Methoden der statistischen Inferenz: Projektarbeit

Bernd Prostmaier (Matrikelnummer: 12004113)

11.08.2021

Contents

ABSTRACT

Die vorliegende Projektarbeit wendet die aus der Lehrveranstaltung *Methoden der statistischen Inferenz* erlernten Methoden auf reale Daten an. Hierfür wird der SQF Datensatz des 'Stop-question-and-frisk'-Programms des New York Police Department verwendet.

Der Hauptteil der Projektarbeit teilt sich in zwei Settings: Classification & Regression. Im Classification Setting wurden neben der logistischen Regression ebenso Classification Trees und Random Forest Modelle verwendet, um die Wahrscheinlichkeit einer Verhaftung, einer Durchsuchung sowie einer erhöhten Kontrolldauer (über 10 Min) zu schätzen. Hierbei war ersichtlich, dass es in keinem der Modelle Hinweise auf die statistische Signifikanz der ethnischen Herkunft gab. Der Vorabverdacht der Kontrolleure, die Bewaffnung des Kontrollierten, die Tageszeit sowie das Geschlecht erklärten im logistischen Regressionsmodell hauptsächlich die abhängige Variable (Verhaftung). Der Classification Tree lieferte Hinweise, dass sich sowohl der Vorabverdacht als auch der Wochentag signifikant auf die Kontrolldauer (als binäre Variable: über 10 Min & unter 10 Min) auswirken.

Durch die Anwendung der Ridge und Lasso Regression konnte zusätzlich herausgefunden werden, dass vor allem bei Terrorismus oder Mordverdacht die durchschnittliche Kontrolldauer extrem erhöht wird. Weiterhin werden z.B. Männer bei gleichen Bedingungen kürzer kontrolliert. Der zusätzliche Fit einer Polynomregression des Alters auf die Kontrolldauer lieferte keine Hinweise für Nichtlinearitäten.

Insgesamt konnte eine Vielzahl der erlernten statistischen Verfahren angewendet werden.

1 Einführung in die Projektarbeit

1.1 Stop-and-frisk New York City Datensatz

Das 'Stop-question-and-frisk'-Programm oder 'Stop-and-frisk' in New York City ist eine Praxis des New York City Police Department, bei der Zivilpersonen und Verdächtige auf der Straße vorübergehend festgehalten, befragt und manchmal auch nach Waffen und anderer Schmuggelware durchsucht werden. An anderen Orten in den Vereinigten Staaten ist dies als 'Terry Stop' bekannt.

Im Jahr 2016 wurden Berichten zufolge 12,404 Kontrollen im Rahmen des Stop-and-frisk-Programms durchgeführt. Das Programm wurde in der Vergangenheit in einem viel größeren Umfang durchgeführt. Zwischen 2003 und 2013 wurden jährlich über 100,000 Personen angehalten, auf dem Höhepunkt des Programms im Jahr 2011 waren es 685.724 Personen.

Aktualisierte Formen des Datensatzes können forlaufend unter **diesem Link** abgerufen werden.

1.2 Datenaufbereitung

Zum Zeitpunkt dieser Arbeit wurde der Datenzsatz aus dem Jahr 2020 verwendet. Dieser besteht aus 9,544 Observationen mit insgesamt 83 unterschiedlichen Variablen. Aus Gründen der besseren Nachvollziehbarkeit, wurde der Datensatz zunächst auf die nachfolgenden Variablen eingeschränkt:

Variable	Erklärung
MONTH2	Monat der Kontrolle
STOP_FRISK_TIME	Uhrzeit der Kontrolle
DAY2	Wochentag der Kontrolle
SUSPECTED_CRIME_DESCRIPTION	Verdächtigtes Verbrechen
OBSERVED_DURATION_MINUTES	Dauer der vorherigen Observation
STOP_DURATION_MINUTES	Dauer der Kontrolle
FRISKED_FLAG	Gefilzt bei der Kontrolle
WEAPON_FOUND_FLAG	Schusswaffe während Kontrolle gefunden
SUSPECT_REPORTED_AGE	Alter der kontrollierten Person
SUSPECT_SEX	Geschlecht der kontrollierten Person
SUSPECT_RACE_DESCRIPTION	Ethnische Herkunft der kontrollierten Person
SUSPECT_ARRESTED_FLAG	Fand eine Verhaftung statt

Weiterhin wurde der Datensatz um nachfolgende Variablen ergänzt

Variable	Erklärung
DAYNIGHT	Fand die Kontrolle tags- oder nachtsüber statt
DURATION	Erfolgte die Kontrolle länger oder kürzer als 10 Min

2 Deskriptive Analyse

Um ein Gespür für den Datensatz zu bekommen, erfolgen vor der Analyse durch die statistischen Methoden aus der Lehrveranstaltung verschiedene deskriptive Statistiken.

Wie bereits erwähnt enthält der Datensatz eine Reihe an Observationen bezüglich verschiedener Kontrollanlässe. Bild 1 gibt Auskunft darüber, unter welchem Motiv die jeweiligen Personen kontrolliert wurden.

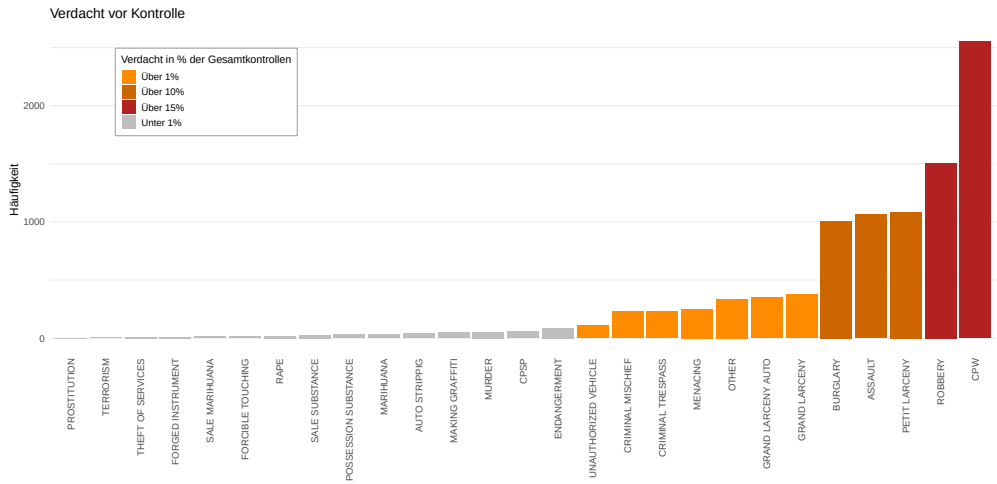

Figure 1: Verdacht bei Kontrolle

Es zeigt sich, dass der geringste Kontrollverdacht im Falle der Prostitution bzw. des Terrorismus besteht. Aus meiner Sicht verwunderlich ist, dass weniger als 1% der Kontrollen aufgrund eines Marihuana Verdachts stattfinden. Raubüberfälle und sog. "CPW" (Criminal Posession of a Weapon) stellen in über 15% der Fälle die meiste Ursache für eine Kontrolle dar.

Ein Blick auf die Untersuchungen der jeweiligen Wochentage zeigt, dass die meisten Kontrollen Mittwochs stattfinden und die wenigsten Kontrollen Montags durchgeführt werden (Bild 2).

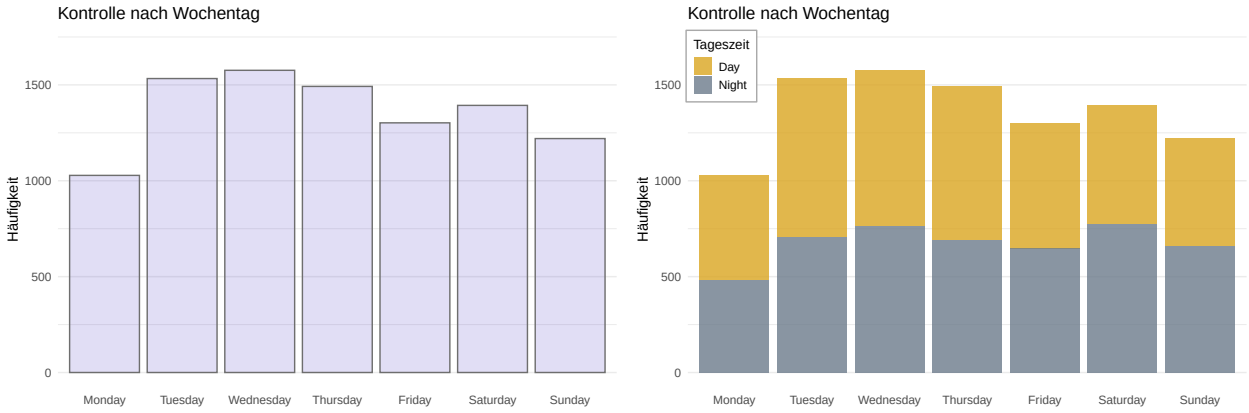

Figure 2: Kontrollen nach Wochentag und Tageszeit

Weiterhin ist in Bild 2 ersichtlich, dass in etwa die Hälfte der Kontrollen tagsüber und die Hälfte der Kontrollen in der Nacht durchgeführt werden. Hierbei lässt sich kein auffälliges Muster erkennen.

Bild 3 zeigt uns einen Boxplot bzgl. der Dauer der Kontrollen an den jeweiligen Wochentagen. Der Interquartilsabstand, welcher die zentralen 50% der Verteilung angibt, ist in etwa gleich groß über alle Wochentage hinweg. Die zentralen 50% der Beobachtungen bezüglich der Kontrolldauer betragen somit ca. 5-8 Min über alle Wochentage hinweg. Der Interquartilsabstand steigt lediglich leicht am Mittwoch. An diesem Tag finden ebenso, wie in Bild 2 ersichtlich, die meisten Kontrollen statt. Die oberen Whisker (1.5 x IQR) befinden sich ebenso für alle Wochentage auf einem ähnlichen Niveau. Man kann also nicht davon ausgehen, dass an einem gewissen Wochentag ungewöhnlich lange Kontrollen die Regel sind. Der Median ist

am Montag am höchsten, was bedeutet, dass die längsten 50% der Kontrollen Montags zu Stande kommen, wobei auch hier gesagt werden muss, dass der Median über alle Wochentage hinweg ebenfalls relativ konstant zu sein scheint.

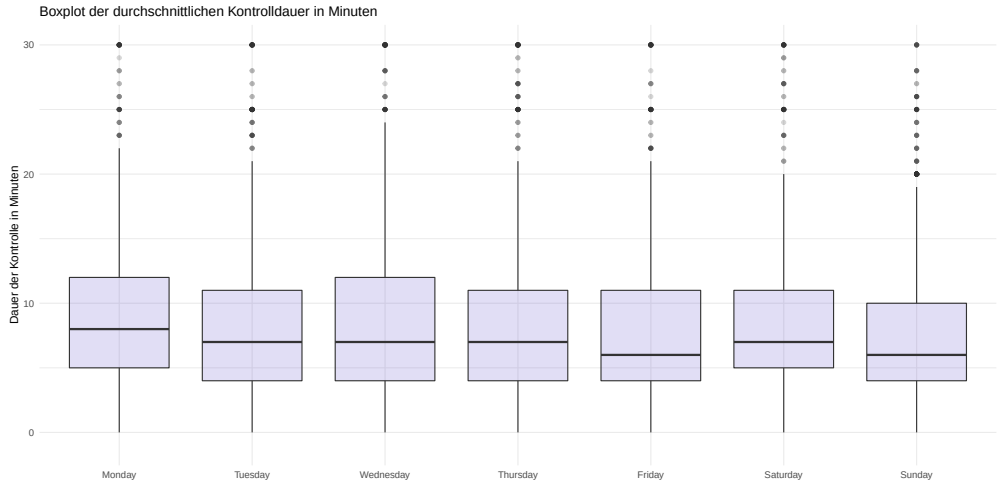

Figure 3: Boxplot Kontrolldauer nach Wochentagen

Bild 4 liefert Informationen über die durchschnittliche Kontrolldauer sowie das durchschnittliche Alter der kontrollierten Personen, gruppiert nach ihrer jeweiligen ethnischen Herkunft. Hierbei ist ersichtlich, dass vor allem jüngere Menschen mit 'black hispanic' bzw. 'black' Herkunft kontrolliert werden. Jedoch ist für die genannten Personengruppen die durchschnittliche Kontrolldauer am geringsten. Am Längsten werden Menschen der Gruppe 'Asian / Pacific Islander' kontrolliert.

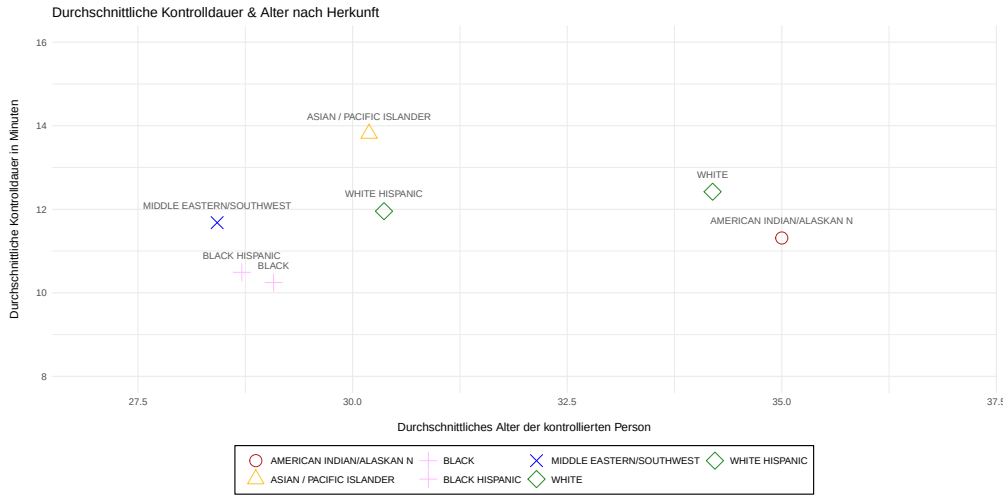

Figure 4: Kontrolldauer und Alter nach ethnischer Herkunft

Im Allgemeinen ergibt sich bzgl. der durchschnittlichen Kontrolldauer ebenfalls ein spannendes Bild. Bild 5 zeigt das Histogramm der Kontrolldauer.

