TDDE01 - Lab 2

David Forslöf - davfo018, assignment 2
Martin Friberg - marfr370, assignment 3
Filip Frenning - filfr933, assignment 1

2020/12/01

1 Assignment 1. LDA and logistic regression

1.1 Task 1

The Sepal Length and Sepal Width from dataset Iris was made into a scatterplot as seen here:

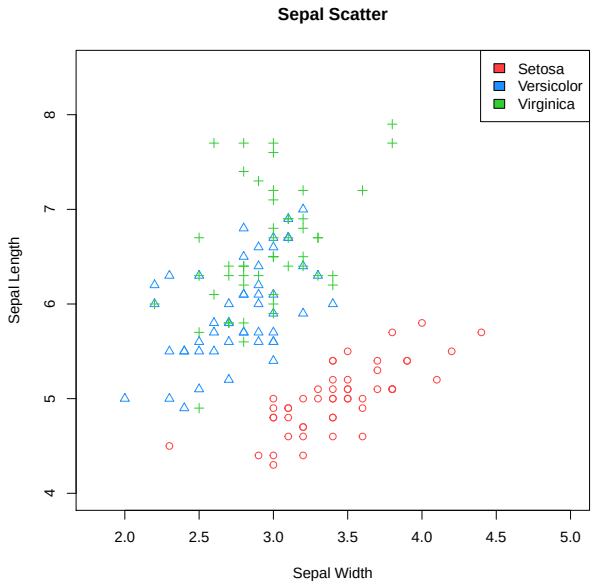

Figure 1: Scatterplot of Sepal Length and Sepal Width

The data is both easy and hard to classify with Linear Discriminant Analysis. We want to reduce the dimension of the data and by the use of some line have the datapoints separate from each other. It seems that "Setosa" is easy to separate from the other two species but seeing as the data of "Versicolor" and "Virginica" are mixed, it would be quite difficult to completely separate and classify the data with Linear Discriminant Analysis. It could on the other hand be done to some degree.

1.2 Task 2

1.2.1 Setosa

For the class Setosa the mean matrix was computed to being:

	Setosa
Mean Sepal Length	5.006
Mean Sepal Width	3.428

The Setosa covariance matrix was calculated using the cov() function and is seen below:

	Sepal Length	Sepal Width
Sepal Length	0.1242490	0.0992163
Sepal Width	0.0992163	0.1436898

1.2.2 Versicolor

For the class Versicolor the mean matrix was computed to being:

	Versicolor
Mean Sepal Length	5.936
Mean Sepal Width	2.770

The Versicolor covariance matrix was calculated using the cov() function and is seen below:

	Sepal Length	Sepal Width
Sepal Length	0.2664326	0.0851837
Sepal Width	0.0851837	0.0984694

1.2.3 Virginica

For the class Virginica the mean matrix was computed to being:

	Virginica
Mean Sepal Length	6.588
Mean Sepal Width	2.974

The Virginica covariance matrix was calculated using the cov() function and is seen below:

	Sepal Length	Sepal Width
Sepal Length	0.4043429	0.0937633
Sepal Width	0.0937633	0.1040041

For all of the classes the prior probabilities were 50/150.

1.2.4 Pooled Covariance

The pooled covariance was computed as a combination of the classes own covariance matrices and the result was:

	Sepal Length	Sepal Width
Sepal Length	0.2650082	0.0927211
Sepal Width	0.0927211	0.1153878

1.2.5 Probabilistic Model

The probabilistic model for the LDA is as seen in the lectures:

$$X|Y = C_i, \mu_i, \Sigma \sim N(\mu_i, \Sigma)$$

where

$$Y|\pi \sim Multinomial(\pi_1, ..., \pi_k)$$

1.2.6 Decision boundaries

By computing the discriminant functions for each class k with the function

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k$$

we could compute the decision boundaries between the classes by equalling the descriminant functions for two classes. The sigma is the same for all classes since it is the pooled covariance matrix.

