Data Science

Group 6

University of Southern Denmark

Report

Niels Alexander Hvid
nhvid13@student.sdu.dk

Matic Novak
maticnova@gmail.com

Lucas Østergaard Jensen
lujen14@student.sdu.dk

Simon Hjortshøj Larsen
sila114@student.sdu.dk

Mads Stagelund Mogensen
mmoge14@student.sdu.dk

December 2017

1 Alcohol related car accidents

At the beginning, we prepared the received data to use it in the language *R*. We exported the .xlsx file to a .csv file. All categorical values were replaced with numbers and used as a factor:

- Gender: Male = 1; Female = 0
- Accident: Yes = 1; No = 0
- Socioeconomic_status: Lower = 0; Middle = 1; Upper = 2

We also removed the column *Subject* because it does not bring any value to the data - it is not a meaningful attribute.

Question 1

The three other variables given apart from BAC are age, gender and socioeconomic status. The likelihood that a certain group has in getting into an accident could be seen bellow.

$$\begin{gathered}
\text{Age} <= 40 : \frac{22}{125} = 0.18 \\
\text{Age} > 40 : \frac{35}{71} = 0.49 \\
\text{Females} : \frac{11}{85} = 0.13 \\
\text{Males} : \frac{46}{111} = 0.41 \\
\text{Lower socioeconomic status} : \frac{19}{70} = 0.27 \\
\text{Middle socioeconomic status} : \frac{14}{49} = 0.29 \\
\text{Upper socioeconomic status} : \frac{24}{77} = 0.31
\end{gathered} \tag{1}$$

This shows that males are more than 3 times as likely to be in an accident compared to females, and people above 40 are more than twice as likely to be in an accident compared to people below 40.

This suggests that for gender, male is a risk factor while female is a protective factor. With age, people at 41 or older is a risk factor while 40 or below is a protective factor.

Question 2 and 4

After loading the data in R, we used the GLM function to get the logistics models. The crude model was given the BAC data, while the adjusted model was given all the data. We then extracted the betas and standard errors to calculate OR, higher, and lower bounds. The code for this can be seen in appendix 1.

The plots for crude and adjusted models on a linear scale can be seen on figure 1 and 2 respectively along with the adjusted model on a logarithmic scale on figure 3.

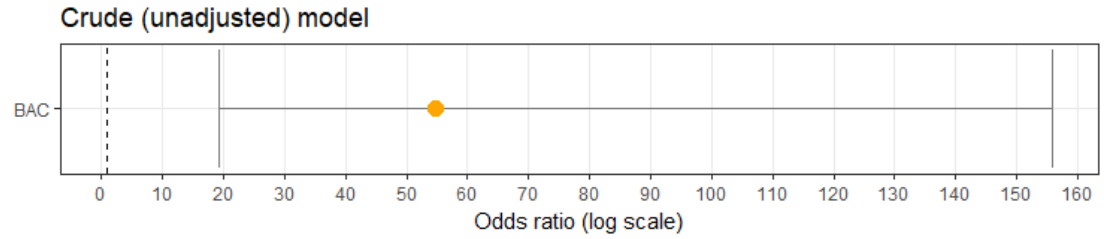

Figure 1: Crude model (linear scale)

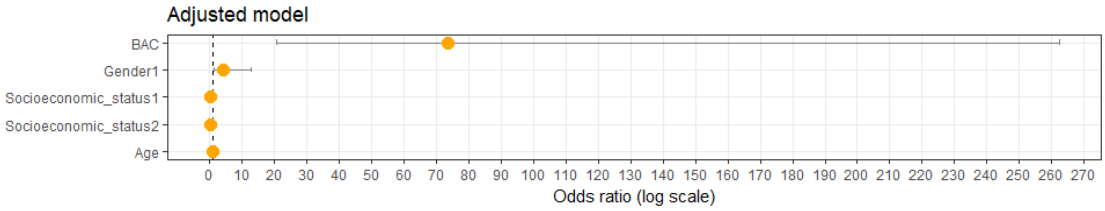

Figure 2: Adjusted model (linear scale)

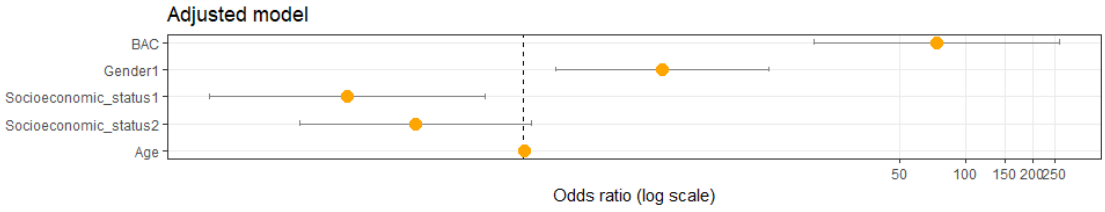

Figure 3: Adjusted model (logarithmic scale)

The adjusted model indicates that BAC has more significance than the crude model indicates. This is interesting as it is easy to make the assumption that other factors such as age and gender would decrease the significance of BAC. Males are also more likely than females to be in an accident. The socioeconomic status does not have a significant association with accidents.

