Statistik hochdimensionaler und komplexer Daten

Fuat Sarp Olcay 11933989

03 02 2022

Aufgabe 1)

```
#Daten einlesen und in dazugehörende Typen umformen.
vertragsdaten <- read.csv("C:/Users/ASUS/Documents/Vertragsdaten.csv", sep = ";")
vertragsdaten$policy_number <-  as.numeric(vertragsdaten$policy_number)
vertragsdaten$start_date <- as.Date(vertragsdaten$start_date, format ="%d.%m.%Y")
vertragsdaten$end_date <- as.Date(vertragsdaten$end_date, format ="%d.%m.%Y")
vertragsdaten$premium <- gsub(",", ".", vertragsdaten$premium)
vertragsdaten$premium <- as.numeric(vertragsdaten$premium)
vertragsdaten$discount <- gsub(",", ".", vertragsdaten$discount)
vertragsdaten$discount <- as.numeric(vertragsdaten$discount)
vertragsdaten$year <- as.numeric(vertragsdaten$year)
# Ein Trick für den 2. Teil der Datenaufbereitung.
vertragsdaten[vertragsdaten$cancellation_date == "",]$cancellation_date <- "01.01.2001"
vertragsdaten$cancellation_date <- as.Date(vertragsdaten$cancellation_date, format ="%d.%m.%Y")
```

```
#Filtern Sie alle Policen heraus, die einen negativen oder fehlenden Praemienbetrag aufweisen.
vertragsdaten1 <- vertragsdaten[!is.na(vertragsdaten$premium), ]
vertragsdaten2 <- vertragsdaten1[!vertragsdaten1$premium < 0, ]
#Filtern Sie alle Policen heraus, deren Stornodatum nach dem Enddatum liegt.
vertragsdaten3 <- vertragsdaten2[!vertragsdaten2$cancellation_date > vertragsdaten2$end_date,]
#Filtern Sie alle Policen heraus, bei denen das Startdatum nach dem Enddatum liegt.
vertragsdaten4 <- vertragsdaten3[vertragsdaten3$start_date < vertragsdaten3$end_date,]
#Filtern Sie alle Policen heraus, bei denen das Jahr (year) fehlt.
vertragsdaten5 <- vertragsdaten4[!is.na(vertragsdaten4$year),]
#Entfernen Sie alle Zeilenduplikate.
vertragsdaten6 <- unique(vertragsdaten5)
Vertragsdaten_korrigiert.csv <- vertragsdaten6 #Die Tabelle umbenennen.
```

Frage 1: Wie groß ist der Unterschied in der Gesamtpraemie zwischen den ursprünglichen Daten und den korrigierten Daten?

```
sum(vertragsdaten$premium, na.rm = TRUE) - sum(Vertragsdaten_korrigiert.csv$premium)
```

```
## [1] 10391.62
```

Der Unterschied zwischen den beiden Datensaetzen bezüglich der Gesamtpraemie liegt bei 2981326.

Frage 2: In welchem Jahr gibt es die größte Anzahl von Annullierungen (Storno)? Wie hoch ist in diesem Jahr der durchschnittliche Rabatt?

```
temp <- Vertragsdaten_korrigiert.csv[!is.na(Vertragsdaten_korrigiert.csv$cancellation_date),]
#Das Jahr 2001 nicht betrachten, da dieses Jahr "Pseudo" ist.
sort(table(substring(temp$cancellation_date, 1, 4)), decreasing = TRUE)
```

```
##
## 2001 2017 2018 2019 2020
## 8106 3592 2945 1884   28
```

Im Jahr 2017 gab es die höchste Anzahl der Annulierungen.

```
mean(Vertragsdaten_korrigiert.csv[Vertragsdaten_korrigiert.csv$year == 2017,]$discount, na.rm = TRUE)
```

```
## [1] 4.788565
```

Der durchschnittliche Rabatt in diesem Jahr betraegt 4.788565.