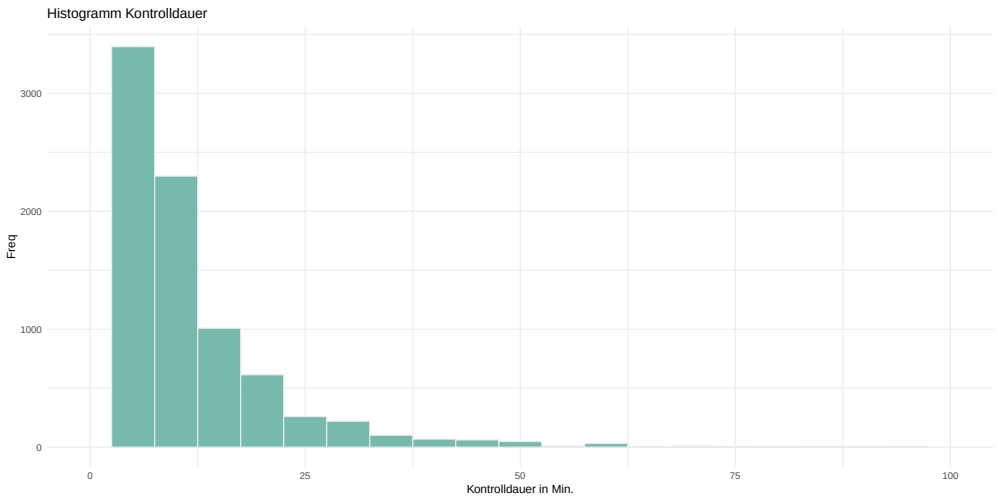

Figure 5: Histogramm Kontrolldauer

Die Achsen in Bild 5 wurden fixiert, da innerhalb der Variable extreme Ausreisser existieren. Der Mittelwert der Kontrolldauer beträgt 10.8 Minuten. Da der Maximalwert 845 Minuten und das 0.75 Quantil der Variable nur 13.0 Minuten beträgt, würde sich zur Beschreibung der Verteilung rund um das Zentrum der Variable der Median anbieten, um den Einfluss der Ausreisser zu minimieren. Der Median der Kontrolldauer beträgt 8.0 Minuten.

```
summary(SQF2$STOP_DURATION_MINUTES)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     5.0     8.0    10.8    13.0   845.0
```

3 Fortgeschrittene Statistische Methoden

Im Hauptkapitel dieser Arbeit sollen vor allem die Methoden aus der Lehrveranstaltung 'Methoden der statistischen Inferenz' angewendet werden, um möglichst spannende Erkentnisse aus dem Datensatz zu ziehen.

3.1 Classification

Nachfolgend wird versucht die Variable 'SUSPECT_ARRESTED_FLAG' mit Hilfe eines logistischen Regressionsmodells und die Variable 'DURATION' sowie 'FRISKED_FLAG' mit Hilfe eines Klassifikation-Baums zu erklären. Es wird also u.a. versucht zu erklären, wann eine Verhaftung stattfindet und wann eine Kontrolle länger oder kürzer als 10 Minuten andauert.

3.1.1 Logistic Regression

```
# Model Setup
model = SUSPECT_ARRESTED_FLAG ~ STOP_DURATION_MINUTES + WEAPON_FOUND_FLAG +
  SUSPECT_REPORTED_AGE + SUSPECT_SEX + SUSPECT_RACE_DESCRIPTION + DAYNIGHT

# Logit
logit = glm(model, data = SQF3,
            family = binomial(link = "probit"),
            na.action = na.exclude)

stargazer(logit, type = "text",
          keep.stat = c("n", "adj.rsq", "ll"))
```

```
##
## ================================================================================
##                                                    Dependent variable:
##                                                -----------------------------
##                                                   SUSPECT_ARRESTED_FLAG
## --------------------------------------------------------------------------------
## STOP_DURATION_MINUTES                                      -0.005***
##                                                             (0.001)
##
## WEAPON_FOUND_FLAGY                                         1.229***
##                                                             (0.045)
##
## SUSPECT_REPORTED_AGE                                        0.002**
##                                                             (0.001)
##
## SUSPECT_SEXMALE                                            -0.139***
##                                                             (0.049)
##
## SUSPECT_RACE_DESCRIPTIONASIAN / PACIFIC ISLANDER            -0.186
##                                                             (0.344)
##
## SUSPECT_RACE_DESCRIPTIONBLACK                               0.029
##                                                             (0.331)
##
## SUSPECT_RACE_DESCRIPTIONBLACK HISPANIC                      0.004
```

```
##                                                                 (0.334)
##
## SUSPECT_RACE_DESCRIPTIONMIDDLE EASTERN/SOUTHWEST                -0.257
##                                                                 (0.352)
##
## SUSPECT_RACE_DESCRIPTIONWHITE                                    0.157
##                                                                 (0.334)
##
## SUSPECT_RACE_DESCRIPTIONWHITE HISPANIC                           0.069
##                                                                 (0.332)
##
## DAYNIGHTNight                                                    0.012
##                                                                 (0.028)
##
## Constant                                                        -0.364
##                                                                 (0.336)
##
## ------------------------------------------------------------------------
## Observations                                                     8,749
## Log Likelihood                                                -5,420.261
## ========================================================================
## Note:                                          *p<0.1; **p<0.05; ***p<0.01
```

Da es sich bei der abhängigen Variable (SUSPECT_ARRESTED_FLAG) um eine binäre Variable handelt (Y = Es fand eine Verhaftung statt; N = Es fand keine Verhaftung statt), wird das Logit Modell verwendet, um eine binomiale logistische Regression für dichotome (binäre) abhängige Variablen durchzuführen. Die unabhängigen Variablen weißen ein beliebiges Skalenniveau auf. Diskrete Variablen mit mehr als zwei Ausprägungen werden hierbei in eine Serie binärer Dummy-Variablen zerlegt. Die Entscheidung fällt für das Logit Modell, weil die Einflüsse auf die diskrete abhängige Variable nicht mit dem Verfahren der klassischen Regressionsanalyse (lm) untersucht werden können, da dieses auf wesentlichen Anwendungsvoraussetzungen basiert, welche nicht gegeben sind (Homoskedastizität, Normalverteilung der Residuen). Weiterhin führt uns das lineare Wahrscheinlichkeitsmodell zu unzuverlässigen Vorhersagen, da es zu Vorhersagen außerhalb des 0 & 1 Intervalls kommen kann ($P(Y_i = 1) \notin [0, 1]$). Die logistische Regression löst dieses Problem durch die Transformation des Erwartungswerts der abhängigen Variable $P(Y_i = 1)$ (Bild 6). Die Verteilung des oben berechneten Logit Modells ist Bild 6 zu entnehmen. Bild 6 verdeutlicht den Vorteil der logistischen Regression durch die sigmuide Verteilungsfunktion, sodass stehts gilt: $P(Y_i = 1) \in [0, 1]$. Das (binomiale) logistische Regressionmodell lautet:

$$P(Y = 1|X = x_i) = P(Y_i = 1) = \frac{exp(x_i\beta)}{1 + exp(x_i\beta)} = \frac{1}{1 + exp(x_i\beta)}$$

Das Modell geht von der Idee der Odds, d.h. dem Verhältnis der Wahrscheinlichkeit ($P(Y_i = 1)$) zur Gegenwahrscheinlichkeit ($1 - P(Y_i = 1)$) aus.

$$Logit(Y) := ln(\frac{P(Y_i = 1)}{1 - P(Y_i = 1)})$$

Die Koeffizienten des Logit Modells lassen sich in der oben dargestellten Stargazer Tabelle entnehmen. Aufgrund des nichtlinearen Einflusses der erklärenden Variablen auf die Eintrittswahrscheinlichkeit π_i für Y_i werden die geschätzten Koeffizienten $\hat{\beta}$ nicht wie beim linearen Regressionsmodell als direkte Einflussfaktoren auf die Wahrscheinlichkeit $Y_i = 1$ interpretiert. Die Vorzeichen der geschätzten Betakoeffizienten geben Auskunft über die Richtung der Wirkung: Bei einem negativen Vorzeichen verringert sich die Wahrscheinlichkeit für das Eintreten von $Y_i = 1$ und bei positiven Werten der erklärenden Variablen ist der umgekehrte Effekt der Fall.

geschätzter Koeffizient $\hat{\beta}$	Odds $(\frac{P(Y=1)}{P(Y=0)})$	Wahrscheinlichkeit $P(Y=1)$
$\hat{\beta} > 0$	Steigen um $exp(\hat{\beta})$	Steigt
$\hat{\beta} < 0$	Sinken um $exp(\hat{\beta})$	Sinkt
$\hat{\beta} = 0$	Bleiben gleich	Bleibt gleich

Wir sehen also, dass bei Erhöhung der Kontrolldauer um eine Einheit (Minute), die Odds um $exp(-0.005)$ sinken, was auf eine Verringerung der Wahrscheinlichkeit einer Verhaftung führt. Wenig überraschend führt das Vorfinden einer Schusswaffe zu einer Erhöhung der Odds um $exp(1.229)$, was im Umkehrschluss logischerweise zu einer Erhöhung der Wahrscheinlichkeit verhaftet zu werden führt. Weiterhin hat das Geschlecht (weiblich) einen signifikant negativen Effent auf die Wahrscheinlichkeit und das Alter einen signifikant positiven Effekt. Die ethnische Herkunft scheint keinen statistisch signifikanten Einfluss, auf die Wahrscheinlichkeit verhaftet zu werden, zu haben.

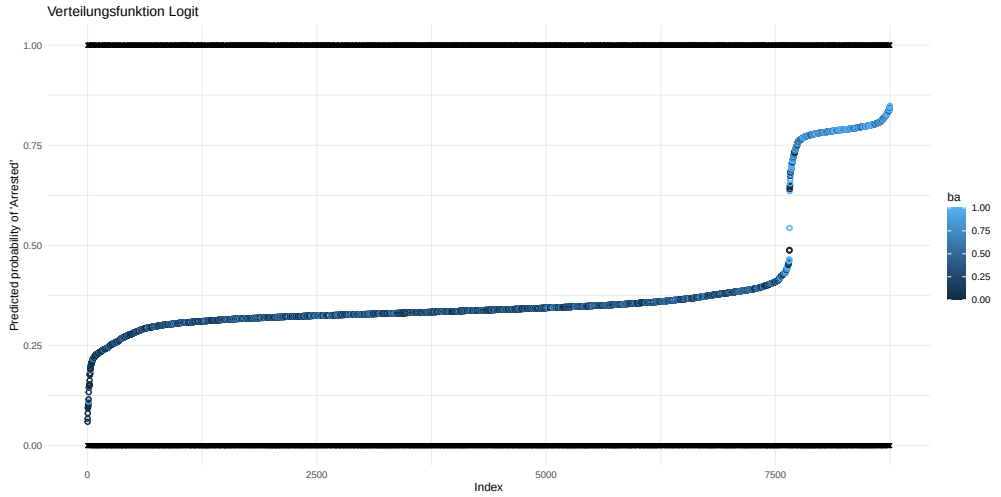

Figure 6: Logistische Verteilung des berechneten Logit Modells

Da die direkte Interpretation anhand der Odds zwar möglich, aber dennoch sehr umständlich ist, werden nachfolgend die marginalen Effekte berechnet um eine einfacherere Interpretation zu ermöglichen.

```
## AME - Durschschnittliche marginale Effekte
summary(margins(logit))
```

```
##                                       factor     AME     SE       z      p
##                                DAYNIGHTNight  0.0044 0.0100  0.4369 0.6622
##                          STOP_DURATION_MINUTES -0.0018 0.0004 -4.4058 0.0000
##  SUSPECT_RACE_DESCRIPTIONASIAN / PACIFIC ISLANDER -0.0625 0.1194 -0.5237 0.6005
##                     SUSPECT_RACE_DESCRIPTIONBLACK  0.0103 0.1156  0.0893 0.9288
##            SUSPECT_RACE_DESCRIPTIONBLACK HISPANIC  0.0013 0.1166  0.0111 0.9912
##  SUSPECT_RACE_DESCRIPTIONMIDDLE EASTERN/SOUTHWEST -0.0852 0.1215 -0.7015 0.4830
##                     SUSPECT_RACE_DESCRIPTIONWHITE  0.0560 0.1166  0.4805 0.6309
##            SUSPECT_RACE_DESCRIPTIONWHITE HISPANIC  0.0243 0.1159  0.2094 0.8342
##                               SUSPECT_REPORTED_AGE  0.0008 0.0004  2.0598 0.0394
##                                   SUSPECT_SEXMALE -0.0498 0.0176 -2.8233 0.0048
##                                 WEAPON_FOUND_FLAGY  0.4524 0.0135 33.4608 0.0000
##    lower   upper
##  -0.0152  0.0239
```

```
##  -0.0026 -0.0010
##  -0.2966  0.1715
##  -0.2162  0.2369
##  -0.2272  0.2298
##  -0.3234  0.1529
##  -0.1724  0.2844
##  -0.2028  0.2514
##   0.0000  0.0016
##  -0.0843 -0.0152
##   0.4259  0.4789
```

Nun sind die Effekte direkt zu interpretieren. Wenn beispielsweise die Kontrolle Nachts stattfindet, erhöht sich die Wahrscheinlichkeit verhaftet zu werden im Durchschnitt um 0.44 Prozentpunkte. Mit jeder Minute mit der die Kontrolldauer steigt, verringert sich die durchschnittliche Wahrscheinlichkeit verhaftet zu werden um -0.18 Prozentpunkte. Handelt es sich um eine Frau, sinkt die durchschnittliche Wahrscheinlichkeit einer Verhaftung um -0.498 Prozentpunkte.

Die marginalen Effekte bzw. diskreten Veränderungen in der Wahrscheinlichkeit sind Bild 7 zu entnehmen.