The OR of the adjusted model for BAC is 73 while the unadjusted is 54.

In question 1 we got the impression that age could be a risk factor, but the model predicts that age has little impact on the probability of crashing. We made a scatterplot of Age versus BAC to see if this was really true.

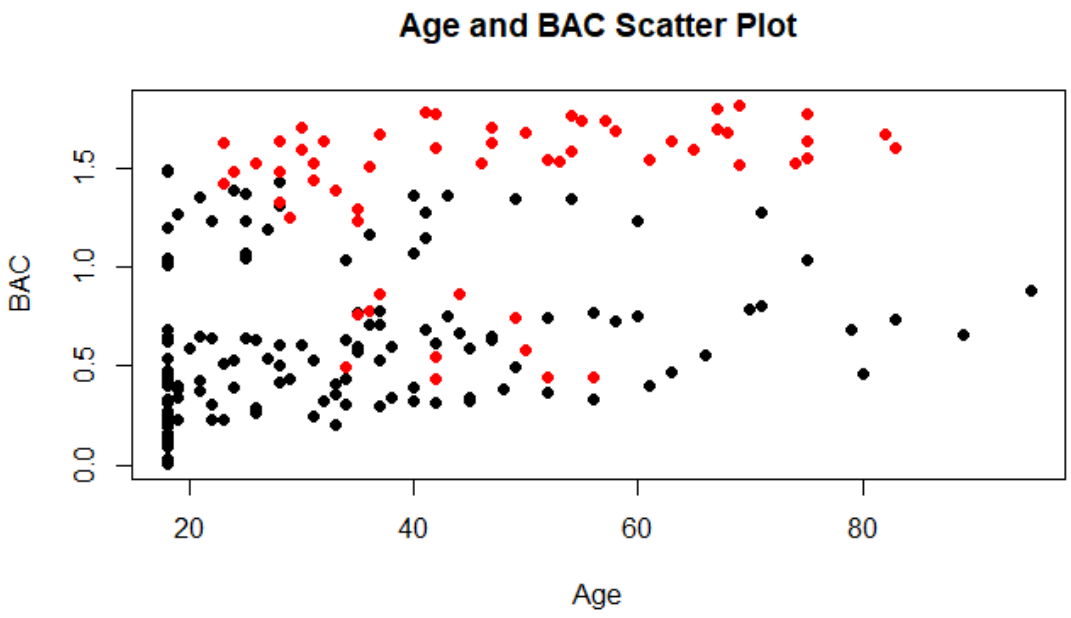

Figure 4: A plot of age versus BAC. Red dots are accidents.

It can be seen that higher age does not necessarily correspond to higher probability of crashing. We can also see that there are many samples of 18 year olds with low BAC that did not crash, which could explain our initial thoughts in question 1.

Question 3

It's a recurring problem in Denmark that a high number of people choose to drive with a high BAC during Christmas. Especially when going home from Christmas dinners which are notoriously known for being celebrated with a high alcohol consumption. On top of the higher BAC, the Christmas dinners are held during the winter where the roads are more likely to be slippery hence more accidents could happen, not because of high BAC, but because of poor road conditions. Other holidays or festivity seasons could likewise influence the amount of accidents. One way to circumvent this is by including the season in which the drive took place.

Another possible confounding variable is drugs. Some drugs have a much more significant impact on cognitive abilities than alcohol and people who take drugs likely consumes alcohol at the same time. A false correlation between alcohol and traffic accidents could be found. This could be accounted for by including what types of drugs (if any) were taken when the drive took place.

The drugs could be one of the confounders because they fulfill these three conditions:

- People with higher BAC are more likely to take drugs than nondrinkers.
- Drugs are a risk factor for the accidents, even for the people with low BAC (or who do not drink).
- Drugs are not an intermediate factor (i.e. BAC or drinking does not cause taking drugs).