Aufgabe 2)

Führen Sie die Auswahl der besten Teilmenge (best subset selection) manuell durch, indem Sie in jedem Schritt R2 berechnen und im letzten Schritt Mallows Cp verwenden (manuelles Berechnen anhand der Formel aus dem Kurs). Frage 1: Welches Modell schneidet am besten ab (mit welchen Regressoren)?

```
housing <- read.csv("C:/Users/ASUS/Documents/housing_data.csv")#Daten einlesen.
housing$price <- log(housing$price)
attach(housing)
set.seed(1)
```

Best Sub-set selection

```
nullmodel <- lm(price~ NULL) #Das Nullmodell
model1 <- lm(price ~ bedrooms)
model2 <- lm(price ~ grade)
model3 <- lm(price ~ sqft_living)
model12 <- lm(price ~ bedrooms + grade)
model13 <- lm(price ~ bedrooms + sqft_living)
model23 <- lm(price ~ grade + sqft_living)
model123 <- lm(price ~ bedrooms + grade + sqft_living)
```

R^2 für das Nullmodell:

```
summary(nullmodel)$r.squared
```

```
## [1] 0
```

R^2 für das Modell mit “Bedrooms”

```
summary(model1)$r.squared
```

```
## [1] 0.1141674
```

R^2 für das Modell mit “Grade”

```
summary(model2)$r.squared
```

```
## [1] 0.4878086
```

R^2 für das Modell mit “Sqft_living”

```
summary(model3)$r.squared
```

```
## [1] 0.4785024
```

R^2 für das Modell mit “Bedrooms” und “Grade”

```
summary(model12)$r.squared
```

```
## [1] 0.4971192
```

R^2 für das Modell mit “Bedrooms” und “Sqft_living”

```
summary(model13)$r.squared
```

```
## [1] 0.482773
```

R^2 für das Modell mit “Grade” und “Sqft_living”

```
summary(model23)$r.squared
```

```
## [1] 0.5471079
```

R^2 für das Modell mit allen Kovariablen

```
summary(model123)$r.squared
```

```
## [1] 0.5478007
```

Das Modell mit alle Kovariablen und das Modell mit den Kovariablen “Grade” und “Sqft_living” besitzen die größte R^2-Werte. Wir berechnen den Mallows Cp für diese Modelle.

Frage 2: Wie groß ist Mallows Cp für dieses Modell? Mallows Cp berechnen

```
n <- nrow(housing)
p <- ncol(housing)
sigma_hat2<- deviance(model123)/(n-p)
mallows_full <- deviance(model123)/n+2*sigma_hat2 *(p/n)
mallows_12 <- deviance(model23)/n+2*sigma_hat2 * (3/n) # p=3
```

Mallows Cp für das Modell mit allen Kovariablen

```
mallows_full
```

```
## [1] 0.1266714
```

Mallows Cp für das Modell mit den Kovariablen “Grade” und “Sqft_living”

```
mallows_12
```

```
## [1] 0.1268401
```

Das Vollmodell wird hier ausgewählt. Mallows Cp betraegt 0.1266714.

Nun wollen wir ein Ridge-, Lasso- und ein elastisches Netzmodell (mit alpha=0,5) mit Hilfe der Funktion glmnet und einer automatischen Wahl der Lambdas anpassen. Teilen Sie zunaechst die Daten in 80-20% auf. Passen Sie alle drei Modelle an die Trainingsdaten (80%) an und prüfen Sie die Leistung bei 20% der Daten, d.h. berechnen Sie den TestMSE auf die übliche Weise.

Frage 3: Welches Modell schneidet am besten ab? Und wie groß ist der MSE für dieses Modell?

```
library(caret)
set.seed(1)
housing <- housing[,-1]
housing.data <- data.matrix(housing)
set.seed(1)#Datensatz als 80%-20% teilen.
folds1 <- createDataPartition(housing.data[,1], p = 0.8, list = FALSE)
datatrain <- housing.data[folds1,]
datatest <- housing.data[-folds1,]
```

```
library(glmnet) #Fits
fit.lasso <- glmnet(datatrain, datatrain[,1], alpha = 1, family = "gaussian")
fit.ridge <- glmnet(datatrain, datatrain[,1], alpha = 0, family = "gaussian")
fit.elas <- glmnet(datatrain, datatrain[,1], alpha = 0.5, family = "gaussian")
#Predictions
pred.lasso <- predict(fit.lasso, newx = datatest, s = fit.lasso$lambda.1se)
pred.ridge <- predict(fit.ridge, newx = datatest, s = fit.lasso$lambda.1se)
pred.elas <- predict(fit.elas, newx = datatest, s = fit.lasso$lambda.1se)
#MSE's berechnen
error.lasso <- pred.lasso - datatest[,1]
error.ridge <- pred.ridge - datatest[,1]
error.elas<- pred.elas - datatest[,1]
mse.lasso <- mean(error.lasso^2)
mse.ridge<- mean(error.ridge^2)
mse.elas <- mean(error.elas^2)
data.frame(c("mse.lasso", "mse.ridge", "mse.elas"), c(mse.lasso, mse.ridge, mse.elas))
```

```
##   c..mse.lasso....mse.ridge....mse.elas.. c.mse.lasso..mse.ridge..mse.elas.
## 1                              mse.lasso                          0.04216267
## 2                              mse.ridge                          0.16465976
## 3                               mse.elas                          0.05331393
```

Das LASSO Modell schneidet am besten ab. Der MSE betraegt 0.2050216.