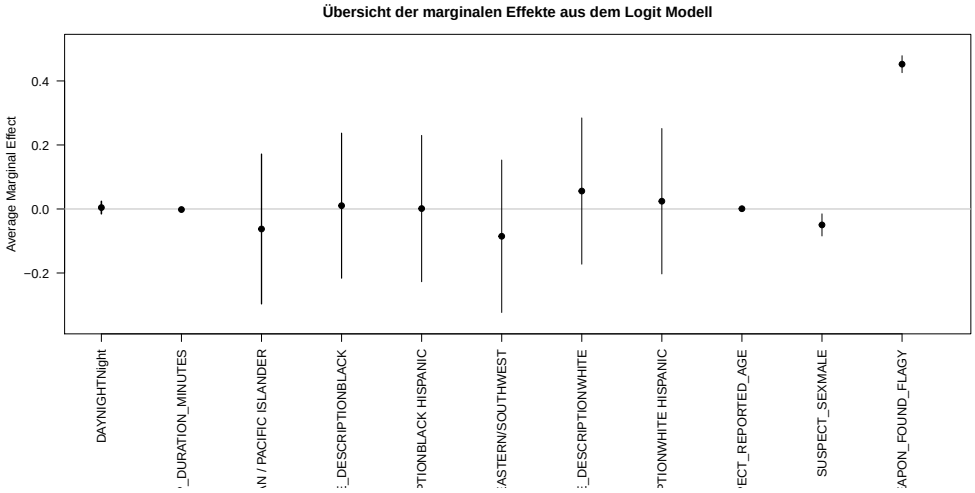

Figure 7: Übersicht der marginalen Effekte aus dem Logit Modell

Da wir gesehen haben, dass die Ausprägungen der Variable der ethnischen Herkunft nicht statistisch signifiant sind, könnten wir zur finalen Modellselektion, wie in der Vorlesung gesehen, unterschiedliche Gütekriterien heranziehen. Hierfür wird u.a. das Akaike-Informationskriterium herangezogen, welches der Idee von Ockhams Rasiermesser folgt und somit Modellkomplexität bestraft.

$$AIC = -2 * LL + 2 * K$$

Die Modellselektion findet hierbei anhand des Wertes der Log Likelihood (LL) statt, welcher umso größer ist, je besser das Modell die abhängige Variable erklärt. Neben der Log Likelihood wird allerdings die Anzahl der gerschätzten Parameter (K) als Strafterm mit berücksichtigt.

$$BIC = -2 * LL + log(N) * K$$

Zusätzlich zum AIC wird das Bayes Informations Kriterium (BIC) herangezogen, welches eine ähnliche Funktionsweise wie das AIC aufweist. Das BIC unterscheidet sich vom AIC, obwohl gezeigt werden kann, dass sie proportional zueinander sind. Im Gegensatz zum AIC bestraft das BIC das Modell stärker für seine

Komplexität, was bedeutet, dass komplexere Modelle eine noch größere (schlechtere) Punktzahl haben und damit weniger wahrscheinlich ausgewählt werden. Das Modell mit dem niedrigsten AIC/BIC wird ausgewählt.

```
# Unrestringiertes Modell
model1 = SUSPECT_ARRESTED_FLAG ~ STOP_DURATION_MINUTES + WEAPON_FOUND_FLAG +
  SUSPECT_REPORTED_AGE + SUSPECT_SEX + SUSPECT_RACE_DESCRIPTION + DAYNIGHT

# Restringiertes Modell ohne ethnische Herkunft
model2 = SUSPECT_ARRESTED_FLAG ~ STOP_DURATION_MINUTES + WEAPON_FOUND_FLAG +
  SUSPECT_REPORTED_AGE + SUSPECT_SEX + DAYNIGHT

# Logit Schätzungen
logit.a = glm(model1, data = SQF3,
              family = binomial(link = "logit"))

logit.b = glm(model2, data = SQF3,
              family = binomial(link = "logit"))
```

	AIC	BIC	ROC	Pseudo R2
Logit.a	10864.55	10949.47	0.6325916	0.0739411
Logit.b	10872.67	10915.13	0.6259581	0.0722225

Zusätzlich zum Akaike-Informationskriterium und Bayes Informationskriterium wird die ROC-Kurve herangezogen. Die vertikale Achse der ROC Kurve stellt die wahr-positiv Rate (Sensitivität) und die horizontale Achse die falsch-positiv Rate (Spezifizität) dar. Die ROC Kurve (grün) des unrestringierten Logit Modells ist Bild 8 zu entnehmen. Je weiter die ROC-Kurve von der Diagonalen (welche einer rein zufälligen Zuordnung durch Münzwurf entspricht) nach oben abweicht, desto 'besser' ist das Modell. Als Güte Maß dient die Fläche unter der Kurve (AUC - Area under the Curve). Diese Fläche liegt zwischen 0.5 (zufällige Zuordnung) und 1 (perfekte Zuordnung). Wir entscheiden uns für das Modell, für welches die Fläche unter der Kurve am höchsten ist. Zusätzlich sehen wir im Mc Fadden Pseudo R^2, dass sich die Log Likelihood verglichen mit dem Nullmodell im unrestringierten Modell stärker erhöht. Sowohl das AIC-Informationskriterium, ROC als auch Mc Faddens Pseudo R^2 bevorzugen das unrestringierte Modell!

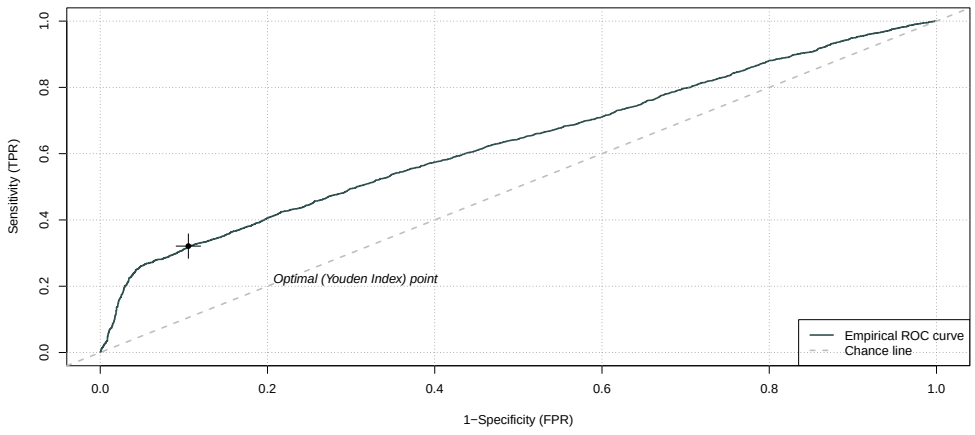

Figure 8: ROC Kurve

3.1.2 Classification Tree

Nachfolgend wird mit Hilfe des eines Classification Trees versucht zu erklären welche Variablen auf die selbst erstellte Variable DURATION einwirken (siehe Kapitel 1). D.h. wir wollen erkären, wann Kontrollen über 10 Minuten andauern und wann nicht.

Die Grundidee von Bäumen ist, den Predictorenraum in eine Anzahl von einfachen Unterräumen $R_1, R_2, ..., R_j$ zu segmentieren. Bei Regression Trees nutzt man für die Prediction üblicherweise den Mittelwert oder Mode des Trainingsdatensatzes in der Region, zu welcher die Observation gehört. Bei Classification Trees hingegen, wird jede Observation zur am öftesten vorkommenden Klasse klassifiziert ('majority vote').

Die Unterräume nennt man terminal nodes oder leaves. Die Knoten des Baumes, bei denen sich der Predictorenraum teilt, nennt man internal nodes oder branches. Trees werden immer upside down modelliert, sodass die leaves am Boden des Baumes sind.

Wie werden die Regionen R_j gebildet?
Man nutzt das sog. 'Recursive Binary Splitting'. Hierbei beginnt man an der Spitze des Baumes und splittet den Prediktorenraum sukzessive. Jeder Split enthält zwei neue Branches. Hierbei handelt es sich jedoch um einen 'greedy' Ansatz, da bei jedem Step der bestmögliche Split an genau diesem Step gewählt wird, anstatt in die Zukunft zu schauen und einen Split zu wählen der in einem zukünftigen Step zu einem besseren Tree führt. Während man im Regression Tree den Predictor X_j und den Cutpoint s so wählt, dass der Split des Preditkrorenraums in die Regionen R_1 und R_2

$$R_1(j,s) = X|X_j < s; R_2(j,s) = X|X_j \geq s$$

zur größtmöglichen Reduktion in den RSS

$$\sum_{i:x_i \in R_{1(j,s)}} (y_i - \hat{y}_{R1})^2 + \sum_{i:x_i \in R_{2(j,s)}} (y_i - \hat{y}_{R3})^2$$

führt, kann im Classification Setting natürlich kein RSS berechnet werden. Als Alternative für den RSS dient die Classification Error Rate. Da wir planen die Vorhersage für eine Variable zur am häufigsten vorkommenden Klasse der Region zuzuweisen, ist die Classification Error Rate einfach der Anteil der Trainingsobservationen in der Region, der nicht zu der 'most common class' gehört - also die Missspezifikation! Die Error Rate ist also nichts anderes als das Verhältnis der Fehler, die wir machen, wenn wir unsere Schätzung $\hat{f}(x)$ auf die Daten anwenden:

$$\frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

Als Alternative zur Missklassifizierungsrate und um noch sensitivere Ergebnisse zu erzielen, kann als Gütemaß ebenso der Gini index

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

oder der sog. Entropy

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk})$$

herangezogen werden, wobei $\hat{p}_{mk}$ den Anteil der Trainingsdaten in der $m$ten Region aus der $k$ten Klasse darstellt.

Cost complexity pruning
Man hört dann auf zu splitten, wenn beispielsweise keine Regionen mehr gebildet werden können oder eine vorher definierte Grenze erreicht ist. Das Problem bei dieser Vorgehensweise ist, dass das Modell zwar gut geeignet ist um Vorhersagen auf dem Trainingsdatenset zu machen, aber es zu einer schwachen Testset Performance führt, da es schnell zum Overfitting verleitet. Das kann passieren, da der Baum schnell zu komplex wird. Was wir also eigentlich wollen ist ein kleinerer Baum, mit weniger Splits (also weniger Regionen). Dies führt zu geringerer Varianz und besserer Interpretation auf Kosten eines kleinen Bias. Um dies zu erreichen kreiert man einen sehr großen Baum T_0 und schneidet diesen zurück (prune) um Subtrees zu erhalten. Hierfür verwenden wir das sog. 'cost complexity pruning'. Anstelle jeden möglichen subtree auszuwählen, wählen wir eine Sequenz von Trees, indexiert bei einem nicht negativen tuning Parameter Alpha.

Es gibt also für jeden Wert von α einen Subtree $T \subset T_0$, sodass

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

so klein wie möglich ist. |T| ist hierbei die Anzahl der Terminal nodes. Der Tuning Parameter Alpha kontrolliert den trade-off zwischen der Komplexität und den Fit zu den Daten. Wenn wir so beispielsweise $\alpha = 0$ wählen, wird der Subtree $T = T_0$ sein. Wenn α steigt, wird ein Preis gezahlt für einen Baum mit vielen Terminal Nodes. Somit tendiert die o.g. Gleichung zu einem kleineren Subtree, weil es das Ziel ist den Ausdruck zu minimieren. Wenn wir Alpha also von 0 aus steigen lassen, werden die Branches 'pruned'. Man kann also die Sequenz von Subtrees aus einer Funktion von Alpha ableiten. Für unterschiedliche Werte von Alpha kann man schließlich Cross Validation nutzen.

Modellimplementierung in R
Nachfolgend erfolgt die Schätzung eines Classification Trees für die abhängige Variable 'DURATION'. Als erklärende Variablen dienen alle im Datensatz SQF3 enthaltenen Variablen bis auf 'STOP_DURATION_MINUTES', da sich die abhängige Variable aus dieser zusammensetzt.

```
# Classification tree
tree.duration = tree(DURATION ~. -STOP_DURATION_MINUTES, SQF3)

# Plot Tree
plot(tree.duration)
text(tree.duration, pretty = 0, cex = 0.6)
```

Die Visualisierung des Trees zeigt, dass wir aus dem Modell 14 Terminal Nodes erhalten, in dem fünf Variablen zur Erklärung der 'DURATION' herangezogen werden: SUSPECTED_CRIME_DESCRIPTION, SUSPECT_RACE_DESCRIPTION, SUSPECT_REPORTED_AGE, OBSERVED_DURATION_MINUTES, DAY2 . Bild 9 zeigt den ungeprunten Tree.

Um die Performance des Classification Trees zu messen, müssen wir den Testerror anstatt des Trainingserrors berechnen. Daher splitten wir die Observationen im Nachfolgenden in ein Test- und ein Trainingsset. Der Tree wird anschließend auf die Trainingsdaten gefittet und die Performance wird anhand der Testdaten gemessen.

```
# Sample Trainingsobservations
set.seed(2)
train = sample(1:nrow(SQF3), nrow(SQF3)/2)  # Zufällige Auswahl
SQF3.test = SQF3[-train, ]                  # Testobservationen

# Response Variable
DURATION_TEST = SQF3$DURATION[-train]
```

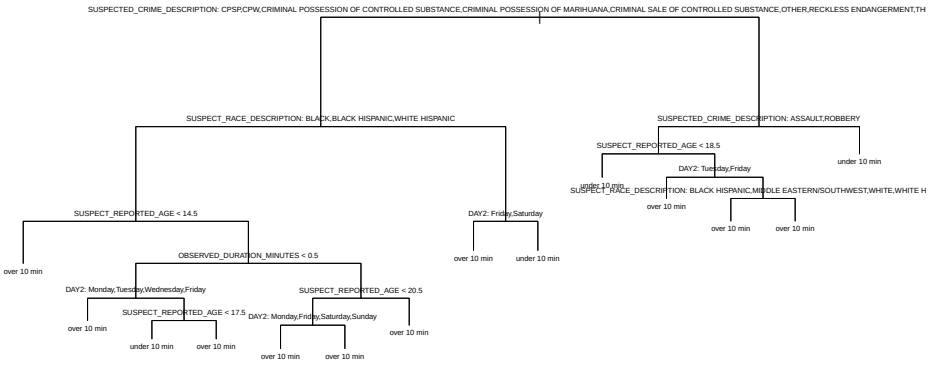

Figure 9: Classification Tree Ergebnisse

```
# Fitting Model auf Trainingsdaten
tree.duration = tree(DURATION ~. -STOP_DURATION_MINUTES, SQF3, subset = train)

# Performance auf Testdaten
tree.pred = predict(tree.duration, SQF3.test, type = "class")
```

Bild 10 zeigt das Ergebnis der Vorhersagen aus dem Classification Tree. Wir erhalten eine Misclassification Error rate von 33.4% mit dem Modell 'tree.duration' in den Testdaten.