Question 5

The model gave the following predictions for a man with a BAC of 1.0%:

- 40 years old: 59.5%
- 50 years old: 63.5%
- 60 years old: 67.3%
- 70 years old: 71%
- 80 years old: 74.3%

For each 10 year that passes the probability increases by roughly 3.8 percentage points. Although the change is almost linear, it decreases slowly. This makes sense as the probability would otherwise eventually exceed 100% which of course is not possible.

Question 6

- Accuracy: 76.47%
- Sensitivity: 88.89%
- Specificity: 62.50%
- Precision: 72.73%

	Actual NO	Actual YES
Predicted NO	8	3
Predicted YES	1	5

Table 1: Confusion Matrix

The performance of the model is decent but not super good. This is both because of the small training set *and* the small validation set. Maybe it's also because of the aforementioned confounding variables.

2 LDA, k-NN, GLM

The *Boston* dataset is used for this part of the assignment. The median crime rate is calculated and applied as a class to each row. Rows with crime rate above the median are classified with class 1 and those below the median with class 0. The crime rate column is then removed from the dataset and all data except the class is scaled. A seed is chosen so the results are reproducible. The dataset is divided into training (75%) and testing (25%) sets. For training the rows with either class are equally represented.

Question 1a

The Boston dataset comprises 14 variables. Each one of them has a short description as listed below:

crim	per capita crime rate by town.
zn	proportion of residential land zoned for lots over 25,000 sq.ft.
indus	proportion of non-retail business acres per town.
chas	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
nox	nitrogen oxides concentration (parts per 10 million).
rm	average number of rooms per dwelling.
age	proportion of owner-occupied units built prior to 1940.
dis	weighted mean of distances to five Boston employment centres.
rad	index of accessibility to radial highways.
tax	full-value property-tax rate per $10,000.
ptratio	pupil-teacherratio by town.
black	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
lstat	lower status of the population (percent).
medv	median value of owner-occupied homes in $1000s.

Table 2: Boston dataset variable description as found 12/12-2017 on https://www.cs.toronto.edu/ delve/data/boston/bostonDetail.html

Question 1b

Looking at the correlation matrix we find that certain variables have high correlation.

The correlation between *nox* and *indus* was 0.778. This makes sense, as many industries release more nitrogen oxide gasses than the average household.

The correlation between *indus* and *tax* was 0.774. This could indicate that the government in the area enforces higher property taxes on industrial properties.

As a result there is also a pretty high correlation between *nox* and *tax* (of 0.707). To make a subset, it would make sense to remove *nox* and *tax*, as *indus* is so highly

correlated with both of them.

Question 1c

The following table shows the accuracy of the different models. Both the test set and validation set is shown as well as the accuracy for the test set of the subset. The test error rates can simply be calculated as $1 - accuracy$.

	Logistic Regression	LDA	k-NN
Test set	87.67%	85.95%	87.80%
Test subset	84.76%	83.12%	85.38%
Validation set	92.86%	86.51%	90.48%

Table 3: Accuracy for the different models

The Logistic regression performed the best closely followed by k-NN.

	Logistic Regression	LDA	k-NN
Sensitivity	92.96%	95.24%	85.71%
Specificity	93.65%	77.78%	90.48%
Precision	92.06%	93.55%	90.00%

Table 4: performance of the different models

The performances shown in table 4 were found using the validation set.

By performing calculations of odds ratios and boundaries of the attributes in logistic regression, we get the impression that especially *dis* and *rad* have high influence on the crime rate.

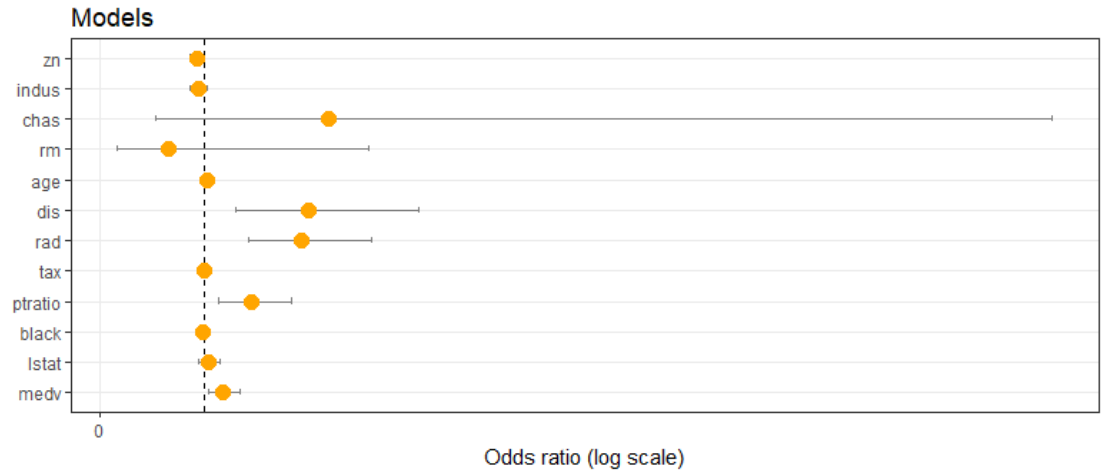

Figure 5: Odds ratios with boundaries

Question 2a

$$\beta_0 = -6, \beta = \begin{bmatrix} 0.05 \\ 1 \end{bmatrix}$$
$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{matrix} \rightarrow \text{time} \\ \rightarrow \text{grade point average} \end{matrix}$$

$$x_1 = 40 \text{ hours} \; ; \; x_2 = 3.5 \text{ grade} => \text{scoring A}$$

$$\begin{aligned} \ln(\frac{p}{1-p}) &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \\ \ln(\frac{p}{1-p}) &= 6 + 0.05 \cdot 40 + 1 \cdot 3.5 = -0.5 \end{aligned} \tag{2}$$

$$\begin{aligned} \frac{p}{1-p} &= e^{-0.5} \\ p &= (1-p) \cdot 0.6 \\ p &= 0.6 - 0.6p \\ p + 0.6p &= 0.6 \\ p = \frac{0.6}{1.6} &= 0.377 = \underline{\underline{37.8\%}} \end{aligned}$$

Figure 6: The probability that a student working 40 hours with a C^+ (3.5) average grade scores an A (5) in the finals.

Question 2b

$$
\begin{aligned}
p &= 80\% \\
\ln(\frac{p}{1-p}) &= -6 + 0.05 \cdot H + 1 \cdot 3.5 \\
\ln(\frac{0.8}{0.2}) &= -2.5 + 0.05H \\
1.38 &= -2.5 + 0.05H \\
3.88 &= 0.05H \\
H &= \underline{\underline{77.73}} \text{ Hours per week}
\end{aligned}
\tag{3}
$$

Figure 7: The number of hours the student must spend per week to have an 80% probability of scoring an A.

Test:

$$
\begin{aligned}
\ln(\frac{p}{1-p}) &= -6 + 0.05 \cdot 77.73 + 3.5 \\
\ln(\frac{p}{1-p}) &= 1.38 \\
\frac{p}{1-p} &= e^{1.38} \\
p &= 4(1-p) \\
p &= 4 - 4p \\
5p &= 4 \\
\underline{\underline{p &= 0.8}}
\end{aligned}
\tag{4}
$$

Figure 8: Test to confirm the 80% probability of scoring an A.

3 Artist Identification

Paintings from the different artist from the *impresionism* period were downloaded from *www.wikiart.org*. The different artists drew different amount of paintings:

- Edouard Manet: 121
- Edgar Degas: 605
- Pierre-Auguste Renoir: 1377
- Claude Monet: 1324

To train a network to answer "Is this painting the work of x?" four different networks has to be trained, one for each artist. The networks are all made using Matlab with these steps:

1. Load paintings of the artist to be detected.
2. Load same amount of paintings spread across the 3 other artists.
3. Load the pre-trained CNN *AlexNet*.
4. Resize all paintings to 227 by 227 to fit *AlexNet*.
5. Divide the dataset into a training set and a test set (75% and 25% respectively).
6. Extract features of training and test datasets from the layer before the classifier in *AlexNet*.
7. Train an SVM classifier using the extracted features.
8. Predict the labels of the test set to evaluate the performance of the classifier.

The amount of paintings vary from artist to artist. Manet only has 121 paintings. So the network designed to detect Manets paintings is trained on a dataset consisting of just 242 paintings (50% from Manet and 50% from others). The other networks can use datasets of size 726.

After the 4 networks finished training, we run the 4 paintings through all the networks resulting in the following response. We used Google reverse image search tool to find the real author of the paintings.

	Manet network	Degas network	Renior network	Monet network
Painting 1	**No**	**No**	**Yes**	**No**
Painting 2	**No**	**No**	**No**	*Yes*
Painting 3	**Yes**	**No**	**No**	**No**
Painting 4	*Yes*	**Yes**	**No**	**No**

Table 5: Network response, bold text is correct response

The accuracy of the networks could likely be increased by increasing the size of the datasets used to train the networks. Furthermore another pretrained network instead of *AlexNet* could be used which could result in a better performance (GoogleNet, ResNet50...).

	Manet network	Degas network	Renior network	Monet network
Accuracy	85%	86%	86%	90%

Table 6: Accuracy of the networks on the test set

Figure 9: (P1) Pierre Auguste Renoir

Figure 10: (P2) Jackson Pollock

Figure 11: (P3) Manet

Figure 12: (P4) Edgar Degas

Appendix 1 - Car accidents

```
library(car)
library("caret")
library("ggplot2")

# Read and prepare the data.
filePath = "Data_Car_accidents_196.csv"
data <- read.csv(file=filePath, header=TRUE, sep=";", stringsAsFactors = TRUE)
data <- data[,2:6]  # remove IDs
data$Gender <- as.factor(data$Gender)
data$Accident <- as.factor(data$Accident)
data$Socioeconomic_status <- as.factor(data$Socioeconomic_status)
summary(data)

plot(data$Age, data$BAC, main="Age and BAC Scatter Plot",
     xlab="Age", ylab="BAC", col=data$Accident, pch=19)


### METHODS
getBetas <- function(logistic_model) {
  betas <- c()
  for (i in 1:length(logistic_model$coefficients)) {
    betas <- c(betas, logistic_model$coefficients[i])
  }
  return (betas)
}

getOddsRatios <- function(betas) {
  oddsRatios <- c()
  for (i in 1:length(betas)) {
    oddsRatios <- c(oddsRatios, exp(betas[i]))
  }
  return (oddsRatios)
}

getStandardErrors <- function(logistic_model) {
  standardErrors <- c()
  for (i in 1:length(summary(logistic_model)$coefficients[, 2])) {
```

```
    standardErrors <- c(standardErrors, summary(logistic_model)
    $coefficients[, 2][i])
  }
  return (standardErrors)
}

getBounds <- function(oddsRatios, betas, standardErrors, confidence_number = 1.96)
{
  bounds <- data.frame()
  for (i in 1:length(betas)) {
    low_bound <- exp(betas[i] - confidence_number * standardErrors[i])
    high_bound <- exp(betas[i] + confidence_number * standardErrors[i])
    entry <- data.frame(OR = oddsRatios[i], LB = low_bound, HB = high_bound, beta
    bounds <- rbind(bounds, entry)
  }
  return (bounds)
}

plotBounds <- function(bounds, numf_of_columns_from_bounds = 3) {
  zeros <- rep(0, numf_of_columns_from_bounds)
  num_of_plots <- nrow(bounds)
  attach(mtcars)
  par(mfrow=c(num_of_plots, 1)) # number of rows and columns on the same graph

  for (i in 1:nrow(bounds)) {
    dots <- as.numeric(bounds[i,1:numf_of_columns_from_bounds])
    plot(dots, zeros, main=rownames(bounds)[i])
    #boxplot(dots)
  }
}

forestPlot <- function(df, boxLabels, yAxis, modelTitle) {
  #dev.off()
  p <- ggplot(df, aes(x = boxOdds, y = yAxis))
  p + geom_vline(aes(xintercept = 1), size = .25, linetype = "dashed") +
    geom_errorbarh(aes(xmax = boxCIHigh, xmin = boxCILow), size = .5, height = .2,
    color = "gray50") +
    geom_point(size = 3.5, color = "orange") +
    theme_bw() +
```

```
    theme(panel.grid.minor = element_blank()) +
    scale_y_continuous(breaks = yAxis, labels = boxLabels) +
    scale_x_continuous(breaks = seq(0,270,10) ) +
    #coord_trans(x = "log10") +
    ylab("") +
    xlab("Odds ratio (log scale)") +
    #annotate(geom = "text", y =1.1, x = 3.5, label ="Model p < 0.001\nPseudo
    R^2 = 0.10", size = 3.5, hjust = 0) +
    ggtitle(modelTitle)
}
###


### 1) Identify demographic characteristics of the drivers that are risk
###    (or protective) factors of car accidents.
onlyAccidents <- data[data$Acciden == 1, ]

gender <- data[data$Gender == 1, ]
genderAcc <- onlyAccidents[onlyAccidents$Gender == 1, ]
nrow(genderAcc)/nrow(gender)

genderF <- data[data$Gender == 0, ]
genderFAcc <- onlyAccidents[onlyAccidents$Gender == 0, ]
nrow(genderFAcc)/nrow(genderF)

ageLessThan41 <- data[data$Age < 41, ]
ageLessThan41Acc <- onlyAccidents[onlyAccidents$Age < 41, ]
nrow(ageLessThan41Acc)/nrow(ageLessThan41)

ageMoreThan40 <- data[data$Age > 40, ]
ageMoreThan40Acc <- onlyAccidents[onlyAccidents$Age > 40, ]
nrow(ageMoreThan40Acc)/nrow(ageMoreThan40)

status0 <- data[data$Socioeconomic_status == 0, ]
status0Acc <- status0[status0$Accident == 1, ]
nrow(status0Acc)/nrow(status0)

status1 <- data[data$Socioeconomic_status == 1, ]
status1Acc <- status1[status1$Accident == 1, ]
```

```r
nrow(status1Acc)/nrow(status1)

status2 <- data[data$Socioeconomic_status == 2, ]
status2Acc <- status2[status2$Accident == 1, ]
nrow(status2Acc)/nrow(status2)


### 2) Obtain the model relating BAC and car accidents.
### a) Crude (not adjusted) OR's.
lmBac <- glm(Accident ~ BAC, family=binomial(logit), data=data)
summary(lmBac)
betasBac <- getBetas(lmBac)
oddsRatiosBac <- getOddsRatios(betasBac)
standardErrorsBac <- getStandardErrors(lmBac)
boundsBac <- getBounds(oddsRatiosBac, betasBac, standardErrorsBac)
plotBounds(boundsBac)

dev.off()
boxLabels = c("BAC")
yAxis = length(boxLabels):1
df <- data.frame(
  yAxis = length(boxLabels):1,
  boxOdds = c(boundsBac[2,1]),
  boxCILow = c(boundsBac[2,2]),
  boxCIHigh = c(boundsBac[2,3])
)
forestPlot(df, boxLabels, yAxis, "Crude (unadjusted) model")


### b) Adjusted OR's.
lmAdj <- glm(Accident ~ BAC + Gender + Socioeconomic_status + Age, family=binomial
summary(lmAdj)
betasAdj <- getBetas(lmAdj)
oddsRatiosAdj <- getOddsRatios(betasAdj)
standardErrorsAdj <- getStandardErrors(lmAdj)
boundsAdj <- getBounds(oddsRatiosAdj, betasAdj, standardErrorsAdj)
plotBounds(boundsAdj)

dev.off()
```

```r
boxLabels = c("BAC", "Gender1", "Socioeconomic_status1", "Socioeconomic_status2",
yAxis = length(boxLabels):1
df <- data.frame(
  yAxis = length(boxLabels):1,
  boxOdds = c(boundsAdj[2,1],boundsAdj[3,1],boundsAdj[4,1],boundsAdj[5,1],
  boundsAdj[6,1]),
  boxCILow = c(boundsAdj[2,2],boundsAdj[3,2],boundsAdj[4,2],boundsAdj[5,2],
  boundsAdj[6,2]),
  boxCIHigh = c(boundsAdj[2,3],boundsAdj[3,3],boundsAdj[4,3],boundsAdj[5,3],
  boundsAdj[6,3])
)
forestPlot(df, boxLabels, yAxis, "Adjusted model")


### 5) What is the probability that a 40 yr male whose BAC is >1%, causes a car ac
###    What will be the probability, 10, 20, 30 and 40 years later? Is this change
linear?
probabilityOfX <- function(BAC, Gender, Age, betas) {
  topFraction <- exp(betas[1] + betas[2] * BAC + betas[3] * Gender + betas[6] *
  Age)
  return (topFraction / (1 + topFraction))
}
p <- probabilityOfX(1, 1, 40, betasAdj)
# p = 0.595
# BetaAge = 1.017061034 --> exp(BetaAge) = 2.765056
# Every year increase in age, makes response being true more likely with
factor 2.765056.

p10 <- probabilityOfX(1, 1, 50, betasAdj)
p20 <- probabilityOfX(1, 1, 60, betasAdj)
p30 <- probabilityOfX(1, 1, 70, betasAdj)
p40 <- probabilityOfX(1, 1, 80, betasAdj)


### 6) Evaluate the predictive performance of the model.
testFilePath = "Data_Car_accidents_17.csv"
testData <- read.csv(file=testFilePath, header=TRUE, sep=";",
stringsAsFactors = TRUE)
testLabels <- testData[,3]
```

```r
testData <- testData[,c("Gender", "Age", "Socioeconomic_status", "BAC")]
testData$Gender <- as.factor(testData$Gender)
testData$Socioeconomic_status <- as.factor(testData$Socioeconomic_status)
summary(testData)
predicted <- predict(lmAdj, testData, type="response")

classFromProbabilities <- function(x) {
  if (x > 0.5) return (1)
  else return (0)
}

predictedClass <- sapply(predicted, classFromProbabilities)
performanceTable <- table(predictedClass, testLabels)
confussionMatrix <- confusionMatrix(performanceTable)

confussionMatrix
# precision = TP / (TP + FP)
```

Appendix 2 - Crime Rate

```
library(ISLR)
library(caret)
library(car)
library(MASS)
set.seed(7)
data <- Boston

# Calculate median crime rate and set class to each row regarding
# if it is above or below median crime rate. Scale the orher data.
median_crim_rate <- median(data$crim)
classFromCrimRate <- function(x) {
  if (x > median_crim_rate) return (1)
  else return (0)
}
calculatedClass <- sapply(data$crim, classFromCrimRate)
# data <- data.frame(scale(data)) #Scaling is done before learning by method train
data$Class <- calculatedClass
data$Class <- as.factor(data$Class)
data <- data[,-1]
nrow(data[data$Class == 1,])


# Prepare training and test set.
TrainingDataIndex <- createDataPartition(data$Class, p=0.75, list=FALSE)
trainingData <- data[TrainingDataIndex,]
testData <- data[-TrainingDataIndex,]
trainingLabels <- trainingData$Class
testLabels <- testData$Class
testDataWithoutClass <- testData[,-14]
prop.table(table(trainingLabels))


## Find the correlations between variables.
nearZeroVariables <- nearZeroVar(trainingData, saveMetrics = TRUE)
correlations <- cor(trainingData[,-14])
highCorrelations <- findCorrelation(correlations, cutoff = 0.75)

```

```
trainingDataCorr <- trainingData[,-drop(c(4,9))]
testDataCorr <- testData[,-drop(c(4,9))]
testDataCorrWithoutClass <- testDataCorr[,-12]


# Logistic Regression
lr_model_all <- train(Class~., data=trainingData,
                      method='glm', family=binomial(link='logit'),
                      preProcess=c('scale', 'center'))
lr_pred_all <- predict(lr_model_all, testData[,-14])
confusionMatrix(lr_pred_all, testData$Class)

lr_model_corr <- train(Class ~ rad + nox,
                       data=trainingData,
                       method='glm', family=binomial(link='logit'),
                       preProcess=c('scale', 'center'))
lr_pred_corr <- predict(lr_model_corr, testData[,-14])
confusionMatrix(lr_pred_corr, testData$Class)
vif(lr_model_corr$finalModel)


# LDA
LDA_model_all <- train(Class~., data=trainingData,
                       method='lda',
                       preProcess=c('scale', 'center'))
LDA_pred_all <- predict(LDA_model_all, testData[,-14])
confusionMatrix(LDA_pred_all, testData$Class)

LDA_model_corr <- train(Class~., data=trainingDataCorr,
              method='lda',
              preProcess=c('scale', 'center'))
LDA_pred_corr <- predict(LDA_model_corr, testDataCorr[,-12])
confusionMatrix(LDA_pred_corr, testDataCorr$Class)


# k-NN
knnGrid <- expand.grid(.k=c(2))
knn_model_all <- train(x=trainingData[,-14], method='knn',
                       y=trainingData$Class,
```

```
                        preProcess=c('center', 'scale'),
                        tuneGrid = knnGrid)
knn_model_corr <- train(x=trainingDataCorr[,-12], method='knn',
             y=trainingDataCorr$Class,
             preProcess=c('center', 'scale'),
             tuneGrid = knnGrid)
knn_pred <- predict(knn_model_all, testData[,-14])
confusionMatrix(knn_pred, testLabels)


# Plot the odds ratios and boundaries for logistic regression.
betasBac <- getBetas(summary(lr_model_all))
oddsRatiosBac <- getOddsRatios(betasBac)
standardErrorsBac <- getStandardErrors(lr_model_all)
boundsBac <- getBounds(oddsRatiosBac, betasBac, standardErrorsBac)
boundsBac<-boundsBac[!(boundsBac$OR > 10000),]

dev.off()
boxLabels = c("zn", "indus", "chas", "rm", "age", "dis", "rad", "tax", "ptratio",
yAxis = length(boxLabels):1
df <- data.frame(
  yAxis = length(boxLabels):1,
  boxOdds = c(boundsBac[2,1],boundsBac[3,1],boundsBac[4,1],boundsBac[5,1],
  boundsBac[6,1],boundsBac[7,1],boundsBac[8,1],boundsBac[9,1],
  boundsBac[10,1],boundsBac[11,1],boundsBac[12,1],boundsBac[13,1]),
  boxCILow = c(boundsBac[2,2],boundsBac[3,2],boundsBac[4,2],boundsBac[5,2],
  boundsBac[6,2],boundsBac[7,2],boundsBac[8,2],boundsBac[9,2],
  boundsBac[10,2],boundsBac[11,2],boundsBac[12,2],boundsBac[13,2]),
  boxCIHigh = c(boundsBac[2,3],boundsBac[3,3],boundsBac[4,3],boundsBac[5,3],
  boundsBac[6,3],boundsBac[7,3],boundsBac[8,3],boundsBac[9,3],
  boundsBac[10,3],boundsBac[11,3],boundsBac[12,3],boundsBac[13,3])
)
forestPlot(df, boxLabels, yAxis, "Models")
```

Appendix 3 - Artist Identification

```
%% Load Images
rootFolder = fullfile(pwd, 'images');
categories = {'degas', 'notDegas'};
% ImageDatastore operates on image file locations and help you manage the data.
images = imageDatastore(fullfile(rootFolder, categories),
'LabelSource', 'foldernames');
% Summarize the number of images per category.
numOfImagesPerCategory = countEachLabel(images)


%% Balance the number of images in the training set
% Determine the smallest amount of images in a category.
minNumOfImagesPerCategory = min(numOfImagesPerCategory{:,2});
% Use splitEachLabel method to trim the set.
images = splitEachLabel(images, minNumOfImagesPerCategory, 'randomize');
% Notice that each set now has exactly the same number of images.
countEachLabel(images)


%% Load Pre-trained AlexNet Network
net = alexnet()

%% Pre-process Images For CNN
% net can only process RGB images that are 227-by-227.
% imds.ReadFcn pre-process images on-the-fly.
% The imds.ReadFcn is called every time an image is read from the ImageDatastore.
images.ReadFcn = @(filename)readAndPreprocessImage(filename);


%% Prepare Training and Test Image Sets
% Pick 75% of images from each set for the training data and the remainder,
% 25%, for the validation data. Randomize the split to avoid biasing the results.
[train, test] = splitEachLabel(images, 0.75, 'randomize');


%% Extract Training Features Using CNN
% Extract features from one of the deeper layers using the activations method.
```

```
% Layer right before the classification layer.
% In net, this layer is named 'fc7'.
featureLayer = 'fc7';
trainingFeatures = activations(net, train, featureLayer, ...
    'MiniBatchSize', 32, 'OutputAs', 'columns');


%% Train A Multiclass SVM Classifier Using CNN Features
% Use the CNN image features to train a multiclass SVM classifier.

% Get training labels from the trainingSet.
trainLabels = train.Labels;

% Train multiclass SVM classifier using a fast linear solver, and set
% 'ObservationsIn' to 'columns' to match the arrangement used for training
% features.
classifier = fitcecoc(trainingFeatures, trainLabels, ...
    'Learners', 'Linear', 'Coding', 'onevsall', 'ObservationsIn', 'columns');


%% Evaluate Classifier
% Extract image features from test set using the CNN.
testFeatures = activations(net, test, featureLayer, 'MiniBatchSize', 32);

% Pass CNN image features to trained classifier.
[predictedLabels] = predict(classifier, testFeatures);

% Get the known labels.
testLabels = test.Labels;

% Tabulate the results using a confusion matrix.
confusionMatrix = confusionmat(testLabels, predictedLabels);

% Convert confusion matrix into percentage form.
confusionMatrix = bsxfun(@rdivide, confusionMatrix, sum(confusionMatrix, 2))

% Display the mean accuracy.
mean(diag(confusionMatrix))

```

```
77
78  %% Try the Newly Trained Classifier on "Validation" Images
79
80  % Load test images
81  testImagesFolder = fullfile(rootFolder, 'testImages');
82  d = dir([testImagesFolder, '\*.bmp']);
83  fileNames = char(d.name);
84
85  % Classify all files.
86  for idx = 1:length(d)
87      newImage = fullfile(testImagesFolder, fileNames(idx, :))
88
89      % Pre-process the images as required for the CNN.
90      img = readAndPreprocessImage(newImage);
91
92      % Extract image features using the CNN.
93      imageFeatures = activations(net, img, featureLayer);
94
95      % Make a prediction using the classifier.
96      label = predict(classifier, imageFeatures)
97  end
98
99
100 %% Helper Functions
101 function preparedImage = readAndPreprocessImage(file)
102     img = imread(file);
103
104     % Some images may be grayscale.
105     % Replicate the image 3 times to create an RGB image.
106     if ismatrix(img)
107         img = cat(3, img, img, img);
108     end
109
110     % Resize the image as required for the CNN.
111     preparedImage = imresize(img, [227 227]);
112 end
```