Gehen wir nun zurück zu 100% der Daten und verwenden wir die Funktion cv.glmnet, um diese 3 Modelle anzupassen. Verwenden Sie die automatische Wahl von Lambdas und die 10-fache Kreuzvalidierung. Frage 4: Welches Modell schneidet jetzt am besten ab? Und wie groß ist der MSE (the mean cross-validated error, cvm) für dieses Modell?

```
#Kreuzvalidierung
cv.lasso <- cv.glmnet(housing.data, housing.data[,1], alpha = 1)#lasso
cv.ridge <- cv.glmnet(housing.data, housing.data[,1], alpha = 0)#Ridge
cv.elas <- cv.glmnet(housing.data, housing.data[,1], alpha = 0.5)#Elastischer Netz
lambda_best.lasso <- cv.lasso$lambda.min
lambda_best.ridge <- cv.ridge$lambda.min
lambda_best.elas <- cv.elas$lambda.min
#CVM für optimale Lambdas
i1 <- which(cv.lasso$lambda == cv.lasso$lambda.min)#Lasso
mse.min.lasso <- cv.lasso$cvm[i1]
i2 <- which(cv.ridge$lambda == cv.ridge$lambda.min)#Ridge
mse.min.ridge <- cv.ridge$cvm[i2]

i3 <- which(cv.elas$lambda == cv.elas$lambda.min)#Elastischer Netz
mse.min.elas <- cv.elas$cvm[i3]
data.frame(c("CVM.Lasso", "CVM.Ridge", "CVM.Elastischer.Netz"), c(mse.min.lasso, mse.min.ridge, mse.min
```

```
##   c..CVM.Lasso....CVM.Ridge....CVM.Elastischer.Netz..
## 1                                          CVM.Lasso
## 2                                          CVM.Ridge
## 3                                  CVM.Elastischer.Netz
##   c.mse.min.lasso..mse.min.ridge..mse.min.elas.
## 1                                  0.0002381446
## 2                                  0.0037516544
## 3                                  0.0002705375
```

Hier würden wir auch LASSO bevorzugen. Der CVM betraegt 0.001170546.

Aufgabe 3) Sehen Sie sich den Datensatz insurance fraud.csv an, in dem eine Liste von Versicherungsansprüchen enthalten ist und mit der Variablen fraud reported angegeben wird, ob ein Betrug vermutet wird oder nicht.

```
fraud <- read.csv("C:/Users/ASUS/Documents/insurance_fraud.csv")
```

Frage 1: Wie hoch ist der Prozentsatz der Betrugsfaelle in den Daten?