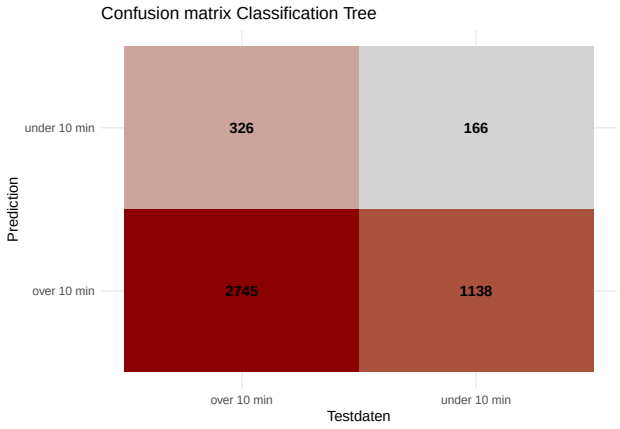

Figure 10: Confusion Matrix Classification Tree

Um bessere Ergebnisse zu erhalten und die Error Rate zu senken, wird im Nachfolgenden versucht den Tree zu prunen (d.h. zu schneiden). Hierfür nutze ich die Funktion *cv.tree()*, welche Cross Validation nutzt um das optimale Level des Baumes zu bestimmen. Das Funktionsargument *FUN = prune.misclass* wählt die Classification Error Rate als Gütemaß, anstatt der deviance.

```
# Prune
set.seed(7)
cv.duration = cv.tree(tree.duration, FUN = prune.misclass)
cv.duration$dev  # Classification Error
```

```
## [1] 32 32 29 29 31
```

```
cv.duration$size # Nodes
```

```
## [1] 14  6  4  3  1
```

Bild 11 zeigt die grafische Aufbereitung der Cross Validation Ergebnisse:

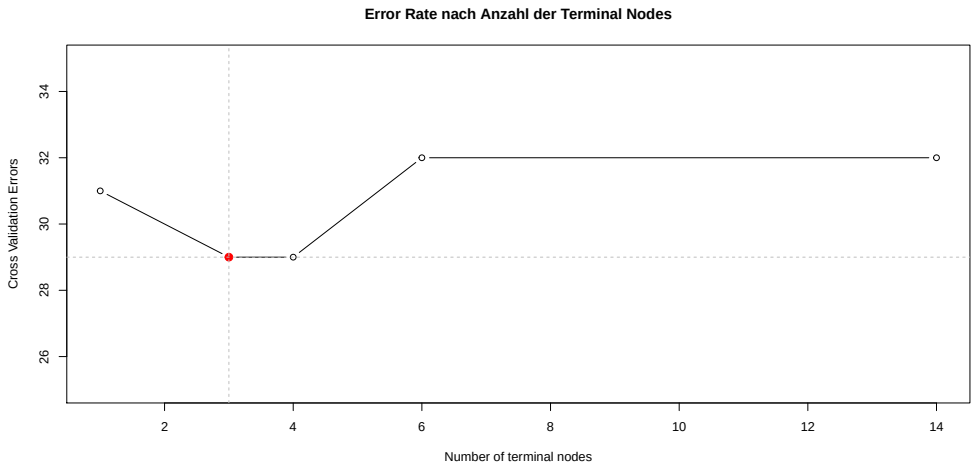

Figure 11: MSE nach Terminal Nodes

Es ist klar ersichtlich, dass der Baum mit 3 Terminal Nodes den kleinsten Cross Validation Error ergibt. Jetzt können wir die *prune.misclass()* Funktion verwenden, um den Baum zu schneiden und auf 3 Terminal Nodes zu regularisieren.

Bild 12 gibt einen Überblick über den geprunten Tree. Dieser ist bereits deutlich einfacher zu interpretieren als der Baum in Bild 9. Wir sehen also, dass der Verdacht vor der Kontrolle die wichtigste Variable ist, um die Dauer der Kontrolle vorherzusagen. Handelt es sich im Vorhinein um einen Verdacht im Bereich: CPW, Marihuana oder Mord, dauert die Kontrolle voraussichtlich über 10 Minuten. Handelt es sich nicht um einen solchen Vorverdacht, wird erneut unterschieden in den Wochentag. Sofern es sich um einen Donnerstag oder Freitag handelt, wird die Kontrolle voraussichtlich unter 10 Minuten dauern, ansonsten über 10 Minuten. Der Baum aus Bild 12 zeigt also nun 3 Regionen des Predictorenraums.

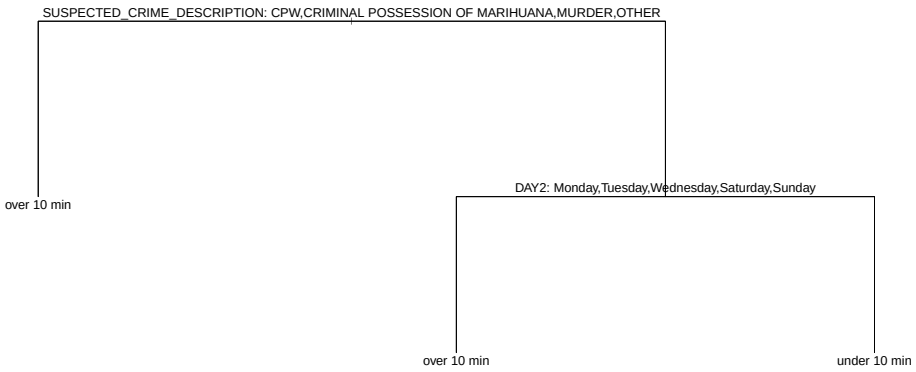

Figure 12: Pruned Tree: DURATION

Bild 13 zeigt erneut die Confusion Matrix des geprunten Trees. Wir sehen, dass die Error Rate leicht gesenkt wurde. Diese beträgt im Testset mit Hilfe des geprunten Trees 31.7% (ungepruned: 33.4%). Der geprunte Tree sorgt also nicht nur für ein besseres Testset Ergebnis, sondern er verbessert auch deutlich die Interpretation des Baumes, welches ebenso ein Hauptvorteil von Bäumen an sich darstellt. Im Gegensatz zum Logistischen Regressionsmodell sehen wir deutlich den Vorteil der Klassifikations Bäume, da wir die Ergebnisse und das Modell deutlich einfacher interpretieren können.

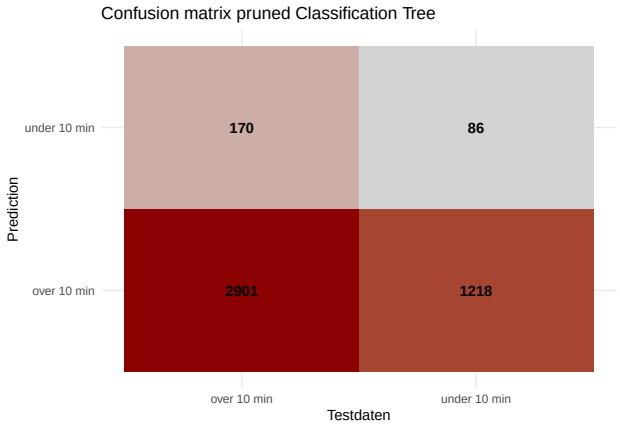

Figure 13: Confusion Matrix pruned Classification Tree

3.1.3 Classification Tree 2

Nachfolgend soll ein zweiter Classification Tree erklären, wann eine Durchsuchung der kontrollierten Person stattfand.

```
# Frisked - Tree
tree.frisked = tree(FRISKED_FLAG2 ~. , data = SQF4)
summary(tree.frisked)
```

```
##
## Classification tree:
## tree(formula = FRISKED_FLAG2 ~ ., data = SQF4)
## Variables actually used in tree construction:
## [1] "SUSPECTED_CRIME_DESCRIPTION"
## Number of terminal nodes:  3
## Residual mean deviance:  1.168 = 10220 / 8746
## Misclassification error rate: 0.3079 = 2694 / 8749
```

Vorangegangener Classification Tree nutzt 3 Terminal Nodes. Wie dem R Befehl zu entnehmen ist, wird ausschließelich eine Variable genutzt, um Vorherzusagen ob eine Filzung stattfindet oder nicht. Hierbei handelt es sich um die Variable SUSPECTED_CRIME_DESCRIPTION, d.h. das vorab vermutete Verbrechen entscheidet als alleiniges Kriterium im Modell über die Durchsuchung der Person. Dies führt bereits zu einer Genauigkeit von knapp 70%.

Entfernen wir die Variable 'SUSPECTED_CRIME_DESCRIPTION' aus dem Modell, verbleiben 3 Nodes bei den Variablen 'WEAPON_FOUND_FLAG' und 'SUPECT_SEX'. Aufgrund des bereits sehr kleinen Baumes wird auf ein prunen des Trees an dieser Stelle verzichtet.

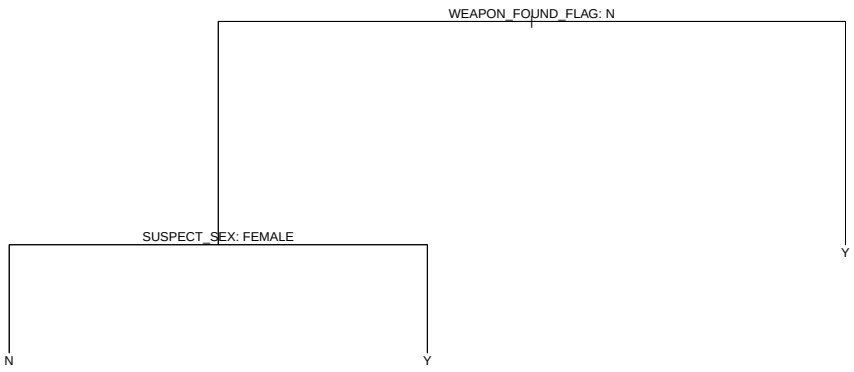

Figure 14: Classification Tree: FRISKED

Der Baum indiziert, dass sofern eine Waffe gefunden wurde unabhängig vom Geschlecht eine Durchsuchung stattgefunden hat (wenig verwunderlich). Sofern keine Waffe ersichtlich war, fand trotzdem in den meisten Fällen eine Durchsuchung statt, wenn es sich um eine männliche Person gehandelt hat.

3.1.4 Random Forest

Grundsätzlich leiden Entscheidungsbäume an einer hohen Varianz. Die Idee bei Random Forest Modellen (ähnlich wie beim Bagging) ist es, eine Anzahl an Entscheidungsbäumen auf bootstrapped Samples aufzubauen. Die Idee ist $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ Bäume mit Hilfe von B bootstrapped Trainingsdatensatze zu schätzen, welche anschließend gemittelt werden um ein Modell mit geringer Varianz zu erhalten:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

Da bei qualitativen Responses (wie im vorliegenden Fall) kein Durchschnitt genommen werden kann, wird, nachdem jede Klasse der B Bäume ermittelt wurde, das sog. 'majority vote' angewandt.

Das Problem von Bagging ist, dass in der Sammlung der bagged Bäume die meisten (oder alle Bäume) einen starken Predictor im oberen Split verwenden. Folglich sehen alle verpackten Bäume einander sehr ähnlich. Bagging würde also eine hohe Korrelation aller geschätzten Bäume aufweisen. Die Mittelung hoch korrelierter Größen, führt leider nicht zu einer signifikanten Reduktion der Varianz. Wie wir in den vorangegangenen Klassifikationsbäumen gesehen haben, ist die Variable 'SUSPECTED_CRIME_DESCRIPTION' ein starker Predictor, welcher vermutlich in allen bagged Bäumen im obersten Split enthalten wäre. Beim Bilden der Entscheidungsbäume in Random Forest Modellen gibt es daher einen elementaren Unterschied zum Bagging.

Bei der Erstellung der unterschiedlichen Entscheidungsbäume wird jedes Mal, wenn eine Aufteilung in einem Baum in Betracht gezogen wird, eine Zufallsstichprobe von m Predictoren als Aufteilungskandidaten aus dem vollständigen Satz von p Predictoren ausgewählt. Dem Split wird erlaubt nur einen der m Prediktoren zu nutzen. Mit anderen Worten: Bei der Erstellung eines Random Forest darf der Algorithmus bei jedem Split im Baum nicht die Mehrheit der verfügbaren Predictoren berücksichtigen, sondern nur ein Subset. Dies erlaubt, dass auch dann unterschiedliche Splits gewählt werden, wenn es einen oder mehrere sehr starke Prediktoren gibt! Somit sehen im Random Forest die Mehrheit aller Bäume unterschiedlich aus - was zu einer gewissen 'dekorrelation' führt!

Modellimplementierung in R
Nachfolgend wird erneut versucht die Variable 'DURATION' erneut mit einem Random Forest Modell zuschätzen. Das Testset Ergebnis wird anschließend mit dem einfachen Klassifikation-Baum verglichen.

```
# Sample Observations
set.seed(2)
train = sample(1:nrow(SQF4), nrow(SQF4)/2)

## Forest 1 - mtry = 10
rf = randomForest(DURATION ~. -STOP_DURATION_MINUTES,
                  mtry = 10,
                  SQF4, subset = train)

print(rf)
```

```
##
## Call:
##  randomForest(formula = DURATION ~ . - STOP_DURATION_MINUTES,      data = SQF4, mtry = 10, subset = t
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 29.61%
## Confusion matrix:
```

```
##              over 10 min under 10 min class.error
## over 10 min         2664          391   0.1279869
## under 10 min         904          415   0.6853677
```

Innerhalb des Random Forest Modells können wir nun festlegen wieviele Prediktoren an jedem Split verfügbar sein dürfen (m - mtry). Insgesamt wurden 500 Bäume geschätzt. Wie wir der Confusion Matrix des Random Forest Modells entnehmen können, erhalten wir eine 70.39%ige Genauigkeit im Testset bei der Schätzung nach der binären Kontrolldauer, wenn wir $m = 10$ setzen. Die Error Rate ist somit mit 29.61% nochmals niedriger als im Modell des geprunten Classification Trees (31.7%).

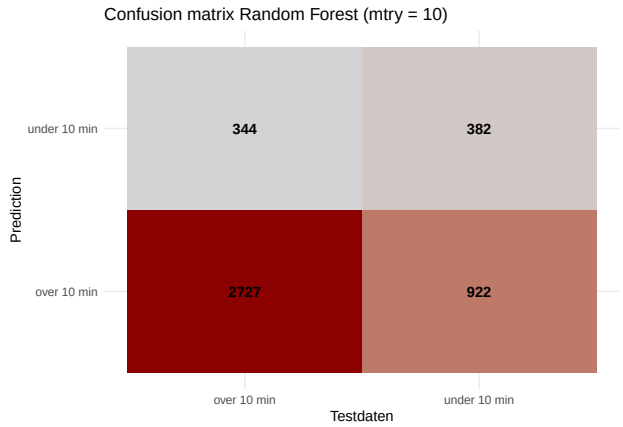

Figure 15: Confusion Matrix Random Forest

Nachfolgend wird ein zweites Random Forest Modell geschätzt. Setzen wir $mtry = 3$, berücksichtigen wir lediglich 3 Prediktoren an jedem Split im Tree. Hierbei zeigt Bild 15 dass wir die Error Rate nochmals deutlich senken. Im Modell 'rf2' beträgt die Error Rate somit nurnoch 28.12% und verbessert sich somit merklich gegenüber dem einfachen Klassifikationsbaum.

```
# Random Forest Modell 2 - mtry = 7
set.seed(2)
rf2 = randomForest(DURATION ~. -STOP_DURATION_MINUTES,
                   mtry = 3, ntree = 750,
                   SQF4, subset = train)
rf2
```

```
##
## Call:
##  randomForest(formula = DURATION ~ . - STOP_DURATION_MINUTES,      data = SQF4, mtry = 3, ntree = 75
##                Type of random forest: classification
##                      Number of trees: 750
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 28.12%
## Confusion matrix:
##              over 10 min under 10 min class.error
## over 10 min         2809          246  0.08052373
## under 10 min         984          335  0.74601971
```

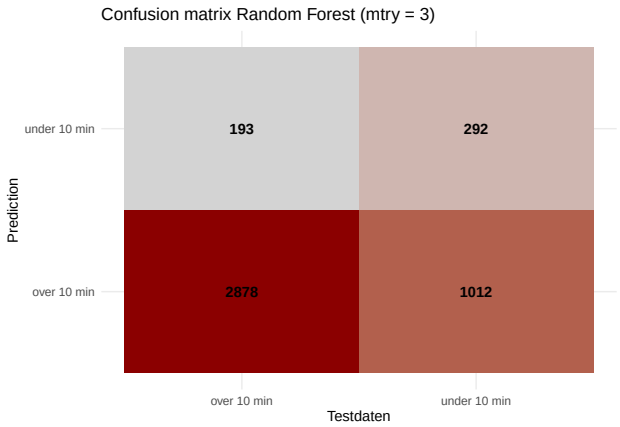

Figure 16: Confusion Matrix Random Forest

Vorteil der Random Forest Modelle ist, dass wir die Error Raten einfach ohne Cross Validation berechnen können, indem wir die sog. 'Out-of-Bag Error Estimation' anwenden. Beim Bootstrap verwenden wir ca. 2/3 der Observationen. Die restlichen 1/3 Observationen werden 'out of bag' (OOB) genannt. Wir können also Predictions nur für jene Observationen O_i berechnen, für die die Beobachtung i OOB war. Hierbei ist es einfach den Test Error zu berechnen, da der geschätzte Response nur auf Basis der Trees geschätzt wurde, die nicht für das Fitting verwendet wurden. Der Fit des Modells 'rf2' ist Bild 17 zu entnehmen.

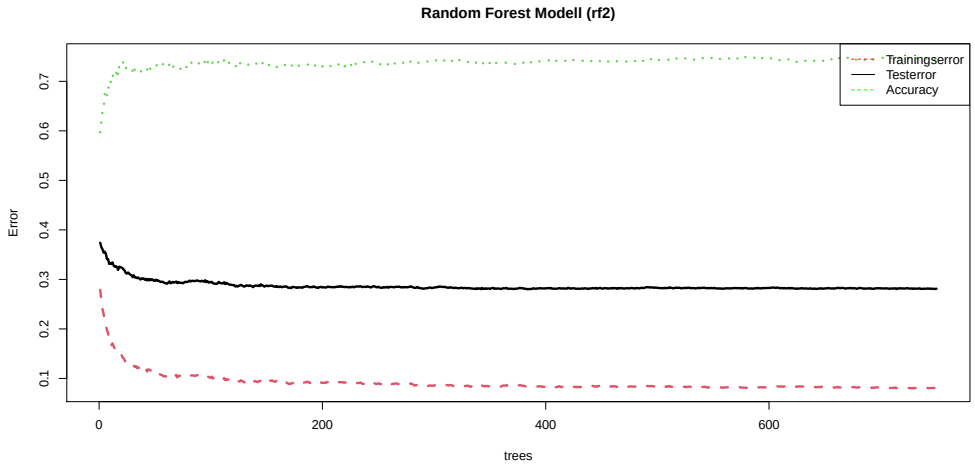

Figure 17: Fit Random Forest (rf2 Modell)

Nachteil der Random Forest Modelle ist jedoch, dass es schwerer ist das resultierende Modell zu interpretieren. Wenn wir eine große Anzahl von gebootstrappten Trees mitteln, ist es nicht länger möglich die Vorhersagen mit einem einzigen Baum zu erklären. Wir können nur eine 'overall summary' generieren, die uns die Wichtigkeit der einzelnen Prediktoren wiedergibt, indem wir auf den Gini Index (bei Classification) oder RSS (bei Regression) abstellen (siehe Bild 18).

Wir sehen in Bild 18, dass im Random Forest Modell das Alter, der Kontrollverdacht sowie der Wochentag ausschlaggebend für die Schätzung der Kontrolldauer sind. Dies passt ebenfalls zu den Erkentnissen der vorangegangenen Kapitel. Weiterhin scheinen die Observierungsdauer, sowie die ethische Herkunft ebenso eine Rolle in vielen Trees zu spielen.

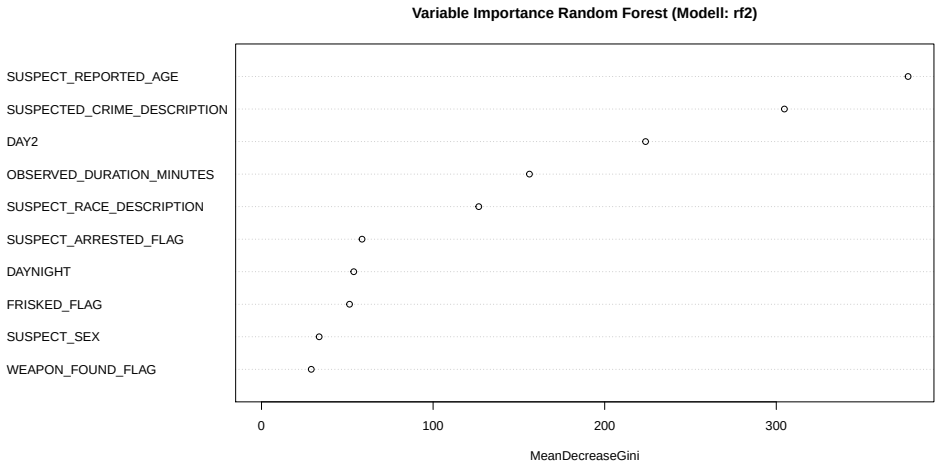

Figure 18: Variablenselektion Random Forest (rf2 Modell)

3.2 Linear Model Selection and Regularization

Nachdem in den vorherigen Kapiteln Klassifikation das vorherrschende Thema war, werden nun Regressionen durchgeführt. Hierbei wird versucht die exakte Kontrolldauer in Minuten zu erklären.

3.2.1 Subset selection

Bei der Subset Selection möchten wir das Modell finden, welches bestmöglich auf den Testdaten performt. Um Subset Selection zu machen und somit das Modell zu finden das den besten Fit und den höchsten Erklärungsgehalt hat, nutzen wir die Funktion *regsubsets()* (Teil des leaps packages). Der Befehl sucht das 'beste' Modell aus der jeweiligen Klasse (2 Predictoren, 3 Predictoren, etc.). 'Das beste Modell' ist in diesem Fall jenes Modell, welches den geringsten RSS in der jeweiligen Klasse hat!

Insgesamt existieren 2^p unterschiedliche Modelle! Wir fitten eine least squares Regression für jede mögliche Kombination der p-Predictoren! Wir fitten also alle Modelle mit exakt einem Predictor, dann mit allen möglichen 2er Kombinationen, 3er Kombinationen etc. Wir nehmen dann das Modell, welches in seiner Subset-klasse (also mit seiner Anzahl an Predictoren) am besten performt (kleinster RSS oder größtes R2)!

```
# Subset Selection
set.seed(2)
regfit.full = regsubsets(STOP_DURATION_MINUTES ~. -DURATION, data = SQF4,
                         nvmax = 10,
                         na.action = na.exclude)
reg.summary = summary(regfit.full)
reg.summary$rsq
```

```
##  [1] 0.01756412 0.03460263 0.05010660 0.06335506 0.07004286 0.07489115
##  [7] 0.07811522 0.07996911 0.08134096 0.08261491
```

Das Ergebnis des o.g. Befehls liefert uns Informationen über das R^2 für jedes Modell aus der besten Klasse. Mit steigender Anzahl der Predictoren steigt das R^2 natürlich automatisch, daher brauchen wir nun ein anderes Maß, um die finale Modellselektion zu betreiben. Hierfür dient uns z.B. das $Adj.R^2$, C_p oder BIC. Vorteil ist, dass an dieser Stelle nun nurnoch $(p+1)$ Modelle zur Auswahl sind.

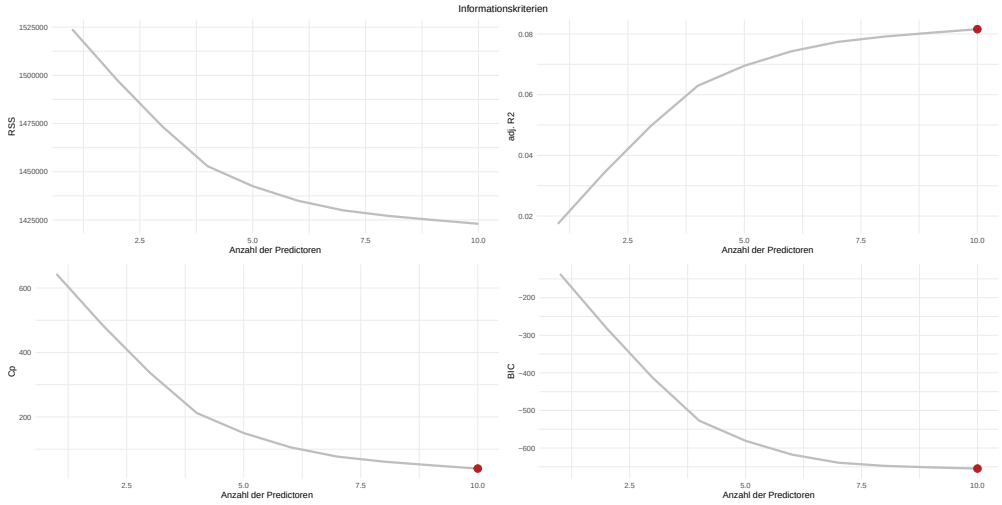

Figure 19: Vergleich unterschiedlicher Güßtemaße

Wir sehen, dass wir die besten Ergebnisse mit einem Modell, welches 10 Variablen inkludiert, erhalten. Die Built-in Plot Funktion liefert uns zusätzliche Informationen über welche Variablen es sich handelt (Bild 20).

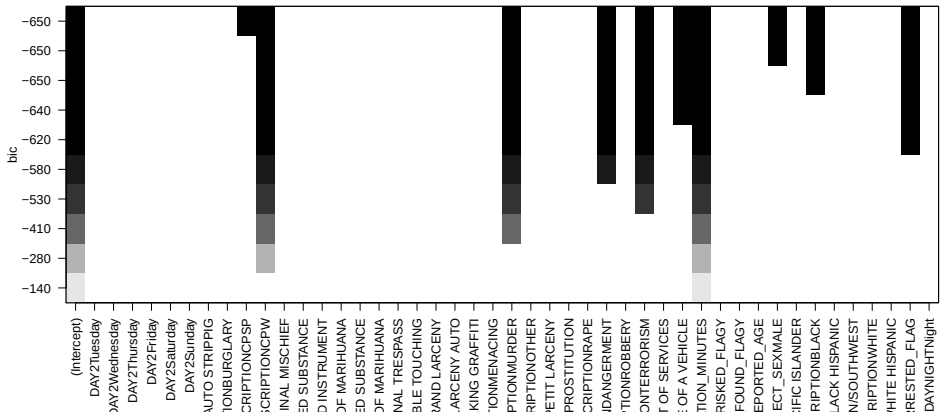

Figure 20: Variablenselektion Best Subset Selection

Die genauen Koeffizienten des Modells sind nachfolgendem R Befehl zu entnehmen. Hierfür wurden die Koeffizienten für das '10-Variablen Modell' ausgewählt:

```
coef(regfit.full, 10)
```

```
##                                  (Intercept)
##                                  14.35930439
##                  SUSPECTED_CRIME_DESCRIPTIONCPSP
##                                   6.02155942
##                   SUSPECTED_CRIME_DESCRIPTIONCPW
##                                  -3.48686807
##                 SUSPECTED_CRIME_DESCRIPTIONMURDER
##                                  22.29896436
```

```
##          SUSPECTED_CRIME_DESCRIPTIONRECKLESS ENDANGERMENT
##                                                11.81300095
##                       SUSPECTED_CRIME_DESCRIPTIONTERRORISM
##                                                63.32329598
## SUSPECTED_CRIME_DESCRIPTIONUNAUTHORIZED USE OF A VEHICLE
##                                                 6.83729758
##                                    OBSERVED_DURATION_MINUTES
##                                                 0.02589493
##                                           SUSPECT_SEXMALE
##                                                -1.75934189
##                                SUSPECT_RACE_DESCRIPTIONBLACK
##                                                -1.15588028
##                                     SUSPECT_ARRESTED_FLAG
##                                                -1.92070525
```

Wir sehen, dass wenn es sich um beispielsweise um den Verdacht des Mordes handelt, steigt die durchschnittliche Kontrolldauer bei sonst gleichbleibenden Bedingungen um 22.3 Minuten. Handelt es sich hingegen um den Verdacht des Terrorismus, steigt die Kontrolldauer durchschnittlich bei sonst gleichbleibenden Bedingungen um 63.32 Minuten. Interressant ist weiterhin, dass mit jeder Minute mit der die vorherige Observation steigt, die Kontrolldauer durchschnittlich um 0.03 Minuten steigt. Das männliche Geschlecht wird offensichtlich durchschnittlich (bei sonst gleichbleibenden Bedingungen) kürzer kontrolliert als Frauen. Ebenso werden Personen schwarzer Hautfarbe durchschnittlich kürzer kontrolliert (-1.16 Minuten).

3.2.2 Ridge

Als Alternative zur Subset Selection kann man Modelle fitten, die alle p Predictoren enthalten, indem wir eine Technik anwenden, die die Koeffizientenschätzungen regularisiert. Ridge & Lasso shrinken die Koeffizientenschätzer Richtung 0.

Ridge ist dabei sehr ähnlich zu OLS. Es schätzt die Koeffizienten durch Minimierung des nachfolgenden Ausdrucks:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

Wie in OLS, sucht die Ridge Regression zunächst Koeffizienten, die gut zu den Daten fitten indem die RSS so klein wie möglich gesetzt werden (erster Teil der Gleichung). Der zweite Teil, der sog. 'Penalty Term', ist klein, wenn β nahe 0 ist. So entsteht der Effekt des 'shrinkens'. Man nimmt also alle Betas, quadriert sie und bestraft um ihre Summe (l2 Norm!). Die Höhe der Bestraftung hängt von Lambda ab.

Der Tuning Parameter λ dient zur Kontrolle des relativen Einflusses auf die Koeffizientenschätzung. Wenn $\lambda = 0$ ist, hat der Penalty Term keinen Effekt und die Ridge Regression liefert die gleichen Ergebnisse wie der OLS Schätzer. Je größer Lambda ist, desto mehr Einfluss nimmt der shrinkage Penalty Term. Geht Lambda gegen unendlich, gehen die Koeffizientenschätzungen gegen 0. Damit die Interpetation der Ridge Regression als 'scale equivariant' zu verstehen ist, müssen die Daten zudem vorher standardisiert werden. Während OLS nur ein Set von Koeffizientenschätzer ergibt, produziert die Ridge Regression unterschliedliche Sets von Koeffizientenschätzungen für jeden Wert von λ. Optimale Werte für Lambda können über Cross Validation gefunden werden.

Beispiel im einfachen Linearen Modell: Wenn nur eine erklärende Variable die unabhängige Variable erklären soll, flacht die Regressionsgerade der Ridge Regression mit steigendem Lambda ab, weil der Koeffizient ja mit steigendem Lambda immer kleiner wird, somit wird die Steigung niedriger. Wenn Lambda gegen unendlich geht, gehen die Koeffizientenschätzer gegen 0. Im einfachen Fall würde das heißen, dass keine Steigung existiert und somit die erklärende Variable keinen Erklärungsgehalt besitzt! Das bedeutet, dass

sofern Lambda durch Cross Validation richtig gewählt wird, die Ridge Regression zu einer Erhöhung im Bias führt, aber gleichzeitig zu einer signifikanten Reduktion der Varianz im Testdatenfit führt.

Modellimplementierung in R

Nachfolgend wird erneut die Kontrolldauer geschätzt. Nachdem wir eine Modelmatrix erstellt haben und ein Y-Vektor definiert ist, kann mit Hilfe der *glmnet()* Funktion sowohl die Ridge als auch Lasso Regressionen durchgeführt werden. Das Funktionsargument 'alpha = 0' indiziert eine Ridge Regression. die *glmnet()* Funktion standardisiert die Daten bereits automatisch. Um den optimalen Wert für Lambda zu finden, nutzen wir Cross Validation.

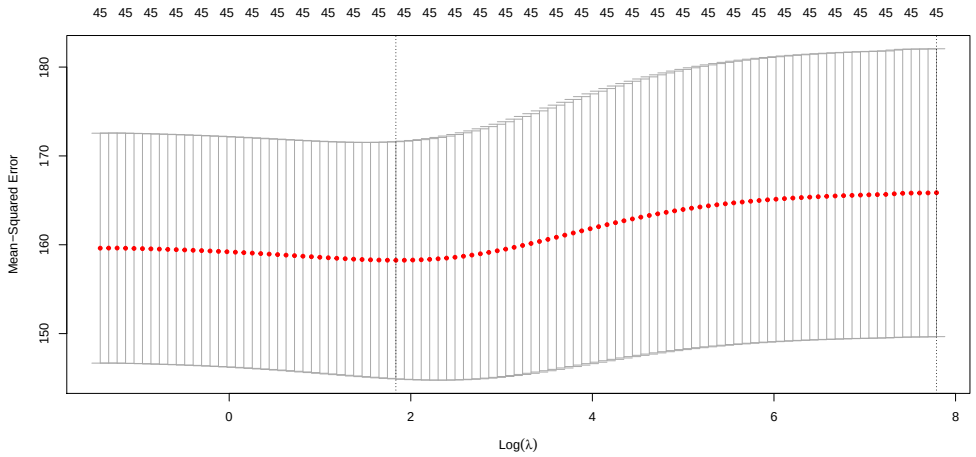

Figure 21: Ridge Regression Ergebnisse

Bild 21 zeigt den Cross Validation Error für das Modell bei unterschiedlichen Werten für λ. Nachfolgend möchten wir natürlich wissen, für welchen Wert von Lambda wir den geringsten CV Error erhalten:

```
bestlam = cv.out$lambda.min
log(bestlam)
```

```
## [1] 1.835839
```

Mit $log(\lambda) = 1.835839$ erhalten wir den geringsten Cross Validation Error. Die Frage ist nun, wie hoch der Test Error bei eben diesem Wert von Lambda ist:

```
ridge.mod = glmnet(x[train, ], y[train], alpha = 0)
ridge.pred = predict(ridge.mod, s = bestlam, newx = x[test, ])
mean((ridge.pred-y.test)^2)
```

```
## [1] 179.3977
```

Der geringste Test MSE beträgt also 179.3977, wenn $log(\lambda) = 1.835839$ ist. Nun können wir die Ridge Regression auf den kompletten Datensatz fitten, indem wir den Wert für Lambda aus der Cross Validation nutzen. Nachfolgender R-Befehl zeigt uns schließlich die Koeffizienten des Modells:

```
out = glmnet(x,y, alpha = 0)
predict(out, type = "coefficients", s = bestlam)
```

```
## 47 x 1 sparse Matrix of class "dgCMatrix"
##                                                                             1
## (Intercept)                                                        13.42396021
## (Intercept)                                                         .
## DAY2Tuesday                                                        -0.20502132
## DAY2Wednesday                                                      -0.02645332
## DAY2Thursday                                                       -0.24822558
## DAY2Friday                                                         -0.75063445
## DAY2Saturday                                                       -0.38141630
## DAY2Sunday                                                         -0.21862774
## SUSPECTED_CRIME_DESCRIPTIONAUTO STRIPPIG                            1.51435999
## SUSPECTED_CRIME_DESCRIPTIONBURGLARY                                 0.99727103
## SUSPECTED_CRIME_DESCRIPTIONCPSP                                     4.38701718
## SUSPECTED_CRIME_DESCRIPTIONCPW                                     -2.21811114
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL MISCHIEF                        0.19897431
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL POSSESSION OF CONTROLLED SUBSTANCE -1.65068152
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL POSSESSION OF FORGED INSTRUMENT     4.82306810
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL POSSESSION OF MARIHUANA            -2.46578676
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL SALE OF CONTROLLED SUBSTANCE       -1.52582081
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL SALE OF MARIHUANA                  -4.44055429
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL TRESPASS                           -0.44709345
## SUSPECTED_CRIME_DESCRIPTIONFORCIBLE TOUCHING                           -1.99403661
## SUSPECTED_CRIME_DESCRIPTIONGRAND LARCENY                            0.53727423
## SUSPECTED_CRIME_DESCRIPTIONGRAND LARCENY AUTO                       0.96688067
## SUSPECTED_CRIME_DESCRIPTIONMAKING GRAFFITI                         -0.58718223
## SUSPECTED_CRIME_DESCRIPTIONMENACING                                -0.15320195
## SUSPECTED_CRIME_DESCRIPTIONMURDER                                  15.50644678
## SUSPECTED_CRIME_DESCRIPTIONOTHER                                    1.04804230
## SUSPECTED_CRIME_DESCRIPTIONPETIT LARCENY                            0.43507604
## SUSPECTED_CRIME_DESCRIPTIONPROSTITUTION                             8.90505041
## SUSPECTED_CRIME_DESCRIPTIONRAPE                                     0.96510566
## SUSPECTED_CRIME_DESCRIPTIONRECKLESS ENDANGERMENT                    8.31495016
## SUSPECTED_CRIME_DESCRIPTIONROBBERY                                 -0.33154406
## SUSPECTED_CRIME_DESCRIPTIONTERRORISM                               43.75797435
## SUSPECTED_CRIME_DESCRIPTIONTHEFT OF SERVICES                       -1.80497184
## SUSPECTED_CRIME_DESCRIPTIONUNAUTHORIZED USE OF A VEHICLE            4.91084802
## OBSERVED_DURATION_MINUTES                                           0.01770569
## FRISKED_FLAGY                                                      -0.06466741
## WEAPON_FOUND_FLAGY                                                 -0.34902497
## SUSPECT_REPORTED_AGE                                               -0.01355711
## SUSPECT_SEXMALE                                                    -1.21954507
## SUSPECT_RACE_DESCRIPTIONASIAN / PACIFIC ISLANDER                    1.67125260
## SUSPECT_RACE_DESCRIPTIONBLACK                                      -0.56954955
## SUSPECT_RACE_DESCRIPTIONBLACK HISPANIC                             -0.33677417
## SUSPECT_RACE_DESCRIPTIONMIDDLE EASTERN/SOUTHWEST                    0.02142731
## SUSPECT_RACE_DESCRIPTIONWHITE                                       0.57098910
## SUSPECT_RACE_DESCRIPTIONWHITE HISPANIC                              0.46471128
## SUSPECT_ARRESTED_FLAG                                              -1.16331761
## DAYNIGHTNight                                                       0.08645670
```

Es ist ersichtlich, dass ebenso wie bei der Best Subset Selection die Dummy Ausprägungen für den Terrorismusverdacht sowie dem Verdacht des Mordes die absolut größte Auswirkung auf die durchschnittliche Kontrolldauer haben. Handelt es sich beispielsweise um einen Terrorismusverdacht, steigt bei sonst gleichbleibenden Bedingungen die durchschnittliche Kontrolldauer um 43.76 Minuten. Auch dieses Modell zeigt,

dass gegeben dass es sich um eine Person mit schwarzer Hautfarbe oder eine männliche Person handelt, die durchschnittliche Kontrolldauer kürzer ist. Wir erhalten also recht ähnliche Ergebnisse zur Best Subset Selection. Dennoch sehen wir einen deutlichen den Nachtil der Ridge Regression. Ridge ergibt ein Modell mit allen p Predictoren. Es werden stets alle erklärende Variablen in das Modell übernommen. Auch wenn Lambda erhöht wird, werden zwar Koeffizienten Richtung 0 gehen, aber wir werden Variablen nie komplett ausschließen können. Insgesamt erhalten wir hier zwar ein reguliertes, aber dennoch recht unübersichtliches Modell.

3.2.3 Lasso

Lasso schafft es im Gegensatz zu Ridge effektive Variablenselektion zu betreiben. Es ist hierbei möglich Koeffizientenschätzer direkt auf 0 zu setzen. Die l1 Norm shrinkt um den Absolutbetrag der β_j. Die Lasso Koeffizienten, β_λ^L, minimieren nachfolgenden Ausdruck:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1} p\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |\beta_j| = RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

Wenn der Tuning Parameter λ ausreichend groß ist, schafft es Lasso die Koeffizientenschätzer exakt auf 0 zu setzen. Es findet also Variablenselektion statt. Das Ergebnis der Lasso Regression sind sog. 'sparse models'.

Modellimplementierung in R
Um ein Lasso Modell zu schätzen, wird erneut die *glmnet()* Funktion verwendet, diesmal mit dem Funktionsparameter *alpha = 1.*

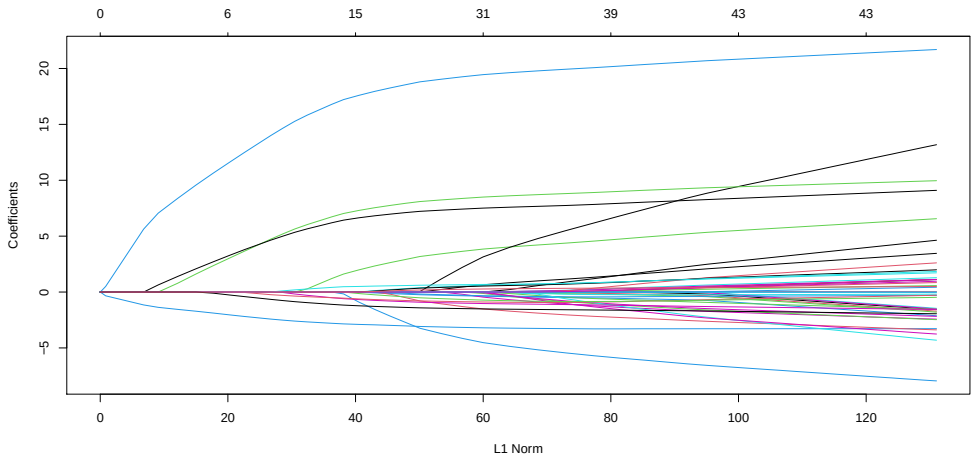

Figure 22: Koeffizienten Shrinkage Lasso

Bild 22 zeigt, bei welchem Wert von Lambda die Variablen geshrinkt werden. Abhängig von Lambda ist klar ersichtlich, dass einige Koeffizienten in dem geschätzten Modell der Kontrolldauer exakt auf 0 gesetzt werden.

Nachfolgend soll erneut durch Cross Validation ein optimaler Lambda Wert gefunden werden (gemessen am Cross Validation Test error).

Bild 23 zeigt erneut den MSE abhängig von $log(\lambda)$.

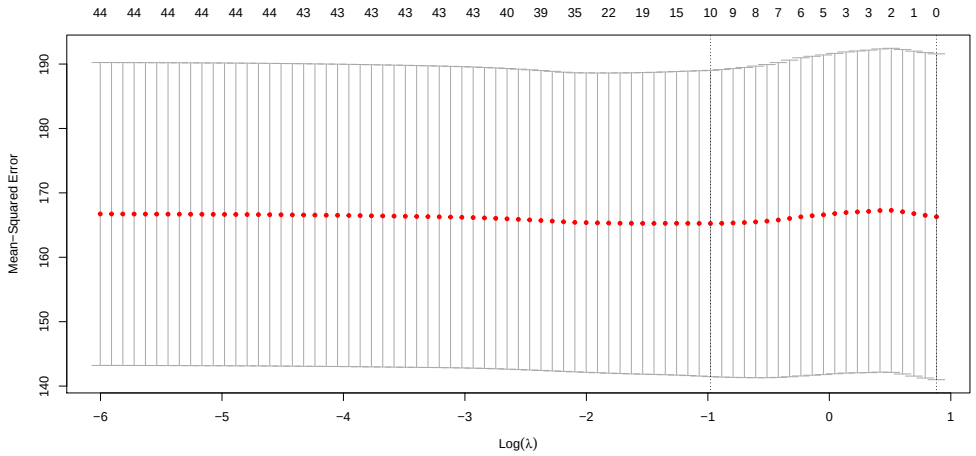

Figure 23: Lasso Regression Ergebnisse

```
bestlam = cv.out$lambda.min
log(bestlam)
```

```
## [1] -0.9784318
```

Den geringsten MSE erreichen wir bei: $log(\lambda) = -0.9784318$.

```
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test, ])
mean((lasso.pred-y.test)^2)
```

```
## [1] 179.9387
```

Wir erhalten einen Test MSE der Ridge Regression , welcher sehr ähnlich zum Test MSE der Ridge Regression ist, gegeben $log(\lambda) = 0.3759001$. Trotzdem erhalten wir wie Eingangs erwähnt einen enormen Vorteil in der Interpretation der Lasso Regression. Nachfolgend können wir sehen, dass eine Vielzahl der Koeffizienten exakt auf 0 gesetzt wurden. Wir erhalten also ein Modell welches im Vergleich zur Ridge Regression, aufgrund der Variablenselektion, deutlich einfacher zu interpretieren ist.

Ähnlich wie bei der OLS (Best Subset Selection) bzw. Ridge Methode, erhalten wir relativ große Koeffizienten für die Dummy Ausprägung des Terrorismusverdachts. So erhöht sich die Kontrolldauer durchschnittlich, bei sonst gleichbleibenden Bedingungen, um 48.36 Minuten, wenn es sich im Vorfeld um Terrorismusverdacht handelt (bzw. 17.43 Minuten bei Mordverdacht). Desweiteren sinkt die Kontrolldauer durchschnittlich um -2.83 Minuten bei Verdacht auf "CPW" Verbrechen. In Kapitel 2 haben wir gesehen, dass die meisten Kontrollen aufgrund eines CPW Verdachts durchgeführt werden. Hierbei handelt es sich um "criminal possession of a weapon", welches vermutlich aufgrund der Häufigkeit eine Standardkontrolle darstellt. Auch das Lasso Modell ergibt, dass Männer im Durchschnitt -0.52 Minuten kürzer kontrolliert werden als Frauen.

```
out = glmnet(x,y,alpha = 1)
lasso.coef = predict(out, type = "coefficients", s = bestlam)
lasso.coef
```

```
## 47 x 1 sparse Matrix of class "dgCMatrix"
##                                                                            1
## (Intercept)                                                       12.52027524
## (Intercept)                                                        .
## DAY2Tuesday                                                        .
## DAY2Wednesday                                                      .
## DAY2Thursday                                                       .
## DAY2Friday                                                         .
## DAY2Saturday                                                       .
## DAY2Sunday                                                         .
## SUSPECTED_CRIME_DESCRIPTIONAUTO STRIPPIG                           .
## SUSPECTED_CRIME_DESCRIPTIONBURGLARY                                0.07573869
## SUSPECTED_CRIME_DESCRIPTIONCPSP                                    1.28208331
## SUSPECTED_CRIME_DESCRIPTIONCPW                                    -2.83005874
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL MISCHIEF                       .
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL POSSESSION OF CONTROLLED SUBSTANCE  .
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL POSSESSION OF FORGED INSTRUMENT     .
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL POSSESSION OF MARIHUANA         .
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL SALE OF CONTROLLED SUBSTANCE    .
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL SALE OF MARIHUANA               .
## SUSPECTED_CRIME_DESCRIPTIONCRIMINAL TRESPASS                        .
## SUSPECTED_CRIME_DESCRIPTIONFORCIBLE TOUCHING                        .
## SUSPECTED_CRIME_DESCRIPTIONGRAND LARCENY                            .
## SUSPECTED_CRIME_DESCRIPTIONGRAND LARCENY AUTO                       .
## SUSPECTED_CRIME_DESCRIPTIONMAKING GRAFFITI                          .
## SUSPECTED_CRIME_DESCRIPTIONMENACING                                 .
## SUSPECTED_CRIME_DESCRIPTIONMURDER                                 17.43219416
## SUSPECTED_CRIME_DESCRIPTIONOTHER                                    .
## SUSPECTED_CRIME_DESCRIPTIONPETIT LARCENY                            .
## SUSPECTED_CRIME_DESCRIPTIONPROSTITUTION                             .
## SUSPECTED_CRIME_DESCRIPTIONRAPE                                     .
## SUSPECTED_CRIME_DESCRIPTIONRECKLESS ENDANGERMENT                    7.86280691
## SUSPECTED_CRIME_DESCRIPTIONROBBERY                                  .
## SUSPECTED_CRIME_DESCRIPTIONTERRORISM                               48.36137025
## SUSPECTED_CRIME_DESCRIPTIONTHEFT OF SERVICES                        .
## SUSPECTED_CRIME_DESCRIPTIONUNAUTHORIZED USE OF A VEHICLE            3.61581483
## OBSERVED_DURATION_MINUTES                                          0.02044632
## FRISKED_FLAGY                                                      .
## WEAPON_FOUND_FLAGY                                                 .
## SUSPECT_REPORTED_AGE                                               .
## SUSPECT_SEXMALE                                                   -0.52969431
## SUSPECT_RACE_DESCRIPTIONASIAN / PACIFIC ISLANDER                   0.05584614
## SUSPECT_RACE_DESCRIPTIONBLACK                                     -0.49742265
## SUSPECT_RACE_DESCRIPTIONBLACK HISPANIC                             .
## SUSPECT_RACE_DESCRIPTIONMIDDLE EASTERN/SOUTHWEST                   .
## SUSPECT_RACE_DESCRIPTIONWHITE                                      .
## SUSPECT_RACE_DESCRIPTIONWHITE HISPANIC                             .
## SUSPECT_ARRESTED_FLAG                                             -1.07881817
## DAYNIGHTNight                                                      .
```

3.3 Beyond Linearity

3.3.1 Polynomial Regression

Nachfolgend schätze ich ein Modell mit Hilfe der *lm()* Funktion, um die Kontrolldauer vorherzusagen. Hierfür verwende ich ein Polynom 4. Grades (Alter der kontrollierten Person) als erklärende Variable.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \epsilon_i$$

Die polynomiale Regression erweitert das lineare Modell durch Hinzufügen zusätzlicher Predictoren, die durch Erhöhen jedes der ursprünglichen Predictoren auf eine Potenz erhalten werden.

```
fit = lm(STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 4), data = SQF4)
coef(summary(fit))
```

```
##                                 Estimate Std. Error    t value  Pr(>|t|)
## (Intercept)                   10.9374786  0.1423876 76.8148091 0.0000000
## poly(SUSPECT_REPORTED_AGE, 4)1 -4.4558629 13.3183820 -0.3345649 0.7379614
## poly(SUSPECT_REPORTED_AGE, 4)2 11.3485157 13.3183820  0.8520942 0.3941852
## poly(SUSPECT_REPORTED_AGE, 4)3 -0.7965485 13.3183820 -0.0598082 0.9523098
## poly(SUSPECT_REPORTED_AGE, 4)4 -3.8637987 13.3183820 -0.2901102 0.7717388
```

Ein Grid von Werten für das Alter wird festgelegt, um später Vorhersagen für diese Werte zu erhalten. Das Funktionsargument *se = TRUE* liefert uns die Standardfehler der Vorhersagen.

```
agelims = range(SQF4$SUSPECT_REPORTED_AGE)      #| Min-Max Wertebereich age
age.grid = seq(from = agelims[1], to = agelims[2])
preds = predict(fit, newdata = list(SUSPECT_REPORTED_AGE = age.grid), se = TRUE)
se.bands = cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)
```

Zuletzt können die Daten visualisiert werden. Hierbei wird der Fit des Polynoms 4. Grades in Bild 24 in blau eingezeichnet.

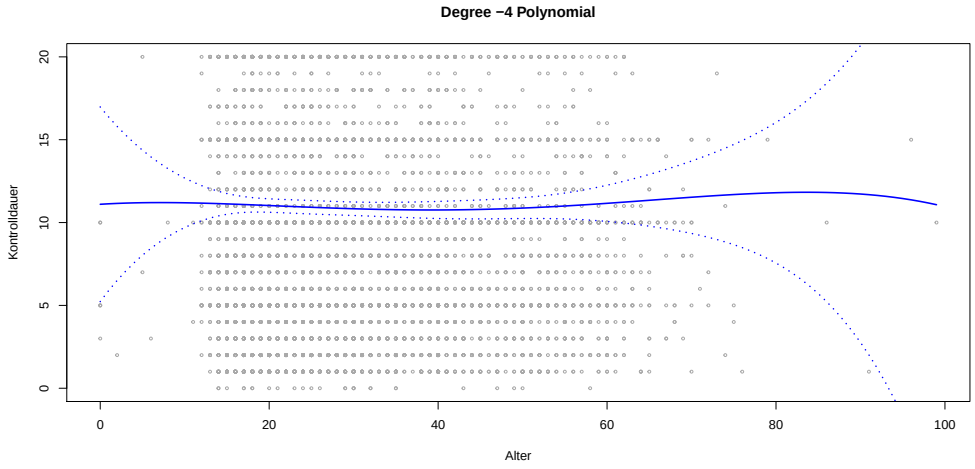

Figure 24: Polynomial Regression

Wir sehen, dass wir an den Rändern 'strange shapes' erhalten. Die Konfidenzintervalle gehen extrem auseinander. Dies liegt daran, dass Polynome eine globale Struktur erzwingen und wir an den Rändern wenig

Observationen haben. Obwohl die Polynomial Regression eine lineare Regression wie jede andere ist, sind wir meistens nicht an den individuellen Koeffizienten, sondern an dem Zusammenhang der Kontrolldauer und des Alters interessiert.

```
summary(SQF4$STOP_DURATION_MINUTES)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00    8.00   10.94   14.00  212.00
```

Es scheint als würde die Kontrolldauer in Bild 24 aus zwei unterschiedlichen Populationen stammen: Es scheint als gäbe es eine Gruppe an Personen die extrem lange Kontrolliert wurden (über 20 min) und eine große Gruppe an Personen, die unter 20 min kontrolliert wurden. Der Plot in Bild 21 ist in der Y Achse, aufgrund extremer Ausreisser, limitiert. Wir können daher die Kontrolldauer (ähnlich wie in Kapitel 3.1.2) auch als binäre Variable behandeln, indem wir den Wertebereich der Variable faktorisieren. In anderen Worten, kann ebenso nachfolgendes Modell gefittet werden (Polynomial Logit):

$$Pr(y_i > 20|x_i) = \frac{exp(\beta_0 + \beta_1 x_i + ... + \beta_d x_i^d)}{1 + exp(\beta_0 + \beta_1 x_i + ... + \beta_d x_i^d)}$$

Das Ergebnis o.g. Ausdrucks ist Bild 25 zu entnehmen. Hierbei sehen wir, dass im Schnitt nur ca 5% der Kontrollen über 30 Minuten andauern. Desweiteren scheint das Alter ziehmlich unabhängig von der Kontrolldauer zu sein. Erneut finden sich an den Rändern unpräzise Schätzungen.

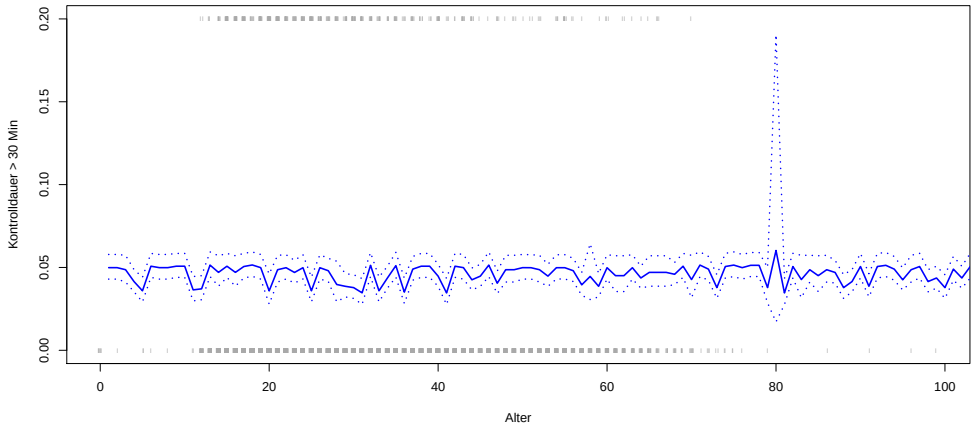

Figure 25: Logistic Polynomial Regression

Um eine Polynom Regression durchzuführen, müssen wir den Grad des Polynoms bestimmen, welches wir verwenden möchten. Ein Weg dies zu machen sind Hypothesentests. Dies bedeutet, dass wir Modelle in einer Range eines linearen Modells hin zu einem Polynom 5. Grades fitten. Anschließend wählen wir das simpelste Modell, welches hinreichend Erklärungsgehalt für die Beziehung zwischen der Kontrolldauer und dem Alter besitzt.

Dafür nutzen wir die *anova()*-Funktion, welche eine Analyse der Varianz (ANOVA nutzt F-Test) performt. Es wird die Nullhypothese getestet, dass ein Model M_1 ausreichend Erklärungsgehalt besitzt, gegen die Alternativhypothese, dass ein komplexeres Modell M_2 benötigt wird. Um die *anova()*-Funktion zu nutzen, müssen M_1 und M_2 geschachtelt (nested) sein! Die Predictoren aus M_1 müssen also ein Subset der Predictoren aus M_2 sein. In diesem Fall fitten wir also 5 unterschiedliche Modelle.

```r
# ANOVA Methode
fit.1 = lm(STOP_DURATION_MINUTES ~ SUSPECT_REPORTED_AGE, data=SQF4)
fit.2 = lm(STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 2), data=SQF4)
fit.3 = lm(STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 3), data=SQF4)
fit.4 = lm(STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 4), data=SQF4)
fit.5 = lm(STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 5), data=SQF4)
anova(fit.1, fit.2, fit.3, fit.4, fit.5)
```

```
## Analysis of Variance Table
##
## Model 1: STOP_DURATION_MINUTES ~ SUSPECT_REPORTED_AGE
## Model 2: STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 2)
## Model 3: STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 3)
## Model 4: STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 4)
## Model 5: STOP_DURATION_MINUTES ~ poly(SUSPECT_REPORTED_AGE, 5)
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1   8747 1551149
## 2   8746 1551020  1   128.789 0.7260 0.3942
## 3   8745 1551020  1     0.634 0.0036 0.9523
## 4   8744 1551005  1    14.929 0.0842 0.7717
## 5   8743 1550974  1    30.951 0.1745 0.6762
```

Ähnlich wie es Bild 24 und Bild 25 vermuten lassen, gibt uns das Ergebnis des ANOVA Tests keinen Hinweis auf eine Verbesserung der Schätzung durch Hinzufügen eines Polynoms auf das Alter der kontrollierten Person! Es sind also keine signifikanten Nichtlinearitäten vorhanden!

Weiterhin soll nachfolgend untersucht werden, ob das Alter einen Einfluss darauf hat, ob man gefilzt wurde oder nicht. Das Ergebnis eines ANOVA Tests sagt, dass ein Polynom 5. Grades die abhängige Variable am besten beschreibt.

```
## Analysis of Deviance Table
##
## Model 1: FRISKED_FLAG2 ~ SUSPECT_REPORTED_AGE
## Model 2: FRISKED_FLAG2 ~ poly(SUSPECT_REPORTED_AGE, 2)
## Model 3: FRISKED_FLAG2 ~ poly(SUSPECT_REPORTED_AGE, 3)
## Model 4: FRISKED_FLAG2 ~ poly(SUSPECT_REPORTED_AGE, 4)
## Model 5: FRISKED_FLAG2 ~ poly(SUSPECT_REPORTED_AGE, 5)
## Model 6: FRISKED_FLAG2 ~ poly(SUSPECT_REPORTED_AGE, 6)
## Model 7: FRISKED_FLAG2 ~ poly(SUSPECT_REPORTED_AGE, 7)
##   Resid. Df Resid. Dev Df Deviance
## 1      8747     2151.7
## 2      8746     2150.1  1  1.68857
## 3      8745     2149.1  1  0.99287
## 4      8744     2148.9  1  0.11589
## 5      8743     2148.9  1  0.04678
## 6      8742     2148.6  1  0.29027
## 7      8741     2148.4  1  0.21563
```

Bild 26 zeigt das Ergebnis der Analyse. Wir sehen, dass beispielsweise bei älteren Leuten (80 Jahre) die Wahrscheinlichkeit deutlich geringer ist gefilzt zu werden (ca 30%), als bei jüngeren Leuten (ca 60%). Jedoch befinden sich kaum Observationen im Sample von Personen über 70 Jahren, weswegen die Ergebnisse als nicht stabil angesehen werden können.

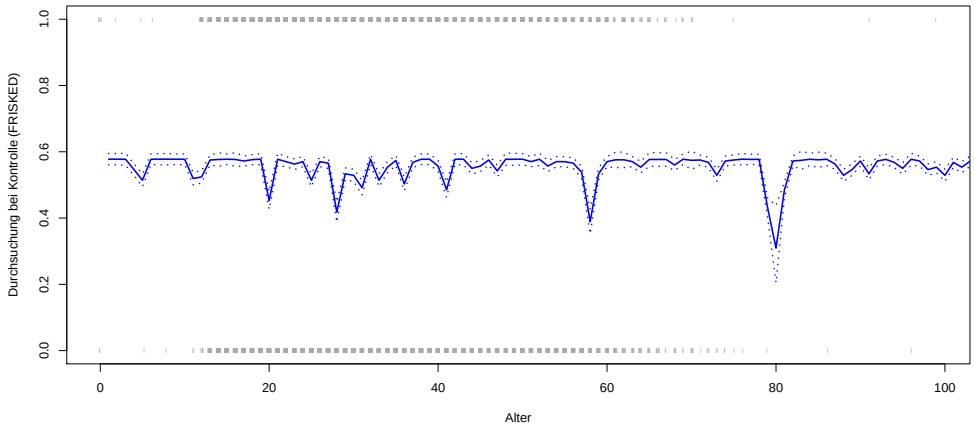

Figure 26: Polynomial Logistic Regression - Frisked Flag

3.3.2 Splines

Wie in Kapitel 3.3.1 beschrieben, ist eines der Hauptprobleme der Polynomischen Regression die globale Struktur (siehe Bild 24). Anstatt ein hochdimensionales Polynom über die komplette Range von X zu fitten, können wir mit der 'piecewise polynomial regression' niedriger dimensionale Polynome über unterschiedliche Regionen von X fitten.

Wir können also beispielsweise eine kubische Regression der Form:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i$$

fitten, bei dem die Beta Koeffizienten sich dadurch unterscheiden, dass sie auf unterschiedliche Teile der Range von X angewenet werden. Der Punkt an dem sich die Koeffizienten ändern, wird Knoten (c) genannt. Ein stückweises kubisches Polynom mit einem Knoten am Punkt c nimmt also beispielsweise nachfolgende Form an:

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & \text{wenn: } x_i = c \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & \text{wenn: } x_i >= c \end{cases}$$

Hierdurch fitten wir also zwei unterschiedliche Polynomfunktionen. Je mehr Knoten wir nutzen, desto flexibler wird die Schätzung. Wenn wir K Knoten setzen, müssen wir $K + 1$ unterschiedliche Polynomials fitten.

Ist der Knoten *unconstraint* bestehen keine Einschränkungen im Fit, d.h. es entstehen Sprünge. Durch Anwendung sogenannter *Constraints* (Einschränkungen), können wir **Splines** fitten. Hierbei setzen wir 3 Einschränkungen auf unsere Schätzung: Kontinuierlicher Fit sowie kontinuierliche erste und zweite Ableitungen. Die Einschränkungen sorgen dafür, dass das 'piecewise polynomial' nicht nur kontinuierlich an den Knotenpunkten, sondern auch *smooth* sein muss, da die 1. und 2. Ableitung ebenfalls kontinuierlich sind.

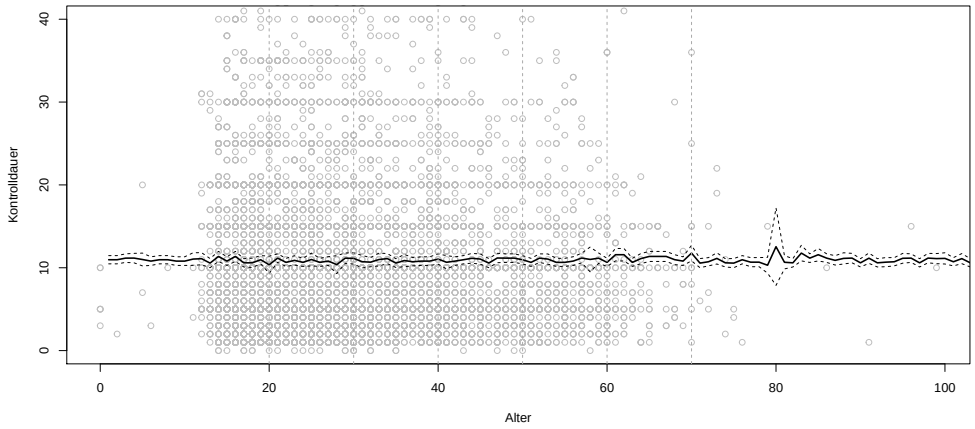

Figure 27: Spline Regression Kontrolldauer vs. Alter

Bild 27 zeigt ein geschätztes **Cubic Spline** Modell. Hierfür wurden Knotenpunkte an jedem Jahrzent des Alters in einer Sequenz von 10 Jahren, zwischen 20 - 70 Jahren, gesetzt. Es wurden also 7 Knoten gesetzt, was einen Spline mit 14 Basisfunktionen produziert.

Wir sehen nun in Bild 28 eine Gegenüberstellung der Polynom Regression und des Cubic Splines. Der Cubic Spline liefert vor allem an den Randverteilungen einen deutlich besseren Fit als die einfache Polynomregression.

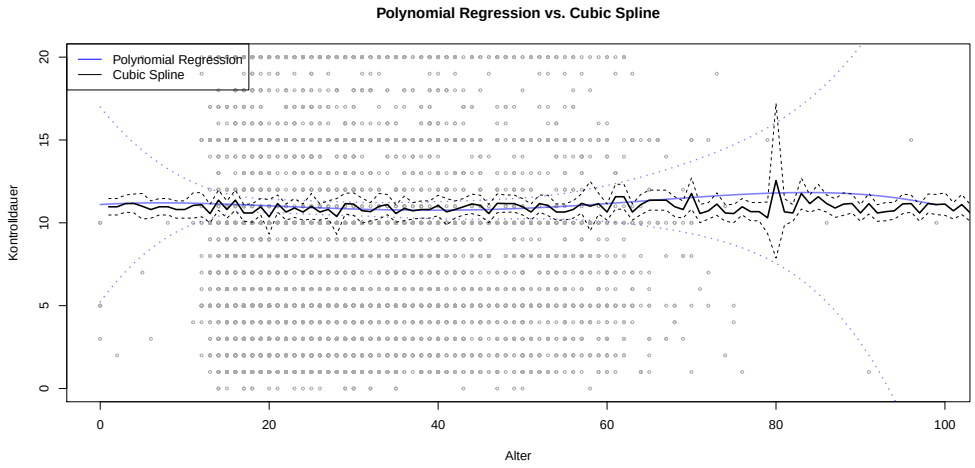

Figure 28: Polynomial Regression vs. Cubic Spline

Ein Spezialfall der Regression Splines stellt der Natural Spline dar. Dieser beinhaltet einen sog. 'boundary constraint', d.h. die Funktion muss am 'Boundary' linear sein, also in dem bereich, in dem X kleiner als der kleinste Knoten oder größer als der größte Knoten ist. Dieser Constraint sorgt dafür, dass natural splines generell stabilere Schätzungen an den Rändern ergeben. Nachfolgend wird ein solcher Natural Spline mit vier Freiheitsgraden geschätzt.

Das Ergebnis der Gegenüberstellung des Cubic Splines und Natural Splines kann Bild 29 entnommen werden. Hierbei sind die oben beschriebenen Vorteile des zusätzlichen Constraints des Natural Splines deutlich ersichtlich.

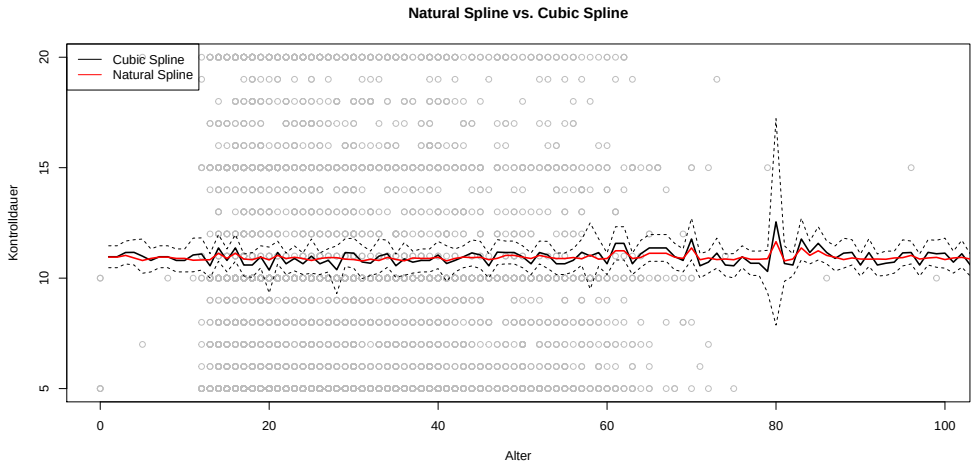

Figure 29: Natural Spline vs. Cubic Spline

3.3.3 Smoothing Splines

Bei Regression Splines spezifizieren wir ein Set von Knoten, produzieren eine Basis Funktion und nutzen OLS um die Spline Koeffizienten zu schätzen. Wir wollen eine Funktion $g(x)$ finden, die die beobachteten Daten gut fittet. Hierfür müssen natürlich die RSS $((y - g(x))^2)$ minimiert werden. Durch Interpolation könnten wir g(x) so wählen, dass die RSS immer 0 sind. Dann würden wir aber Overfitten. Was wir also eigentlich wollen, ist eine Funktion g, die die RSS so klein wie möglich macht und ebenso smooth ist. Wir wollen eine Funktion finden, die

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

minimiert, indem λ ein nicht negativer Tuningparameter ist. Die Funktion g, die oben genannten Ausdruck minimiert nennt man Smoothing Spline.

Die zweite Ableitung von g ist ein Maß der Krümmung. Sie ist im Absolutwert groß, wenn $g(t)$ in der Nähe von t sehr krumm ist und andernfalls ist sie nahe Null. Das Integral ist eine Messung der Veränderung in der Funktion $g'(t)$ über ihren kompletten Bereich. Hieraus resultieren 2 Cases:

- Ist g sehr smooth, dann wird $g'(t)$ nahezu konstant sein und das Integral über $g''(t)^2 dt$ nimmt einen kleinen Wert an
- Wenn g eine hohe Varianz hat, wird $g'(t)$ sehr variabel sein und das Integral über $g''(t)^2 dt$ nimmt einen großen Wert an.

Der Penalty Term begünstigt es also, wenn g smooth ist. Je höher der Wert von Lambda, desto smoother wird g sein. Der Penalty Term bestraft also Abweichungen von der Linearität. Hierbei ist ebenso zwischen 2 Cases zu unterscheiden:

- $\lambda = 0$: Der Penalty term hat keinen Effekt und die Funktion g wird sprunghaft sein und die Trainingsdaten interpolarieren
- $\lambda = \infty$: g wird perfekt smooth sein. Es wird nur eine gerade Linie sein, die so nah wie möglich die Trainingspunkte passiert. g wird in diesem Fall die OLS Ergebnisse liefern, da die Loss Funktion die RSS minimiert - wie bei OLS!

Figure 30: Smoothing Spline

Es kann gezeigt werden, dass wenn wir an jedem X Wert eine Stützstelle aufstellen und einen 'piecewise cubic spline' integrieren, mit kontinuierlichen 1. und 2. Ableitungen an jedem Knoten, der Ausdruck von oben minimiert wird. Ein Smoothing Spline hat also viele Knoten, was viele Freiheitsgrade benötigt. Lambda kontrolliert hierbei die 'roughness' des Splines und somit auch die effektive Anzahl der Freiheitsgrade. Der optimale Wert für Lambda wurde in Bild 30 mit Hilfe von Cross Validation ermittelt. Wir sehen, deutliche Unterschiede zwischen den Freiheitsgraden und erhalten mit dem Lambda Wert aus der CV Methode (blaue Linie) deutlich stabilere Ergebnisse. Die rote Kurve ist der Smoothing Spline den wir erhalten, wenn wir 16 effektive Freiheitsgrade setzen. Die blaue Kurve ist der Smoothing Spline den wir erhalten, wenn wir λ über Cross Validation ermitteln - in diesem Fall beinhaltet Lambda 2 effektive Freiheitsgrade.

4 Conclusion

Die Analyse des vorliegenden Datensatzes lieferte spannende Ergebnisse bezüglich der verhafteten Personen, durchschnittlichen Kontrolldauer und Untersuchungen anhand einer Vielzahl erklärender Variablen.

Die unterschiedlichen Methoden der Lehrveranstaltung zeigen, dass wir mit einer Vielzahl unterschiedlicher Methoden zu recht ähnlichen Ergebnissen kommen können. Hierbei gilt es abzuwägen, welche Methodik im Einzelfall als hinreichend gilt.

Während die Lasso Regression meiner Meinung nach eine einfache Interpretationsmöglichkeit für die durchschnittliche Kontrolldauer geliefert hat, konnten Bäume ebenfalls valide Klassifikationsergebnisse erzielen. Letztendlich ist es wohl von größter Wichtigkeit die Hyperparameter der einzelnen statistischen Verfahren bewusst zu wählen, um den bestmöglichen Fit und somit valide Ergebnisse zu erzielen.

Zusammenfassend kann gesagt werden, dass im vorliegenden Datensatz in keinem der Modelle statistische Signifikanz der Kontrollen bezogen auf die ethnische Herkunft nachgewiesen werden konnte. Vielmehr ist beispielsweise bei der Kontrolldauer das Geschlecht, die Observationszeit oder das vermutete Verbrechen von Bedeutung. So konnte gezeigt werden, dass vor allem bei Terrorismus- oder Mordverdacht die durchschnittliche Kontrolldauer extrem ansteigt. Bei der Verhaftung spielen die Tageszeit, die Bewaffnung und das Geschlecht ebenfalls eine signifikante Rolle. Durchsuchungen finden hauptsächlich bei männlichen Personen statt. Sollte eine Waffe gefunden werden, findet wenig überraschend zu 100% eine Durchsuchung statt.

5 Reference

[1] Hastie, T., James, G., Tibshirani, R., Witten, D. (2021): An Introduction to Statistical Learning with Applications in R (second Edition)
[2] Hofmarcher, P. (2021): Lecture Notes Methoden der statistischen Inferenz