DAV

DEUTSCHE
AKTUARVEREINIGUNG e.V.

Praktische Prüfung im Vertiefungswissen

Actuarial Data Science Completion

gemäß Prüfungsordnung 4
der Deutschen Aktuarvereinigung e. V.

Lösungsvorschlag Gesamt (Block A, B, C)

Prüfungszeitraum: 06.10.2021 – 05.11.2021

Block A (Allgemeine Fragen) [20 Punkte]

Digitalisierung

Aufgabe A1: a) [Lernziel 1.1.2] [2 Punkte]

Welche Vorteile könnte der Einsatz von Cloud Computing bei einem Projekt im Kontext von Notebook B bieten?

- Das Bottleneck an vielen Stellen dieses Projekts ist die Laufzeit der Algorithmen
- Beispielsweise ist ein effizientes Hyperparameter Tuning auf dem lokalen Rechner deswegen kaum möglich.
- In der Cloud steht eine deutlich höhere Rechenleistung zur Verfügung
- Nachdem potenzieller Verbesserungsbedarf einzelner Modelle lokal identifiziert wurde, kann für diese Einzelmodelle ein Hyperparameter Tuning in der Cloud erfolgen
- Ebenso kann auch eine deutlich erhöhte Anzahl von Trainingsepochen durchgeführt werden, sofern die Modelle beim lokalen Training noch Instabilitäten aufweisen

Aufgabe A1: b) [Lernziel 1.1.2] [2 Punkte]

Wie müssten mögliche Risiken im Zuge des Einsatzes von Cloud Computing hier konkret abgefedert werden?

- Im Versicherungskontext werden oft sensible persönliche Daten verwendet
- Eine Anonymisierung / Aggregation der Daten ist sicherzustellen
- Gegebenenfalls ist zu überprüfen, ob der Serverstandort (z.B. in den USA) mit der DSGVO konform ist

Entwicklungsmethoden

Aufgabe A2: a) [Lernziel 2.1.1] [3 Punkte]

Diskutieren Sie, in welcher Form sich die vorliegende Aufgabenstellung von Notebook C als Scrum-Projekt in einer Versicherung umsetzen ließe.

- Grundsätzlich ist eine Umsetzung als Scrum-Projekt denkbar
- Zunächst müssten die Verantwortlichkeiten (Product Owner, Scrum Master, Entwickler, etc.) definiert werden
- Anforderungen, ähnlich der vorliegenden Aufgabenstellung, müssten als Product Backlog definiert werden
- Hieraus müsste der Sprint Backlog abgeleitet werden, der anschließend in einem Sprint umgesetzt wird
- Ziel ist es, eine lauffähige, inkrementell verbesserte Software zu erhalten
- Der Prozess wiederholt sich

Aufgabe A2: b) [Lernziel 2.1.1] [2 Punkte]

Würden Sie sich schlussendlich für Scrum entscheiden oder eine andere Entwicklungsmethode vorziehen?

- Es ist stark vom Kontext abhängig, welche Entwicklungsmethode am zielführendsten ist
- Scrum bietet sich hier tendenziell nicht an, weil es um fest definierte Fragen handelt, die beantwortet werden sollen
- Sollte es sich hingegen um wiederkehrende Fragestellungen handeln und eine permanente Verbesserung der Analyse angestrebt sein, kann Scrum eine gute Wahl sein.

Kodierungstheorie

Aufgabe A3 [Lernziel 2.2.1] [2 Punkte]

Skizieren Sie, welche Methoden der Kryptograpie eingesetzt werden könnte, um Texte zwischen verschiedenen Systemen sicher zu übertragen.

- In der Kryptographie unterscheidet man zwischen symmetrischen und asymmetrischen Verschlüsselungsverfahren
- Tendenziell haben symmetrische Verfahren Vorteile bezüglich der Laufzeit während asymmetrische Verfahren tendenziell sicherer sind (public key)
- In der Praxis kommen daher meist Hybrid Verfahren zum Einsatz
- Konkret sollte das Verfahren vom notwendigen Geheimhaltungsgrad der Nachricht und der Möglichkeit für einen (sicheren) Schlüsselaustausch abhängig gemacht werden

Big Data Analytics

Aufgabe A4 [Lernziel 3.1.2] [3 Punkte]

Bei welchen Teilen der vorliegenden Aufgabenstellung (Notebook B, C) wäre der Einsatz von Apache Spark eine denkbare Alternative? Begründen Sie Ihre Antwort.

- Spark ist leicht skalierbar und arbeitet im Arbeitsspeicher, dass ermöglicht eine sehr schnelle Verarbeitung von enormen, heterogenen Datenmengen, auch mit Datenströmen. Grundsätzlich könnten die In-Memory-Verarbeitung, das DataScource API, die funktionale Codierung und die Machine Learning Pipeline bei der Lösung der vorliegenden Aufgabe nützlich sein.
- Diese Big Data Technologien sind für die hier vorliegenden tabularen, vergleichsweise kleinen Datensätze jedoch nicht notwendig und würden die Komplexität der Lösung deutlich erhöhen, ohne deren Vorteile ausspielen zu können.

Thema: Korrelation & kausale Inferenz

Aufgabe A5: a) [Lernziel 4.1.4] [3 Punkte]

Diskutieren Sie kurz, ob LIME & SHAP eher Korrelationen identifizieren oder tatsächliche kausale Zusammenhänge.

- Bei LIME werden Ersatzmodelle auf Basis einzelner Beobachtungen erstellt. Bei SHAP steht die Rekonstruktion von Marginalverteilungen aller Merkmale im Vordergrund. Beide Erklärungsmethoden beruhen auf der Verwendung des zugrundeliegenden Machine Learning Modells. Daraus wird unmittelbar klar, dass LIME & SHAP nur Abhängigkeiten innerhalb des gegebenen Modells erklären. Folglich werden im Allgemeinen keine kausalen Zusammenhänge gefunden. Damit sind Ergebnisse eher im Sinne von Korrelation zu interpretieren

Aufgabe A5: b) [Lernziel 4.1.4] [3 Punkte]

Welche zusätzlichen Elemente würden Sie gegebenenfalls benötigen, um kausale Zusammenhänge nachzuweisen?

- Sofern sichergestellt wäre, dass das zugrundeliegende Modell tatsächlich kausale Zusammenhänge abbildet, würden diese Zusammenhänge durch LIME & SHAP (mit ihren jeweiligen Besonderheiten) mutmaßlich abgebildet werden. Um kausale Zusammenhänge zu identifizieren, gibt es beispielsweise ökonometrische Ansätze, deren Einsatzfähigkeit allerdings vom konkreten Kontext abhängt und im Einzelfall konkret geprüft werden muss. Meistens können nur Detailfragen auf kausale Zusammenhänge überprüft werden. Die Möglichkeiten zum Einsatz für die Analyse großer Datensätze, was ja das Haupteinsatzgebiet der Data Science ist, sind sehr beschränkt.

Lösungsvorschlag zur Prüfungsaufgabe VADS Completion 2021, Block B: Notebook Mortalität

Contents

B: Notebook Mortalität: Neuronale Netze

Quellenangaben zu den verwendeten R-Codes:

RW2018: Ronald Richman und Mario V. Wüthrich (2018), "A Neural Network Extension of the Lee-Carter Model to Multiple Populations", SSRN-Preprint Nr. 3270877, https://www.ssrn.com/

RW2019: Ronald Richman und Mario V. Wüthrich (2019), "Lee and Carter go Machine Learning: Recurrent Neural Networks", SSRN-Preprint Nr. 3441030, https://www.ssrn.com/ sowie R-Skript "6 - Lee and Carter go Machine Learning Recurrent Neural Networks/02_a Lee Carter prediction.R" unter https://github.com/JSchelldorfer/ActuarialDataScience

Li2020: Huifeng Li et al. (2020), "Neuronale Netze treffen auf Mortalitätsprognose", https://aktuar.de/unsere-themen/big-data/anwendungsfaelle/Seiten/anwendungsfall3.aspx sowie zugehöriges R-Notebook https://github.com/DeutscheAktuarvereinigung/Mortality_Modeling

Vorbemerkungen:

Die Datenbasis enthält für jedes Lebensalter von 0 bis 100 Jahren und getrennt nach Geschlechtern (M/W) logarithmierte Mortalitätsraten für die Länder Japan, Australien, USA, Spanien, Italien, Frankreich, Schweiz, Polen, Dänemark, Deutschland Gesamt sowie für Deutschland Ost und Deutschland West ('JPN', 'AUS', 'USA', 'ESP', 'ITA', 'FRATNP', 'CHE', 'POL', 'DNK', 'DEUT','DEUTE','DEUTW') im Zeitraum 1958 bis 2014. Um die Betrachtung zu vereinfachen sollen im folgenden nur logaritmierte Mortralitätsraten modelliert, visualiert, bewertet und interpretiert werden.

```
# 0: Benötigte Bibliotheken einbinden und reproduzierbare Ergebnisse sicher stellen
suppressMessages(library(data.table))
suppressMessages(library(tidyverse))
```

```r
suppressMessages(library(reshape2))
suppressMessages(library(tensorflow))
suppressMessages(library(keras))

# Einheitliche Zufallszahl (R, Python, Numpy, TensorFlow)
use_session_with_seed(42) # Default: Deaktiviert GPU und CPU-Parallelisierung
```

```
## Set session seed to 42 (disabled GPU, CPU parallelism)
```

1. Daten einlesen und visualisieren

[Lernziel 6.1, 2 Punkte] Aufgabe B1: a) Lesen Sie die Mortalitätsdaten ein und vergewissern Sie sich, dass alle Datensätze korrekt verarbeitet wurden. Wie viele Datensätze und Features sind vorhanden, gibt es hier Unterschiede zwischen den Ländern? Zeigen Sie zehn zufällig ausgewählte Datensätze an.

Lösungsvorschlag:

```r
# Mortalitätsdaten einlesen und Dateigröße (Zeilen & Spalten) ausgeben:
data <- fread(file="D:/MORT.csv")
data$Gender <- as.factor(data$Gender)

dim(data)
```

```
## [1] 138168      5
```

```r
table(data$Country) # Anzahl Datensätze je Land
```

```
##
##    AUS    CHE   DEUT  DEUTE  DEUTW    DNK    ESP FRATNP    ITA    JPN    POL
##  11514  11514  11514  11514  11514  11514  11514  11514  11514  11514  11514
##    USA
##  11514
```

Der Datensatz enthält 5 Features und 138.168 Datensätze, einheitlich 11.514 Datensätze für jedes Land.

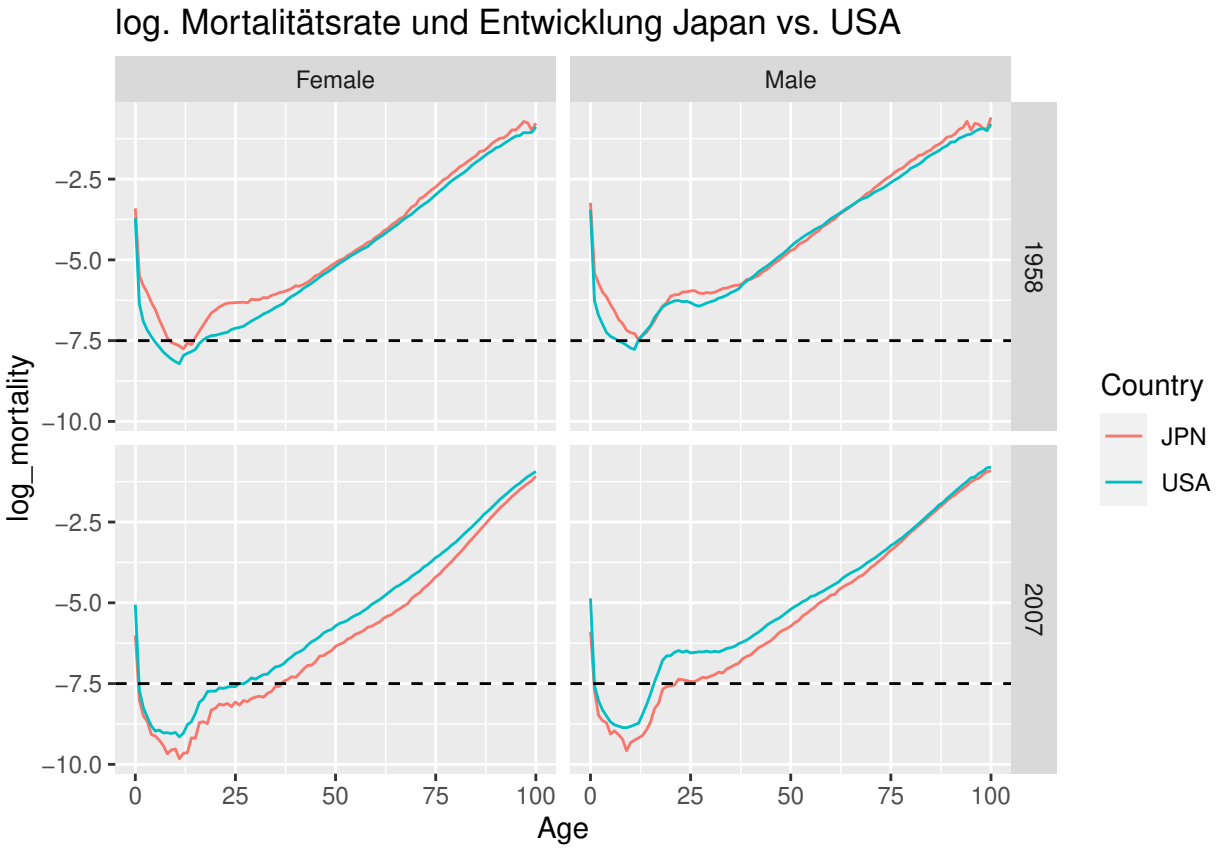
```r
# Zur Plausibilisierung Zufallsausgabe von 10 Datensätzen
set.seed(42)
sample_n(data, 10)
```

```
##     Country Year Gender Age log_mortality
##  1:     DNK 1983 Female   4     -8.220799
##  2:     ESP 1980 Female  86     -1.973742
##  3:     CHE 1999   Male  70     -3.648313
##  4:     JPN 1988 Female  25     -7.995428
##  5:     CHE 1977   Male  50     -5.104397
##  6:     DNK 1983   Male  95     -1.150897
##  7:   DEUTE 1988   Male  10     -8.547652
##  8:   DEUTE 2000 Female  45     -6.470791
##  9:     ITA 1983 Female  26     -7.834096
## 10:     ESP 1982   Male  71     -3.289164
```

[Lernziel 6.1, 2 Punkte] Aufgabe B1: b) Stellen Sie die logarithmierten Mortalitätsdaten von Japan und den USA im Altersbereich 0 bis 100 Jahre jeweils für die Jahre 1958/2007 und die Geschlechter Männlich/Weiblich gegenüber (4 Graphiken) und interpretieren Sie kurz das Ergebnis.

Lösungsvorschlag:

```
ggplot(data[which(data$Year %in% c(1958, 2007) & data$Country %in% c("JPN","USA")),],
       aes(Age, log_mortality, color=Country)) +
  geom_line() +
  geom_hline(yintercept=-7.5,col="black",lty=2) +
  facet_grid(Year ~ Gender)+
ggtitle("log. Mortalitätsrate und Entwicklung Japan vs. USA")
```



Die Sterblichkeit ist in beiden Ländern im betrachteten Zeitraum zurückgegangen, allerdings in Japan deutlich stärker als in den USA. Während die Mortalitätsraten Japans im Jahre 1958 in weiten Altersbereichen über denen der USA liegen, ist es im Jahr 2007 umgekehrt. Diese Entwicklung wird auch in Aufgabe B5 (Embeddings) transparent werden.

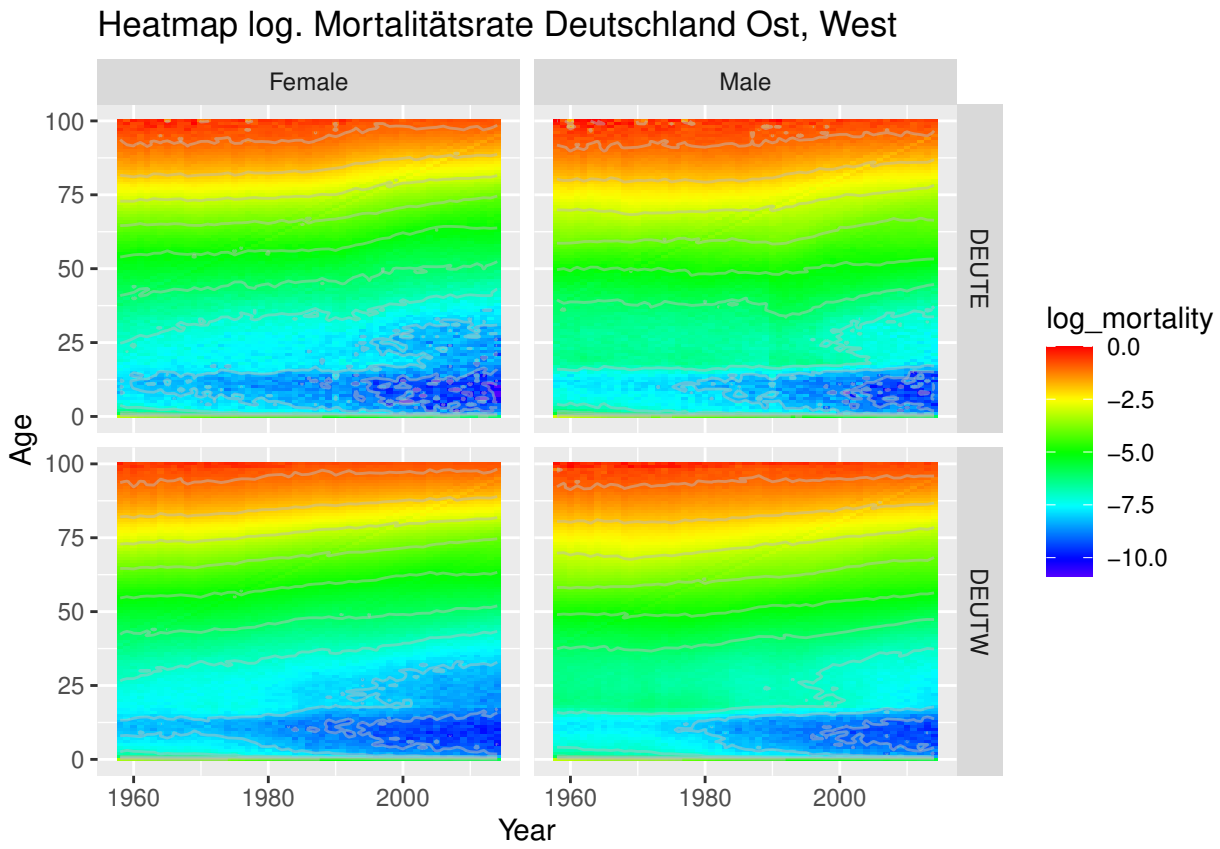
2. Auffälligkeitsprüfung: Mortalitätsunterschiede Ost/West-Deutschland aufspüren

[Lernziel 6.1, 2 Punkte] Aufgabe B2: a) Stellen Sie in vier Heatmaps die Mortalitätsentwicklung von Ost- und Westdeutschland (analog wie in 1.2 getrennt nach Geschlecht männlich /

weiblich) für alle verfügbaren Datenpunkte kompakt gegenüber. Verwenden Sie hierzu die Regenbogenfarbskala von blau für geringe bis rot für hohe Mortalität.

Lösungsvorschlag:

```
data %>% filter(Country %in% c("DEUTE","DEUTW")) %>%
  ggplot(aes(Year, Age, fill=log_mortality)) +
  geom_tile() +
  scale_fill_gradientn(colors=rev(rainbow(n=60, start=0, end=.72))) +
  geom_contour(aes(z=log_mortality), col="grey", alpha=0.5) +
  facet_grid(Country ~ Gender) +
  ggtitle("Heatmap log. Mortalitätsrate Deutschland Ost, West")
```

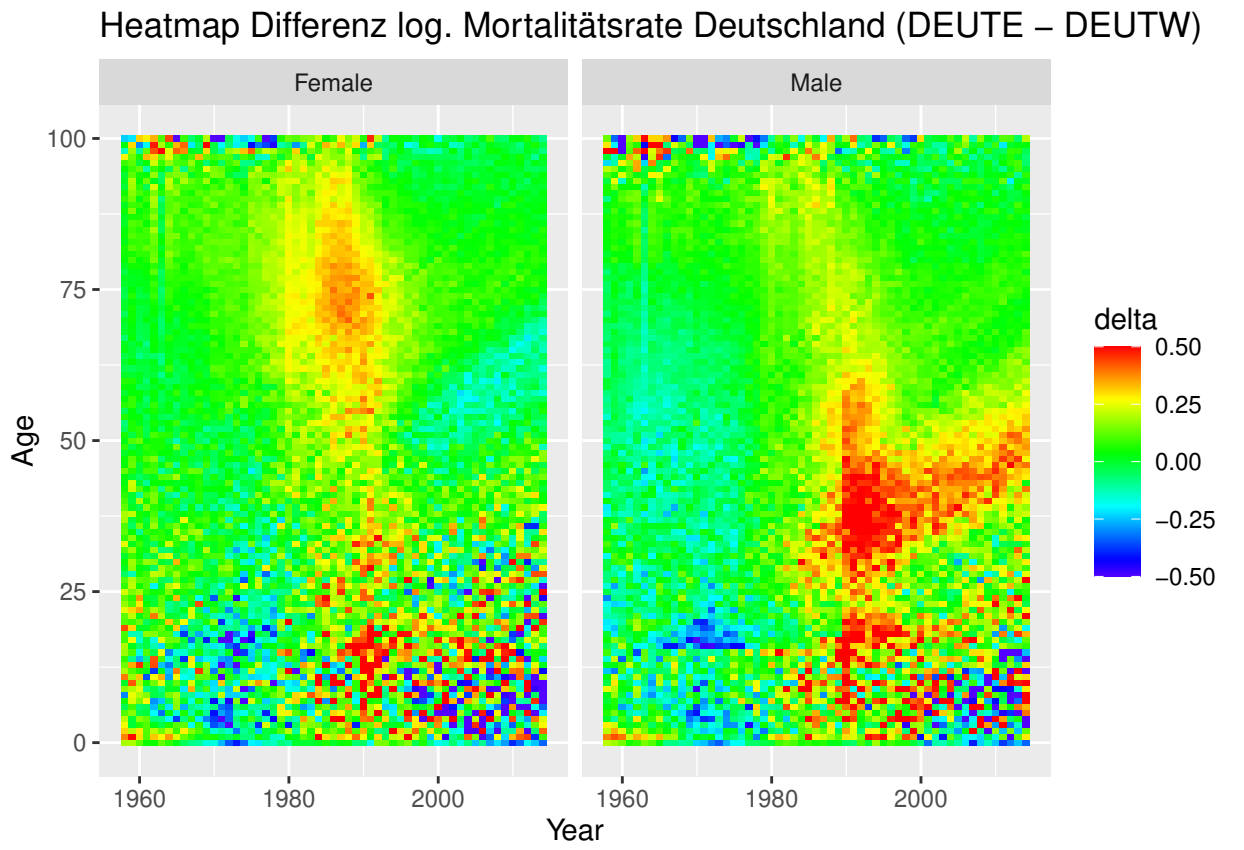


[Lernziel 6.1, 2 Punkte] Aufgabe B2: b) Berechnen Sie die Differenz der logarithmierten Mortalitäten von Ost- und Westdeutschland und zeigen Sie diese als Heatmaps nach Geschlechtern getrennt an. Verwenden Sie verschiedene Schwellenwerte als Obergrenze für das jeweilige Ende der Farbskala (blau/rot). Begründen Sie ihre Wahl und interpretieren Sie das Ergebnis.