Setosa and Versicolor:

$$x^T\Sigma^{-1}\mu_{setosa} - \frac{1}{2}\mu_{setosa}^T\Sigma^{-1}\mu_{setosa} + log\pi_{setosa} = x^T\Sigma^{-1}\mu_{versicolor} - \frac{1}{2}\mu_{versicolor}^T\Sigma^{-1}\mu_{versicolor} + log\pi_{versicolor}$$

Setosa and Virginica:

$$x^T\Sigma^{-1}\mu_{setosa} - \frac{1}{2}\mu_{setosa}^T\Sigma^{-1}\mu_{setosa} + log\pi_{setosa} = x^T\Sigma^{-1}\mu_{virginica} - \frac{1}{2}\mu_{virginica}^T\Sigma^{-1}\mu_{virginica} + log\pi_{virginica}$$

Virginica and Versicolor:

$$x^T\Sigma^{-1}\mu_{virginica} - \frac{1}{2}\mu_{virginica}^T\Sigma^{-1}\mu_{virginica} + log\pi_{virginica} = x^T\Sigma^{-1}\mu_{versicolor} - \frac{1}{2}\mu_{versicolor}^T\Sigma^{-1}\mu_{versicolor} + log\pi_{versicolor}$$

In the LDA we assume that the covariance matrix is identical for each of the classes k. If we make this assumption the classifier becomes linear. In our case the covariance matrices do not seem to fulfill our assumptions since the class-specific covariance matrix for each class is generally different from another. That is why we compute the pooled covariance matrix.

1.3 Task 3

After comparing the discriminant functions for the tree classes for each datapoint, we get the following scatterplot.

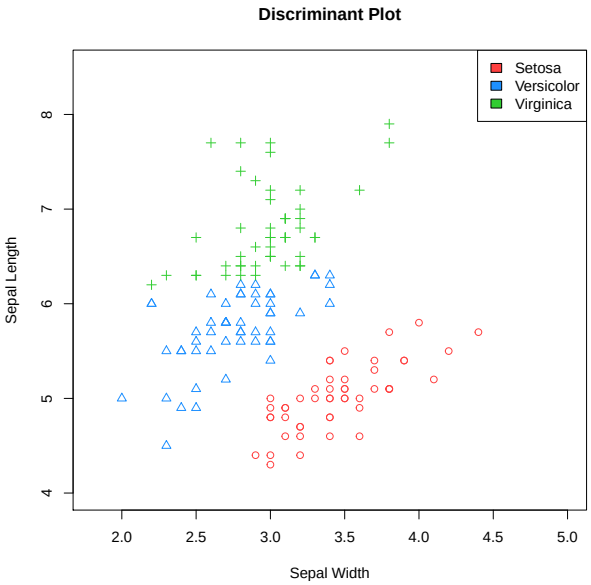

Figure 2: Scatterplot of Sepal Length and Sepal Width. LDA.

A confusion matrix for the prediction was computed and the misclassification rate was estimated to 0.2. Our prediction therefore had a 20% error rate which is quite large and the prediction is definitely not perfect. Through the confusion matrix we also see that the LDA predicted all of the Setosa correctly except for one datapoint and the rest of the misclassifications were split evenly between Versicolor and Virginica. This shows some indication of a good prediction that at the same time has room for improvement.

After using the predefined lda function in r we obtained the exact same confusion matrix, as well as the same misclassification rate of 0.2. This is expected since both methods should be the same type of Linear Discriminant Analysis. What this also does is provide some legitimacy to our calculations and shows that we did the correct predictions and assumptions.

1.4 Task 4

The scatterplot with the new generated data is shown below.

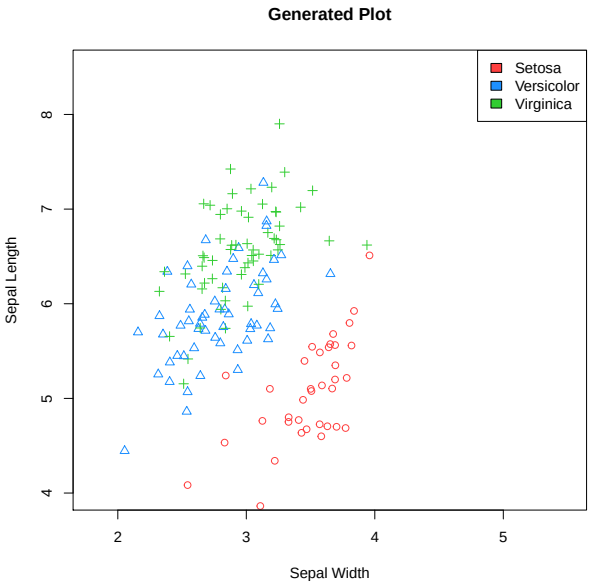

Figure 3: Scatterplot of Sepal Length and Sepal Width. Generated values.