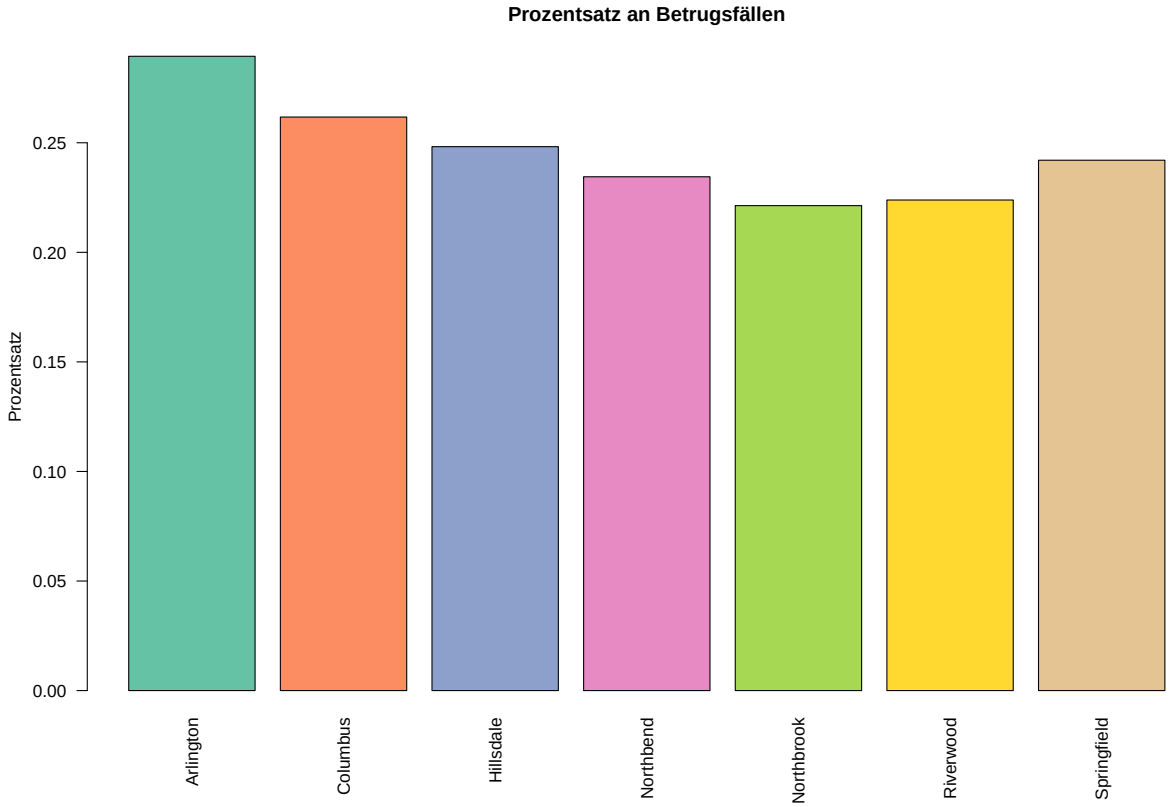
```
mean(fraud$fraud_reported)
```

```
## [1] 0.247
```

Der Prozentsatz der Betrugsfälle betraegt 24.7%.

Frage 2: In welcher Stadt gibt es den höchsten Prozentsatz an Betrugsfaellen?

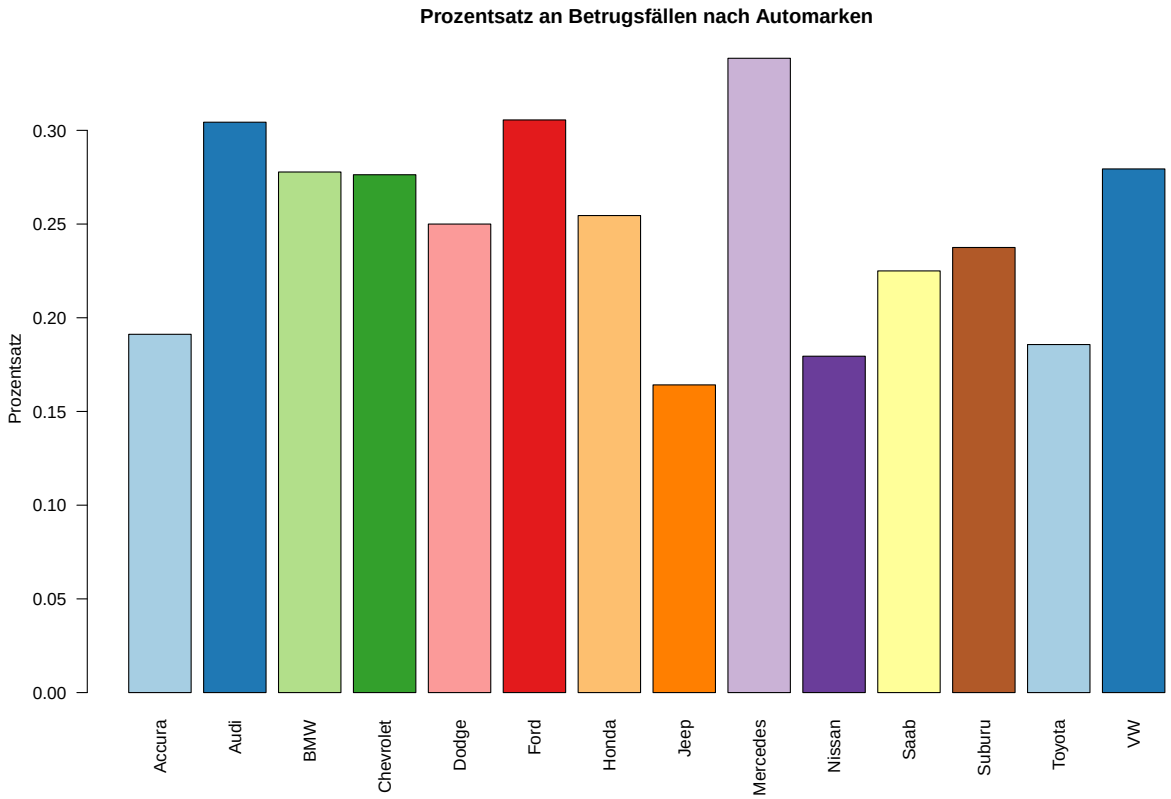
```
state.fraud <- aggregate(fraud_reported ~ incident_city, FUN = mean, data = fraud)#Mittelwert für jede
library(RColorBrewer)
coul <- brewer.pal(length(state.fraud$incident_city), "Set2")
barplot(state.fraud$fraud_reported, names.arg = state.fraud$incident_city, col = coul, ylab = "Prozents
```