Lösungsvorschlag:

```
# Berechne die Unterschiede zwischen Deutschland Ost vs. West
DO <- data[which(data$Country == "DEUTE"),]
DW <- data[which(data$Country == "DEUTW"),]
DW$delta <- pmax(pmin((DO$log_mortality - DW$log_mortality),0.5),-0.5)
#sample_n(DW, 10)
```

```r
ggplot(DW, aes(Year, Age, fill=delta)) +
  geom_tile() +
  scale_fill_gradientn(colors=rev(rainbow(n=60, start=0, end=.72))) +
  facet_grid(~ Gender) +
  ggtitle("Heatmap Differenz log. Mortalitätsrate Deutschland (DEUTE - DEUTW)")
```



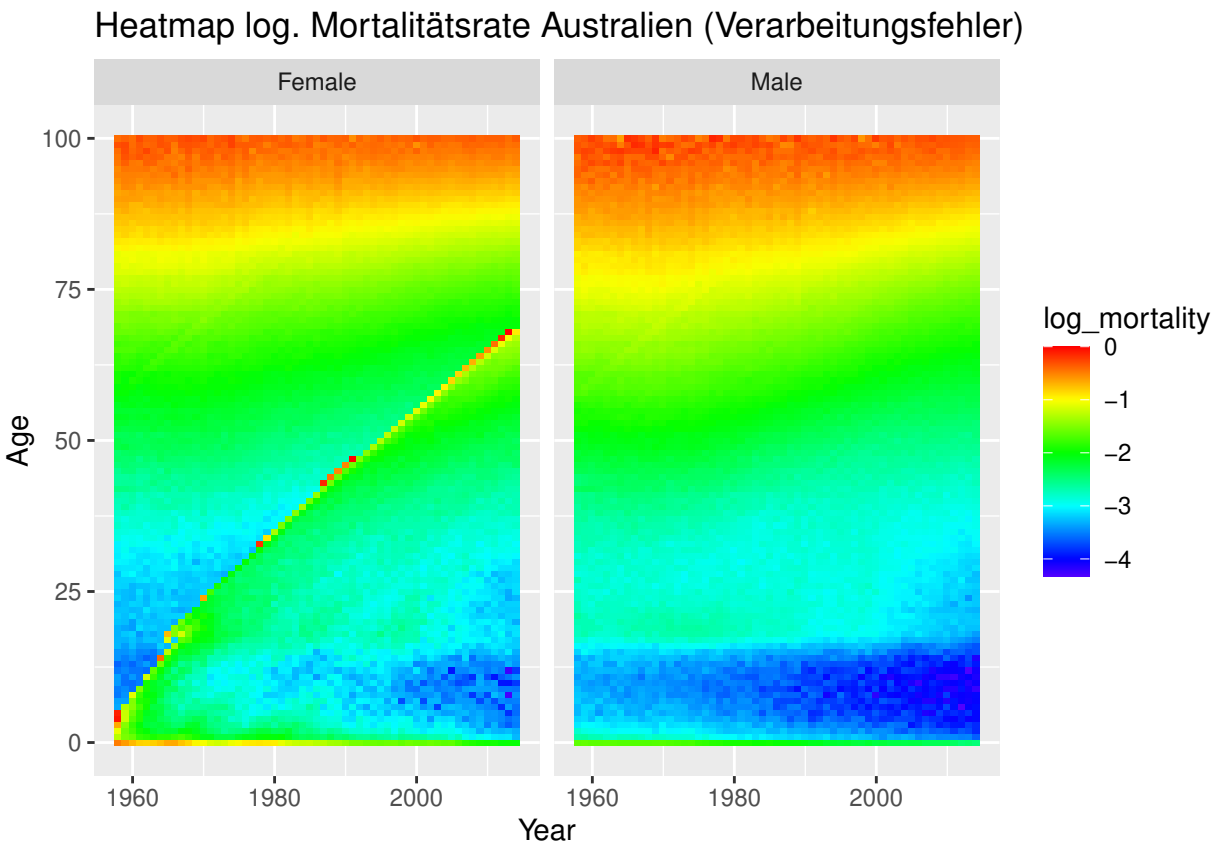
Die Wahl der Obergrenze für das jeweilige Ende der Farbskala hat einen erheblichen Einfluss auf den optischen Eindruck von den Sterblichkeitsunterschieden. Bei einer Kappung der berechneten Differenz bei +/-1 ist nur ein feines Muster zu erkennen, bei +/-0,1 befindet sich hingegen ein erheblicher Teil der Datenpunkte an der Farbgrenze und es kann kaum mehr zwischen geringen und großen Differenzen unterschieden werden. Daher wird ein "mittlerer" Wert von +/-0,5 gewählt, bei dem nur ein kleiner Teil der Datenpunkte von der Kappung betroffen ist, die unterschiedliche Mortalitätsentwicklung zwischen Ost- und West jedoch deutlich sichtbar wird.

[Lernziel 5.2, 3 Punkte] Aufgabe B2: c) Finden Sie im Datensatz ein Land mit stark unplausiblen Mortalitätsentwicklungen bei einem der beiden Geschlechter. Erläutern Sie Ihre Vorgehensweise. Erstellen Sie eine Heatmap mit den Daten des betroffenen Landes, in der das Problem erkennbar wird.

Lösungsvorschlag:

Bei der geringen Anzahl an Ländern können ohne großen Aufwand alle Datenpunkte via Heatmaps angezeigt werden (hier nicht gezeigt). Dabei fällt ein "unnatürliches" Muster bei der Sterblichkeitsenwicklung der Frauen in Australien auf, das auf einen Verarbeitungsfehler beim hier vorliegenden Datensatz zurückzuführen ist.

```r
data %>% filter(Country == "AUS") %>%
  ggplot(aes(Year, Age, fill=log_mortality)) +
  geom_tile() +
  scale_fill_gradientn(colors=rev(rainbow(n=60, start=0, end=.72))) +
  facet_wrap(~Gender) +
ggtitle("Heatmap log. Mortalitätsrate Australien (Verarbeitungsfehler)")
```



3. Lee-Carter Modell (klassisches Vergleichsmodell)

[Lernziel 6.1, 5 Punkte] Aufgabe B3: a) Erstellen Sie auf Basis der Daten der Jahre 1958 bis 2007 für jedes Land eine Mortalitätsprognose mittels des Lee-Carter-Modells. Verwenden Sie hierzu die im Artikel von RW2019 beschriebene Singulärwertzerlegung. Suchen Sie dazu online geeignete Lee-Carter-Programcodes, nennen Sie die Quelle und vergewissern Sie sich über die korrekte Funktionsweise. Extrapolieren Sie die jährlichen Werte ab 2008 bis 2014 mittels eines "Random Walk with Drift" für den späteren Vergleich mit den neuronalen Netzen.

Lösungsvorschlag:

Quellenangaben: Die wesentlichen Teile des im folgenden verwendeten Codes sind im eingangs genannten Artikel *RW2019* bereits aufgelistet. Teile dieses Codes werden auch im Notebook *Li2020* verwendet.

Anpassungen: Neben den angepassten Ländern und Zeiträumen wurde der Code zwecks Ausgabe der log. Mortalitätsraten für die Trainingsdaten ergänzt.

```r
# Lee-Carter-Modell: Getrennte Berechnung für jede Geschlecht-Land-Kombination

country <- c('JPN', 'AUS', 'USA', 'ESP', 'ITA', 'FRATNP', 'CHE', 'POL', 'DNK',
             'DEUT', 'DEUTE', 'DEUTW')

gender <- c("Male", "Female")
Year.min <- 1958
ObsYear <- 2007
lc_ctrain <- list()
lc_cval <- list()
for(c in country){
  lc_gtrain <- list()
  lc_gval <- list()
  for(g in gender){
    train <- subset(data, Year >= Year.min & Year<=ObsYear &  Gender == g & Country == c)
    # Lee-Carter-Modell per SVD berechnen
    train[,ax:= mean(log_mortality), by = (Age)]
    train[,mx_adj:= log_mortality-ax]
    rates_mat<-as.matrix(train %>% dcast.data.table(Age~Year,value.var="mx_adj",sum))[,-1]
    svd_fit <- svd(rates_mat)
    ax <- train[,unique(ax)]
    bx <- svd_fit$u[,1]*svd_fit$d[1]
    kt <- svd_fit$v[,1]
    c1 <- mean(kt)
    c2 <- sum(bx)
    ax <- ax+c1*bx
    bx <- bx/c2
    kt <- (kt-c1)*c2

    # Extrapolation und Vorhersage
    vali <- subset(data, Year>ObsYear &  Gender == g & Country == c)
    t_forecast <- vali[,unique(Year)] %>% length()
    forecast_kt  =kt %>% forecast::rwf(t_forecast, drift = T)
    kt_forecast = forecast_kt$mean

    # log.Mortalitätsrate für Trainings- und Testzeitraum berechnen
    fitted = (ax+(bx)%*%t(kt)) %>% melt
    train$pred_LC = fitted$value
    fitted_vali = (ax+(bx)%*%t(kt_forecast)) %>% melt
    vali$pred_LC =   fitted_vali$value

    lc_gtrain[[g]] <- train
    lc_gval[[g]] <- vali
  }

  lc_ctrain[[c]] <- rbindlist(lc_gtrain)
  lc_cval[[c]] <- rbindlist(lc_gval)
}
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
lc_train <- rbindlist(lc_ctrain)
lc_val <- rbindlist(lc_cval)
```

[Lernziel 5.2, 3 Punkte] Aufgabe B3: b) Berechnen Sie für jede Land-Geschlecht-Kombination den mittleren quadratischen Fehler (mse) über alle in die Modellberechnung eingeflossenen Jahre und Alter. Geben Sie das Ergebnis für die je fünf Kombinationen mit dem höchsten und den niedrigsten Fehler aus und Interpretieren sie das Ergebnis. Eignet sich die mittels des Lee-Carter-Modells angestellte Messfehlerbetrachtung für die Aufspürung von Datenanomalien? Welche Aspekte spielen noch eine Rolle?

Lösungsvorschlag:

```
# Mittleren quadratischen Fehler je Land und Geschlecht berechnen und absteigend sortieren:
mse_lc_train <- lc_train %>%
  mutate(error_LC = (pred_LC-log_mortality)^2)  %>%
  group_by(Country,Gender) %>%
  summarize(mse_LC=mean(error_LC,na.rm=TRUE)) %>%
  arrange(desc(mse_LC))
```

```
## `summarise()` has grouped output by 'Country'. You can override using the `.groups` argument.
```

```
# Die  fünf Kombinationen mit dem höchsten Fehler
head(mse_lc_train,5)
```

```
## # A tibble: 5 x 3
## # Groups:   Country [3]
##   Country Gender mse_LC
##   <chr>   <fct>   <dbl>
## 1 AUS     Female 0.0580
## 2 DNK     Female 0.0487
## 3 CHE     Female 0.0426
## 4 DNK     Male   0.0351
## 5 CHE     Male   0.0292
```

```
# Die  fünf Kombinationen mit dem kleinsten Fehler
tail(mse_lc_train,5)
```

```
## # A tibble: 5 x 3
## # Groups:   Country [4]
##   Country Gender  mse_LC
##   <chr>   <fct>    <dbl>
## 1 JPN     Male   0.00556
## 2 DEUT    Female 0.00485
## 3 USA     Male   0.00358
## 4 USA     Female 0.00283
## 5 AUS     Male   0.00218
```

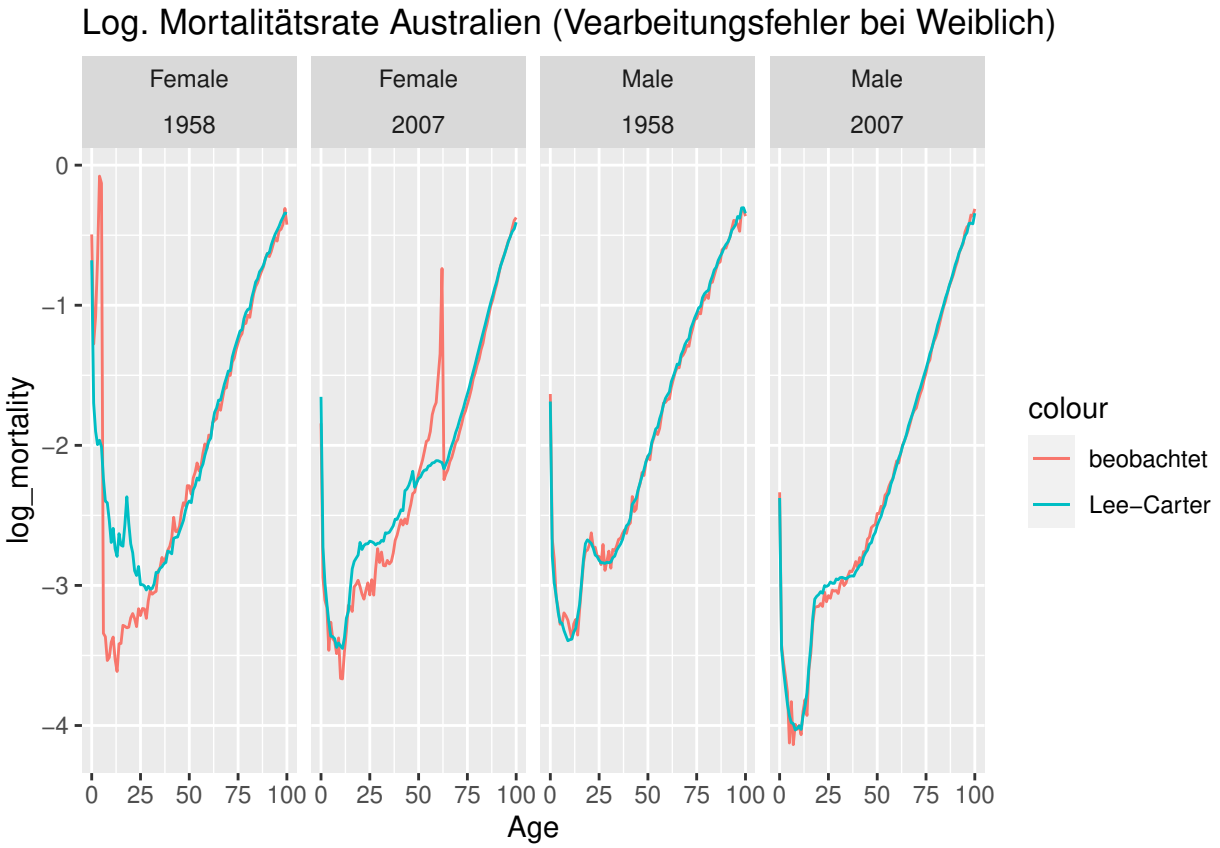
Den größten MSE weisen die Mortalitätsdaten der Kombination Australien-Weiblich auf. Nicht unerwartet passt die Struktur der in Kap. 2 bereits festgestellten Datenannomalie für Australien-Weiblich nicht gut zur Struktur eines Mortalitätsmodells, von daher ist die angestellte Messfehlerbetrachtung grundsätzlich für das Aufspüren von Datenannomalien geeignet. Allerdings wird der Effekt von den hohen natürlichen

Schwankungen der Mortalitätsraten je Alter und Jahr bei wenig bevölkerungsreichen Ländern überlagert. Entsprechend ist der jeweilige MSE der bevölkerungsreichen Länder wie Japan, Deutschland und die USA recht gering, wobei Australien wieder eine Sonderrolle zu spielen scheint.

[Lernziele 5.2 und 6.1, 3 Punkte] Aufgabe B3: c) Stellen Sie den Altersverlauf der gemessenen logarithmierten Mortalitätsrate und den entsprechenden Verlauf des Lee-Carter-Modells in einer Graphik dar, und zwar jeweils für die Jahre 1958 und 2007 sowie für die Land-Geschlecht-Kombination mit dem größten bzw. dem kleinsten Messfehler (4 Felder) und interpretieren Sie kurz das Ergebnis. Führen Sie die Analyse in Aufgabe B4 und B5 ohne die Länder Australien und Deutschland (gesamt) durch und entfernen Sie dazu die Daten mit der Länderkennung "AUS" und "DEUT" aus den relevanten Dateien.

Lösungsvorschlag:

```
lc_train %>%
    filter((Year == 1958 | Year == 2007) & Country == "AUS") %>%
    ggplot(aes(Age)) +
      geom_line(aes(y = log_mortality, colour = "beobachtet")) +
      geom_line(aes(y = pred_LC, colour = "Lee-Carter")) +
      facet_grid(~ Gender + Year)  +
      ggtitle("Log. Mortalitätsrate Australien (Vearbeitungsfehler bei Weiblich)")
```



Für die Kombination mit dem geringsten Meßfehler, Australien-Männlich, stimmten die logarithmierten Mortalitätsraten des Lee-Carter-Modells erwartungsgemäß sehr gut mit den beobachteten Werten überein, sowohl zu Beginn als auch zum Ende des betrachteten Zeitraums 1958 bzw. 2007, d.h. das Lee-Carter-Modell bildet auch die Mortalitätsverringerung 1958/2007 gut ab. Im Gegensatz dazu kann das Lee-Carter-Modell dem "unnatürlichen" Verlauf der stark verzerrten Mortalitätsrate bei Australien-Weiblich nicht folgen, ensprechend groß ist der Meßfehler.

```r
# Kap 3 abschließend Daten für die weiteren Schritte vorbereiten:

# a) Deutschland (gesamt) für Vergleich in Kap. 6 sichern
lc_val_DEUT<-filter(lc_val,Country=='DEUT')

# b) Die in der Aufgabenstellung genannten Länder aus dem Datensatz und den Lee-Carter-Prognosen entfer
data <- filter(data  ,!Country%in%c('AUS','DEUT'))
lc_val<-filter(lc_val,!Country%in%c('AUS','DEUT'))

#table(data$Country)   # 2*101*(2014-1958+1) = 11.514 Datensätze je Land
#table(lc_val$Country) # 2*101*(2014-2008+1) =  1.414
```

4. Tiefes neuronales Netz mit Embeddings (Multipopulationsmodell)

Erstellen Sie mittels eines voll verknüpften tiefen neuronalen Netzes auf Basis der Daten der Jahre 1958 bis 2007 mit (dem TensorFlow-API) Keras eine Mortalitätsprognose für die Jahre ab 2008. Gehen Sie dabei in folgenden Schritten vor:

[Lernziel 6.1, 4 Punkte] Aufgabe B4: a) Bereiten Sie die Merkmale "Gender", "Age" und "Country" als numerische Merkmale für die Embeddings auf. Für jede Kategorie eines Merkmals ist ein eindeutiger, ganzzahligen Wert (fortlaufende Nummerierung 0,1,2,...) zu vergeben und ein entsprechendes Mapping zu erzeugen. Stellen Sie sicher, dass das Merkmal Year numerisch ist. Zeigen Sie eine Zufallsauswahl von zehn aufbereiteten Datensätzen sowie die Mappings an.

Lösungsvorschlag:

Quellenangaben: Die folgende Umsetzung basiert in wesentlichen Teilen auf a) den in *RW2018* aufgelisteten Codes zum Aufbau des neuronalen Netzwerks (das dort aufgelistete Netz hat eine verborgene Schicht mehr, siehe Teil "middle") sowie b) dem im Notebook *Li2020* umgesetzten Modelltraining.

```r
data<- as.data.frame(data)
# Attributwerte der nominalen Merkmale sichern und in Zahlen umwandeln (0,1,..)
cat_features <- c("Country","Gender","Age")
cat_variable <- c("Country_cat","Gender_cat","Age_cat")
data[cat_variable] <- lapply(data[cat_features], factor)
data[cat_features] <- lapply(data[cat_variable], as.integer)
data[cat_features]<-data[cat_features]-1

# Sicherstelle, dass das Merkmal Year numerisch ist
data$Year <- as.numeric(as.character(data$Year))

# Reihenfolge der Spalten vorgeben und 10 zufällig ausgewählte Datensätze ausgeben
data<-data %>% select(Country_cat,Gender_cat,Age_cat,Country,Year,Gender,Age, log_mortality)
sample_n(data,10)
```

```
##    Country_cat Gender_cat Age_cat Country Year Gender Age log_mortality
## 1        DEUTE     Female      46       1 1985      0  46     -5.940391
## 2          ITA       Male      46       6 2012      1  46     -6.328897
## 3          CHE     Female      45       0 1985      0  45     -6.315534
## 4          JPN     Female      75       7 1964      0  75     -2.800824
```

```
## 5        DEUTW       Male      84       2 2010      1  84     -2.227747
## 6        DEUTE     Female      76       1 1972      0  76     -2.691311
## 7        DEUTW     Female      13       2 2011      0  13     -9.304651
## 8       FRATNP     Female      42       5 2010      0  42     -6.837297
## 9          ITA       Male      87       6 2001      1  87     -1.897240
## 10         JPN     Female      75       7 2014      0  75     -4.392239
```

```
setDT(data) # umwandeln in data.table
```

```
# Mapping für Land erstellen und anzeigen
country_mapping=unique(data[,c('Country_cat','Country')])
row.names(country_mapping) <- NULL
country_mapping
```

```
##     Country_cat Country
##  1:         CHE       0
##  2:       DEUTE       1
##  3:       DEUTW       2
##  4:         DNK       3
##  5:         ESP       4
##  6:      FRATNP       5
##  7:         ITA       6
##  8:         JPN       7
##  9:         POL       8
## 10:         USA       9
```

```
# Mapping für Lebensalter erstellen und anzeigen
age_mapping=unique(data[,c('Age_cat','Age')])
row.names(age_mapping) <- NULL
head(age_mapping)
```

```
##    Age_cat Age
## 1:       0   0
## 2:       1   1
## 3:       2   2
## 4:       3   3
## 5:       4   4
## 6:       5   5
```

```
# Mapping für Geschecht erstellen und anzeigen
gender_mapping=unique(data[,c('Gender_cat','Gender')])
row.names(gender_mapping) <- NULL
gender_mapping
```

```
##    Gender_cat Gender
## 1:     Female      0
## 2:       Male      1
```

[Lernziel 6.1, 3 Punkte] Aufgabe B4: b) Bereiten Sie die Trainings-, Validierungs- und Testdaten für die Modellierung vor. Verwenden Sie die Jahre ab 2008 als Testdaten. Unterteilen Sie die Datensätze der Jahre 1958 bis 2007 über eine reproduzierbare Zufallsauswahl in 10%

Validierungs- und 90% Trainingsdaten. Erzeugen Sie für jede der drei Stichproben die Feature-Matrix ("X") aus den in vorangegangenen Aufgabenteil aufbereiteten Merkmalen und den Output-Vektor ("y") aus der logarithmierten Mortalitätsrate. Geben Sie die jeweilige Anzahl an Datensätzen und Variablen an und überprüfen Sie das Ergebnis.