After comparing the generated values with the previous plots we see that the datapoints are reasonable. Since we used the original data and not the predicted data to generate new values the versicolor and virginica are not separated as much as in the predicted graph. The values are also a bit more spread out.

1.5 Task 5

By using the multinom() function and performing a classification by logistic regression we got the following plot.

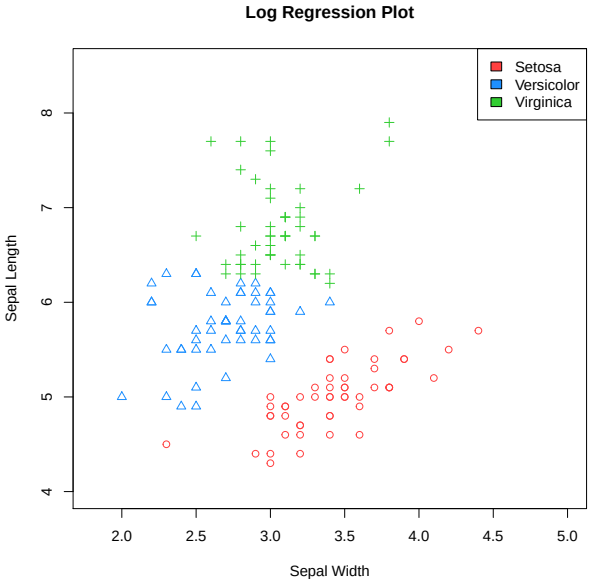

Figure 4: Scatterplot of Sepal Length and Sepal Width. Logistic Regression.

In this case, the computed misclassification error was estimated to 0.1667. Our prediction therefore had a 16.67% error rate which compared to the LDA is slightly better. This can be partly due to the fact that the assumption of a common covariance matrix was not completely satisfied, even though we used the pooled matrix in our calculations. We also know that LDA does not handle outliers quite as well as Logistic Regression, and if we look at the graph we see that the Logistic Regression predict all of the setosa correctly while the LDA misclassified one of the outliers. This also has effect on the error rates and can cause the Logistic Regression to have a better value.

2 Assignment 2. Decision trees and Naïve Bayes for bank marketing

The data was split into training/validation/test as 40%/30%/30%. Three decision trees with different settings were fit and the result is presented below:

Settings	Misclassification rate training	Misclassification rate validation
Default	0.1048	0.1092
Node size > 7000	0.1048	0.1092
Deviance > 0.0005	0.0936	0.1117

The best model seems to be with the default settings, since it has the lowest misclassification rate for the validation data. When changing the node size the number of terminal nodes changed from 6 in the default model, to 5 in the model with node size > 7000. The model with changed deviance had 122 terminal nodes, which is a big difference from the two previous mentioned models. The model got bigger when we accepted a lower minimum deviance since the deviance in each node was lower than the previous.

2.1 Task 3 - Optimal tree depth

The graph below plots the dependence of deviances for the training and validation data on the number of leaves from 1 to 50.

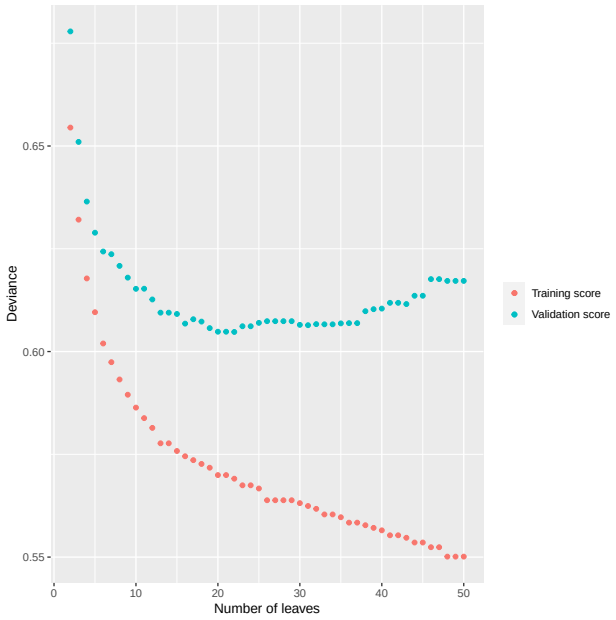

Figure 5: Deviance vs number of leaves

The optimal number of leaves based on the graph above is 22 since it has the lowest deviance on the validation data. The most important variables that seems most important for decision making in the tree are; `poutcome`, `pdays`, `month` and `contact`. If the outcome of the previous marketing campaign was a success and the number of days that has passed since the customer was last contacted was fewer than 94.5 the Y was predicted to be yes. There are lots of more conclusions that can be drawn, but basically if the outcome

of the previous campaign was failure or nonexistent the month was a god indicator together with the contact communication type.