In Arlington gab es der höchste Prozentsatz an Betrugsfaellen mit 28.9%.

Frage 3: Für welche Automarke gibt es den größten Prozentsatz an Betrugsfaellen?

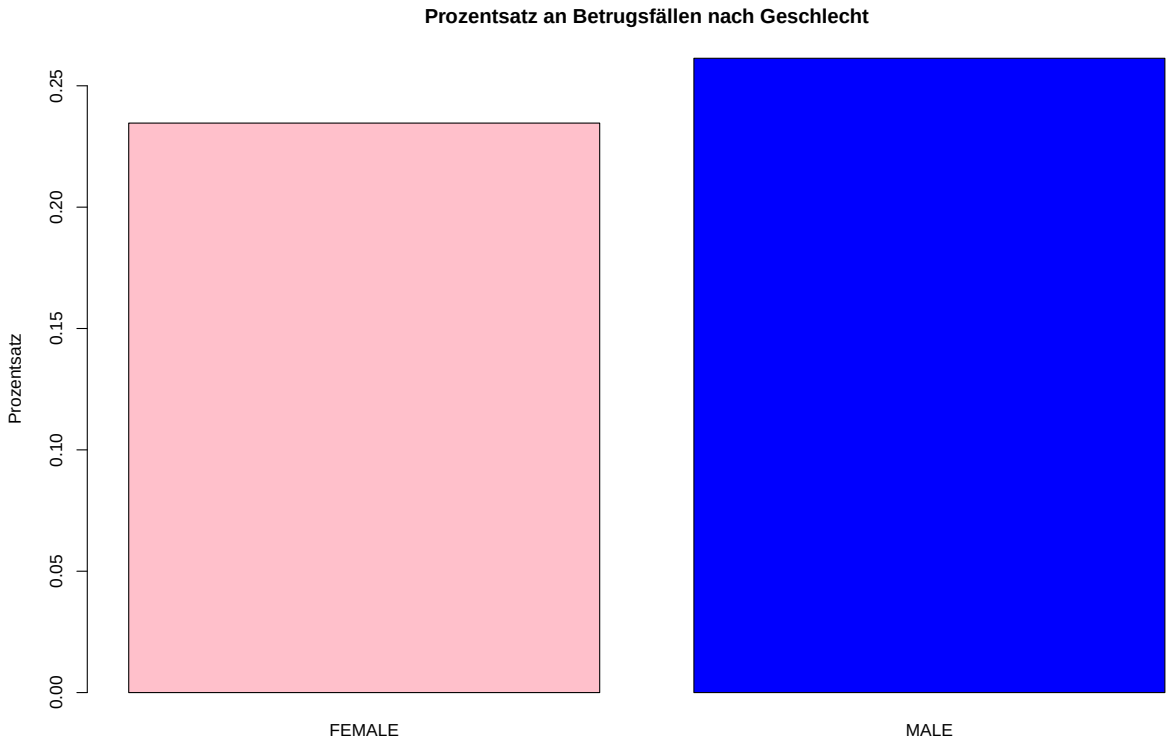
```
set.seed(1)
marke.fraud <- aggregate(fraud_reported ~ auto_make, FUN = mean, data = fraud)#Mittelwert für jede Mark
marke.fraud$auto_make[marke.fraud$auto_make == "Volkswagen"] <- "VW"
library(RColorBrewer)
coul1 <- brewer.pal(length(marke.fraud$auto_make), "Paired")
barplot(marke.fraud $fraud_reported, names.arg = marke.fraud$auto_make, col = coul1, ylab = "Prozentsat
```



Für Mercedes gab es der höchste Prozentsatz der Betrugsfaelle mit 33.8%.

Frage 4: Wer ist nach diesen Daten eher betrugsanfaellig, Männer oder Frauen?

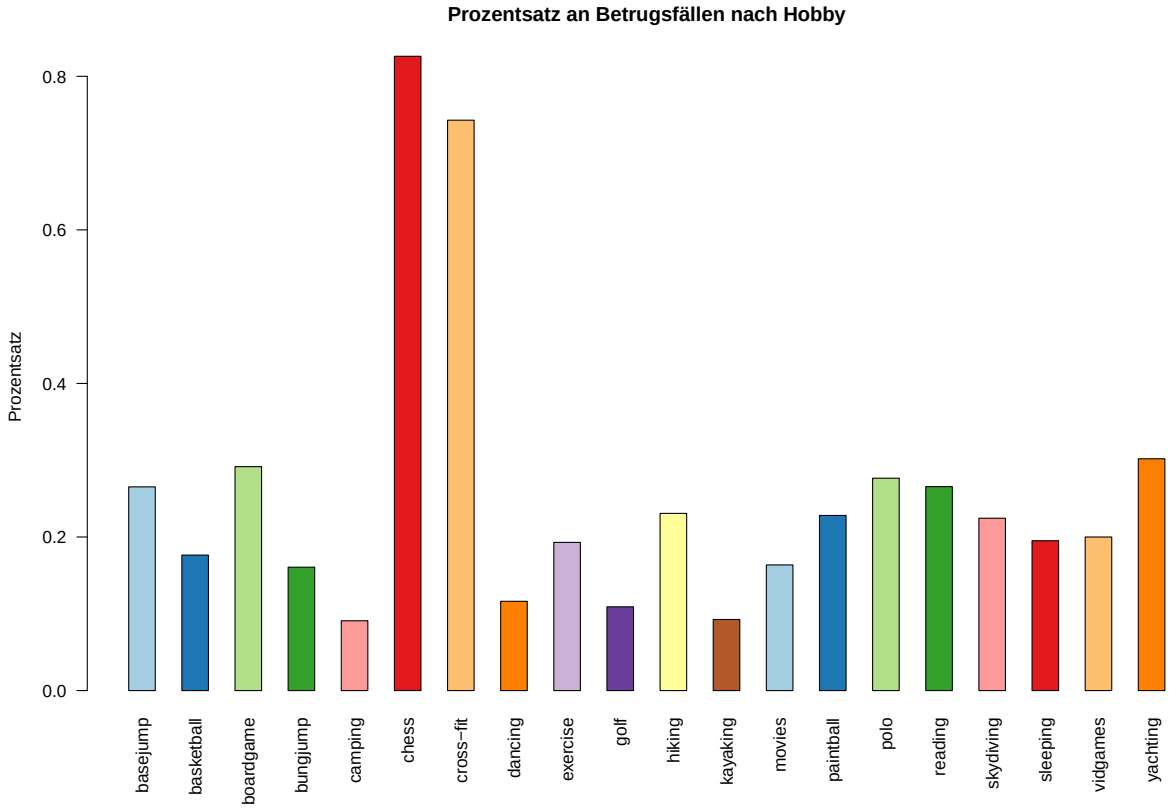
```
sex.fraud <- aggregate(fraud_reported ~ insured_sex, FUN = mean, data = fraud)
barplot(sex.fraud$fraud_reported, names.arg = sex.fraud$insured_sex, col = c("Pink", "Blue"), ylab = "Pr
```



Männer sind eher betrugsanfaellig mit dem Prozentsatz 0.2613391.

Frage 5: Welche Hobbykategorie hat den höchsten Prozentsatz an betrügerischen Forderungen?

```
hobby.fraud <- aggregate(fraud_reported ~ insured_hobbies, FUN = mean, data = fraud)
hobby.fraud$insured_hobbies[1] <- "basejump"
hobby.fraud$insured_hobbies[3] <- "boardgame"
hobby.fraud$insured_hobbies[4] <- "bungjump"
hobby.fraud$insured_hobbies[19] <- "vidgames"
coul2 <- brewer.pal(length(hobby.fraud$insured_hobbies), "Paired")
barplot(hobby.fraud$fraud_reported, names.arg = hobby.fraud$insured_hobbies, col = coul2, ylab = "Prozen
```



Die Hobbykategorie “Chess/Schach” besitzt den höchsten Prozentsatz der Betrugsfaelle mit 82.61%.