Lösungsvorschlag:

```
# Gesamte Trainingdaten selektieren (1958 bis 2007).
training <-dplyr::filter(data,Year%in%1958:2007)

## Validierungsdaten auswählen: 10% der Trainingsdaten
col_vector <- c("Year","Age","Country","Gender","log_mortality")
Training <- training %>% select(one_of(col_vector))
dt = sort(sample(nrow(Training), nrow(Training)*.9))
val<-Training[-dt,]

# Validierungsdaten "X": Features aufbereiten
X_validation <- val[,c("Year","Age","Country","Gender")]
X_val <- list(as.matrix(X_validation$Year),as.matrix(X_validation$Age),
              as.matrix(X_validation$Country),as.matrix(X_validation$Gender))

# Validierungsdaten "y" : log_mortality aufbereiten
y_validation <- val[, "log_mortality"]
y_val <- as.matrix(y_validation)

## "Lerndaten" auswählen:
train<-setdiff(Training,val)

# Lerndaten "X": Features aufbereiten
X_training <- train[,c("Year","Age","Country","Gender")]
X_train <- list(as.matrix(X_training$Year),as.matrix(X_training$Age),
                as.matrix(X_training$Country),as.matrix(X_training$Gender))

# Lerndaten "y" : log_mortality aufbereiten
y_training <- train[, "log_mortality"]
y_train <- as.matrix(y_training)

## Testdaten auswählen (2008 bis 2014)
test <- dplyr::filter(data,Year%in%2008:2014)

# Testdaten "X": Features aufbereiten
X_testing <- test[,c("Year","Age","Country","Gender")]
X_test <- list(as.matrix(X_testing$Year),as.matrix(X_testing$Age),
               as.matrix(X_testing$Country),as.matrix(X_testing$Gender))

# Testdaten "y" : log_mortality aufbereiten
y_testing <- test[, "log_mortality"]
y_test <- as.matrix(y_testing)

# Check der Dimensionen der angelegten Datensätze
dim(X_training);dim(y_training)
```

```
## [1] 90900     4
```

```
## [1] 90900     1
```

```
dim(X_validation);dim(y_validation)
```

```
## [1] 10100     4
```

```
## [1] 10100     1
```

```
dim(X_testing);dim(y_testing)
```

```
## [1] 14140     4
```

```
## [1] 14140     1
```

[Lernziel 6.1, 12 Punkte] Aufgabe B4: c) Legen Sie die Architektur des tiefen neuronalen Netzes fest. Programmieren Sie dazu ein Netz mit vorgeschalteten Embeddings sowie vier folgenden verborgenen Schichten mit jeweils 128 Neuronen, ReLu-Aktivierung, Batch-Normalisierung und 5% Dropout. Verwenden Sie dabei zweidimensionale Embeddings für die Merkmale "Gender", "Country" und "Age". Geben Sie die Modellarchitektur aus (summary-Funktion) und überprüfen Sie das Ergebnis.

Lösungsvorschlag:

```
Year <- layer_input(shape=c(1),dtype="float32",name="Year")
Age <- layer_input(shape=c(1),dtype="int32",name="Age")
Country <- layer_input(shape=c(1),dtype="int32",name="Country")
Gender <- layer_input(shape=c(1),dtype="int32",name="Gender")

# Embeddings definieren und dimensionieren
Age_embed <- Age %>%
  layer_embedding(input_dim = 101,output_dim=2,input_length=1,name="Age_embed") %>%
  keras::layer_flatten()

Gender_embed <- Gender %>%
  layer_embedding(input_dim=2,output_dim=2,input_length = 1,name="Gender_embed") %>%
  keras::layer_flatten()

Country_embed <- Country %>%
  layer_embedding(input_dim=10,output_dim = 2,input_length = 1,name="Country_embed") %>%
  keras::layer_flatten()

# Features (Embeddings, Year) zusammenführen
features <- layer_concatenate(list(Year,Age_embed,Gender_embed,Country_embed))

# Die verborgenen Schichten definieren
middle <- features %>%
      layer_dense(units=128,activation="relu") %>%
      layer_batch_normalization() %>%
      layer_dropout(0.05) %>%

      layer_dense(units=128,activation="relu") %>%
      layer_batch_normalization() %>%
      layer_dropout(0.05) %>%
```

```
      layer_dense(units=128,activation="relu") %>%
      layer_batch_normalization() %>%
      layer_dropout(0.05)

# Alles aneinanderfügen und output definieren
main_output <- layer_concatenate(list(features,middle)) %>%
  layer_dense(units=128,activation="relu") %>%
  layer_batch_normalization() %>%
  layer_dropout(0.05) %>%
  layer_dense(units = 1, name = 'main_output')

# Das Keras Model aufsetzen und die Architektur ausgeben
model <- keras_model(inputs=c(Year,Age,Country,Gender),outputs=c(main_output))
summary(model)
```

```
## Model: "model"
## ____________________________________________________________________________
## Layer (type)                Output Shape      Param #  Connected to
## ============================================================================
## Age (InputLayer)            [(None, 1)]       0
## ____________________________________________________________________________
## Gender (InputLayer)         [(None, 1)]       0
## ____________________________________________________________________________
## Country (InputLayer)        [(None, 1)]       0
## ____________________________________________________________________________
## Age_embed (Embedding)       (None, 1, 2)      202      Age[0][0]
## ____________________________________________________________________________
## Gender_embed (Embedding)    (None, 1, 2)      4        Gender[0][0]
## ____________________________________________________________________________
## Country_embed (Embedding)   (None, 1, 2)      20       Country[0][0]
## ____________________________________________________________________________
## Year (InputLayer)           [(None, 1)]       0
## ____________________________________________________________________________
## flatten (Flatten)           (None, 2)         0        Age_embed[0][0]
## ____________________________________________________________________________
## flatten_1 (Flatten)         (None, 2)         0        Gender_embed[0][0]
## ____________________________________________________________________________
## flatten_2 (Flatten)         (None, 2)         0        Country_embed[0][0]
## ____________________________________________________________________________
## concatenate (Concatenate)   (None, 7)         0        Year[0][0]
##                                                        flatten[0][0]
##                                                        flatten_1[0][0]
##                                                        flatten_2[0][0]
## ____________________________________________________________________________
## dense (Dense)               (None, 128)       1024     concatenate[0][0]
## ____________________________________________________________________________
## batch_normalization (Batc (None, 128)         512      dense[0][0]
## ____________________________________________________________________________
## dropout (Dropout)           (None, 128)       0        batch_normalization[0][0]
## ____________________________________________________________________________
## dense_1 (Dense)             (None, 128)       16512    dropout[0][0]
## ____________________________________________________________________________
## batch_normalization_1 (Ba (None, 128)         512      dense_1[0][0]
```

```
## ____________________________________________________________________________
## dropout_1 (Dropout)       (None, 128)       0         batch_normalization_1[0][0]
## ____________________________________________________________________________
## dense_2 (Dense)           (None, 128)       16512     dropout_1[0][0]
## ____________________________________________________________________________
## batch_normalization_2 (Ba (None, 128)       512       dense_2[0][0]
## ____________________________________________________________________________
## dropout_2 (Dropout)       (None, 128)       0         batch_normalization_2[0][0]
## ____________________________________________________________________________
## concatenate_1 (Concatenat (None, 135)       0         concatenate[0][0]
##                                                       dropout_2[0][0]
## ____________________________________________________________________________
## dense_3 (Dense)           (None, 128)       17408     concatenate_1[0][0]
## ____________________________________________________________________________
## batch_normalization_3 (Ba (None, 128)       512       dense_3[0][0]
## ____________________________________________________________________________
## dropout_3 (Dropout)       (None, 128)       0         batch_normalization_3[0][0]
## ____________________________________________________________________________
## main_output (Dense)       (None, 1)         129       dropout_3[0][0]
## ============================================================================
## Total params: 53,859
## Trainable params: 52,835
## Non-trainable params: 1,024
## ____________________________________________________________________________
```

[Lernziel 6.1, 6 Punkte] Aufgabe B4: d) Kompilieren Sie das Modell. Verwenden Sie dazu Loss mse, Metrik mae und Optimierer adam mit Lernrate 0.1. Kontrollieren Sie den Trainingsprozess über Callbacks. Fitten Sie das neuronale Netz mit Batchsize 500 und mindestens 100 Epochen. Verwenden Sie dabei die in Aufgabenteil b) erstellten Trainings- und Validierungsdaten ("X","y"). Überprüfen Sie die Konvergenz des Fits und adjustieren Sie bei Bedarf die Optimierungsparameter.

Lösungsvorschlag:

```
model %>% compile(optimizer_adam(lr = 0.1),loss='mse',metrics=c('mae'))
```

```
# Callback-Funktion um den Trainingsprozess zu überwachen: Monitoring des Validation Loss
early_stop <- callback_early_stopping(monitor = 'val_loss', patience =20)

lr_reducer <- callback_reduce_lr_on_plateau(monitor = 'val_loss', factor = 0.1,
                                patience = 10, verbose = 0, mode = 'min',
                                 min_delta = 1e-04, cooldown = 0, min_lr = 0)
```

```
# Fitten des Modells (final verbose=0 um die lange Auflistung der Epochen zu entfernen)
{t1 <- proc.time()
history <- model %>% fit(
  x  = X_train,
  y  = y_train,
  batch_size = 500,
  epochs = 100,
  validation_data =list(X_val, y_val),
  verbose = 0,
  callbacks = list(early_stop,lr_reducer)
```

```
)
proc.time()-t1}
```

```
##    user  system elapsed
##  421.22   20.18  450.51
```

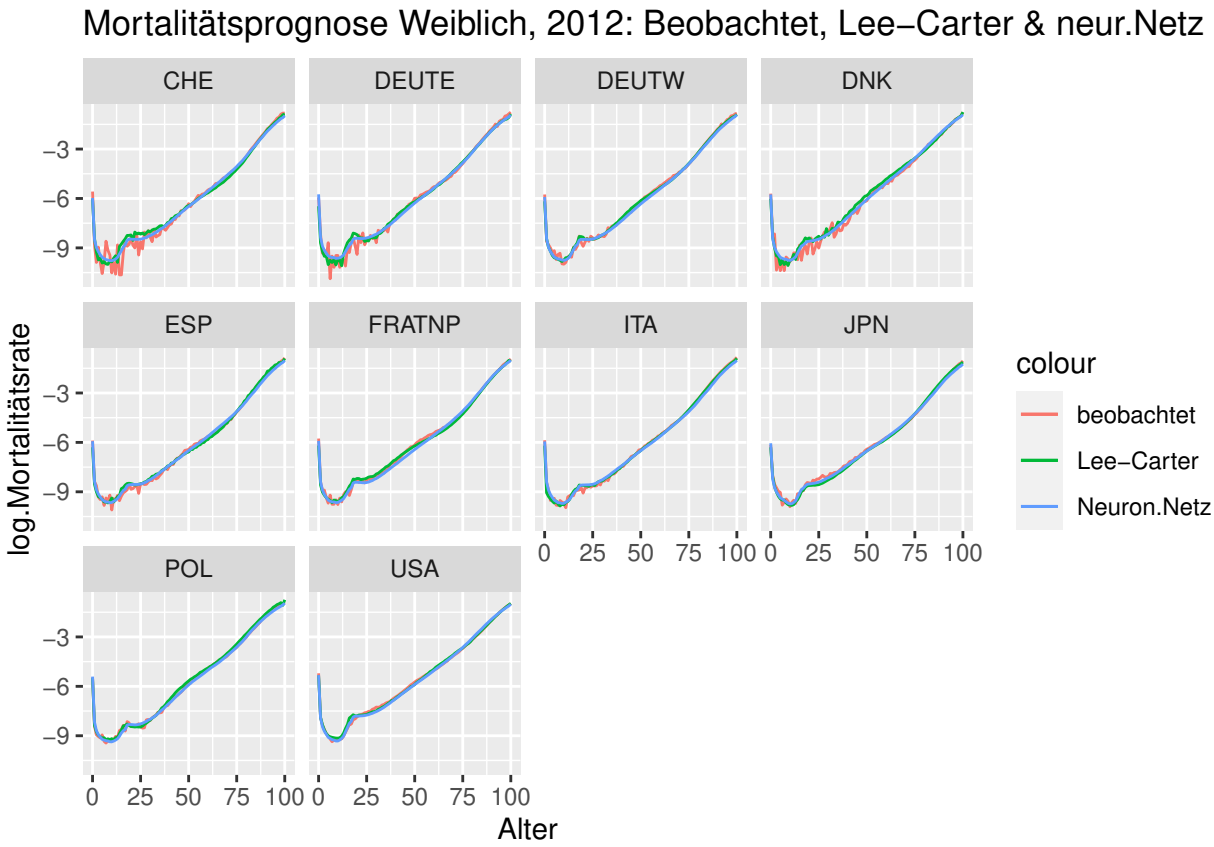
[Lernziel 6.1, 8 Punkte] Aufgabe B4: e) Wenden Sie das Neuronale Netz auf die in Aufgabenteil b) erstellten Testdaten an und prognostizieren Sie die (log.) Mortalitätsentwicklung. Stellen Sie für das Jahr 2012 die prognostizierten Werte des Neuronalen Netzes sowie des Lee-Carter-Modells den beobachteten Werten je Land und Geschlecht graphisch gegenüber und bewerten Sie das Ergebnis Ihrer "Sichtprüfung". Berechnen und vergleichen Sie den mittleren quadratischen Fehler für das Neuronale Netz sowie das Lee-Carter-Modell über den gesamten Prognosezeitraum und stellen Sie abschließend für jede Land-Geschlecht-Kombination den mittleren quadratischen Fehler der beiden Modelle graphisch gegenüber und interpretieren Sie das Ergebnis.

Lösungsvorschlag:

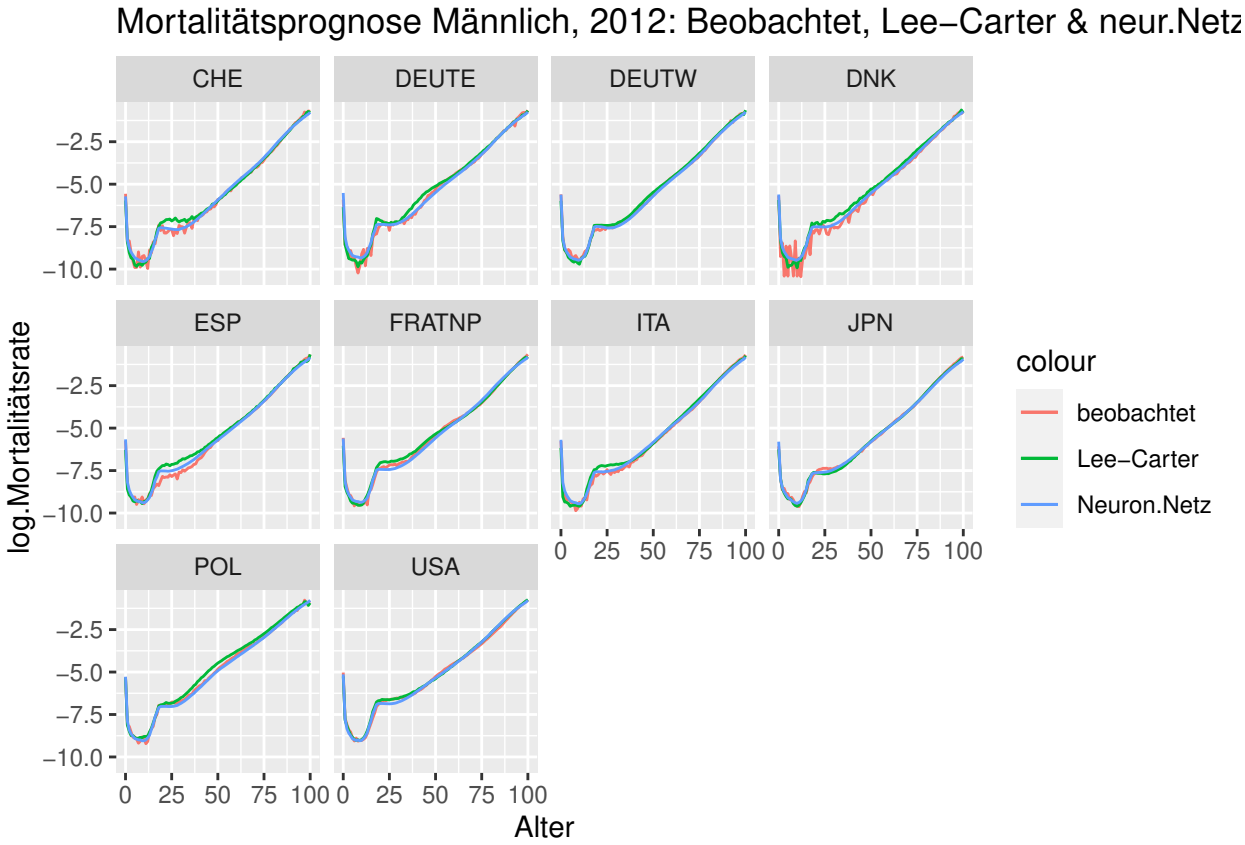
```
# Neuronales Netz auf Testdaten anwenden, Prognosedaten erstellen
pred_NN <- model %>% predict(X_test)
colnames(pred_NN)[1] <- "pred_NN"
nn_prediction <- cbind(test, pred_NN)

# Variablen "sprechend" aufbereiten
nn_prediction$Gender  <- nn_prediction$Gender_cat
nn_prediction$Country <- nn_prediction$Country_cat
nn_prediction <- nn_prediction[,c(4:9)]
nn_prediction <- nn_prediction %>% select(Country,Year,Gender,Age,pred_NN,log_mortality)

#.Mit den Prognosen des Lee-Carter-Modells für die folgenden Vergleiche zusammenführen
predictions <- inner_join(lc_val,nn_prediction[,c(1:5)],by=c("Country","Year","Gender","Age"))
sample_n(predictions,10)
```

```
##     Country Year Gender Age log_mortality   pred_LC   pred_NN
##  1:     USA 2008   Male  17    -7.1994454 -7.010470 -7.085099
##  2:   DEUTE 2012 Female  99    -0.8598253 -1.076448 -1.038663
##  3:     USA 2014   Male  60    -4.4693290 -4.599753 -4.602765
##  4:     USA 2008 Female  46    -6.0414955 -6.163072 -6.152619
##  5:     USA 2013   Male  37    -6.3333916 -6.238070 -6.391706
##  6:     ESP 2013 Female  29    -8.5023046 -8.281562 -8.347521
##  7:     DNK 2008   Male  74    -3.2470181 -3.078091 -3.200653
##  8:  FRATNP 2009   Male  93    -1.3310489 -1.380207 -1.360560
##  9:     ITA 2013 Female  75    -4.0955853 -4.054273 -4.147033
## 10:  FRATNP 2008 Female  88    -2.3118277 -2.312785 -2.242300
```

```
# Mortalitätsprognose Weiblich, 2012
predictions %>% filter(Year==2012 & Gender=="Female") %>%
  ggplot() + xlab("Alter") + ylab("log.Mortalitätsrate") + facet_wrap(~Country) +
  geom_line(aes(Age, log_mortality,color="beobachtet")) +
  geom_line(aes(Age, pred_LC, color="Lee-Carter")) +
  geom_line(aes(Age, pred_NN, color="Neuron.Netz")) +
  ggtitle("Mortalitätsprognose Weiblich, 2012: Beobachtet, Lee-Carter & neur.Netz")
```




```
# Mortalitätsprognose Männlich, 2012
predictions %>% filter(Year==2012 & Gender=="Male") %>%
  ggplot() + xlab("Alter") + ylab("log.Mortalitätsrate") + facet_wrap(~Country) +
  geom_line(aes(Age, log_mortality,color="beobachtet")) +
  geom_line(aes(Age, pred_LC, color="Lee-Carter")) +
  geom_line(aes(Age, pred_NN, color="Neuron.Netz")) +
  ggtitle("Mortalitätsprognose Männlich, 2012: Beobachtet, Lee-Carter & neur.Netz")
```



Die Graphiken zeigen für beide Geschlechter und alle betrachteten Länder auf den ersten Blick eine insgesamt gute Übereinstimmung zwischen den prognostizierten und beobachteten Werten für das Jahr 2012. Insgesamt bilden beide Modelle die Realität gut ab. Am auffälligsten sind in den Graphiken die starken Schwankungen der beobachteten log. Mortalitätsrate Dänemarks und der Schweiz, den beiden Ländern mit der geringsten Bevölkerungszahl. Bei genauerer Betrachtung sind in mehreren Ländern bei jungen erwachsenen Männern Abweichungen zwischen den prognostizierten und beobachteten Mortalitätsraten erkennbar. Dies trifft insbesondere auf das Lee-Carter-Modell zu. Im Falle von Spanien-Männlich wird hier die tatsächliche Mortalitätsrate erkennbar überschätzt. Dies sollte sich auch in der folgenden Betrachtung des mittleren quadratischen Fehlers widerspiegeln.