Based on this a model was fitted and predicted on the test data, which produced the confusion matrix below. The misclassification error for the testing data is 10.89%. This is a lower score than the previous models achieved on the validation data, and can therefore be seen as having similar predictive power as previous models. However, this misclassification error was estimated on a different data set than previous models, so we can not compare them apples to apples. Concluded we can say that this model has similar predictive power compared with earlier proposed models.

Validation data	no	yes
no	11872	107
yes	1371	214

2.2 Task 4 - Loss matrix

Tree classification was made on the test data with the loss matrix as presented in the lab. This yielded the confusion matrix presented below. Compared to the model proposed above (with optimal tree depth), we can see that we better predicted true positives than before. This is because we reduced false negatives by assigning a higher cost to them with the loss matrix. This model did predicted true positives to a higher degree, but did overall have a higher misclassification error than the previous model.

Test data	no	yes
no	11189	790
yes	861	724

2.3 Task 5 - Naïve Bayes

The model proposed in `2.1 Optimal tree depth` was seen as the optimal tree and was used with the Naïve Bayes model to classify the test data, using the following principle:

$$\widehat{Y} = 1 \text{ if } p(Y = \text{'yes'}|X) > \pi \text{ , otherwise } \widehat{Y} = 0$$

where:

$$\pi = 0.05, 0.10, 0.15, ..., 0.90, 0.95$$

The TPR and TFR values for the two models was caclulated and presented as a ROC curve, which is illustrated in the figure below. The conclusions from the ROC curve is that the Optimal Tree model seems to perform better than the Naïve Bayes model, since the AUC is higher for the Optimal Tree model.

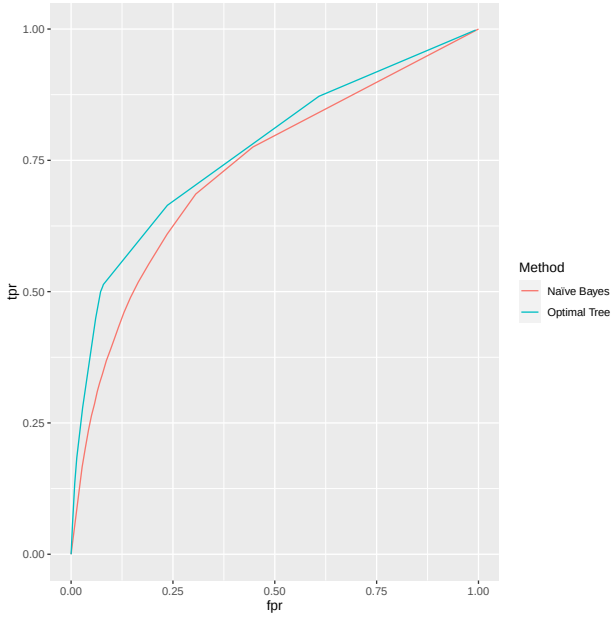

Figure 6: ROC Curve

3 Assignment 3. Principal components for crime level analysis

3.1 Task 1

In order to obtain at least 95% of variance in the original data we need to include 35 principal components. The first two principal components covers only 41.95% of variance in the original data, where PC1 covers 25% and PC2 cover 17% of the variance in the original data.

3.2 Task 2

Many features have a noticeable contribution to PC1. This can be seen by analyzing the absolute values of the loadings for PC1 as shown in the trace plot below. The loadings are defined as the coefficients of the linear model that makes up the PC. Since many of the loadings are similar in value, e.g above 0.10 in absolute value, it can be said that many of the features have a noticeable contribution to PC1.

PCA by using princomp() function on the data and doing a screeplot on the resulting object.