Verwenden Sie das Paket gbm (Bernoulli-Verteilung), um die folgenden Modelle zur Vorhersage der betrügerischen Schäden anzupassen. Verwenden Sie die folgenden Parameter. n.trees = 500, interaction.depth = 2, shrinkage = 0.1, cv.folds = 5, n.cores = 1, verbose = FALSE

```
#Variablentypen zuordnen und GBM fitten.
fraud$incident_city <- as.factor(fraud$incident_city)
fraud$auto_make <- as.factor(fraud$auto_make)
fraud$policy_state <- as.factor(fraud$policy_state)
fraud$insured_hobbies <- as.factor(fraud$insured_hobbies)
fraud$insured_sex <- as.factor(fraud$insured_sex)
fraud$insured_education_level <- as.factor(fraud$insured_education_level)
fraud$insured_occupation <- as.factor(fraud$insured_occupation)
fraud$authorities_contacted <- as.factor(fraud$authorities_contacted)
fraud$incident_state <- as.factor(fraud$incident_state)
fraud$police_report_available <- as.factor(fraud$police_report_available)
fraud <- fraud[,-1]
library(gbm)
set.seed(1)
gbm.fit <- gbm(formula = fraud_reported ~. ,
               distribution = "bernoulli",
               data = fraud,
               n.trees = 500,
               interaction.depth = 2,
               shrinkage = 0.1,
               cv.folds = 5,
               n.cores = 1,
```

```
              verbose = FALSE)
```

```
print(gbm.fit)
```

```
## gbm(formula = fraud_reported ~ ., distribution = "bernoulli",
##     data = fraud, n.trees = 500, interaction.depth = 2, shrinkage = 0.1,
##     cv.folds = 5, verbose = FALSE, n.cores = 1)
## A gradient boosted model with bernoulli loss function.
## 500 iterations were performed.
## The best cross-validation iteration was 33.
## There were 17 predictors of which 7 had non-zero influence.
```

Für 33 Bäume wurde der kleinste CV-Fehler erreicht. Frage 7: Welche Variablen wurden in model1 ausgewaehlt? Welche ist die wichtigste?

```
set.seed(1)
model1.gbm<- gbm(formula = fraud_reported ~. ,
              distribution = "bernoulli",
              data = fraud,
              n.trees = 33,
              interaction.depth = 2,
              shrinkage = 0.1,

              n.cores = 1,
              verbose = FALSE)
```

```
summary(model1.gbm, plotit = FALSE)
```

```
##                                                 var    rel.inf
## insured_hobbies                     insured_hobbies 59.6397107
## total_claim_amount               total_claim_amount 10.5674637
## auto_make                                 auto_make  8.9576481
## insured_occupation               insured_occupation  8.8056495
## incident_state                       incident_state  4.9970201
## insured_education_level     insured_education_level  2.4624650
## authorities_contacted         authorities_contacted  2.4530338
## incident_city                         incident_city  1.1367819
## policy_annual_premium       policy_annual_premium  0.9802273
## months_as_customer             months_as_customer  0.0000000
## age                                           age  0.0000000
## policy_state                         policy_state  0.0000000
## policy_deductable               policy_deductable  0.0000000
## insured_sex                           insured_sex  0.0000000
## witnesses                               witnesses  0.0000000
## police_report_available   police_report_available  0.0000000
## number_of_vehicles_involved number_of_vehicles_involved  0.0000000
```

Es wurden 9 Variablen ausgewaehlt. Die Variablen sind:

```
summary(model1.gbm, plotit = FALSE)[!as.numeric(summary(model1.gbm, plotit = FALSE)$rel.inf) == 0,]$var
```

```
## [1] "insured_hobbies"          "total_claim_amount"
## [3] "auto_make"                "insured_occupation"
## [5] "incident_state"           "insured_education_level"
## [7] "authorities_contacted"    "incident_city"
## [9] "policy_annual_premium"
```

Die wichtigste Variable ist insured_hobbies.

Berechnen Sie unter Verwendung aller Daten (100 %) die Modellvorhersagen mit model1 (als Wahrscheinlichkeiten, type="response") und setzen Sie den Cut-Off-Punkt auf 28%. Das Model sagt also Betrug für Wahrscheinlichkeiten größer als 28% voraus. Berechnen Sie die Konfusionsmatrix (auf den 100% der Daten).

Konfusionsmatrix:

```
library(caret)
set.seed(1)
pred1 <- predict(model1.gbm, type = "response")
data.pred <- (pred1 > 0.28)^1 #TRUE^1 = 1, FALSE^1 = 0
konf1 <- confusionMatrix(factor(data.pred), factor(fraud$fraud_reported), positive = "1")$table
konf1
```

```
##           Reference
## Prediction   0   1
##          0 647 114
##          1 106 133
```

Frage 8: Wie hoch ist die Trefferquote (Accuracy rate)?

```
#Trefferquote berechnen
sum(diag(konf1)) / sum(konf1)
```

```
## [1] 0.78
```

Die Trefferquote betraegt 78%.

Wiederholen Sie dasselbe Verfahren wie oben aber jetzt mit Parametern: n.trees = 1000, interaction.depth = 3, shrinkage = 0.05, cv.folds = 5, n.cores = 1, verbose = FALSE

```
set.seed(1)
gbm.fit1 <- gbm(formula = fraud_reported ~. ,
                distribution = "bernoulli",
                data = fraud,
                n.trees = 1000,
                interaction.depth = 3,
                shrinkage = 0.05,
                cv.folds = 5,
                n.cores = 1,
                verbose = FALSE)
```

Frage 9: Für wie viele Baeume wird der kleinste cv-Fehler erreicht?

```
print(gbm.fit1)
```

```
## gbm(formula = fraud_reported ~ ., distribution = "bernoulli",
##     data = fraud, n.trees = 1000, interaction.depth = 3, shrinkage = 0.05,
##     cv.folds = 5, verbose = FALSE, n.cores = 1)
## A gradient boosted model with bernoulli loss function.
## 1000 iterations were performed.
## The best cross-validation iteration was 38.
## There were 17 predictors of which 9 had non-zero influence.
```

Für 38 Baeume wurde der kleinste CV-Fehler erreicht.

Refitten Sie nun das Modell mit dieser Anzahl von Baeumen neu zusammen und nennen Sie es model2.

```
set.seed(1)
model2.gbm <- gbm(formula = fraud_reported ~. ,
              distribution = "bernoulli",
              data = fraud,
              n.trees = 38,
              interaction.depth = 3,
              shrinkage = 0.05,

              n.cores = 1,
              verbose = FALSE)
```

Frage 10: Welche Variablen wurden in model2 ausgewaehlt? Welche ist die wichtigste?

```
summary(model2.gbm, plotit = FALSE)
```

```
##                                                 var    rel.inf
## insured_hobbies                       insured_hobbies 52.0760488
## insured_occupation                 insured_occupation 15.9476521
## total_claim_amount                 total_claim_amount  9.5738894
## auto_make                                   auto_make  7.1314186
## incident_state                         incident_state  6.0263266
## insured_education_level       insured_education_level  2.0555310
## incident_city                           incident_city  1.9282663
## months_as_customer                 months_as_customer  1.7974331
## authorities_contacted           authorities_contacted  1.7595965
## policy_annual_premium           policy_annual_premium  1.2734972
## witnesses                                   witnesses  0.4303405
## age                                               age  0.0000000
## policy_state                             policy_state  0.0000000
## policy_deductable                   policy_deductable  0.0000000
## insured_sex                               insured_sex  0.0000000
## police_report_available       police_report_available  0.0000000
## number_of_vehicles_involved number_of_vehicles_involved  0.0000000
```

Die Variablen sind:

```
summary(model1.gbm, plotit = FALSE)[!as.numeric(summary(model1.gbm, plotit = FALSE)$rel.inf) == 0,]$var
```

```
## [1] "insured_hobbies"         "total_claim_amount"
## [3] "auto_make"               "insured_occupation"
## [5] "incident_state"          "insured_education_level"
## [7] "authorities_contacted"   "incident_city"
## [9] "policy_annual_premium"
```

Die 2 wichtigste Variablen sind insured_hobbies und insured_occupation.

Wie oben, berechnen Sie die Konfusionsmatrix.

```
library(caret)#Konfusionsmatrix
set.seed(1)
pred2 <- predict(model2.gbm, type = "response")
data.pred2 <- (pred2 > 0.28)^1 #TRUE^1 = 1, FALSE^1 = 0
konf2 <- confusionMatrix(factor(data.pred2), factor(fraud$fraud_reported), positive = "1")$table
```

Konfusionsmatrix:

```
konf2
```

```
##           Reference
## Prediction   0   1
##          0 669 110
##          1  84 137
```

Frage 11: Wie hoch ist die Trefferquote (Accuracy rate) mit model2?

```
sum(diag(konf2)) / sum(konf2)
```

```
## [1] 0.806
```

Die Trefferquote betraegt 80.6%.