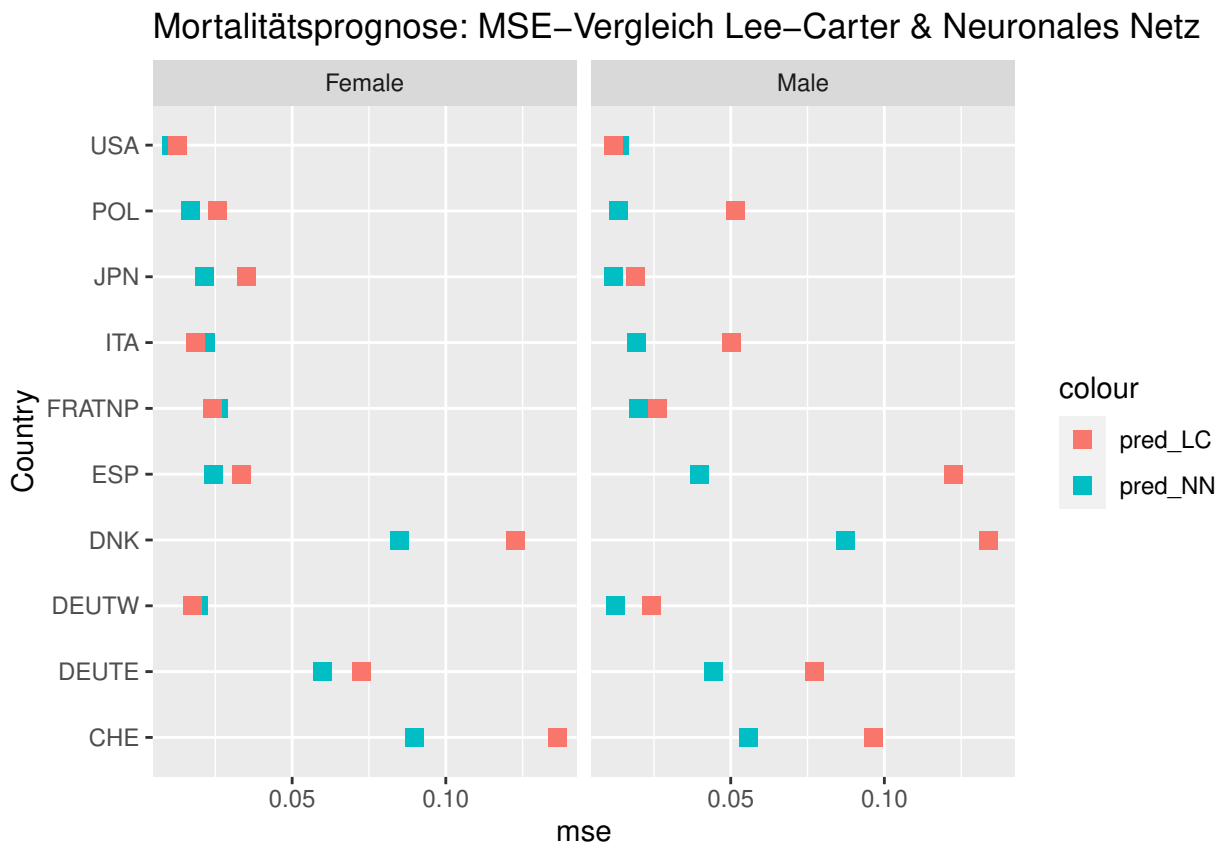
```
# Berechnung des mittleren quadratischen Fehlers über die gesamten Prognosedaten:
mse_pred <- predictions %>%
  mutate(error_NN = (pred_NN-log_mortality)^2,error_LC = (pred_LC-log_mortality)^2)  %>%
  summarize(mse_NN=mean(error_NN,na.rm=TRUE),mse_LC=mean(error_LC,na.rm=TRUE))  %>%
  mutate(rel_NN_LC = (mse_NN/mse_LC))
mse_pred
```

```
##       mse_NN     mse_LC rel_NN_LC
## 1 0.03470246 0.05558951 0.6242627
```

```
predictions %>%
  mutate(error_NN = (pred_NN-log_mortality)^2,error_LC = (pred_LC-log_mortality)^2)  %>%
  group_by(Gender,Country) %>%
  summarize(mse_NN=mean(error_NN,na.rm=TRUE),mse_LC=mean(error_LC,na.rm=TRUE))  %>%
  mutate(rel_NN_LC = (mse_NN/mse_LC))  %>%
  ggplot() + coord_flip() + facet_wrap(~Gender) +
```

```
    geom_point(aes(Country,mse_NN,color="pred_NN"),size=3,shape=15) +
    geom_point(aes(Country,mse_LC,color="pred_LC"),size=3,shape=15) +
    xlab("Country") + ylab("mse") +
    ggtitle("Mortalitätsprognose: MSE-Vergleich Lee-Carter & Neuronales Netz")
```

```
## `summarise()` has grouped output by 'Gender'. You can override using the `.groups` argument.
```



Der mittlere quadratische Fehler des neuronalen Netzes ist über die gesamten Prognosedaten deutlich geringer als der des Lee-Carter-Modells. Die Prognoseverbesserung gegenüber dem Lee-Carter-Modell trifft auf die überwiegende Mehrheit der Länder differenziert nach Geschlechtern zu. Die hohen Meßfehler von Dänemark, der Schweiz und in geringerem Maße auch von Deutschland-Ost sind zum Teil den aus der geringen Bevölkerungsanzahl resultierenden Schwankungen der beobachteten Werte geschuldet und wirken sich sowohl auf den MSE des Neuronalen Netzes als auch den des Lee-Carter-Modells aus. Auffällig ist der gegenüber dem Neuronalen Netz um ein mehrfaches höherere MSE des Lee-Carter-Modells bei Spanien-Männlich. Dies passt zur oben bebachteten Überschätzung bei jungen Männern in 2012.

Da die Ergebnisse von neuronalen Netzen deutlichen Schwankungen unterworfen sein können wurde die vorliegende Analyse mehrfach mit verschieden Zufallszahlen wiederholt. Es wurden dabei überwiegend sehr ähnliche Ergebnisse erzielt, sofern das Modell ausreichend trainiert wurde. Abschließend kann festgehalten werden, dass das verwendete neuronale Netz die log. Mortalitätsrate im Prognosezeitraum um rund 35% genauer als das Lee-Carter-Modell vorhersagt und daher ein potentiell genaueres, allerdings auch deutlich aufwändiger zu erstellendes und berechnendes Mortalitätsmodell ist.

5. Dimensionsreduzierung und Modellinterpretation mit Embeddings

[Lernziel 6.1, 3 Punkte] Aufgabe B5: a) Beschreiben Sie die Funktionsweise und Anwendungsmöglichkeiten von "Entity Embeddings".

Beschreiben Sie die Funktionsweise und Anwendungsmöglichkeiten von "Entity Embeddings".**

Lösungsvorschlag:

Mit einem "Entity Embedding" wird bei nominal skalierten Merkmalen jede Merkmalskategorie durch einen mehrdimensionalen Vektor ersetzt. Diese Embeddings können im Vergleich zur One-Hot-Codierung nicht nur die Speichernutzung drastisch reduzieren und die Berechnung neuronale Netze beschleunigen, sondern zeigen vor allem die intrinsischen Eigenschaften der kategorialen Variablen auf, indem ähnliche Werte im Einbettungsraum nahe beieinander abgebildet werden.

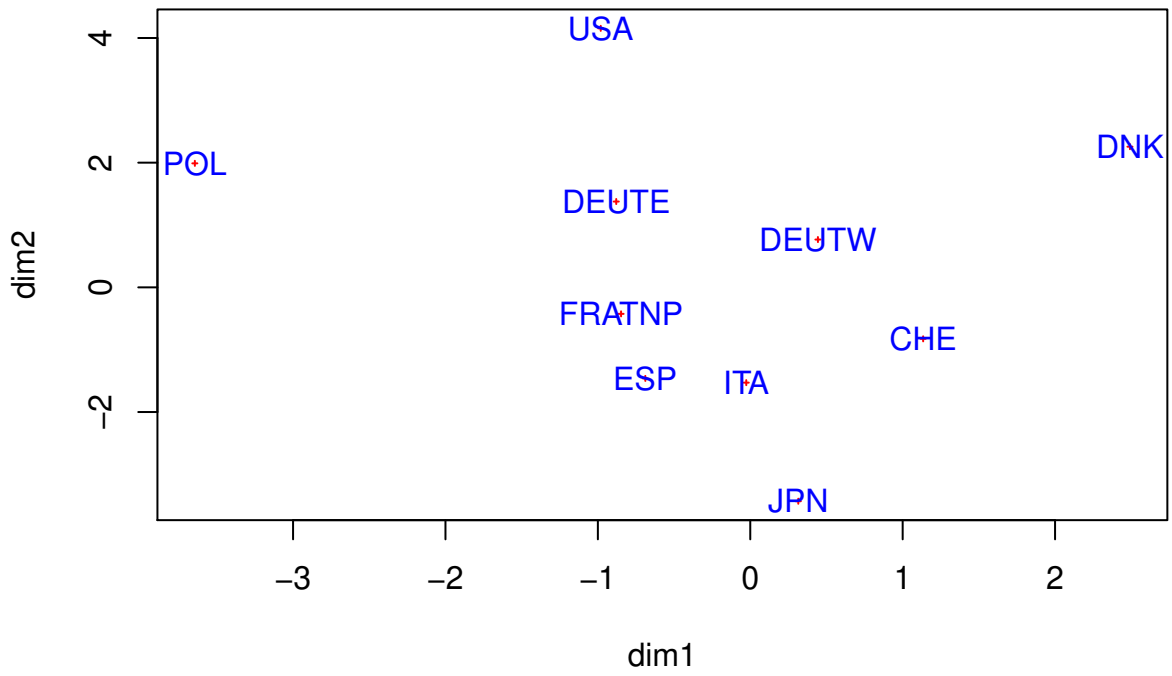
Mit Embeddings lassen sich hochkardinale Merkmale wie Tätigkeitsgruppen, Branchen, Postleitzahlen und Krankheitsdiagnosen mittels einer deutlich kleineren Matrix abbilden und beispielsweise über eine sogenannte Embedding-Layer in einem neuronalen Netz berechnen. Dieses "Mapping" kann interpretiert, extrahiert und sogar in anderen Verfahren des überwachten Lernens verwendet werden.

Wie wir oben gesehen haben ist es sogar möglich, metrische Merkmale wie das Alter über ein mehrdimensionales Embedding abzubilden.

[Lernziel 6.1, 6 Punkte] Aufgabe B5: b) Extrahieren ("get_weights") und visualisieren Sie das oben erstellte zweidimensionale Embedding für das Merkmal „Country" und interpretieren Sie das Ergebnis.

Lösungsvorschlag:

```
# Embedding für "Country" aus den Modellgewichten extrahieren und visualisieren
emb_country <- cbind(country_mapping,get_weights(model)[[3]])
colnames(emb_country)<-c("CountryName","CountryID","dim1","dim2")
plot(data=emb_country, dim2~dim1, pch=3, col="red",cex=0.25)
with(data=emb_country, text(dim2~dim1, labels=CountryName, col="blue"), pos=1)
```



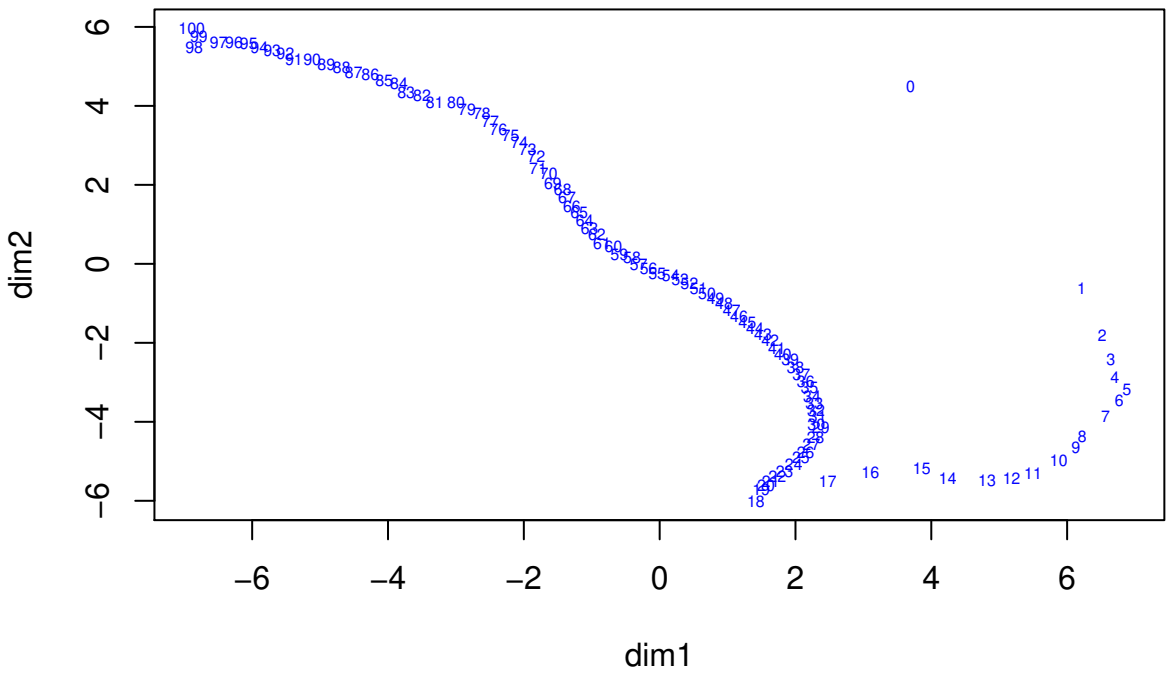
Das Embedding zeigt sogenannte Gewichte des neuronalen Netzes bei der Berechnung der Mortalitätsrate. Auf der einen Seite Japan, das zu Beginn des Betrachungszeitraums deutlich überdurchschnittliche und am Ende deutlich unterdurchschnittliche Mortalitäsraten und damit einen vergleichsweise starken Rückgang der Mortalitätsraten aufweist. Entgegengesetzt hierzu die USA, ein Land, das im Gegensatz zu Japan zu Beginn deutlich unterdurchschnittliche und am Ende überdurchschnittliche Mortalitätsraten und damit einen vergleichsweise geringen Rückgang der Mortalitätsraten aufweist. Die Achse USA in Richtung Japan könnte man als Achse der Mortalitätsverbesserung (Rückgang) interpretieren und Abweichungen von dieser Achse könnten verschiedene Mortalitätsniveaus anzeigen.

Dieses Muster kann bei einer Wiederholung des Fits an den Achsen gespiegelt sowie rotiert auftreten.

[Lernziel 6.1, 6 Punkte] Aufgabe B5: c) Extrahieren ("get_weights") und visualisieren Sie das oben erstellte zweidimensionale Embedding für das Merkmal „Age" und interpretieren Sie das Ergebnis.

Lösungsvorschlag:

```
# Embedding für "Age" aus den Modellgewichten extrahieren und visualisieren
emb_age <- cbind(age_mapping,get_weights(model)[[1]])
emb_age <- emb_age[,c(1:4)] # Einschränkung dim1 & dim2 (falls >2dim)
colnames(emb_age)<-c("AgeName","AgeID","dim1","dim2")
plot(data=emb_age, dim2~dim1, pch=1, col="black",cex=0.0)
with(data=emb_age, text(dim2~dim1, labels=AgeName, col="blue", cex=.5), pos=1)
```



Dieses Alters-Embedding zeigt ebenfalls sogenannte Gewichte des neuronalen Netzes bei der Berechnung der Mortalitätsrate. Während die aufeinander folgenden Lebensalter hier recht deutlich einer Linie folgen, liegt das Alter 0 weit entfernt von allen anderen Lebensaltern und zeigt seine Besonderheit.

Die Achse von rechts unten nach links oben kann man als Achse der Mortalitätsverschlechterung (Mortalitätsniveau) interpretieren. Die Abweichung von dieser Achse nach rechts oben bzw. links unten ist weniger offensichtlich.

Prüfung im Vertiefungswissen

Actuarial Data Science Completion 2021

Musterlösung für Aufgabenteil C

Hinweis zur Musterlösung: Im Notebook sind einige Arbeitsschritte dargestellt (bspw. im Rahmen der Datenanalyse: Korrelationsplots, WordCloud), die in der Aufgabenstellung nicht gefordert sind. Diese wurden zu Informationszwecken in der Musterlösung belassen und gehen nicht in die Punktevergabe ein.

Daten einlesen, Explorative Datenanalyse & Visualisierung, Basismodell Regression

Aufgabe C01 [Lernziel 3.4.5] [10 Punkte]

Bevor die eigentliche Verarbeitung beginnt, sollen die notwendigen Programmbibliotheken importiert sowie der Datensatz aus der Datei data.csv eingelesen und die ersten Datensätze (vollständig) angezeigt werden. Im Anschluss sind die Daten geeignet aufzubereiten und ein Basismodell (LGBM Regressor) für die Zielgröße "Endkosten" zu erstellen.

Hinweis: Dieses Modell wird als Benchmark fungieren und muss nicht zwingend ausoptimiert sein.

Import

In [ ]:

```
# Import nötiger Pakete

# Basis
import sys

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

# Numerik / DataFrames
import pandas as pd
import numpy as np

# Plotting Bibliotheken
import matplotlib.pyplot as plt
plt.rcParams.update({'font.size': 12})
import seaborn as sns

# Datenaufbereitung
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

# Modellierungsbilbiotheken
import lightgbm as lgb
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_precision_recall_curve
from sklearn.metrics import average_precision_score

# Pakete für NLP
from wordcloud import WordCloud
import gensim
import gensim.corpora as corpora

# Pakete für stopwords, stemming
import nltk
#ltk.download('stopwords') # start the NLTK Downloader and download the stopwords
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer

import pyLDAvis
import pyLDAvis.gensim_models

# Modellpersistenz
from joblib import dump, load

# Erklärbarkeitsbibs
import shap
from sklearn.inspection import plot_partial_dependence
import lime
```

In [2]:
```
print(sys.version)
```

```
3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
```

Explorative Datenanalyse & Visualisierung

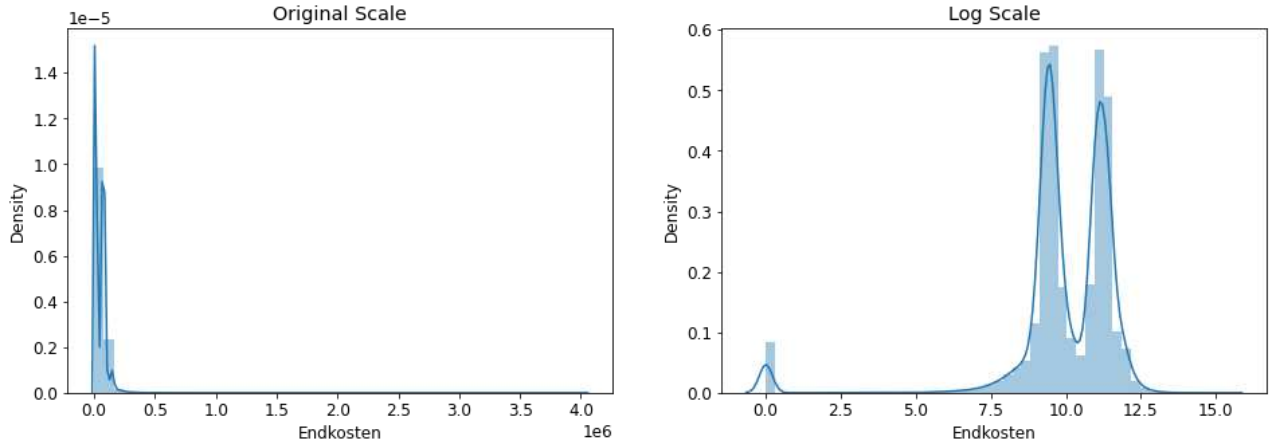
In [3]:
```
# Einlesen der Daten
data = pd.read_csv("data.csv")
data = data.drop(data.columns[0], axis=1)

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

# Zeige die ersten 5 Zeilen der Daten um einen Überblick zu gewinnen
data.head(5)
```

Out[3]:

	ID	Ereignisdatum	Meldedatum	Alter	Geschlecht	Familienstand	Kinder	AndereHausstandsmitglieder	LohnWoche	Arbeitszei
0	WC3693607	1992-11-07T16:00:00Z	1993-01-07T00:00:00Z	56	F	M	0	0	635.230258	
1	WC6792814	1999-12-02T07:00:00Z	1999-12-15T00:00:00Z	36	M	M	0	0	767.044987	
2	WC3373583	1991-01-21T09:00:00Z	1991-02-06T00:00:00Z	32	M	M	0	0	750.103486	
3	WC3931490	1992-04-23T09:00:00Z	1992-05-09T00:00:00Z	30	M	M	0	0	808.507288	
4	WC9943635	2005-08-28T11:00:00Z	2005-09-16T00:00:00Z	18	M	S	0	0	953.180178	

In [4]:
```
# Kopie der unverarbeiteten Daten erstellen für Auswertung unten (Validierung Daten-ID)
data_original = data.copy(deep=True)
```

In [5]:
```
# Quelle: https://www.kaggle.com/msafi04/predict-actuarial-loss-eda-and-baseline

# Verteilung des Zielwertes
plt.figure(figsize = (16, 5))
plt.subplot(1, 2, 1)
plt.title('Original Scale')
sns.distplot(data['Endkosten'])
plt.subplot(1, 2, 2)
plt.title('Log Scale')
sns.distplot(np.log1p(data['Endkosten']));
```



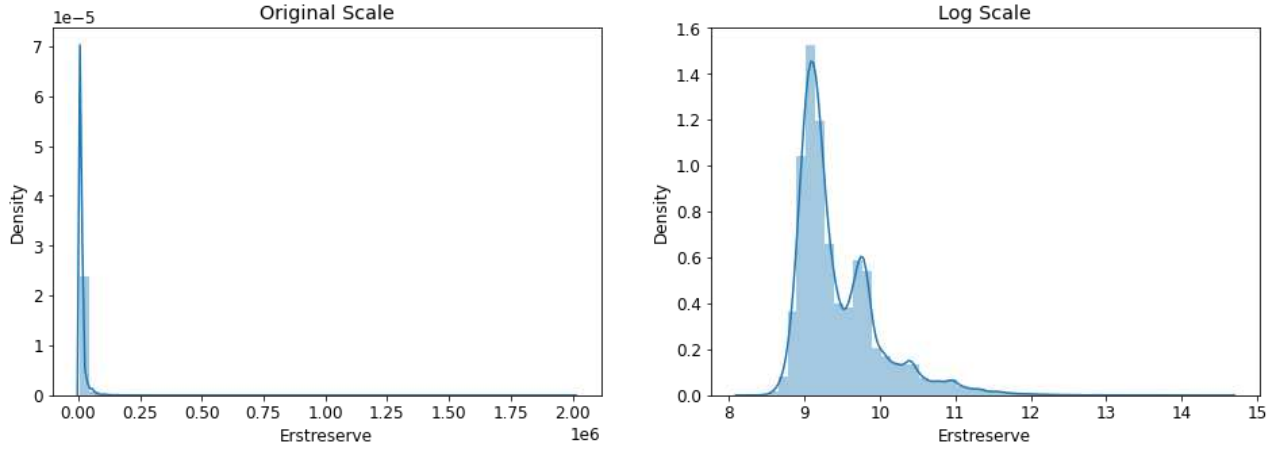
In [6]:
```
# Inputation
data['Familienstand'].fillna('U', inplace = True)

# Check
data.isna().any().sum()
```

Out[6]:
```
0
```

In [7]:
```
# Typecast Datumstypen
data['Ereignisdatum'] = pd.to_datetime(data['Ereignisdatum'])
```

```
data['Meldedatum'] = pd.to_datetime(data['Meldedatum'])
```

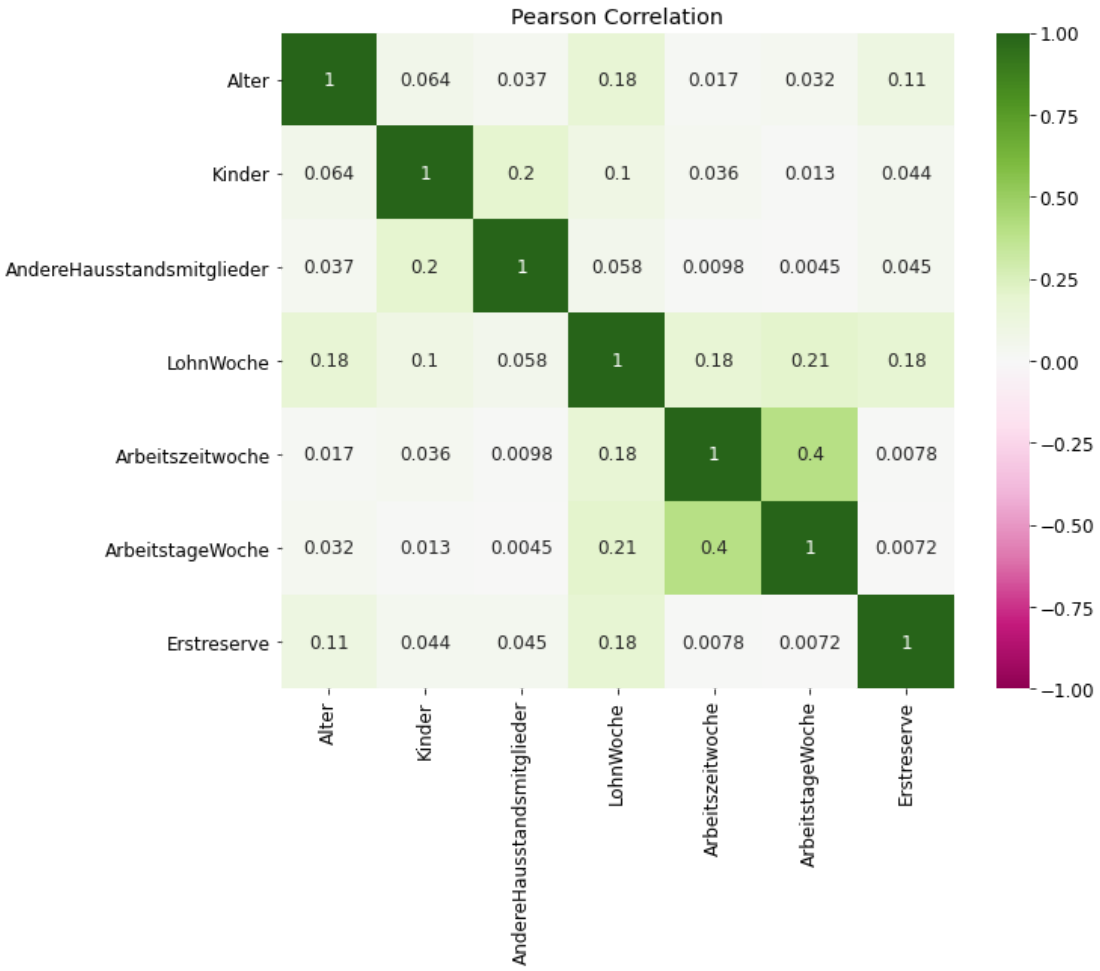
In [8]:
```
# Verteilung Initial Incurred CalimsCost
plt.figure(figsize = (16, 5))
plt.subplot(1, 2, 1)
plt.title('Original Scale')
sns.distplot(data['Erstreserve'])
plt.subplot(1, 2, 2)
plt.title('Log Scale')
sns.distplot(np.log1p(data['Erstreserve']));
```



In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54000 entries, 0 to 53999
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          54000 non-null  object
 1   Ereignisdatum               54000 non-null  datetime64[ns, UTC]
 2   Meldedatum                  54000 non-null  datetime64[ns, UTC]
 3   Alter                       54000 non-null  int64
 4   Geschlecht                  54000 non-null  object
 5   Familienstand               54000 non-null  object
 6   Kinder                      54000 non-null  int64
 7   AndereHausstandsmitglieder  54000 non-null  int64
 8   LohnWoche                   54000 non-null  float64
 9   Arbeitszeitkategorie        54000 non-null  object
 10  Arbeitszeitwoche            54000 non-null  float64
 11  ArbeitstageWoche            54000 non-null  int64
 12  Schadenbeschreibung         54000 non-null  object
 13  Erstreserve                 54000 non-null  float64
 14  Endkosten                   54000 non-null  float64
dtypes: datetime64[ns, UTC](2), float64(4), int64(4), object(5)
memory usage: 6.2+ MB
```

In [10]:
```
# Korrelation der Features
numerical_features = [c for c in data.columns if data[c].dtype in ['float64', 'int64']
                      if c not in ["Meldedatum", "Ereignisdatum", "ID", "Schadenbeschreibung"]]

categorical_features = [c for c in data.columns if c not in numerical_features
                        if c not in ["Meldedatum", "Ereignisdatum", "ID", "Schadenbeschreibung"]]

numerical_features, categorical_features

numerical_features.remove('Endkosten')


corr1 = data[numerical_features].corr(method = 'pearson')
fig = plt.figure(figsize = (10, 8))

sns.heatmap(corr1, mask = None, annot = True, cmap = 'PiYG', vmin = -1, vmax = +1)
plt.title('Pearson Correlation')
plt.xticks(rotation = 90)
plt.show()
```



In [11]:
```
# Erzeuge eine Wordcloud um einen Überblick der häufigen Beschreibungen zu erhalten
long_string = ','.join(list(data['Schadenbeschreibung'].values))
wordcloud = WordCloud(background_color="white", max_words=1000, contour_width=5, contour_color='steelblue')
wordcloud.generate(long_string)
wordcloud.to_image()
```

Out[11]:

In [12]:
```
# Standardisierung
data['Erstreserve'] = np.log1p(data['Erstreserve'])
target = np.log1p(data_original['Endkosten'])

data.drop(['ID', 'Schadenbeschreibung', 'Endkosten'], axis = 1, inplace = True)
data.shape
```

Out[12]: (54000, 12)

In [13]:
```
std = StandardScaler()

data[numerical_features] = std.fit_transform(data[numerical_features])
```

In [14]:
```
data
```

Out[14]:

	Ereignisdatum	Meldedatum	Alter	Geschlecht	Familienstand	Kinder	AndereHausstandsmitglieder	LohnWoche	Arbeitszei
0	1992-11-07 16:00:00+00:00	1993-01-07 00:00:00+00:00	1.827878	F	M	-0.230187	-0.090944	-0.793016	
1	1999-12-02 07:00:00+00:00	1999-12-15 00:00:00+00:00	0.177992	M	M	-0.230187	-0.090944	-0.263632	

	Ereignisdatum	Meldedatum	Alter	Geschlecht	Familienstand	Kinder	AndereHausstandsmitglieder	LohnWoche	Arbeitszei
2	1991-01-21 09:00:00+00:00	1991-02-06 00:00:00+00:00	-0.151985	M	M	-0.230187	-0.090944	-0.331671	
3	1992-04-23 09:00:00+00:00	1992-05-09 00:00:00+00:00	-0.316974	M	M	-0.230187	-0.090944	-0.097114	
4	2005-08-28 11:00:00+00:00	2005-09-16 00:00:00+00:00	-1.306905	M	S	-0.230187	-0.090944	0.483910	
...	...	...	...	...	...	...	...	...	
53995	2000-10-10 07:00:00+00:00	2000-11-03 00:00:00+00:00	1.580395	M	U	-0.230187	-0.090944	0.761699	
53996	1994-04-03 10:00:00+00:00	1994-05-20 00:00:00+00:00	-0.729445	M	S	-0.230187	-0.090944	-0.219194	
53997	1992-09-21 13:00:00+00:00	1992-10-01 00:00:00+00:00	0.013004	M	M	-0.230187	-0.090944	1.026159	
53998	1988-09-18 10:00:00+00:00	1988-10-08 00:00:00+00:00	-0.646951	M	S	-0.230187	-0.090944	-0.022638	
53999	1988-10-18 08:00:00+00:00	1988-11-22 00:00:00+00:00	-0.976928	M	M	-0.230187	-0.090944	-0.244588	

54000 rows × 12 columns

Erstellung des Basismodells Regression

In [15]:
```
X = data
y = target

X_pre_enc = data[['Geschlecht','Familienstand','Arbeitszeitkategorie']]
X_num = data.drop(['Meldedatum','Ereignisdatum','Geschlecht','Familienstand','Arbeitszeitkategorie'],axis=1)

# Categoricals encoden
enc = OneHotEncoder(handle_unknown='ignore')
fit = enc.fit(X_pre_enc)
X_post_enc = pd.DataFrame(enc.transform(X_pre_enc).toarray()
                          ,columns=enc.get_feature_names(['Geschlecht', 'Familienstand', 'Arbeitszeitkategorie']))
X = X_num.join(X_post_enc)
```

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

```
      Alter    Kinder  AndereHausstandsmitglieder  LohnWoche  \
0  1.827878 -0.230187                   -0.090944  -0.793016
1  0.177992 -0.230187                   -0.090944  -0.263632
2 -0.151985 -0.230187                   -0.090944  -0.331671
3 -0.316974 -0.230187                   -0.090944  -0.097114
4 -1.306905 -0.230187                   -0.090944   0.483910

   Arbeitszeitwoche  ArbeitstageWoche  Erstreserve  Geschlecht_F  \
0         -2.922770         -5.262874     0.545216           1.0
1          0.021078          0.170688     0.826888           0.0
2          0.021078          0.170688    -0.677993           0.0
3          0.021078          0.170688    -0.543440           0.0
4          0.578022          0.170688     1.911642           0.0

   Geschlecht_M  Geschlecht_U  Familienstand_M  Familienstand_S  \
0           0.0           0.0              1.0              0.0
1           1.0           0.0              1.0              0.0
2           1.0           0.0              1.0              0.0
3           1.0           0.0              1.0              0.0
4           1.0           0.0              0.0              1.0

   Familienstand_U  Arbeitszeitkategorie_F  Arbeitszeitkategorie_P
0              0.0                     0.0                     1.0
1              0.0                     1.0                     0.0
2              0.0                     1.0                     0.0
3              0.0                     1.0                     0.0
4              0.0                     1.0                     0.0
```

Training des Basis-Modells

In [17]:
```
# Und ab ins Training...
lgb_instance = lgb.LGBMRegressor(num_iterations=500
                                 , silent=False
                                )

num_round = 200
lgb_basis = lgb_instance.fit(X,y)
```

```
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001576 seconds.
You can set `force_row_wise=true` to remove the overhead.
```

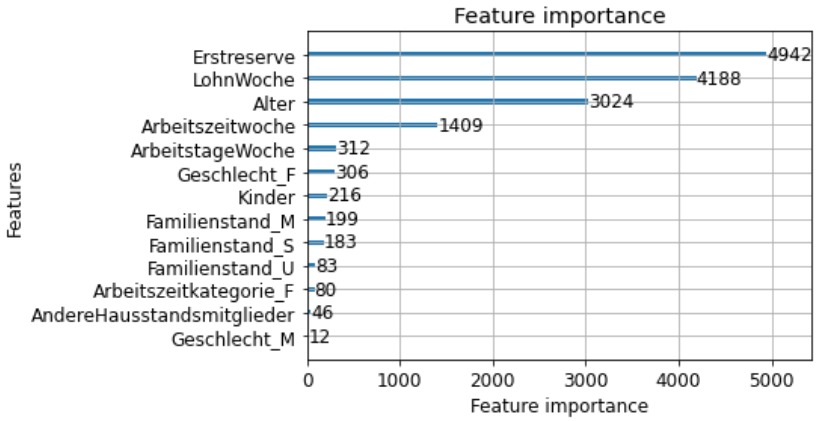
```
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 748
[LightGBM] [Info] Number of data points in the train set: 54000, number of used features: 14
[LightGBM] [Info] Start training from score 9.972913
```

Performance des Basis-Modells

In [18]:
```
y_pred = lgb_basis.predict(X, num_iteration=lgb_basis.best_iteration_)
rmse = np.sum(np.sqrt(np.power(y_pred - y,2)))/y_pred.size
print('The rmse of prediction is:', rmse)
lgb.plot_importance(lgb_basis)
```

```
The rmse of prediction is: 0.9738763988087364
```

Out[18]: <AxesSubplot:title={'center':'Feature importance'}, xlabel='Feature importance', ylabel='Features'>



Basismodell Klassifikation (Großschaden)

Aufgabe C02 [Lernziel 6.1.1] [10 Punkte]

Es ist ein Modell zur Klassifikation von Groß- und Kleinschäden zu erstellen: Teilen Sie den eingelesenen Datensatz anhand der Zielgröße „Endkosten" geeignet in Groß- und Kleinschäden ein (bspw. 95%-Quantil), erstellen Sie die Trainings- und Testdaten und führen Sie das Training aus. Die Verwendung aller Merkmale bis auf die textuelle Beschreibung ist zulässig.

Hinweis: Auch dieses Modell wird als Benchmark fungieren und muss nicht zwingend ausoptimiert sein. Die Wahl der Methode ist freigestellt und die Imbalance zwischen Groß- und Normschäden muss nicht behandelt werden.

In [19]:
```
# Großschäden schwellwertbasiert markieren
claims = y
major_claims_95pc_quantile = np.quantile(claims,0.95)
```

Schwellwert für Großschäden basierend auf 95%-Quantil:

In [20]:
```
np.exp(major_claims_95pc_quantile)
```

Out[20]: 124516.80295531332

Das 95%-Quantil der Schadenshöhen liegt bei ~124.000, also etwas höher als man es klassicherweise vermuten würde.

In [21]:
```
def claims_mapper(claim_amount,threshold=11.7):
    if(claim_amount > threshold):
        return 1
    return 0

major_claims = np.zeros((len(claims)))
major_claims = claims.apply(claims_mapper)
major_claims_y = pd.DataFrame(list(major_claims),columns=['Is_Major_Claim'])

sample = int(len(X)*0.8)
X_train = X[:sample]
X_test = X[sample:]
major_claims_y_train = major_claims_y[:sample]
major_claims_y_test = major_claims_y[sample:]

# Train dummy model
clf_basis = svm.SVC(kernel='linear',degree=2,class_weight={0:1, 1: 10})
#(weight sollte 1:10 sein, wenn wir das 90%-quantil verwenden und so ein Verhältnis von 1:9 bei den Schäden haben.
#Für 95%-Quantil entsprechend 1:19. Ist aber gar nicht so gut wie 1:2 oder 1:3)

clf_basis.fit(X_train, major_claims_y_train)
print(clf_basis.score(X,major_claims_y))
```

```
y_pred = clf_basis.predict(X_test)
print(confusion_matrix(major_claims_y_test, y_pred))
```

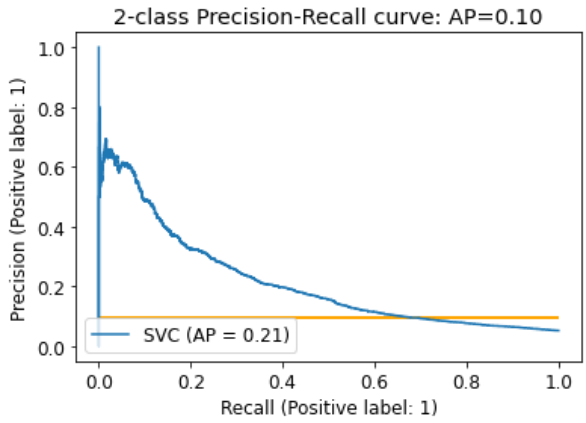
```
0.889537037037037
[[9412  840]
 [ 361  187]]
```

Deutlich aussagekräftiger als die Confusion-Matrix ist bei unbalancierten Datensätzen ein Precision-Recall-Plot.

In [22]:
```
average_precision = average_precision_score(major_claims_y_test, y_pred)

print('Average precision-recall score: {0:0.2f}'.format(
      average_precision))

disp = plot_precision_recall_curve(clf_basis, X_train, major_claims_y_train)
plt.hlines(y=average_precision,xmin=0.0,xmax=1.0,linewidth=2, color = 'orange')
disp.ax_.set_title('2-class Precision-Recall curve: '
                   'AP={0:0.2f}'.format(average_precision))
```

```
Average precision-recall score: 0.10
```

Out[22]:
```
Text(0.5, 1.0, '2-class Precision-Recall curve: AP=0.10')
```



Text Mining

Topic (LDA) des Schadentextes ermitteln

Hinweis: Dieser Aufgabenteil beschäftigt sich mit der Spalte 'Schadenbeschreibung' des Datensatzes data.csv.

Aufgabe C03 [Lernziel 5.1.2 - 5.1.4] [12 Punkte]

Erläutern Sie die Text-Mining-Anwendung "Topic Models". Als Vorbereitung für die Textanalyse sind die Textbestandteile des Datensatzes in einer Abfolge von Preprocessing-Schritten zu bearbeiten, um eine Topic-Analyse vorzubereiten. Erläutern Sie eine sinnvolle Vorverarbeitung und geben Sie im Anschluss zehn bearbeitete Texte aus, um den Unterschied zu den Rohdaten zu verdeutlichen.

Hinter dem Begriff "Topic Models" stehen statistische Modelle, welche im "Natural Language Processing" (NLP) verwendet werden. Sie dienen dazu, übergreifende Themen ("Topics") und latente semantische Strukturen zu identifizieren, die in (einer Sammlung von) Text-Dokumenten auftaucht.

Ein frühes Topic Model ist die "Probabilistic Latent Semantic Analysis" (PLSA). Eines der derzeit gebräuchlichsten Verfahren, die "Latent Dirichlet Allocation" (LDA), ist eine Verallgemeinerung der PLSA und basiert darauf, in bezug auf Wort-Vektoren eine Hauptkomponentenanalyse der Worthäufigkeiten relativ zueinder durchzuführen. Der limitierende Aspekt hierbei besteht darin, dass nicht tausende von Wörtern einbezogen werden sollten, sondern nur häufig auftretende. Dies kann allerdings dazu führen, dass wichtige, aber seltene Worte nicht identifiziert werden.

In [23]:
```
# Selecting some columns to work with
Claim_description = data_original[['ID', 'Schadenbeschreibung']].copy()
```

In [24]:
```
# remove stopwords
stop_words = stopwords.words('english') # Selecting english stopwords
stop_words.extend(['left', 'right', 'leav'])
stop_words.extend(['injuri', 'lacer', 'pain', 'lower'])
```

In [25]:
```
stemmer = SnowballStemmer("english")
'''
Write a function to perform the pre processing steps on the entire dataset
'''
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text, pos='v'))
```

```
# Tokenize and lemmatize
def preprocess(text):
    result=[]
    for token in gensim.utils.simple_preprocess(text) :
        if lemmatize_stemming(token) not in stop_words and len(token) > 3:
            result.append(lemmatize_stemming(token))
    return result
```

In [26]:
```
data_words = []

for word in Claim_description.Schadenbeschreibung.values:
    data_words.append(preprocess(word))

#Show first 10 tokens
for i in range(1, 11):
    print(data_words[i])
```

```
['bend', 'take', 'sampl', 'strain', 'back']
['strike', 'move', 'object', 'fractur', 'forearm']
['strike', 'walk', 'cooler', 'bruis', 'hand']
['hammer', 'hand', 'hand']
['catch', 'finger', 'cylind', 'press', 'middl', 'finger']
['lift', 'heavi', 'object', 'back', 'strain', 'shoulder']
['pole', 'bruis', 'abras', 'cornea']
['catch', 'basket', 'finger', 'shoulder', 'ankl']
['cut', 'food', 'chef', 'knife', 'middl', 'finger']
['walk', 'foot', 'slip', 'knee', 'strain', 'knee']
```

Aufgabe C04 [Lernziel 5.1.3] [10 Punkte]

Auf Basis der Latent Dirichlet Allocation (LDA) ist die Analyse der Topics mit "gensim" (Python) bzw. "topicmodels" (R) für sechs Topics durchzuführen. Die Zusammensetzung der Topics ist grafisch kompakt mit LDAvis darzustellen (Python: pyLDAvis, R: LDAvis).

In [27]:
```
# Quelle: https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24
# Create Dictionary
id2word = corpora.Dictionary(data_words)

# filter words that show up less than 15 times, more than 50% of the time and keep it to 10.000 Words
id2word.filter_extremes(no_below=15, no_above=0.50, keep_n=10000)

# Create Corpus
texts = data_words

# Term Document Frequency
# Converts a collection of words to  a list of (word_id, word_frequency) 2-tuples.
corpus = [id2word.doc2bow(text) for text in texts]

# View first description
print(corpus[0])
```

```
[(0, 1), (1, 1), (2, 1), (3, 2), (4, 1)]
```

In [28]:
```
def get_model(corpus, id2word, num_topics, random_state=0):
    lda_model = gensim.models.LdaMulticore(corpus = corpus,
                                           id2word = id2word,
                                           num_topics = num_topics,
                                           random_state = random_state
                                           )
    return lda_model
```

In [29]:
```
pyLDAvis.enable_notebook()
```

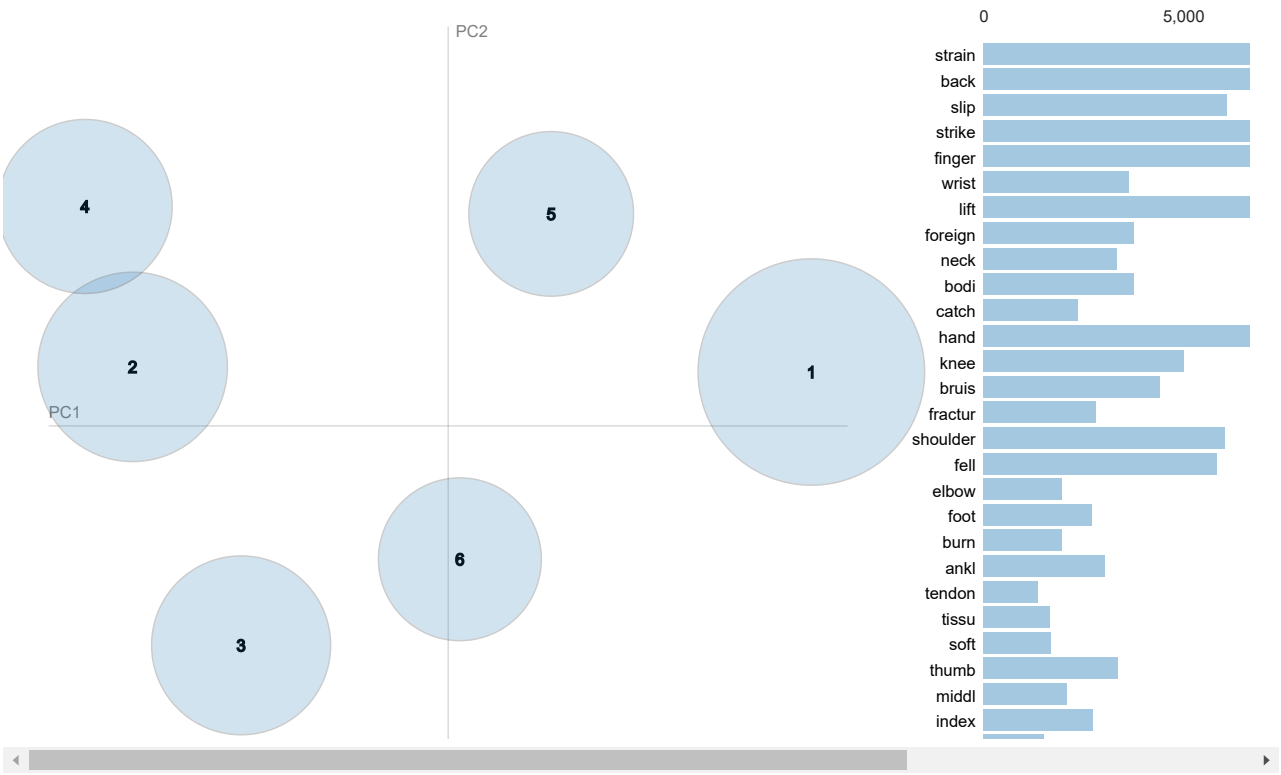
In [30]:
```
num_topics = 6
lda_model = get_model(corpus=corpus, id2word=id2word, num_topics=num_topics, random_state=2)
LDAvis_prepared = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word)
LDAvis_prepared
```

Out[30]:

Selected Topic: 0 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)

λ = 1



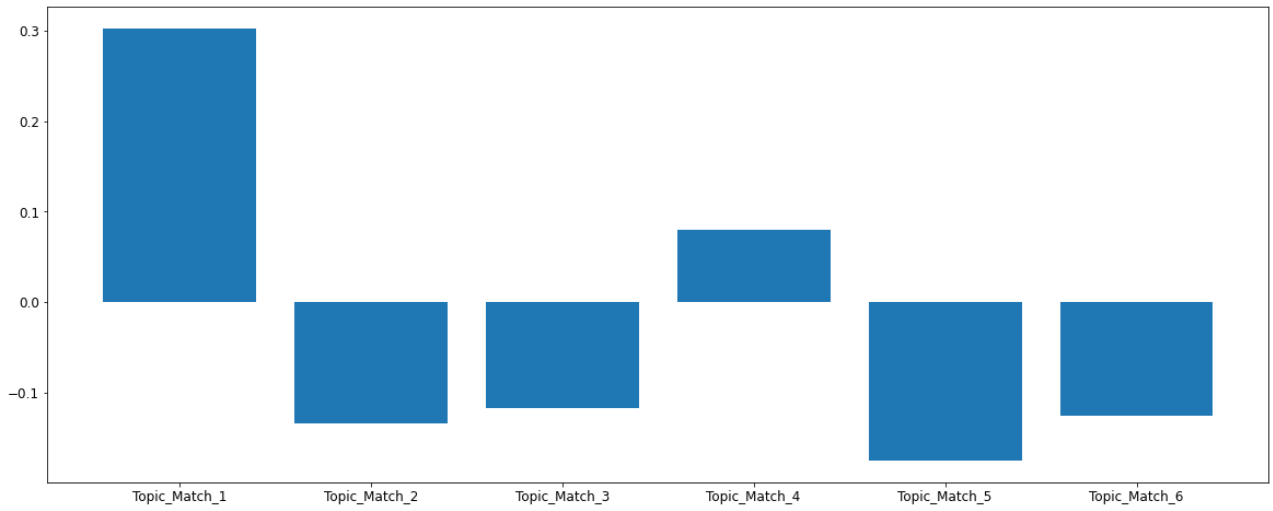
Anhand der Visualisierung der identifizierten Topics ist natürlich der Einfluss auf das Target nicht sichtbar, zumal die Häufigkeiten der einzelnen Wörter ziemlich gering sind und keine "dominanten" Effekte offensichtlich sind. Eine Differenzierung in Hauptkomponenten ist aber ein guter Indikator, dass zumindest eine heuristische Trennung möglich ist. Ob diese auch in bezug auf das Target etwas hergibt, muss im nächsten Schritt geprüft werden.

Aufgabe C05 [Lernziel 6.1.1] [12 Punkte]

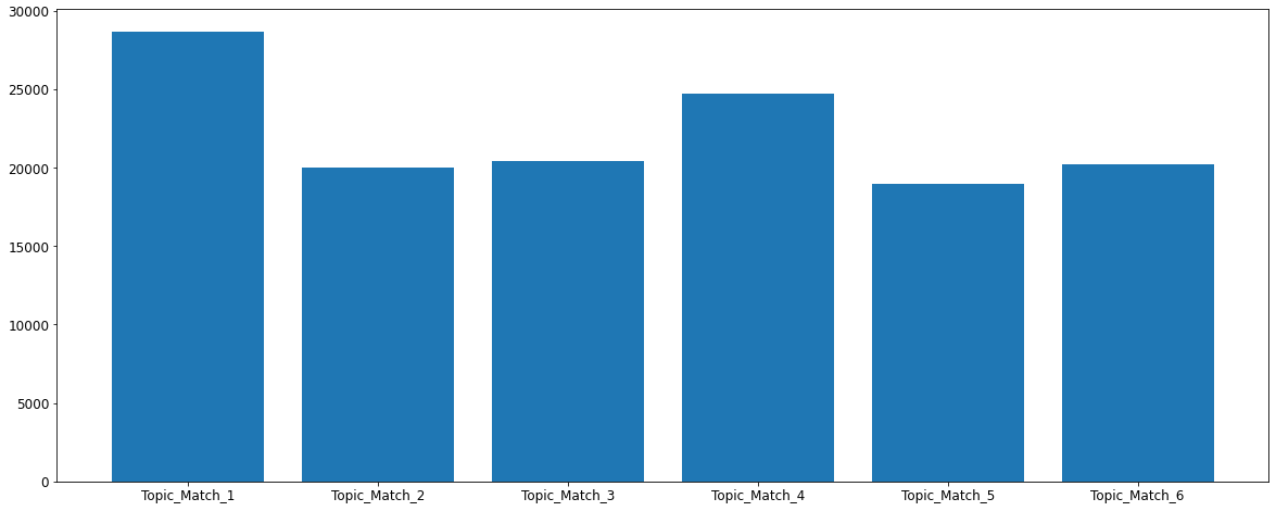
Überprüfen Sie, ob die Hinzunahme einer geeigneten Anzahl Topics die Prädiktionsgüte des Großschadenmodells zu verbessern vermag. Adressieren Sie auch die Imbalance der Trainingsdaten zwischen Großschaden "ja/nein": Was ist eine geeignete Methode des Umgangs und wie wirkt sich dieser auf das verbesserte Modell aus?

In [31]:
```
topics = lda_model.print_topics()

# Vorgeschlagener Algorithmus: Für jedes Topic schauen wir, ob die darin enthaltenen Wörter in der Beobachtung sind.
# Dann bilden wir also je nach Anzahl Topics Gruppen der Art Topic_Member_i für das i-te Topic.
# Diese neu gewonnenen Flag überführen wir dann in die Klassifikation.

topic_word_dict = {}
num_topics = len(topics)
for j in range(num_topics):
    splitted_topic = str.split(topics[j][1],'"')
    topic_word_dict[j] = [ (splitted_topic[i]).upper() for i in range(1,len(splitted_topic),2)]

# Als nächstes nehmen wir die Beobachtungen und checken, welche davon wo enthalten sind.

def topic_check(input,topic_num=num_topics,topics=topic_word_dict):
    output_vector = np.zeros(topic_num,dtype=np.int)
    for check_id in range(topic_num):
        if any(x in input for x in topics[check_id]):
            output_vector[check_id] = 1
    return output_vector

topic_flag_series = data_original['Schadenbeschreibung'].apply(topic_check)
topic_column_names = []
for i in range(1,num_topics+1):
    topic_column_names.append("Topic_Match_"+str(i))
topic_df = pd.DataFrame.from_dict(dict(zip(topic_flag_series.index, topic_flag_series.values))).transpose()
topic_df.columns=topic_column_names

# Nun ergänzen wir den Trainings-DataFrame um die Topic-Klassen und schauen, ob die Analyse besser wird.
# Da wir die Reihenfolge nicht verändert haben, können wir genauso wie zuvor die Train- und Test-Datensätze erzeugen.

df_train_topics = X.copy()
X_train_topics = X_train.join(topic_df[:sample])
X_test_topics = X_test.join(topic_df[sample:])
```

In [32]:
```
# Darstellung des mittleren Schadenbedarfs je Topic
overall_mean_claims = np.mean(y)
claims_size_diff_from_mean = {}
for n in topic_column_names:
    claims_size_diff_from_mean[n] = overall_mean_claims - np.mean(
        (df_train_topics.join(topic_df).join(y)).loc[X_train_topics[X_train_topics[n]==1].index]['Endkosten'])
fig, ax = plt.subplots(figsize=(20, 8))
plt.bar(range(len(claims_size_diff_from_mean)), list(claims_size_diff_from_mean.values()), align='center')
plt.xticks(range(len(claims_size_diff_from_mean)), list(claims_size_diff_from_mean.keys()))
plt.show()
```



In [33]:
```
# Darstellung des mittleren Schadenbedarfs je Topic
# Target exponentiert, da oben der log genommen wurde
overall_mean_claims = np.mean(np.exp(y))
claims_size_diff_from_mean = {}
for n in topic_column_names:
    claims_size_diff_from_mean[n] = overall_mean_claims - np.exp(
        np.mean((df_train_topics.join(topic_df).join(y)).loc[X_train_topics[X_train_topics[n]==1].index]['Endkosten']
fig, ax = plt.subplots(figsize=(20, 8))
plt.bar(range(len(claims_size_diff_from_mean)), list(claims_size_diff_from_mean.values()), align='center')
plt.xticks(range(len(claims_size_diff_from_mean)), list(claims_size_diff_from_mean.keys()))
plt.show()
```



In [34]:
```
print("== Radial, Weight 1:3 ===")
clf_topr = svm.SVC(kernel='rbf',class_weight={0:1, 1: 3})
clf_topr.fit(X_train_topics, major_claims_y_train)
print(clf_topr.score(X_test_topics,major_claims_y_test))
y_pred_topr = clf_topr.predict(X_test_topics)
print(confusion_matrix(major_claims_y_test, y_pred_topr))
```

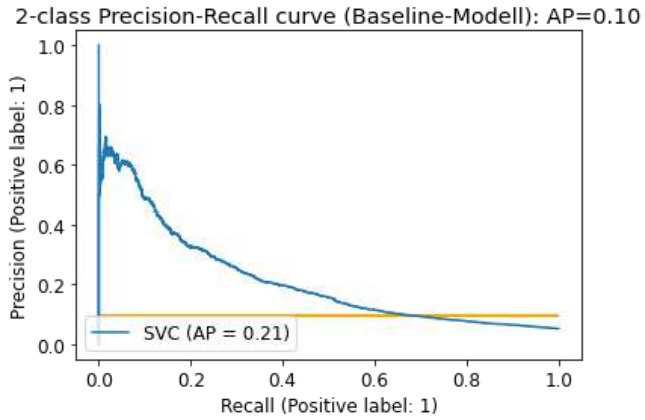
```
== Radial, Weight 1:3 ===
0.9300925925925926
[[9759  493]
 [ 262  286]]
```

In [35]:
```
average_precision = average_precision_score(major_claims_y_test, y_pred)

print('Average precision-recall score: {0:0.2f}'.format(
      average_precision))

disp = plot_precision_recall_curve(clf_basis, X_train, major_claims_y_train)
plt.hlines(y=average_precision,xmin=0.0,xmax=1.0,linewidth=2, color = 'orange')
```

```
disp.ax_.set_title('2-class Precision-Recall curve (Baseline-Modell): '
                   'AP={0:0.2f}'.format(average_precision))
```

```
Average precision-recall score: 0.10
```

Out[35]:
```
Text(0.5, 1.0, '2-class Precision-Recall curve (Baseline-Modell): AP=0.10')
```

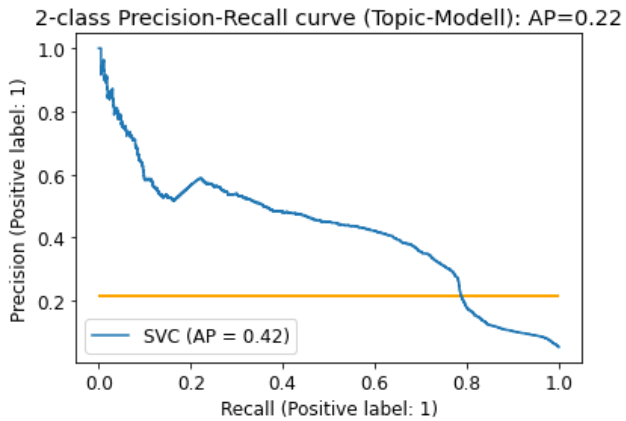


Hier zur Erinnerung / zum Vergleich die Precision-Recall-Kurve des Baseline-Modells.

In [36]:
```
average_precision = average_precision_score(major_claims_y_test, y_pred_topr)

print('Average precision-recall score: {0:0.2f}'.format(
      average_precision))

# PRC berechnen
disp = plot_precision_recall_curve(clf_topr, X_train_topics, major_claims_y_train)
plt.hlines(y=average_precision,xmin=0.0,xmax=1.0,linewidth=2, color = 'orange')
disp.ax_.set_title('2-class Precision-Recall curve (Topic-Modell): '
                   'AP={0:0.2f}'.format(average_precision))
```

```
Average precision-recall score: 0.22
```

Out[36]:
```
Text(0.5, 1.0, '2-class Precision-Recall curve (Topic-Modell): AP=0.22')
```



Es ist zu erkennen, dass im Vergleich zum Basismodell eine deutliche Verbesserung erreicht worden ist. Weitere Verbesserung erfordert ein Hyperparametertuning über Topics und Behandlung der Imbalance der Großschadentabelle über das Eingeben von Weights hinaus, da diese Abhilfe (für das SVC-Beispiel!) jenseits von 1:4-1:5 instabil ist.

Anmerkung: Die Verbesserung hängt auch davon ab,wie viele Topics verwendet werden. Dies kann als Hyper-Hyper-Parameter aufgefasst werden, da es sich um einen veränderlichen Teil der Gesamt-Pipeline handelt, aberkeinen Hyperparameter im engen Sinne als Teil des Algorithmus. Dennoch ist die Anzahl der Topics in eine eventuelle Optimierung einzubeziehen. Erfahrungsgemäß sind 8-20 Topics performant, während bei noch größerenZahlen zwar die Performance noch geringfügug steigt, der Aufwand dies aber oftmals verbietet.

Modellinterpretationsverfahren: Anforderungen und Verfahren

Aufgabe C06 [Lernziele 5.3.1, 5.3.5, 5.3.6] [8 Punkte]

Zur Modellinterpretierbarkeit sind die wesentlichen Anforderungen und Eigenschaften zu nennen:

- Regulatorische und operative Anforderungen an die Modellinterpretierbarkeit sind zu beschreiben.

- Das Kosten-Nutzen-Verhältnis zwischen Modellinterpretierbarkeit und Prognosegüte ist zu erläutern

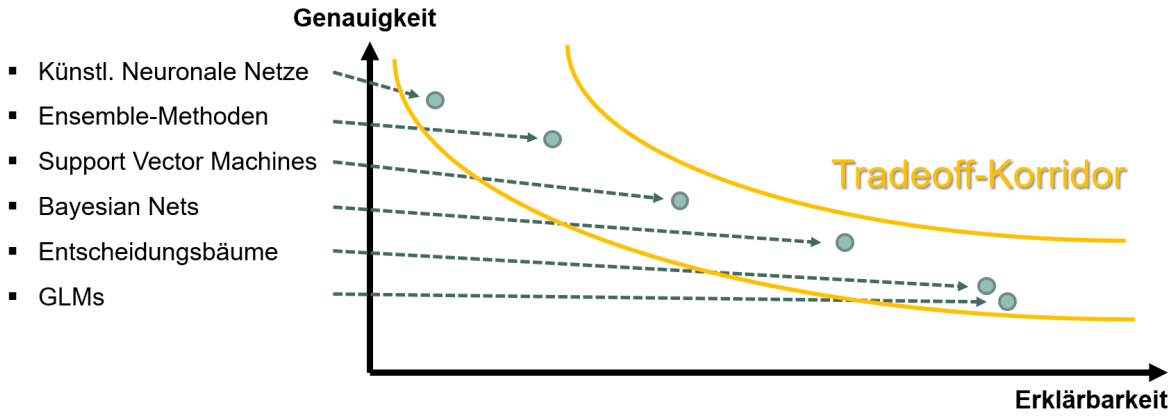
Die Modell-agnostischen Interpretationsverfahren LIME und SHAP sind kurz zu beschreiben, zu vergleichen und Anwendungsgebiete zu benennen.

Regulatorisch: Grundsätzlich gelten alle Anforderungen in Tariferstellung, Reservierung und Controlling wie ohne KI-Methoden auch. Dies stellt uns vor die Herausforderung, dass der Regulator die aus KI-Methoden gewonnenen Einblicke in gleicher Weise prüfen können muss, wie klassische statistische Verfahren. Die genaue Ausgestaltung lässt zwar einen Ermessensspielraum, aber prinzipiell muss für jedes Risiko / jeden Vertrag die Entscheidung basierend auf den zugrunde liegenden Risikomerkmalen nachvollziehbar sein. Zusätzlich ist die Stabilität sowohl bezogen auf die zeitliche Dimension als auch selten auftretende Merkmalskombinationen zu prüfen.

Beispiel: Lt. GDV gibt es in Bezug auf die Kfz-Tarifempfehlung ca. 10^{12} unterscheidbare Tarifzellen - also deutlich mehr als die versicherten $\approx 4 \cdot 10^7$ Fahrzeuge. Während diesem Problem in einem GLM mit theoretischen Annahmen (IID, Homoskedaszität der Fehlerquellen etc.) entgegen getreten wird, ist bei ML-Methoden oft nur eine Nachverdichtung, wie man sie aus dem Leben-Bereich kennt, möglich und muss mit entsprechenden (anschaulichen) Randwertuntersuchungen unterlegt werden. Insbesondere die (näherungsweise) Monotonie der Funktionalabhängigkeiten der Merkmale sind vor diesem Hintergrund relevant.

Operativ:

Es gibt drei Aspekte in Einklang zu bringen: Neben der ökonomischen Perspektive (also Budget für die eigentliche Modellerstellung), die hier keine Rolle spielen soll, ist abzuwägen zwischen einer möglichst guten Performance einerseits und einer guten Erklärbarkeit andererseits. Dass dies überhaupt im Widersprich steht bzw. stehen kann, rührt daher, dass komplexere Modelle mit mehr Freiheitsgraden tendenziell bessere Vorhersageergebnisse erlauben (natürlich gegen Overfitting abzusichern, es geht hier also nicht um Regularisierung) als "simplere" Modelle. Während also einerseits die Frage zu bewerten ist, ob das komplexere Modell genug "Wissenszuwachs" bietet, dass es sich lohnt, dafür auch mehr Arbeit damit zu haben, es erklären und insbesondere mit den Aspekten der Frage A-01 zu verknüpfen, ist andererseits relevant, dass unterschiedliche Methoden auch strukturell bedingt unterschiedlich viel Aufwand zur Erklärbarkeit erfordern, siehe Schaubild.



Zusammenfassend ist für jedes Problem individuell zu prüfen, welche Kombination das Beste Verhältnis liefert; für regulatorisch relevante Modelle gibt es dabei natürlich nur wenig Spielraum, was im Umkehrschluss noch immer dazu führt, dass gewisse Techniken für bestimmte Einsätze de facto nur mit unüberwindbarem Aufwand freigabefähig wären.

LIME

Die Idee bei **LIME** besteht darin, Ersatzmodelle auf Basis einzelner Beobachtungen zu erstellen. Dabei geht man so vor, dass zunächst das *komplexe, schwer interpretierbare* Ausgangsmodell bestimmt wird, das eine exzellente Performance bietet. Danach wird, um eine Entscheidung über eine andere Beobachtung zu erklären, werden die bekannten erklärenden Beobachtungen auf Basis des Abstands zur neuen Beobachtung gewichtet und ein L1-regularisiertes lineares Modell trainiert. Die Regressionskoeffizienten davon sollen zur Erklärung verwendet werden. (*Anmerkung*: Es können auch andere Modelle wie Entscheidungsbäume oder Lasso/Ridge verwendet werden, solange die Interpretierbarkeit gegenüber dem komplexen Modell verhältnismäßig hoch ist.)

LIME versucht dies zu erreichen, indem ein Datensatz aus permutierten Samples erstellt wird. Mathematisch beschreiben wir dies, indem wir den Ansatz machen, dass die Erklärung $\hat{E}$ sich berechnet als

$\hat{E} = \text{argmin}_{g \in G}\, L(f, g(\pi(x))) + \Omega(g)$, wobei L die Loss-Funktion (z.B. MSE) ist, die wir verwenden, um das Surrogatmodell $g(\pi(x))$ gegeben der Vorhersagen $f(x)$ zu optimieren und Ω ein Maß für die Komplexität des Surrogatmodells ist, im einfachsten Falle also bspw. die Anzahl der verwendeten Merkmale. Die Menge G beschreibt die Familie der möglichen *Erklärungen*, also allgemein gesagt alle Surrogatmodelle, die prinzipiell in Frage kommen. $\pi(x)$ indes beschreibt die Umgebung der Beobachtung, aus der Vorhersagen für das Surrogatmodell in Betracht gezogen werden sollen. Hierbei ist zu beachten, dass eine zu große Wahl bei starken Nicht-Linearitäten im Ausgangsmodell natürlich zu einer schlechten Genauigkeit des Surrogats führt, eine zu enge Wahl aber indes oftmals überhaupt keine Zusammenhänge erkennbar macht.

Zusammengefasst bedeutet dies, dass es die Frage nach der "richtigen" Umgebungsgröße ungefähr dual dazu ist, für ein GLM den richtigen Link, bzw. Interaktionen zu bestimmen. Deshalb ist in der Praxis trotz der allgemeinen Limitationen nicht selten ein globales Surrogat zu bevorzugen.

Algorithmus

- Auswahl der Beobachtung x, die erklärt werden soll.
- Perturbationssampling der Umgebung π um x.
- Samples basierend auf Abstand gewichten.
- Lokales Surrogatmodell basierend auf Perturbationssamples erstellen.

Typischerweise muss die maximale Anzahl an Merkmalen, die einbezogen werden sollen, als Hyperparameter gesetzt werden. Eine größere Anzahl verspricht dabei natürlich größere Genauigkeit, aber eine geringere Anzahl wiederum ist besser interpretierbar und schneller zu berechnen.

Anmerkung LIME-Modelle bilden eine echte Untermenge der mittels *Shapley-Values* möglichen Anschauungen. Dieses Konzept erläutern wir im Folgenden:

SHAP

Die Idee hinter der *Shapley Value Contribution* ist eine konsequente Fortführung von *LIME* und *Sensitivitätsanalysen*. *SVC* ergänzt die vorgenannten Methoden um die Möglichkeit, eine komplette Rekonstruktion der Marginalverteilungen aller Merkmale zu erstellen. Der Nachteil dabei liegt auf der Hand, *SVC* zu verwenden ist insbesondere bei großen Datensätzen *extrem* aufwendig.

Im Prinzip ist die Fragestellung hinter *SVC* einfach erklärt: Wir interessieren uns für die Frage, inwieweit jedes Merkmal die jeweiligen Vorhersagen beeinflusst. Eine für erfahrene Aktuare nützliche Perspektive ist es, *SVC* als das ML-Äquivalent einer vollständigen Tomographie der den multivariaten, stochastischen Zustandsraum aufspannenden Copula aufzufassen.

Dies ist analog zum linearen Ansatz recht anschaulich gerade die Marginalverteilung (bspw. bzgl. X) der Merkmale x_1 und x_2. Der resultierende Ausdruck ist natürlich kombinatorisch ziemlich komplex zu berechnen. Die *Shapley*-Kontributionen erfüllen folgende Eigenschaften, die nützlich für die Betrachtung der Gewichtung der Merkmale sind:

- **Symmetrie**: Die Kombination von zwei (oder mehr) Merkmalen muss symmetrisch in dem Sinne sein, dass gleiche Beiträge bezüglich aller möglichen Kombinationen auch gleiche Vorhersagen implizieren.
- **Effizienz**: Die Effizienzbedingung besagt, dass die Beiträge der Merkmale gerade der Differenz zwischen der Vorhersage von x und $E(X)$ betragen.
- **Additivität**: Angenommen, wir haben mehrere Prädiktoren trainiert (bspw. verschiedene Entscheidungsbäume). Dann sollte für die Kombination (die direkte Summe der Merkmale) gelten, dass auch die Vorhersagen und die Shapley-Values additiv sind.
- **Nullidentitäts-Eigenschaft**: Merkmale, die effektiv keinen Beitrag aufweisen, und zwar unabhängig davon, welches Marginal betrachtet wird, sollten *Shapley-Values* identisch Null haben.

Praktische Berechnung

Um die vollständigen Shapley-Koeffizienten für alle beobachtungen zu berechnen müssen alle Kombinationen von Werten bestimmt werden. Das ist natürlich ein großer Aufwand, weshalb es manchmal sinnvoll sein kann, zunächst ein Sub-Sampling durchzuführen. Eine konkrete, moderne Implementierung mittels `shap` erfolgt mittels Monte-Carlo-Sampling der Prädiktoren, um die Komplexität *etwas* niedriger zu halten. Die Lösung approximiert die "echten" Shapley-Koeffizienten dann zwar nur, ist aber in der Praxis ein guter Kompromiss. Die Berechnung erfolgt nach dem folgenden Ansatz:

- Für alle Merkmale j, wiederhole:
  - Für alle $m = 1, \dots, M$:
    - Zufälligen Vektor z aus X ziehen.
    - Zwei neue Instanzen konstruieren:
      - Mit Merkmal j: $x_{-j} = \left(z_1, \dots, z_{j-1}, x_j, z_{j+1}, \dots, z_p\right)$
      - Ohne Merkmal j: $x_{-j} = \left(z_1, \dots, z_{j-1}, z_j, z_{j+1}, \dots, z_p\right)$
    - Marginalkoeffizienten berechnen: $\phi_j^m = \hat{f}\left(x_{+j}\right) - \hat{f}\left(x_{-j}\right)$
  - Shapley-Koeffizienten für Merkmal j als Mittelwert der Monte-Carlo-Samples errechnen: $\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \phi_j^m$

Sampling-Ansätze

- Full (Grid-Search)
- Randomisiert
- Dichtebasiert
  - Beobachtungsabhängig
  - Definitionsbereich

Vor- und Nachteile

- Allem voran ist der verhältnismäßig hohe Aufwand zu nennen, um eine korrekte Approximation der Marginalverteilungen zu erhalten, insbesondere für eine große Anzahl Merkmale. Demgegenüber stehen aber eine Reihe von Vorteilen, die in dieser

Kombination kein anderes Modell beieten kann.

- Die Shapley-Koeffizienten erlauben eine differenzierte Untersuchung von Untermengen, Schnitten und einzelner Punkte. Sie umfassen damit die Funktionalität von *LIME*, ohne an die Linearitätsbedingungen dessen gebunden zu sein.
- Funktioniert mit prinzipiell jedem Modell, auch bei hoher Nicht-Linearität. Insbesondere durch die Axiomatische Monte-Carlo-Definition werden die strukturellen Eigenschaften garantiert. (Anmerkung: Das bedeutet auch, dass man mittels Shapley-Werten sehen kann, wenn ein Modell bezüglich bestimmter Features instabil ist - für die Umkehrung gibt es aber keine Garantie.)
- Fehlende Werte können durch "blind" gesamplete Zufallswerte einbezogen werden, dis erhöht aber natürlich die Varianz der Ergebnisse und Nullidentitäts-Eigenschaft in Frage. Umgekehrt kann man diese Eigenschaft aber verwenden, um die Güte von imputierten Merkmalen zu bewerten.
- Shapley-Koeffizienten entsprechen *nicht* einem linearen Surrogatmodell wie LIME. änderungen in Shapley-Werten sind relativ zueinander, aber die Relation zum vorhergesagten Wert ist unbekannt. (Dafür müssten nicht nur die Marginale rekonstruiert werden, sondern die zugrunde liegende Verteilung, was nicht möglich ist.)

Modellinterpretation auf Merkmalsebene und globaler Ebene

In diesem und den darauffolgenden Aufgabenteilen sind zwei vortrainierte Modelle zu verwenden, die mit der Aufgabenstellung zur Verfügung gestellt wurden. Importieren Sie in Python mittels joblib die Dateien lgb_pretrained_A.joblib sowie lgb_pretrained_B.joblib und in R die Dateien modelA.rds sowie modelB.rds. Gehen Sie bei der Verwendung davon aus, dass ein One-Hot-Encoding der kategorialen Merkmale Geschlecht, Familienstand, Arbeitszeitkategorie mit der Nomenklatur xi_j für das i-te Merkmal x und der j-ten Ausprägung vorgenommen wurde. Hinweis: Eine Verwendung der zuvor erstellten Topics ist nicht nötig und im vorgegebenen Binärmodell nicht vorgesehen.

In [37]:
```
lgb_externalA = load('lgb_pretrained_A.joblib')
lgb_externalB = load('lgb_pretrained_B.joblib')
```

In [38]:
```
lgb_externalA.feature_name_
```

Out[38]:
```
['Alter',
 'Kinder',
 'AndereHausstandsmitglieder',
 'LohnWoche',
 'Arbeitszeitwoche',
 'ArbeitstageWoche',
 'Erstreserve',
 'Geschlecht_0',
 'Geschlecht_1',
 'Geschlecht_2',
 'Familienstand_0',
 'Familienstand_1',
 'Familienstand_2',
 'Arbeitszeitkategorie_0',
 'Arbeitszeitkategorie_1']
```

In [39]:
```
X.columns
```

Out[39]:
```
Index(['Alter', 'Kinder', 'AndereHausstandsmitglieder', 'LohnWoche',
       'Arbeitszeitwoche', 'ArbeitstageWoche', 'Erstreserve', 'Geschlecht_F',
       'Geschlecht_M', 'Geschlecht_U', 'Familienstand_M', 'Familienstand_S',
       'Familienstand_U', 'Arbeitszeitkategorie_F', 'Arbeitszeitkategorie_P'],
      dtype='object')
```

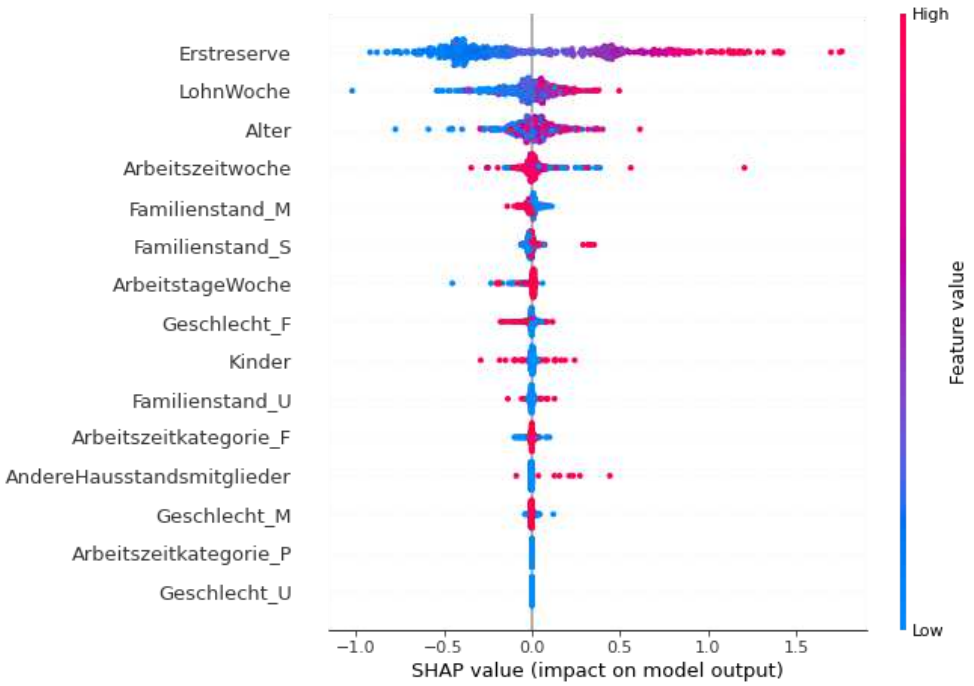
Aufgabe C07 [Lernziel 5.3.5, 5.3.6] [20 Punkte]

Es ist je eine Gesamtdarstellung ("summary plot") der wichtigsten erklärenden Merkmale mit SHAP für das in Aufgabenteil C01 erstellte Basis-Modell sowie die importierten Prognosemodelle zu erstellen. Die jeweils gezeigten Ergebnisse sind zu erläutern, Darstellungen sind zu vergleichen und ggf. Unterschiede zu begründen. Wählen Sie eine geeignete Größenordnung an zufällig ausgewählten Beobachtungen, um Berechnungsaufwand und Erklärbarkeit gegeneinander abzuwägen und begründen Sie Ihre Wahl. Es sind außerdem Partial Dependency Plots (PDP) (R:pdp, Python: pdpbox) kurz zu beschreiben, sowie für die Merkmale Alter, LohnWoche und Erstreserve zu erstellen und zu vergleichen. Anhand der Ergebnisse sind die Vor -und Nachteile von PDPs zu benennen und es ist zu begründen, ob dieses Verfahren für die vorliegenden Daten geeignet ist bzw. welches alternative Verfahren herangezogen werden sollte.

Prüfen Sie, ob das Basis-Modell oder ein importiertes Modell besser geeignet ist, den Zusammenhang der Zielgröße abzubilden. Falls eines der Modelle signifikant besser ist, begründen Sie mittels geeigneter Erklärbarkeitsansätze, wieso dies der Fall ist.

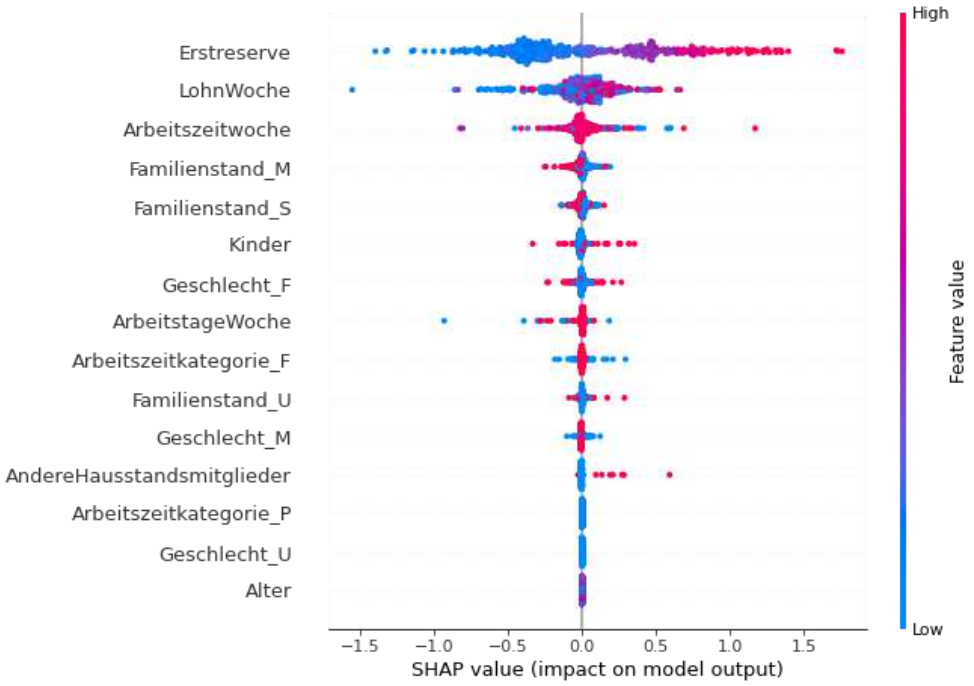
In [40]:
```
# Shap für Basismodell:
shap_basis_explainer = shap.TreeExplainer(lgb_basis,data=X[:500],check_additivity = False)
shap_values_basis = shap_basis_explainer.shap_values(X[:500])
```

In [41]:
```
shap.summary_plot(shap_values_basis, X[:500])
```



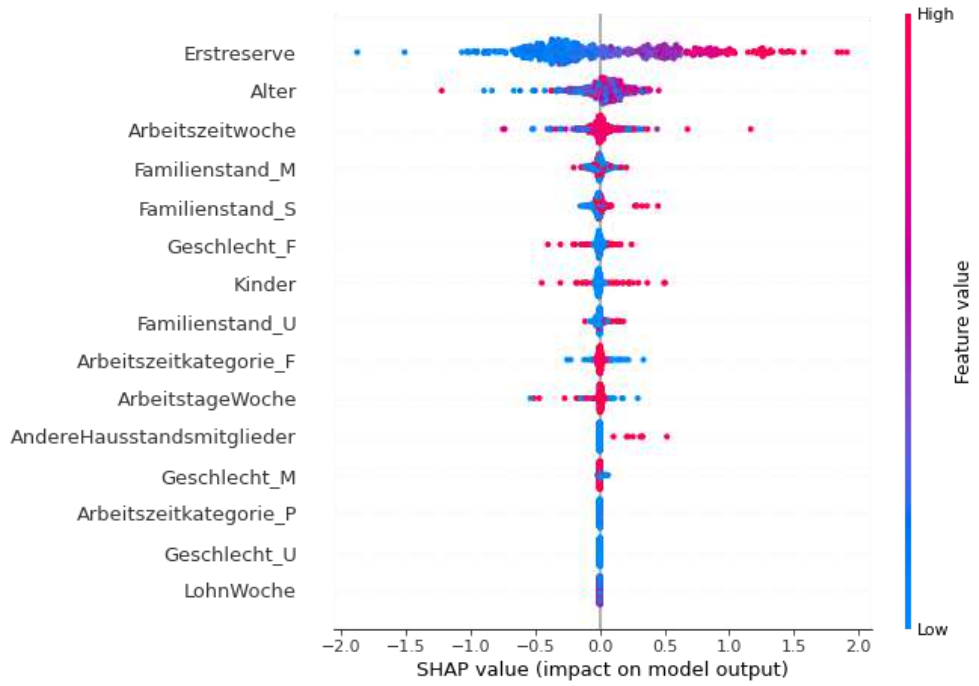
In [42]:
```
# Shap für externes Modell A:

A_shap_external_explainer = shap.TreeExplainer(lgb_externalA,data=X[:500],check_additivity = False)
A_shap_values_external = A_shap_external_explainer.shap_values(X[:500])
```

In [43]:
```
shap.summary_plot(A_shap_values_external, X[:500])
```



In [44]:
```
# Shap für externes Modell B:

B_shap_external_explainer = shap.TreeExplainer(lgb_externalB,data=X[:500],check_additivity = False)
B_shap_values_external = B_shap_external_explainer.shap_values(X[:500])
```

In [45]:
```
shap.summary_plot(B_shap_values_external, X[:500])
```



Partial Dependency Plots

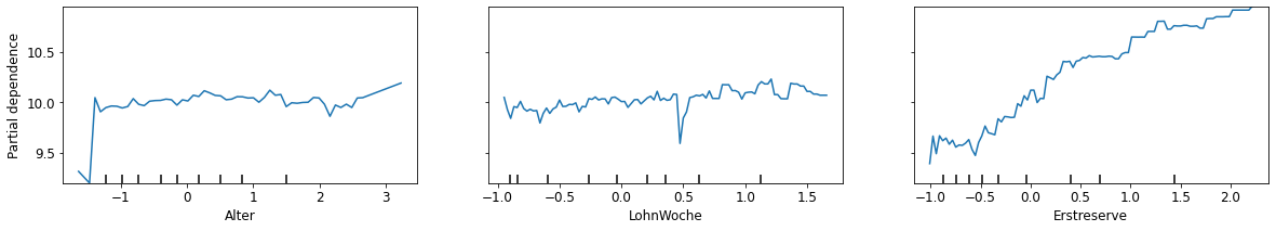
Ein Partial Dependency-Plot (PDP) zeigt die marginale Kontribution von ein bis zwei Merkmalen auf den vorhergesagten Output eines ML-Modells. Man kann daran ablesen, wie der funktionale Zusammenhang zwischen Regressor und Regressand bspw. linear, monoton, quadratisch etc. ist. In einem linearen Modell ohne Artefakte zeigen folglich die PDPs für alle Merkmale eine Linie. Im Falle von Klassifikationen sollte für jede Klasse ein spezifischer PDP erzeugt werden.

In [46]:
```
X.columns
# --> Alter ist index 0, LohnWoche ist index 3, Erstreserve ist index 6
```

Out[46]:
```
Index(['Alter', 'Kinder', 'AndereHausstandsmitglieder', 'LohnWoche',
       'Arbeitszeitwoche', 'ArbeitstageWoche', 'Erstreserve', 'Geschlecht_F',
       'Geschlecht_M', 'Geschlecht_U', 'Familienstand_M', 'Familienstand_S',
       'Familienstand_U', 'Arbeitszeitkategorie_F', 'Arbeitszeitkategorie_P'],
      dtype='object')
```

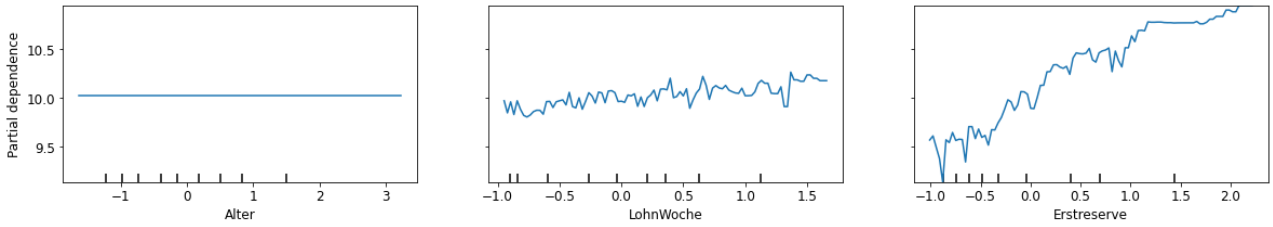
In [47]:
```
features = [0,3,6] # Angabe, für welche Merkmale PDPs erzeugt werden sollen.

plt.rcParams["figure.figsize"] = (20,3) # Größe der Plots auf eine brauchbare Dimension setzen.

display(plot_partial_dependence(lgb_basis,X[:1000], features))
```

```
<sklearn.inspection._plot.partial_dependence.PartialDependenceDisplay at 0x226b10740a0>
```



In [48]:
```
features = [0,3,6] # Angabe, für welche Merkmale PDPs erzeugt werden sollen.

plt.rcParams["figure.figsize"] = (20,3) # Größe der Plots auf eine brauchbare Dimension setzen.

display(plot_partial_dependence(lgb_externalA,X[:1000], features))
```

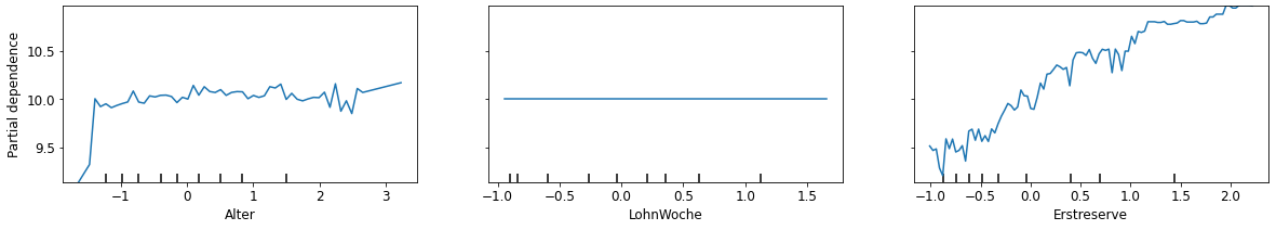
```
<sklearn.inspection._plot.partial_dependence.PartialDependenceDisplay at 0x226b6f26fa0>
```



In [49]:
```
features = [0,3,6] # Angabe, für welche Merkmale PDPs erzeugt werden sollen.

plt.rcParams["figure.figsize"] = (20,3) # Größe der Plots auf eine brauchbare Dimension setzen.
```