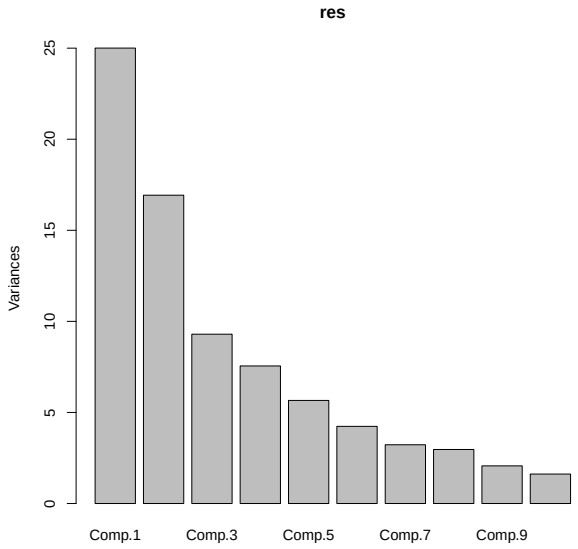

Figure 7: Screeplot of PCs

Scoreplot of the first principal component

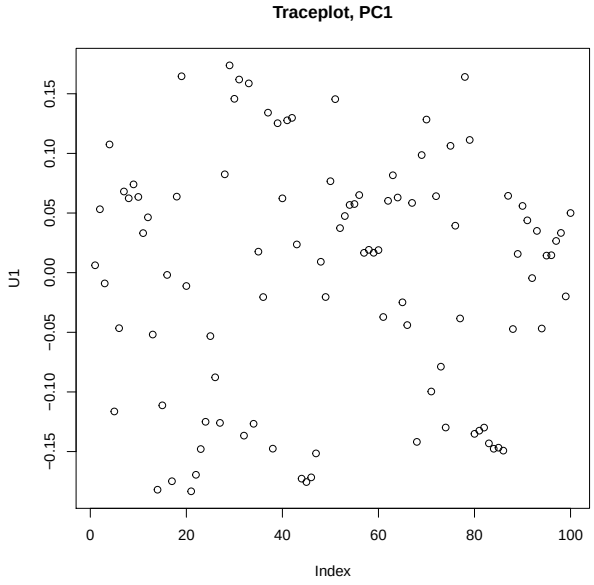

Figure 8: Score plot of PC1

3.2.1 5 features that contribute mostly (by the absolute value) to the first principle component

- PctPopUnderPov: percentage of people under the poverty level
- pctWInvInc: percentage of households with investment / rent income in 1989
- PctKids2Par: percentage of kids in family housing with two parents
- medIncome: median household income
- medFamInc: median family income (differs from household income for non-family households)

All of the features are directly linked to the economic situation of a certain household or a person's personal economic situation which makes it relate to the crime level in the country since it is known that poor people commit more crimes. Kids in households with two parents have one of them at home more often and the household's economy is also often based on two incomes. The features relate to each other since they are all describing the economic situation of a person/family.

Plot of the PC scores of the first and second PCs where colors indicate crime level

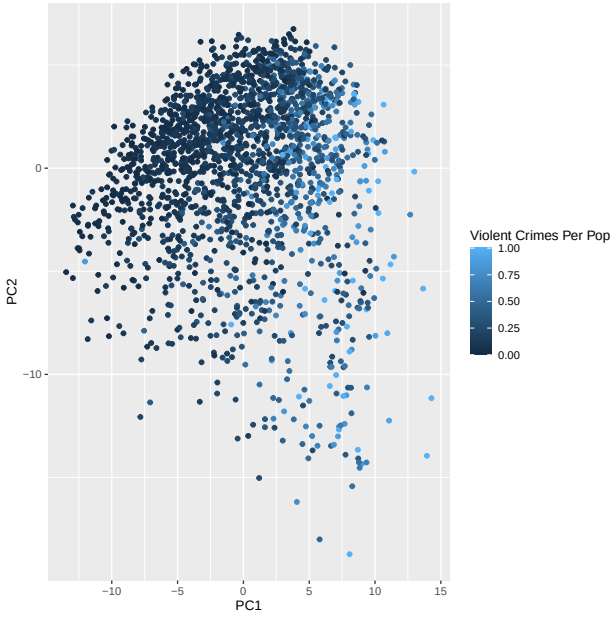

Figure 9: Plot of the PC scores

Analysis of the plot tells us that a higher value of PC2 seem to indicate that the crime level decreases whereas a higher value of PC1 seem to indicate that the crime level increases. PC1 describes the crime level with more accuracy since the crime level often is high when the values of both PC1 and PC2 are high.

3.3 Task 3

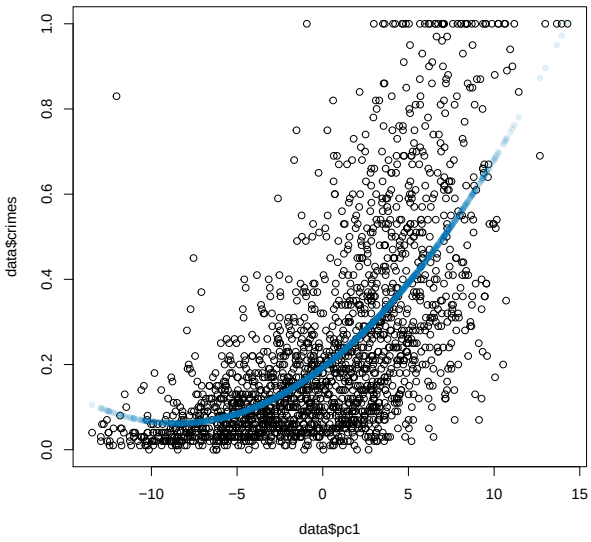

Figure 10: Scatterplot where crimes are the target and PC1 is the feature

Our target seems to be pretty well explained by PC1. A higher value of the PC is fairly linked to a higher crime level. The high variation in crime level for different PC1 scores, especially when PC1 has a high value can be derived from that PC1 only covers roughly 25% of the variance in the original data. The model seem to capture the connection between the target and the feature in a good way. Since we are only using a polynomial regression model of degree 2, we also avoid overfitting, but the model might be underfitted. We can not provide answers to if the model is underfitted since we have not run it against test data.

3.4 Task 4 - Estimation of confidence and prediction bands

The prediction interval is always wider than the confidence interval which is normal since the mean of the values for the crime rate for a specific PC1 should fall in the confidence interval 95% of the times, whereas the interval of the prediction bands describe what range a single values should fall within 95% of the times. The confidence interval is wider for small and large values of PC1 which corresponds well with the spread of the values in our plot. We also have more values for the crime rate where PC1 is close to 0 which makes our estimation of the mean value more certain.

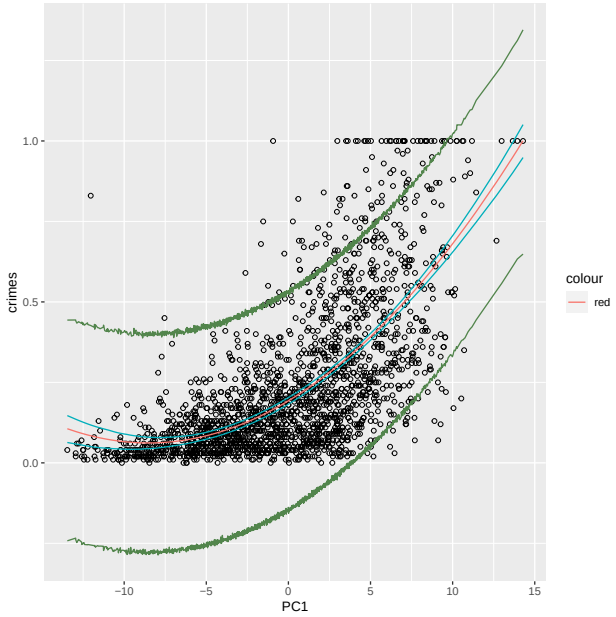

Figure 11: Scatterplot of the crime level where PC1 is the feature. Confidence bands are the blue lines and prediction bands are the green ones

4 Appendix

4.1 Code for Assignment 1

```
my_data <- iris

## Task 1 - Scatterplot Sepal

plot(iris$Sepal.Width, iris$Sepal.Length, pch = c(1,2,3)[iris$Species],
     col=c("brown1","dodgerblue1","limegreen")[iris$Species], main = "Sepal Scatter",
     xlab = "Sepal Width", ylab = "Sepal Length", xlim = c(1.8, 5), ylim = c(4, 8.5))
legend('topright', legend = c("Setosa", "Versicolor", "Virginica"),
       fill = c("brown1","dodgerblue1","limegreen"))

## Task 2 - Linear Discriminant Analysis

## Task 2a
#Setosa
x1 <- iris$Sepal.Length[iris$Species == "setosa"]
x2 <- iris$Sepal.Width[iris$Species == "setosa"]
x_setosa = cbind(x1, x2)
mu_x1 = (sum(x1))/50
mu_x2 = (sum(x2))/50
mu_set = rbind(mu_x1, mu_x2)
cov(x_setosa)
setosa_prior = 50/150

#Versicolor
x3 <- iris$Sepal.Length[iris$Species == "versicolor"]
x4 <- iris$Sepal.Width[iris$Species == "versicolor"]
x_versicolor = cbind(x3, x4)
mu_x3 = (sum(x3))/50
mu_x4 = (sum(x4))/50
mu_ver = rbind(mu_x3, mu_x4)
cov(x_versicolor)
versicolor_prior = 50/150

#Virginica
x5 = iris$Sepal.Length[iris$Species == "virginica"]
x6 = iris$Sepal.Width[iris$Species == "virginica"]
x_virginica = cbind(x5, x6)
mu_x5 = (sum(x5))/50
mu_x6 = (sum(x6))/50
mu_vir = rbind(mu_x5, mu_x6)
cov(x_virginica)
virginica_prior = 50/150

## Task 2b
Pooled_covariance = (cov(x_setosa)*50 + cov(x_versicolor)*50 +
                       cov(x_virginica)*50)/150

## Task 2d
#Setosa
```

```r
x = cbind(iris$Sepal.Length, iris$Sepal.Width)

Dk_set_x = x %*% solve(Pooled_covariance) %*% mu_set -
  ((t(mu_set) %*% solve(Pooled_covariance) %*% mu_set) / 2)[1, 1] + log(setosa_prior)
softmax(Dk_set_x)

Dk_ver_x = x %*% solve(Pooled_covariance) %*% mu_ver -
  ((t(mu_ver) %*% solve(Pooled_covariance) %*% mu_ver) / 2)[1, 1] + log(versicolor_prior)

Dk_vir_x = x %*% solve(Pooled_covariance) %*% mu_vir -
  ((t(mu_vir) %*% solve(Pooled_covariance) %*% mu_vir) / 2)[1, 1] + log(virginica_prior)

## Task 2e

#Between setosa and versicolor, Dk_set_x = Dk_ver_x
x %*% solve(Pooled_covariance) %*% mu_set -
  ((t(mu_set) %*% solve(Pooled_covariance) %*% mu_set) / 2)[1, 1] + log(setosa_prior) =
  x %*% solve(Pooled_covariance) %*% mu_ver -
  ((t(mu_ver) %*% solve(Pooled_covariance) %*% mu_ver) / 2)[1, 1] + log(versicolor_prior)

#Between setosa and virginica, Dk_set_x = Dk_vir_x
x %*% solve(Pooled_covariance) %*% mu_set -
  ((t(mu_set) %*% solve(Pooled_covariance) %*% mu_set) / 2)[1, 1] + log(setosa_prior) =
  x %*% solve(Pooled_covariance) %*% mu_vir -
  ((t(mu_vir) %*% solve(Pooled_covariance) %*% mu_vir) / 2)[1, 1] + log(virginica_prior)

#Between versicolor and virginica, Dk_ver_x = Dk_vir_x
x %*% solve(Pooled_covariance) %*% mu_ver -
  ((t(mu_ver) %*% solve(Pooled_covariance) %*% mu_ver) / 2)[1, 1] + log(versicolor_prior) =
  x %*% solve(Pooled_covariance) %*% mu_vir -
  ((t(mu_vir) %*% solve(Pooled_covariance) %*% mu_vir) / 2)[1, 1] + log(virginica_prior)

## Task 3 - Prediction with Discriminant function

for(i in 1:150) {
  if (Dk_set_x[i, 1] > Dk_ver_x[i, 1] && Dk_set_x[i, 1] > Dk_vir_x[i, 1]) {
    my_data[i, ]$Species <- "setosa"
  } else if (Dk_ver_x[i, 1] > Dk_set_x[i, 1] && Dk_ver_x[i, 1] > Dk_vir_x[i, 1]) {
    my_data[i, ]$Species <- "versicolor"
  } else if (Dk_vir_x[i, 1] > Dk_set_x[i, 1] && