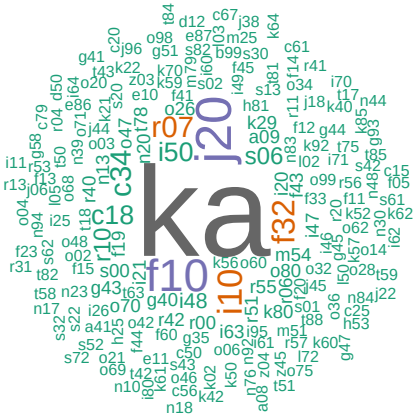


Figure 3: Word Cloud der Diagnosen

Aufgabe 2. [*Business Case Krankenversicherung - Textbasiertes neuronales Netz*] [35 Punkte]

Entwickeln Sie ein neuronales Netz als "Language Model", welches textbasierte Vorhersagen zu einer Person unter Berücksichtigung der medizinischen Historie erstellt. Die medizinische Historie soll als "Satz" aufgefasst werden und das Modell soll die nächsten "Wörter" (Diagnosen) dieses "Satzes" fortschreiben bzw. vorhersagen.

(a) *Datenaufbereitung und Vorbereitung für das Language Model*

Die medizinischen Historien aller Kunden sollen gemeinsam betrachtet werden. Die einzelnen "Sätze" werden also in einen gemeinsamen, längeren "Text" geschrieben. Prognosen werden jeweils nur innerhalb eines "Satzes" getroffen.

```
categories <- unique(data$ID)
#Gesamthistorie je Gruppe
tracs_collect <- c()
for (cats in categories) {
  tracs_df <- data[which(data$ID == cats),]
  tracs <- " "
  tracs <- tracs_df[, 'Historie']
  tracs_collect <- append(tracs_collect, tracs)
}
```

Diskretisieren Sie den Text für die weitere Verarbeitung mit Hilfe eines Tokenizers. Wählen Sie dabei eine zur Fragestellung passende Größe des Vokabulars.

```
#Tokenizer bricht den String in einzelne Tokens
vocab_size <- length(words)+1   #etwas groesser als die anzahl an worten
oov_token <- "<OOV>"
tokenizer <- text_tokenizer(num_words = vocab_size, oov_token = oov_token)
```

```
## Traceback (most recent call last):
##   File "C:\Users\jschupp\DOCUME~1\VIRTUA~1\R-TENS~1\lib\site-packages\tensorflow\python\pywrap_tenso
##     import ssl
##   File "C:\Users\jschupp\AppData\Local\R\win-library\4.3\reticulate\python\rpytools\loader.py", line
##     return _run_hook(name, _hook)
##   File "C:\Users\jschupp\AppData\Local\R\win-library\4.3\reticulate\python\rpytools\loader.py", line
##     module = hook()
##   File "C:\Users\jschupp\AppData\Local\R\win-library\4.3\reticulate\python\rpytools\loader.py", line
##     return _find_and_load(name, import_)
##   File "C:\PROGRA~3\ANACON~1\lib\ssl.py", line 98, in <module>
##     import _ssl             # if we can't import it, let the error propagate
##   File "C:\Users\jschupp\AppData\Local\R\win-library\4.3\reticulate\python\rpytools\loader.py", line
##     return _run_hook(name, _hook)
##   File "C:\Users\jschupp\AppData\Local\R\win-library\4.3\reticulate\python\rpytools\loader.py", line
##     module = hook()
##   File "C:\Users\jschupp\AppData\Local\R\win-library\4.3\reticulate\python\rpytools\loader.py", line
##     return _find_and_load(name, import_)
## ImportError: DLL load failed while importing _ssl: Das angegebene Modul wurde nicht gefunden.
##
##
## Warning: Failed to load ssl module. Continuing without ssl support.
```

```
fit_text_tokenizer(tokenizer, tracs_collect)
#Word Index sortiert nach Häufigkeit, z.B. frau ist am häufigsten, dann nichtraucher, etc.
word_index = tokenizer$word_index

#Sequences mappt zum Text jeder ID die zugehörige Häufigkeit
sequences <- texts_to_sequences(tokenizer, tracs_collect)
```

Im weiteren Verlauf möchten Sie nun aus vier Diagnosen die nun folgende fünfte Diagnose vorhersagen. Zerlegen Sie dazu den Text rollierend in Elemente der Länge 5 (N-Gram-Sequence). Führen Sie diese anschließend in eine Tabelle zusammen. Je versicherter Person entstehen durch diese Vorgehensweise aus den vorliegenden 10 Diagnosen 6 Beobachtungen für die Modellbildung.

```
#N gram Sequence
#Länge der Kette
m = 5

n_gram_sequences <- c()

#IDs mitzählen um nachher die Dimensionen einheitlich zu erhalten
ID = c()
id = 0

for (line in sequences){
  id = id+1
  for (index in 1:(length(line) - (m-1))){
    n_gram_sequences <- append(n_gram_sequences,list(line[index:(index + (m-1))]))
```

```
    ID <- append(ID,list(id))
  }
}


#Transformiere Liste aus Listen (n_gram_sequences)
#zur selben Länge und schreibe es in eine Tabelle

padded <- pad_sequences(n_gram_sequences, maxlen = m)
IDs <- pad_sequences(ID,maxlen = 1)
```

Bereiten Sie den Datensatz für das "Language Model" vor. Zerlegen Sie den Datensatz in Training und Test (stratifiziert).

```
set.seed(10)

#Letzte Spalte definiert Labels
labels <-padded[,ncol(padded)]

#Rest definiert Inputs
inputs <- padded[,(-ncol(padded))]

#One-Hot-Encoding
encoded_labels <- to_categorical(y = labels, num_classes = vocab_size)

#Zerlegen in Training und Testing
test_indices <- createDataPartition(labels, p = .2, list = FALSE)

#Feature nicht One-Hot-Encoden
x_test <- inputs[test_indices,]
x_train <- inputs[-test_indices,]

##weitere Feautures
scale_Age = min(data$Alter)
X_age = data$Alter[IDs]-scale_Age
X_bmi = ifelse(data$BMI[IDs]=="ok",0L,ifelse(data$BMI[IDs]=="extrem",1L,2L))
X_sex= ifelse(data$Geschlecht[IDs]=="Mann",1L,0L)

X_age_test = X_age[test_indices]
X_bmi_test = X_bmi[test_indices]
X_sex_test= X_sex[test_indices]

X_age_train = X_age[-test_indices]
X_bmi_train = X_bmi[-test_indices]
X_sex_train= X_sex[-test_indices]

x_train = list(as.matrix(x_train),
               as.matrix(X_age_train),
               as.matrix(X_bmi_train),
               as.matrix(X_sex_train))

x_test = list(as.matrix(x_test),
              as.matrix(X_age_test),
              as.matrix(X_bmi_test),
```

```
              as.matrix(X_sex_test))



#normal
labels_test = labels[test_indices]
labels_train = labels[-test_indices]

#und für analysen als faktor
labels_test_cat = factor(labels_test)
labels_train_cat = factor(labels_train)

#One-Hot-Encoden (ersten beiden Elemente löschen,
#da "to_categorical" implizit bei 0 startet und nicht bei 2)

y_train = to_categorical(labels_train)[,-c(1,2)]
y_test = to_categorical(labels_test)[,-c(1,2)]
```

(b) *Neuronales Netz*

Definieren Sie ein neuronales Netz und kalibrieren Sie dieses. Wählen Sie eine geeignete Parametrisierung (Hinweis: ein ausführliches Parameter-Tuning ist nicht notwendig). Das Modell soll aus vier Diagnosen die nun folgende fünfte Diagnose vorhersagen. Berücksichtigen Sie folgende Modellstruktur:

Modellstruktur:

- Eingabeschicht
  - vergangene Diagnosen
    - ein Embedding um die Dimension des Vokabulars zu reduzieren (Output-Dimension 2),
    - mehrere LSTM-Units um die sequentielle Struktur der medizinischen Information abzudecken,
  - Informationen zur versicherten Person
    - je ein Embedding für das Alter, das Geschlecht und den BMI (Output-Dimension jeweils 2)
- Zwischenschicht
  - mind. einen weiteren Layer mit Regularisierung,
- Ausgabeschicht
  - Ausgabe von Wahrscheinlichkeiten für alle Diagnosen.

Geben Sie die Modellstruktur aus und kalibrieren Sie anschließend das Modell.

```
#Teil A:

TRACS = layer_input(dtype = 'float32',shape = c(m-1))
modelTRACS = TRACS %>% layer_embedding(input_dim = vocab_size, output_dim = 2)
modelTRACS = modelTRACS %>%  bidirectional(layer_lstm(units = 8))

x_TRACS = keras_model(inputs = c(TRACS), outputs = c(modelTRACS))

#Teil B:

AGE = layer_input(dtype = 'int32',shape = c(1))
modelAGE = AGE %>% layer_embedding(input_dim = n_distinct(data$Alter),
                                   output_dim = 2,input_length = 1) %>%
```

```
  layer_flatten()
x_AGE = keras_model(inputs = list(AGE), outputs = c(modelAGE))


BMI = layer_input(dtype = 'int32',shape = c(1))
modelBMI = BMI %>% layer_embedding(input_dim = 3,
                                   output_dim = 2,input_length = 1) %>%
  layer_flatten()
x_BMI = keras_model(inputs = list(BMI), outputs = c(modelBMI))


SEX = layer_input(dtype = 'int32',shape = c(1))
modelSEX = SEX %>% layer_embedding(input_dim = 2,
                                   output_dim = 2,input_length = 1) %>%
  layer_flatten()
x_SEX = keras_model(inputs = list(SEX), outputs = c(modelSEX))


#Inputs kombinieren und Zwischenlayer anlegen

combined = layer_concatenate(list(x_TRACS$output
                                  ,x_AGE$output,
                                  x_BMI$output,
                                  x_SEX$output)) %>%
  layer_dense(units = 8,
              kernel_regularizer = regularizer_l1_l2(l1 = 0.01,l2 = 0.01))

#Ausgabelayer anlegen
z = combined %>% layer_dense(units = ncol(y_train), activation = 'softmax')

#Layer kombinieren
modelC = keras_model(inputs = list(x_TRACS$input,
                                   x_AGE$input,
                                   x_BMI$input,
                                   x_SEX$input),
                     outputs = c(z))


modelC %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_adam(learning_rate = 0.01),
  metrics = c('accuracy')
)

summary(modelC)
```

```
## Model: "model_4"
## __________________________________________________________________________________
##  Layer (type)           Output Shape            Param   Connected to
##                                                  #
## ==================================================================================
##  input_1 (InputLayer)   [(None, 4)]             0       []
##  input_2 (InputLayer)   [(None, 1)]             0       []
##  input_3 (InputLayer)   [(None, 1)]             0       []
```

```
##  input_4 (InputLayer)   [(None, 1)]             0       []
##  embedding (Embedding)  (None, 4, 2)            424     ['input_1[0][0]']
##  embedding_1 (Embeddin  (None, 1, 2)            94      ['input_2[0][0]']
##  g)
##  embedding_2 (Embeddin  (None, 1, 2)            6       ['input_3[0][0]']
##  g)
##  embedding_3 (Embeddin  (None, 1, 2)            4       ['input_4[0][0]']
##  g)
##  bidirectional (Bidire  (None, 16)              704     ['embedding[0][0]']
##  ctional)
##  flatten (Flatten)      (None, 2)               0       ['embedding_1[0][0]']
##  flatten_1 (Flatten)    (None, 2)               0       ['embedding_2[0][0]']
##  flatten_2 (Flatten)    (None, 2)               0       ['embedding_3[0][0]']
##  concatenate (Concaten  (None, 22)              0       ['bidirectional[0][0]',
##  ate)                                                    'flatten[0][0]',
##                                                          'flatten_1[0][0]',
##                                                          'flatten_2[0][0]']
##  dense (Dense)          (None, 8)               184     ['concatenate[0][0]']
##  dense_1 (Dense)        (None, 210)             1890    ['dense[0][0]']
## ==================================================================================
## Total params: 3306 (12.91 KB)
## Trainable params: 3306 (12.91 KB)
## Non-trainable params: 0 (0.00 Byte)
## __________________________________________________________________________________
```

```
history <- modelC %>% fit(
  x_train[1:4],
  y_train,
  epochs = 50,
  batch_size = 500,
  validation_split = 0.3,
  verbose = 0)
```

Modellvalidierung:

Plausibilisieren und interpretieren Sie die Ergebnisse. Erstellen Sie Modellvorhersagen und vergleichen Sie diese mit der empirischen Beobachtung. Nutzen Sie ein geeignetes Gütemaß, z.B. den Multiclass-AUC für Trainings- und Testdaten.

```
history
```

```
##
## Final epoch (plot to see history):
##         loss: 3.222
##     accuracy: 0.2853
##     val_loss: 3.262
## val_accuracy: 0.2835
```

```
plot(history) + theme_minimal() +
  scale_color_manual(values=c("#70AD47", "#5B9BD5")) +
  scale_fill_manual(values =c ("#A5A5A5","#A5A5A5"))
```

```
#Evaluation auf Trainingsdaten
pred = predict(modelC,x_train)
```

```
## 1500/1500 - 5s - 5s/epoch - 3ms/step
```

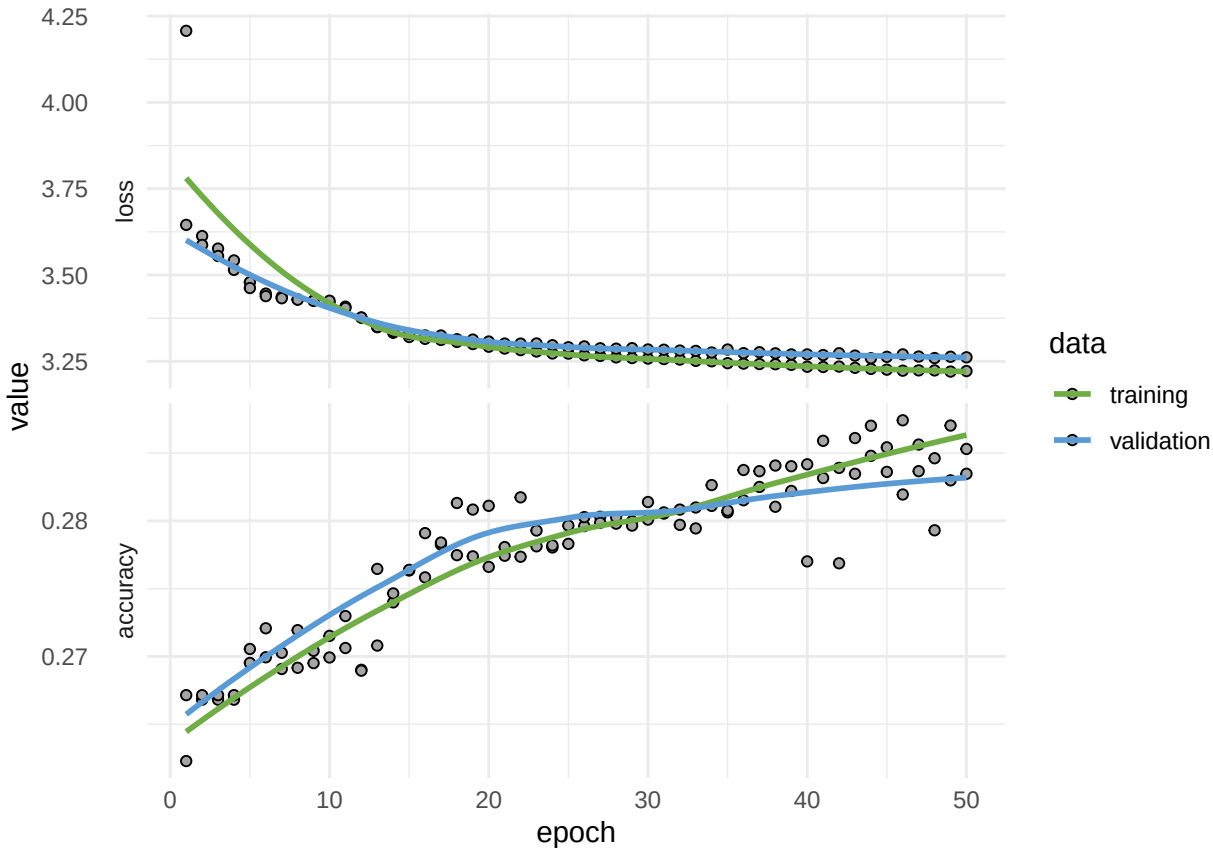


Figure 4: Training und Validierung volles Modell

```
real_train  = labels_train_cat
colnames(pred) = levels(real_train)
multiclass.roc(real_train, pred)
```

```
##
## Call:
## multiclass.roc.default(response = real_train, predictor = pred)
##
## Data: multivariate predictor pred with 210 levels of real_train: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
## Multi-class area under the curve: 0.7466
```

```
#Evaluation auf Testdaten
pred_test = predict(modelC,x_test)
```

```
## 375/375 - 2s - 2s/epoch - 5ms/step
```

```
real_test  = labels_test_cat
colnames(pred_test) = levels(real_train)
multiclass.roc(real_test, pred_test)
```

```
## Warning in multiclass_roc_multivariate(response, predictor, levels, percent, :
## The following classes were not found in 'response':
## 202,203,206,207,208,209,210.
```

```
##
## Call:
## multiclass.roc.default(response = real_test, predictor = pred_test)
##
## Data: multivariate predictor pred_test with 203 levels of real_test: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
## Multi-class area under the curve: 0.7277
```

Antwort [Modellprognose]

- Das Modell ist mit einem AUC von rund 0.7 (auf den Testdaten) in Ordnung - aber noch nicht optimal. Ggf. liegen nicht genügend Daten vor um Muster noch besser zu erkennen oder die Modellarchitektur ist zu einfach um diese komplexen Muster zu erkennen.

(c) *Eingeschränktes Neuronales Netz*

Ziel ist es nun, die Prognose auf einzelne Krankheiten mit besonderer Schwere einzuschränken, um damit das Modell auf die geplante Anwendung maßzuschneidern. Ziel ist es bessere Prognosen zu erzielen. Nach intensiven Diskussionen mit ihren medizinischen Experten einigen Sie sich auf folgende Krankheiten für die Anwendung im Case Management:

- Chronische Herzinsuffizienz (ICD-10 Diagnose I50)
- Depression (ICD-10 Diagnose F32)
- Darmkrebs (ICD-10 Diagnose C18)
- Akute Bronchitis (ICD-10 Diagnose J20)
- Lungenkrebs (ICD-10 Diagnose C34)

Schränken Sie die Zielgröße entsprechend ein. Rekodieren Sie alle anderen Diagnosen auf "ka". Definieren Sie ein zweites neuronales Netz und kalibrieren Sie dieses. Wählen Sie eine geeignete Parametrisierung (Hinweis: ein ausführliches Parameter-Tuning ist nicht notwendig). Verwenden Sie die gleiche Modellstruktur wie in Aufgabe 2b. Verwenden Sie zusätzlich eine Gewichtung im Kalibrierungsschritt um die Unbalananciertheit in der Zielgröße zu berücksichtigten (z.B. ist ka nun deutlich überrepräsentiert).

```
part = as.numeric(which(names(word_index) %in% c("i50","f32","c18","j20","c34")))
default = as.numeric(which(names(word_index) %in% c("ka")))

target_train = ifelse(!labels_train %in% part,default,labels_train)
labels_train_cat = factor(target_train)


target_test = ifelse(!labels_test %in% part,default,labels_test)
labels_test_cat = factor(target_test)



y_train = to_categorical(target_train)[,c(default,part)+1]
y_test =to_categorical(target_test)[,c(default,part)+1]



TRACS = layer_input(dtype = 'float32',shape = c(m-1))
modelTRACS = TRACS %>% layer_embedding(input_dim = vocab_size, output_dim = 2)
modelTRACS = modelTRACS %>%  bidirectional(layer_lstm(units = 8))

x_TRACS = keras_model(inputs = c(TRACS), outputs = c(modelTRACS))


AGE = layer_input(dtype = 'int32',shape = c(1))
modelAGE = AGE %>% layer_embedding(input_dim = n_distinct(data$Alter),
                                   output_dim = 2,input_length = 1) %>%
  layer_flatten()
x_AGE = keras_model(inputs = list(AGE), outputs = c(modelAGE))


BMI = layer_input(dtype = 'int32',shape = c(1))
modelBMI = BMI %>% layer_embedding(input_dim = 3,
                                   output_dim = 2,input_length = 1) %>%
  layer_flatten()
x_BMI = keras_model(inputs = list(BMI), outputs = c(modelBMI))


SEX = layer_input(dtype = 'int32',shape = c(1))
modelSEX = SEX %>% layer_embedding(input_dim = 2,
                                   output_dim = 2,input_length = 1) %>%
  layer_flatten()
x_SEX = keras_model(inputs = list(SEX), outputs = c(modelSEX))




#Inputs kombinieren und Zwischenlayer anlegen
combined = layer_concatenate(list(x_TRACS$output
                                  ,x_AGE$output,
                                  x_BMI$output,
                                  x_SEX$output)) %>%
  layer_dense(units = 8,
```

```
              kernel_regularizer = regularizer_l1_l2(l1 = 0.01,l2 = 0.01))

#Ausgabelayer anlegen
z = combined %>% layer_dense(units = ncol(y_train), activation = 'softmax')


modelC = keras_model(inputs = list(x_TRACS$input,
                                   x_AGE$input,
                                   x_BMI$input,
                                   x_SEX$input),
                     outputs = c(z))


modelC %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_adam(learning_rate = 0.01),
  metrics = c('accuracy')
)

summary(modelC)
```

```
## Model: "model_9"
## __________________________________________________________________________________
##  Layer (type)           Output Shape            Param   Connected to
##                                                  #
## ==================================================================================
##  input_5 (InputLayer)   [(None, 4)]             0       []
##  input_6 (InputLayer)   [(None, 1)]             0       []
##  input_7 (InputLayer)   [(None, 1)]             0       []
##  input_8 (InputLayer)   [(None, 1)]             0       []
##  embedding_4 (Embeddin  (None, 4, 2)            424     ['input_5[0][0]']
##  g)
##  embedding_5 (Embeddin  (None, 1, 2)            94      ['input_6[0][0]']
##  g)
##  embedding_6 (Embeddin  (None, 1, 2)            6       ['input_7[0][0]']
##  g)
##  embedding_7 (Embeddin  (None, 1, 2)            4       ['input_8[0][0]']
##  g)
##  bidirectional_1 (Bidi  (None, 16)              704     ['embedding_4[0][0]']
##  rectional)
##  flatten_3 (Flatten)    (None, 2)               0       ['embedding_5[0][0]']
##  flatten_4 (Flatten)    (None, 2)               0       ['embedding_6[0][0]']
##  flatten_5 (Flatten)    (None, 2)               0       ['embedding_7[0][0]']
##  concatenate_1 (Concat  (None, 22)              0       ['bidirectional_1[0][0]'
##  enate)                                                  , 'flatten_3[0][0]',
##                                                          'flatten_4[0][0]',
##                                                          'flatten_5[0][0]']
##  dense_2 (Dense)        (None, 8)               184     ['concatenate_1[0][0]']
##  dense_3 (Dense)        (None, 6)               54      ['dense_2[0][0]']
## ==================================================================================
## Total params: 1470 (5.74 KB)
## Trainable params: 1470 (5.74 KB)
## Non-trainable params: 0 (0.00 Byte)
## __________________________________________________________________________________
```

```
#add weights
ww = (length(target_train)) / (summary(factor(target_train)))
#sortieren
ww = ww[as.character(c(default,part))]
wt = list()
for(i in ww){
  wt =append(wt,i)
}
names(wt) = seq(0,length(wt)-1)
history <- modelC %>% fit(
  x_train, y_train,
  class_weight = wt,
  epochs = 50,
  batch_size = 500,
  validation_split = 0.3,
  verbose = 0)
```

Modellvalidierung: Plausibilisieren und interpretieren Sie die Ergebnisse. Erstellen Sie Modellvorhersagen und vergleichen Sie diese mit der empirischen Beobachtung. Nutzen Sie ein geeignetes Gütemaß, z.B. den Multiclass-AUC für Trainings- und Testdaten.

```
history
```

```
##
## Final epoch (plot to see history):
##         loss: 6.614
##     accuracy: 0.3298
##     val_loss: 1.683
## val_accuracy: 0.2822
```

```
plot(history) + theme_minimal() +
  scale_color_manual(values=c("#70AD47", "#5B9BD5")) +
  scale_fill_manual(values =c ("#A5A5A5","#A5A5A5"))
```

```
pred = predict(modelC,x_train)
```

```
## 1500/1500 - 6s - 6s/epoch - 4ms/step
```

```
real  = labels_train_cat
colnames(pred) = levels(real)
multiclass.roc(real, pred)
```

```
##
## Call:
## multiclass.roc.default(response = real, predictor = pred)
##
## Data: multivariate predictor pred with 6 levels of real: 2, 3, 6, 8, 9, 10.
## Multi-class area under the curve: 0.8504
```

```
pred_test = predict(modelC,x_test)
```

```
## 375/375 - 1s - 1s/epoch - 4ms/step
```

```
real_test  = labels_test_cat
colnames(pred_test) = levels(real_test)
multiclass.roc(real_test, pred_test)
```

```
##
```

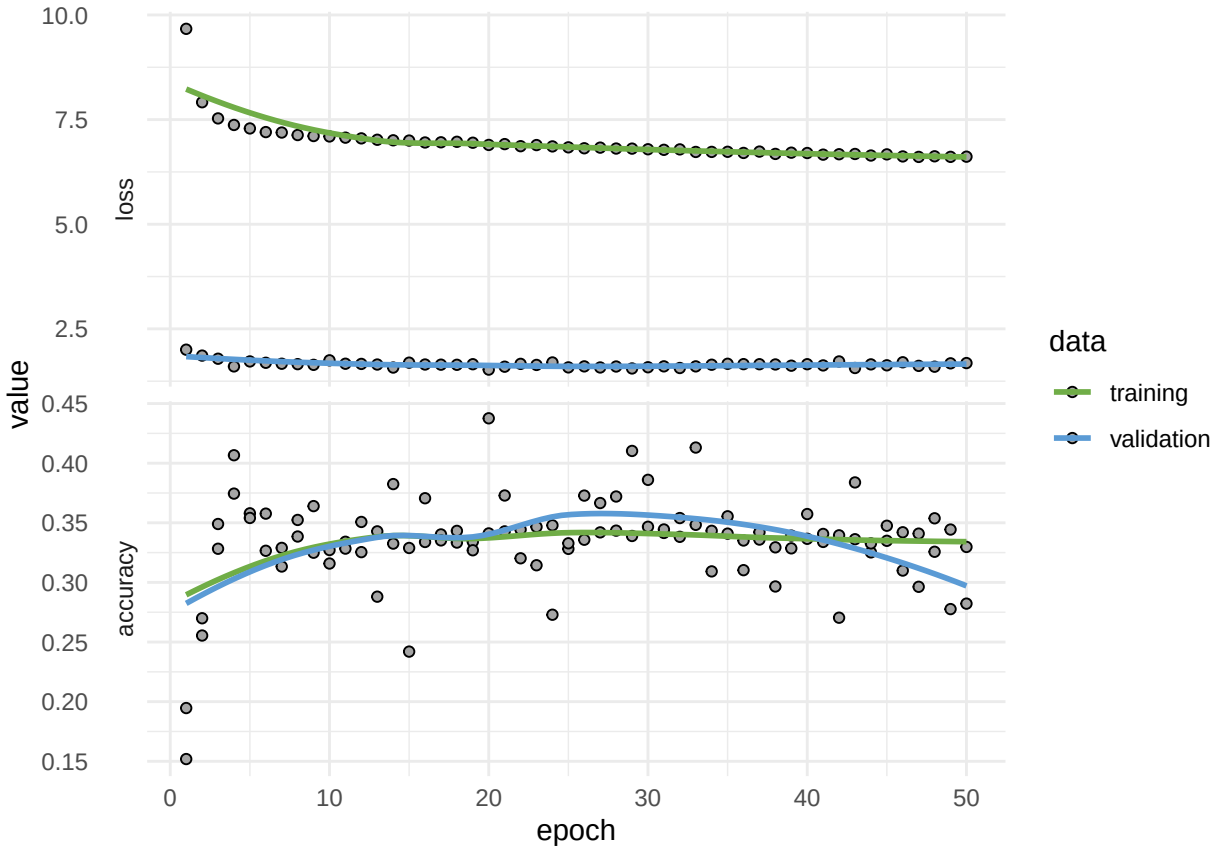


Figure 5: Training und Validierung spezielles Modell

```
## Call:
## multiclass.roc.default(response = real_test, predictor = pred_test)
##
## Data: multivariate predictor pred_test with 6 levels of real_test: 2, 3, 6, 8, 9, 10.
## Multi-class area under the curve: 0.8242
```

Antwort [Modellprognose]

- Die Modellgüte wird besser. Das Modell ist in der Lage die fünf Krankheiten verlässlich zu prognostizieren.

Aufgabe 3. [*Business Case Krankenversicherung - Modellanwendung und Interpretation*] [30 Punkte]

(a) *Anwendung des Modells zur Prognose*

Welche Krankheiten sind laut Modell am wahrscheinlichsten, wenn zuvor

- Bluthochdruck (ICD-10 Diagnose I10)
- Alkoholmissbrauch (ICD-10 Diagnose F10)
- Akute Bronchitis (ICD-10 Diagnose J20)
- keine Erkrankung (kA)

diagnostiziert wurde? Nutzen Sie hierfür exemplarisch einen 60 jährigen Mann mit BMI-Status "OK".

Lesen sie die Datei Neue_Kunden.csv ein. Welche Krankheit ist für den Krankheitsverlauf des ersten neuen Kunden aus dem Datensatz Neue_Kunden.csv am wahrscheinlichsten?

Diskutieren Sie die Anwendbarkeit des Modells in der Praxis.

```
highest_prob = function(input_tracs, input_case){
 test = pad_sequences(texts_to_sequences(tokenizer,
                                         list(input_tracs)), maxlen = m-1)
 case = input_case
 case[[1]]  = case[[1]]-scale_Age
 x_test_case = append(list(test),case)
 pred_probs <- modelC %>% predict(x_test_case,verbose = 0)

 return(names(word_index)[c(c(default,part))[which.max(pred_probs)]])
}

highest_prob(input_tracs = "i10", input_case = list(60,1,0))
```

```
## [1] "i50"
```

```
highest_prob("f10",input_case = list(60,1,1))
```

```
## [1] "c18"
```

```
highest_prob("j20",input_case = list(60,1,1))
```

```
## [1] "c34"
```

```
highest_prob("ka",input_case = list(60,1,1))
```

```
## [1] "i50"
```

```
Kunde_neu = read.csv("Neue_Kunden.csv")
i = 1
K1_input_tracs = Kunde_neu$Historie[i]
K1_input_case = list(Kunde_neu$Alter[i],
```

```
                      ifelse(Kunde_neu$BMI[i]=="ok",
                             0L,
                             ifelse(Kunde_neu$BMI[i]=="extrem",1L,2L)),
                      ifelse(Kunde_neu$Geschlecht[i]=="Mann",1L,0L))



highest_prob(K1_input_tracs,K1_input_case)
```

```
## [1] "ka"
```

Antwort [Modellbewertung für die Anwendung in der Praxis]

- Die Prognosen sind plausibel. Es ist jedoch darauf zu achten, dass es immer nur Wahrscheinlichkeiten sind und keine gesicherten Diagnosen. Einzeleffekte, die nicht erfasst werden, können dazu führen, dass die tatsächlichen Eintrittswahrscheinlichkeiten für einzelne Personen gravierend abweichen. Es ist also auf eine sensible Kontaktaufnahme zu achten.
- Das Case Management ist für die Unsicherheit der Prognosen zu sensibilisieren.

Sie wollen das Modell nutzen, um anhand von vorgegebenen Krankheiten den weiteren Krankheitsverlauf für **mehrere Jahre** vorherzusagen. Für die Schätzung der nächsten Krankheit nutzen Sie wie zuvor die Modellvorhersage für die bekannten Krankheiten. Für die darauf folgende Krankheit werden dann die bekannten Krankheiten und die zuvor geschätzte Krankheit genutzt (rollierend), usw. Für die Vorhersage einer konkreten Krankheit ziehen Sie zufällig eine der fünf Krankheiten bzw. keine Krankheit anhand der geschätzten Wahrscheinlichkeiten des Modells.

Nutzen Sie die Krankheitshistorie der Kunden zwei bis fünf des Datensatzes Neue_Kunden.csv, um jeweils einen Krankheitsverlauf für weitere fünf Jahre vorherzusehen. Sind die Ergebnisse plausibel?

```
write_medical_tracs <- function(input_tracs, input_case, length = 5) {
    case = input_case
    case[[1]]  = case[[1]]-scale_Age

    for (i in 1:length + 1) {
      test = pad_sequences(texts_to_sequences(tokenizer, list(input_tracs)), maxlen = m-1)
      x_test_case = append(list(test),case)
      pred_probs <- modelC %>% predict(x_test_case,verbose = 0)

      #Ziehen einer konkreten Vorhersage
      output_arg <- sample(seq_len(length(pred_probs)), 1, prob = pred_probs)

      #Wandelt in Wort (Krankheit) um
      output_word <- names(word_index)[as.numeric(levels(labels_train_cat))][output_arg]
      input_tracs <- paste(input_tracs, output_word)
    }
    return(input_tracs)
}

input_tracs = "i10"
input_case = list(60,1,0)

write_medical_tracs(input_tracs = "i10", input_case = list(60,1,0))
```

```
## [1] "i10 i50 i50 i50 ka c34"
```

```
K2 = Kunde_neu$Historie[2]
K3 = Kunde_neu$Historie[3]
```

```
K4 = Kunde_neu$Historie[4]
K5 = Kunde_neu$Historie[5]

for(i in 2:5){
  K_input_tracs = Kunde_neu$Historie[i]
  K_input_case = list(Kunde_neu$Alter[i],
                      ifelse(Kunde_neu$BMI[i]=="ok",
                             0L,
                             ifelse(Kunde_neu$BMI[i]=="extrem",1L,2L)),
                      ifelse(Kunde_neu$Geschlecht[i]=="Mann",1L,0L))

print(write_medical_tracs(K_input_tracs, K_input_case))
}
```

```
## [1] "j20 i10 i50 j20 j20 c34 h25 r55 j20 j20 c34 i50 ka ka ka"
## [1] "h25 i10 ka ka i50 j20 i10 i50 j20 f10 c34 c18 i50 i50 j20"
## [1] "j20 ka ka l02 i50 c34 j20 c34 l72 j20 i50 ka c34 i50 j20"
## [1] "i50 k29 i46 ka j20 ka f10 l02 f32 f32 f32 f32 f32 f32 f32"
```

Antwort [Modellbewertung für die Anwendung in der Praxis]

- Für Kundin 2 wird verstärkt C34 (Lungenkrebs) vorhergesagt. Da es sich um eine Raucherin handelt, die bereits wegen akuter Bronchitis (J20) behandelt wurde, ist das Ergebnis plausibel.
- Für die dritte Kundin wird verstärkt C18 (Darmkrebs) vorhergesagt. Dies ist mit letzter Diagnose F10 (Alkoholmissbrauch) plausibel
- Für den vierten Kunden ist es analog zu Kundin 2 (Lungenkrebs als Raucher mit Bronchitis)
- Der letzte Kunde hat verstänrkt Depression (F32). Für mittelalte Männer mit extremen BMI ist das plausibel.

(b) *Modellinterpretation*

Analysieren Sie die Gewichte der Embeddings. Welche Diagnosen sind auffällig, was wird für fehlende Diagnosen verwendet? Begrenzen Sie ggf. auf besonders relevante Diagnosen.

Betrachten Sie auch das Alter-Embedding z.B. unter Verwendung der in Aufgabe 1 verwendeten Altersbereiche.

```
plot_data = cbind(get_weights(modelC)[[1]])
plot_data_small = plot_data[1:50,]

rownames(plot_data_small) = names(word_index)[c(2:51)]

ggplot(data.frame(plot_data_small), aes(x=X1, y=X2)) +
geom_point() +
geom_text_repel(label=rownames(plot_data_small),max.overlaps = 30) +theme_bw()
```

```
#alterskalierung berücksichtigen
plot_data = cbind(get_weights(modelC)[[2]],c(rep("jung",13),
                                             rep("mittel",20),
                                             rep("alt",14)))
rownames(plot_data) = as.character(seq(1,47))

ggplot(data.frame(plot_data), aes(x=X1, y=X2, colour = X3)) +
geom_point() +
theme_bw() +
```

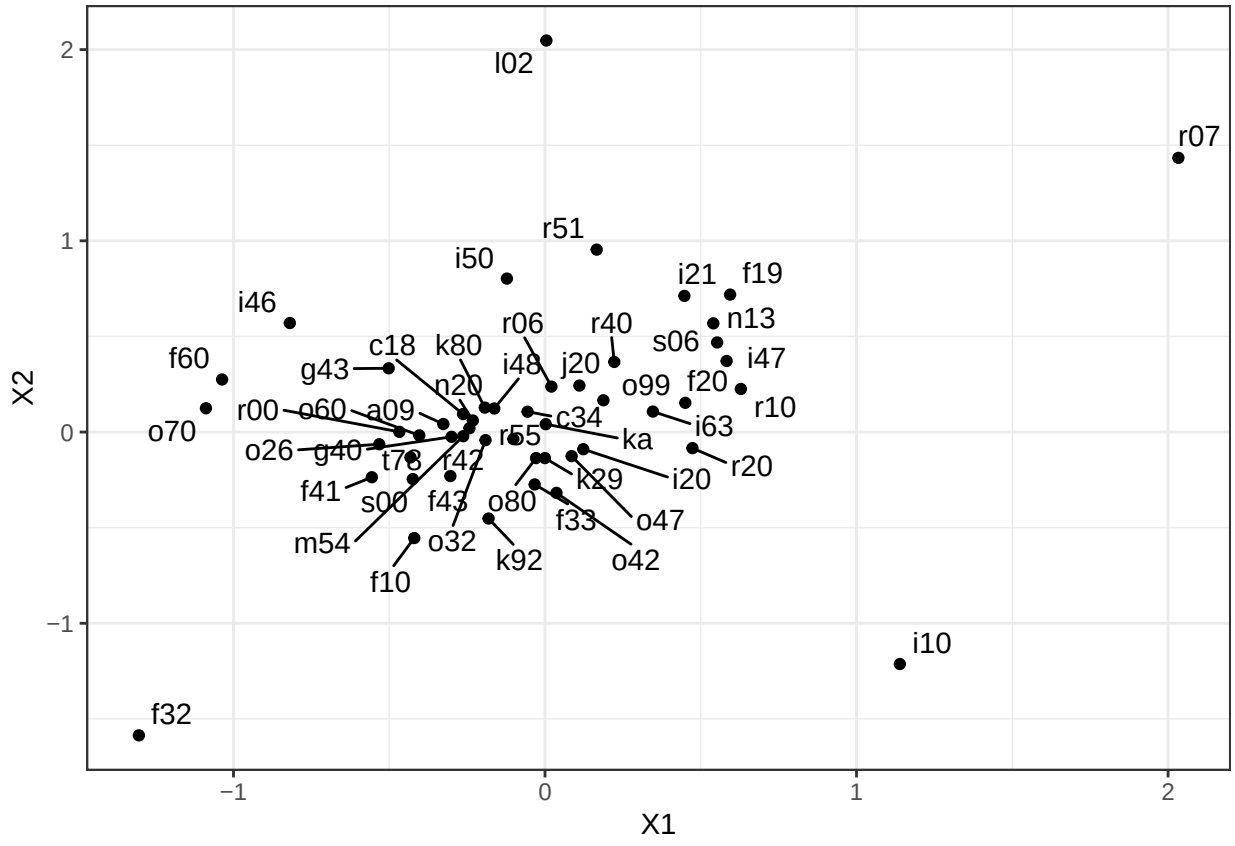


Figure 6: Gewichte des zweidimensionalen Diagnosen Embedding

```
theme(
  axis.text.x = element_blank(),
  axis.text.y = element_blank())
```

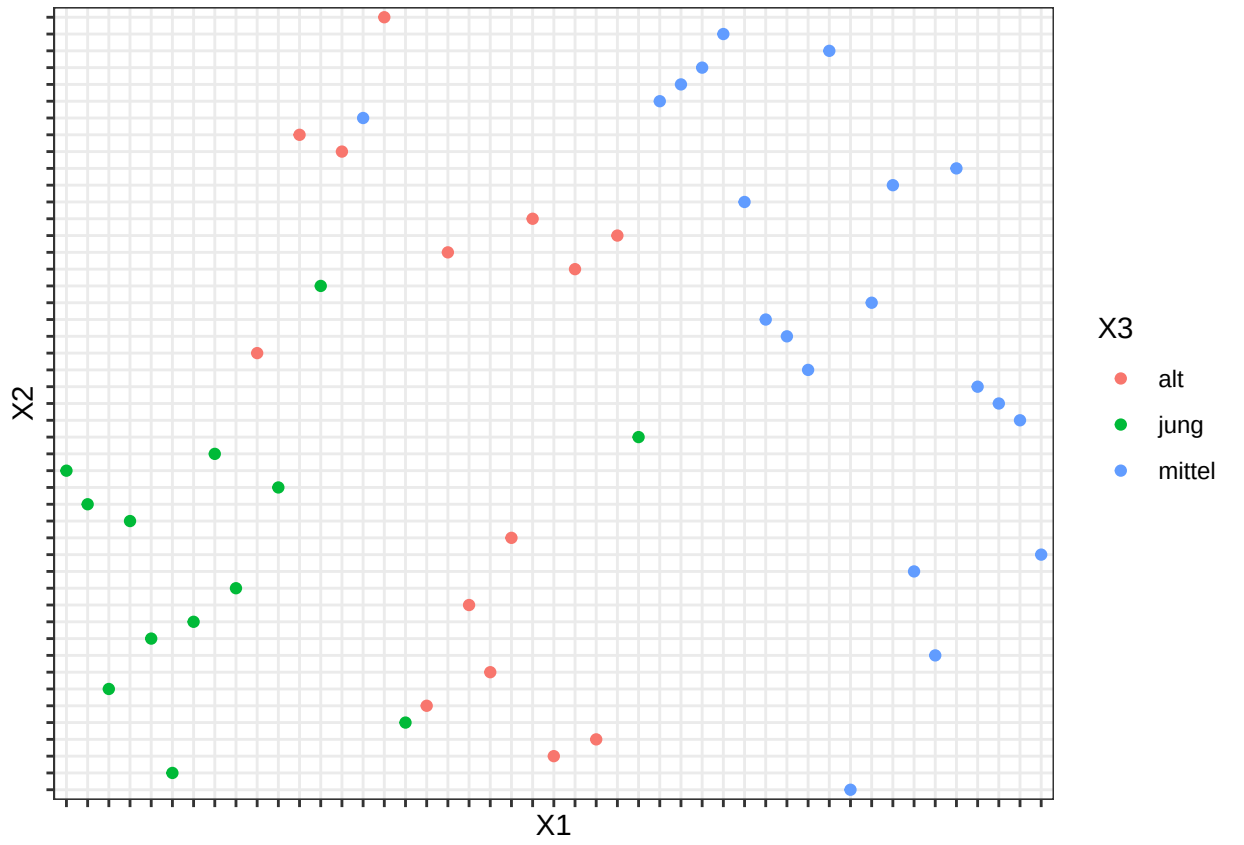


Figure 7: Gewichte des zweidimensionalen Alter Embedding

Antwort [Modellinterpretation]

- Die auffälligen Krankheiten (I10, J20 und F32) scheinen sich am Rand des Embedding-Raums zu befinden. Keine Diagnose (ka) ist hingegen ziemlich am Ursprung.
- die betrachteten Altersbereiche finden sich im Embedding wieder.