```
display(plot_partial_dependence(lgb_externalB,X[:1000], features))
```

```
<sklearn.inspection._plot.partial_dependence.PartialDependenceDisplay at 0x22699436430>
```



Man erkennt aus dem Partial Dependence Plot, dass im importierten Modell A das Merkmal `Alter` und im importierten Modell B das Merkmal `Lohnwoche` keinerlei Interaktion zeigt. Es ist also davon auszugehen, dass es keinen Einfluss auf die Vorhersage hat, bzw. nicht richtig abgebildet ist. Aufgrund dessen ist kein Modell klar vorzuziehen, wenngleich der Zusammenhang beim externen Modell etwas "glatter" erscheint.

Vorteile

- Leicht zu implementieren. Nicht nur gibt es geeignete Bibliotheken, man kann auch praktisch nichts falsch machen.
- Intuitive Interpretation als $x \mapsto f(x)$. Daraus ergibt sich, dass der marginale Anteil der Kontribution des Merkmals abgesehen von nicht-linearen Effekten direkt abgelesen werden kann.

Nachteile

- Unter Verwendung von Density-Plots können maximal zwei Merkmale in einem Graphen abgebildet werden. Wirkliche Nicht-Linearitäten sind daher faktisch nicht zugänglich.
- Einbezug der Merkmalsverteilung (vgl. Exposure) ist wichtig, insbesondere weil man oft Bereiche mit wenigen Werten nicht spezifisch gekennzeichnet sind. Eine Bestimmung von Konfidenzintervallen ist aufgrund der marginalen Natur problematisch.
- Annahme der (linearen) Unabhängigkeit: Partielle Abhängigkeiten sind nicht zu verwechseln mit den tatsächlichen Gesamtwirkungen aller Merkmale (etwa den auf Eigenräumen der PCA operierenden transformierten Merkmalen), die nur mit anderen Methoden zugänglich ist (z.B. Shap).

Spezifisch auf das vorliegende Modell bezogen ist anzumerken, dass aus den PDPs allein nicht klar wird, ob es starke Nichtlinearitäten gibt, weil diese nicht abgebildet werden können. Dies muss seperat geprüft werden, bspw. über einen Test mittels geeigneter Samplings der Shapley-Values, insbesondere auch außerhalb des Bereiches der vorgegebenen Beobachtungen.

Aufgabe C08 [Lernziel 5.3.5] [8 Punkte]

Es sollen die Einflussfaktoren der Prognose des in Aufgabe C-01 erstellten Modells mittels LIME erklärt werden. Für Einzeldarstellungen ist der Datensatz mit der ID WC5746143 zu verwenden.

1. Mit Standardwerten,
2. Mit stark verkleinerter Kernel-Breite,
3. Mit stark erhöhter Kernelbreite.

Die gewählten Parameter sind zu begründen und die Ergebnisse zu vergleichen und zu interpretieren.

Außerdem sollen die Einflussfaktoren der Prognose des in Aufgabe C-01 erstellten Basismodells mittels SHAP (force_plots, waterfall) erklärt werden. Die Ergebnisse für Datensatz WC5746143 sind mit denen vom LIME zu vergleichen.

Erstellen Sie zwei weitere Einzeldarstellungen für die Datensätze WC7248740 und WC5579080 mittels SHAP (force_plots, waterfall) und beschreiben Sie die Unterschiede zwischen den drei Datensätzen. Diskutieren Sie abschließend, ob LIME oder SHAP geeigneter für die Modellinter-pretation erscheint.

LIME

Standard-Kernel

In [50]:
```
# Original WC5746143
data_original.iloc[66,:]
```

Out[50]:

ID	WC5746143
Ereignisdatum	1996-09-18T08:00:00Z
Meldedatum	1996-10-05T00:00:00Z
Alter	35
Geschlecht	F

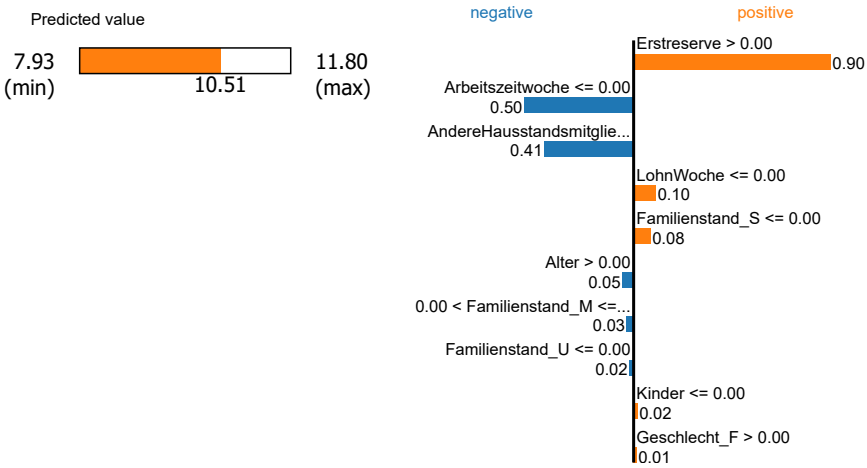
```
Familienstand                                                    M
Kinder                                                           0
AndereHausstandsmitglieder                                       0
LohnWoche                                                820.517047
Arbeitszeitkategorie                                             F
Arbeitszeitwoche                                              30.0
ArbeitstageWoche                                                 4
Schadenbeschreibung          PULLING TARPAULIN STRAINED LOWER BACK STRAIN
Erstreserve                                                16618.77954
Endkosten                                                  57611.688341
Name: 66, dtype: object
```

In [51]:
```
X.iloc[66,:]
```

Out[51]:
```
Alter                         0.095498
Kinder                       -0.230187
AndereHausstandsmitglieder   -0.090944
LohnWoche                    -0.048881
Arbeitszeitwoche             -0.615430
ArbeitstageWoche             -1.640500
Erstreserve                   0.501312
Geschlecht_F                  1.000000
Geschlecht_M                  0.000000
Geschlecht_U                  0.000000
Familienstand_M               1.000000
Familienstand_S               0.000000
Familienstand_U               0.000000
Arbeitszeitkategorie_F        1.000000
Arbeitszeitkategorie_P        0.000000
Name: 66, dtype: float64
```

In [52]:
```
#Lime mit Standardwerten

feature_names=X.columns
explainer_std = lime.lime_tabular.LimeTabularExplainer(X[feature_names].astype(int).values
                                                       , mode='regression'
                                                       , training_labels=y
                                                       , feature_names=feature_names
                                                       , verbose=True
                                                       , kernel_width=np.sqrt(len(X))*0.75)

predict_basis = lambda x: lgb_basis.predict(x).astype(float)
predict_external = lambda x: lgb_external.predict(x).astype(float)

exp = explainer_std.explain_instance(X.iloc[66], predict_basis, num_features=10)
exp.show_in_notebook(show_all=True)
```

```
Intercept 10.662401442627031
Prediction_local [10.76647097]
Right: 10.513732921372213
```

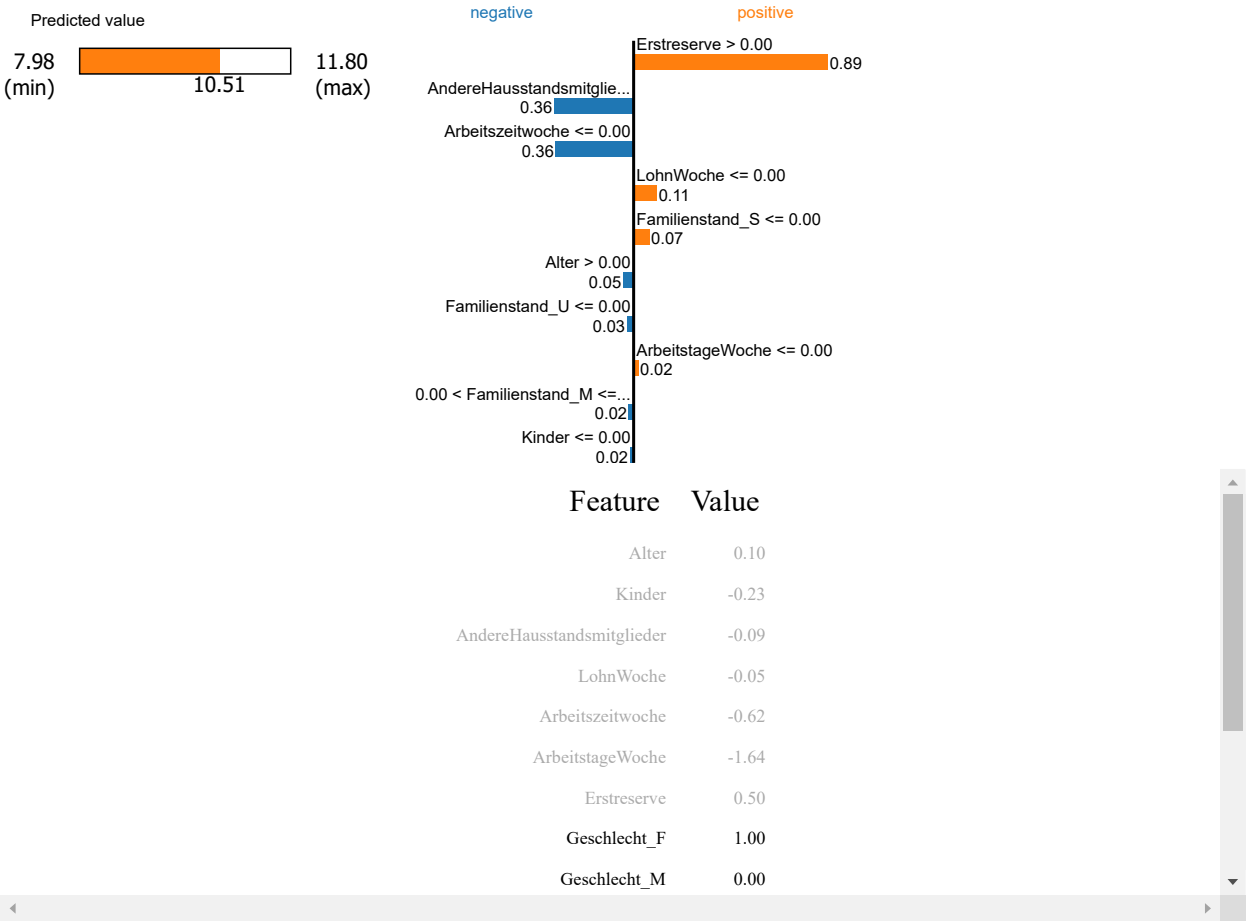


Feature	Value
Alter	0.10
Kinder	-0.23
AndereHausstandsmitglieder	-0.09
LohnWoche	-0.05
Arbeitszeitwoche	-0.62
ArbeitstageWoche	-1.64
Erstreserve	0.50
Geschlecht_F	1.00
Geschlecht_M	0.00

Stark vergrößerter Kernel

In [53]:
```
#Lime mit vergrößertem kernel

feature_names=X.columns
explainer_std = lime.lime_tabular.LimeTabularExplainer(X[feature_names].astype(int).values
                                                       , mode='regression'
                                                       , training_labels=y
                                                       , feature_names=feature_names
                                                       , verbose=True
                                                       , kernel_width=np.sqrt(len(X))*100) # Maximale Kernel-Breite:

exp = explainer_std.explain_instance(X.iloc[66], predict_basis, num_features=10)
exp.show_in_notebook(show_all=True)
```

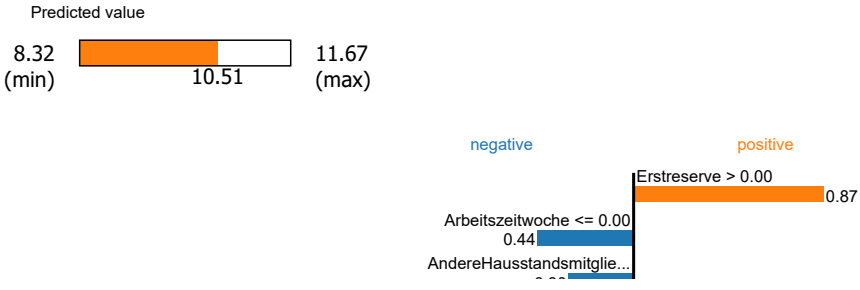
```
Intercept 10.49538918936502
Prediction_local [10.74546836]
Right: 10.513732921372213
```

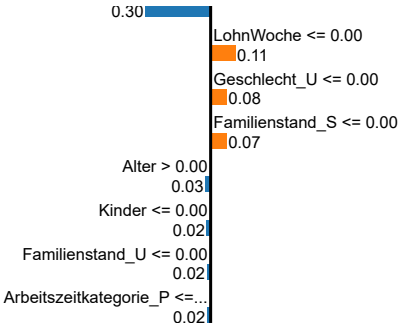


Stark verkleinerter Kernel

In [54]:
```
#Lime mit kleinem Kernel

feature_names=X.columns
explainer_small = lime.lime_tabular.LimeTabularExplainer(X[feature_names].astype(int).values
                                                       , mode='regression'
                                                       , training_labels=y
                                                       , feature_names=feature_names
                                                       , verbose=True
                                                       , kernel_width=np.sqrt(len(X))*0.01)

exp = explainer_small.explain_instance(X.iloc[66], predict_basis, num_features=10)
exp.show_in_notebook(show_all=True)
```

```
Intercept 10.447546096273909
Prediction_local [10.76404932]
Right: 10.513732921372213
```





Feature	Value
Alter	0.10
Kinder	-0.23
AndereHausstandsmitglieder	-0.09
LohnWoche	-0.05
Arbeitszeitwoche	-0.62
ArbeitstageWoche	-1.64
Erstreserve	0.50
Geschlecht_F	1.00
Geschlecht_M	0.00

Es ist zu erkennen, dass die Wahl des Kernels für die gewählten Beobachtung(en) keine bzw. sehr kleine Abweichungen aufweist. Dies deutet darauf hin, dass die Topologie des Zustandsraumes sehr glatt ist und die Merkmale keine bzw. sehr begrenzte Interaktionen aufweisen.

SHAP

In [55]:
```
shap.initjs()
```

In [56]:
```
# Shap für Basismodell:
shap_basis_explainer = shap.TreeExplainer(lgb_basis,data=X[:500],check_additivity = False)
shap_values_basis = shap_basis_explainer.shap_values(X[:500])
```

In [57]:
```
def shap_plot(j):
    p = shap.force_plot(shap_basis_explainer.expected_value, shap_values_basis[j,:], X.iloc[j,:], matplotlib=True)
    return(p)
```

In [58]:
```
def shap_waterfall(j):
    q = shap.plots._waterfall.waterfall_legacy(shap_basis_explainer.expected_value, shap_values_basis[j,:], X.iloc[j,
```

In [59]:
```
# Datensatz
X.iloc[66]
```

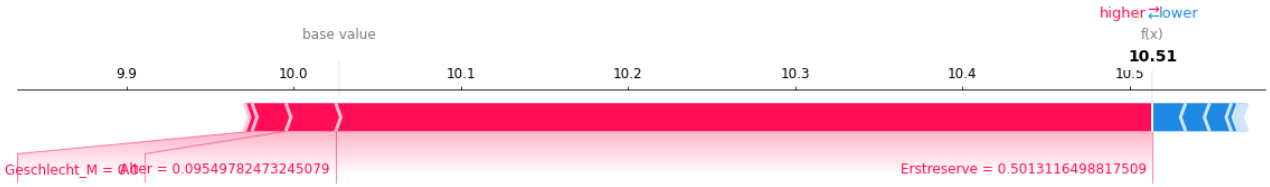
Out[59]:
```
Alter                          0.095498
Kinder                        -0.230187
AndereHausstandsmitglieder    -0.090944
LohnWoche                     -0.048881
Arbeitszeitwoche              -0.615430
ArbeitstageWoche              -1.640500
Erstreserve                    0.501312
Geschlecht_F                   1.000000
Geschlecht_M                   0.000000
Geschlecht_U                   0.000000
Familienstand_M                1.000000
Familienstand_S                0.000000
Familienstand_U                0.000000
Arbeitszeitkategorie_F         1.000000
Arbeitszeitkategorie_P         0.000000
Name: 66, dtype: float64
```

In [60]:
```
#Original WC5746143
data_original.iloc[66,:]
```

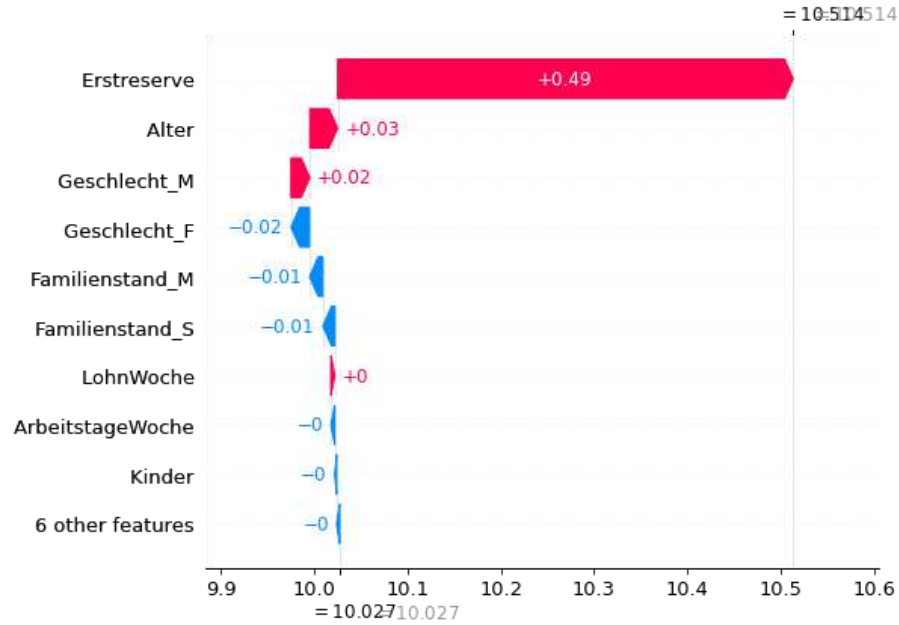
Out[60]:
```
ID                                                  WC5746143
Ereignisdatum                                1996-09-18T08:00:00Z
Meldedatum                                   1996-10-05T00:00:00Z
Alter                                                      35
Geschlecht                                                  F
Familienstand                                               M
Kinder                                                      0
```

```
AndereHausstandsmitglieder                                    0
LohnWoche                                                820.517047
Arbeitszeitkategorie                                          F
Arbeitszeitwoche                                           30.0
ArbeitstageWoche                                              4
Schadenbeschreibung          PULLING TARPAULIN STRAINED LOWER BACK STRAIN
Erstreserve                                              16618.77954
Endkosten                                                57611.688341
Name: 66, dtype: object
```

In [61]:
```
shap_plot(66)
```



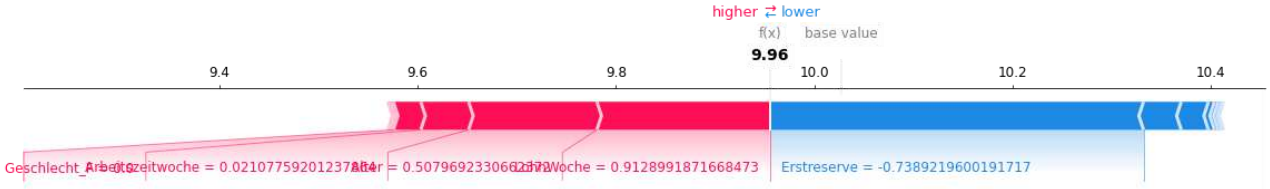
In [62]:
```
shap_waterfall(66)
```



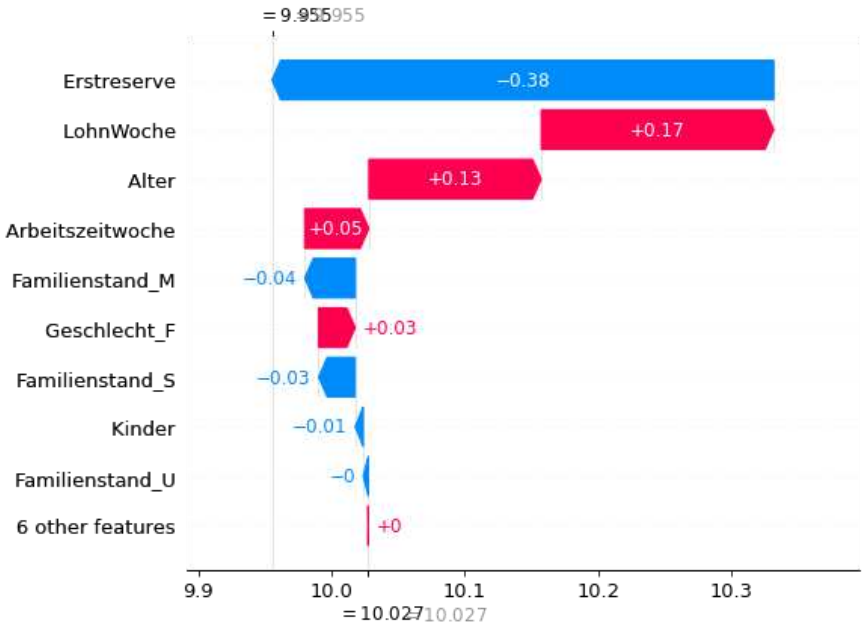
In [63]:
```
# WC7248740
data_original.iloc[97,:]
```

Out[63]:
```
ID                                                      WC7248740
Ereignisdatum                                2000-02-07T12:00:00Z
Meldedatum                                   2000-05-14T00:00:00Z
Alter                                                         40
Geschlecht                                                     M
Familienstand                                                  M
Kinder                                                         0
AndereHausstandsmitglieder                                     0
LohnWoche                                           1059.996837
Arbeitszeitkategorie                                           F
Arbeitszeitwoche                                            38.0
ArbeitstageWoche                                               5
Schadenbeschreibung          HIT PIECE OF STEEL FOREIGN BODY LEFT EYE
Erstreserve                                           8392.103107
Endkosten                                             88249.759333
Name: 97, dtype: object
```

In [64]:
```
shap_plot(97)
```



In [65]:
```
shap_waterfall(97)
```



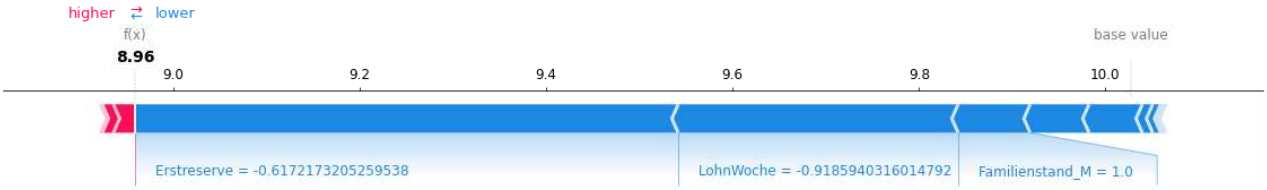
In [66]:
```
# WC5579080
data_original.iloc[40,:]
```

Out[66]:

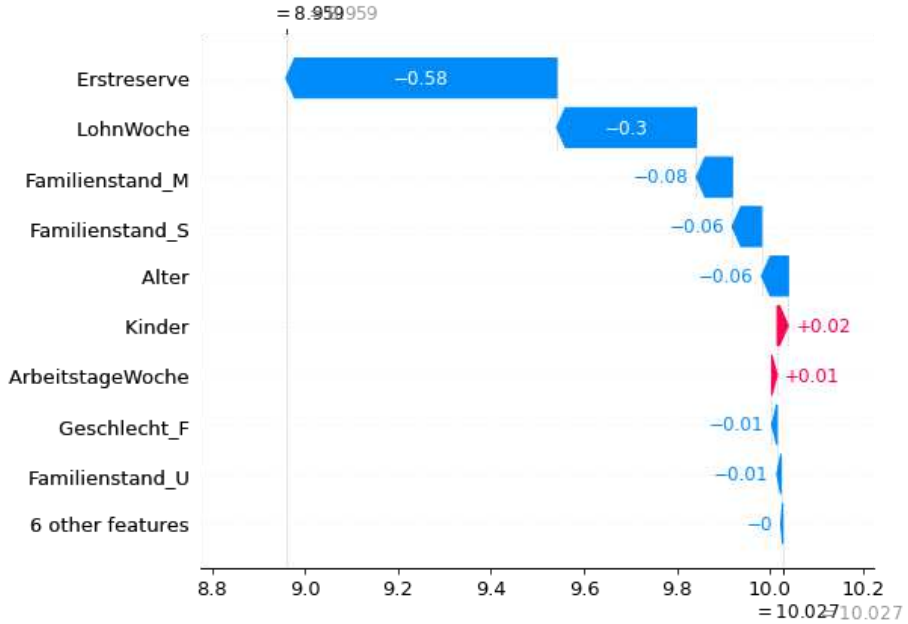
ID	WC5579080
Ereignisdatum	1996-02-09T05:00:00Z
Meldedatum	1996-05-13T00:00:00Z
Alter	38
Geschlecht	M
Familienstand	M
Kinder	0
AndereHausstandsmitglieder	0
LohnWoche	603.961848
Arbeitszeitkategorie	F
Arbeitszeitwoche	38.0
ArbeitstageWoche	5
Schadenbeschreibung	OVER REACHING BRUISED LEFT LEG
Erstreserve	8974.075644
Endkosten	0.0

Name: 40, dtype: object

In [67]:
```
shap_plot(40)
```



In [68]:
```
shap_waterfall(40)
```



Im direkten Vergleich der Modelle ist die große Stärke der Shapley-Values zu erkennen: Es sind differenziertere und prinzipiell granularere Einblicke in die genaue Funktionsweise der Schätzers möglich - dies geht aber auf Kosten der Berechnungszeit und ist im vorliegenden Beispiel daher nur auf einem Sub-Set effektiv anwendbar. Deshalb ist für die globale Vergleichbarkeit ein LIME-Ansatz mit sehr großer Kernel-Breite in Bezug auf die Gesamtaussage in der Kosten-Nutzen-Abwägung zu bevorzugen. Dabei ist allerdings zu bemerken, dass es dabei keine richtigere oder falsche Antwort gibt, sondern lediglich Abwägungen, die auf die zu betrachtenden Randbedingungen abgestellt sein müssen.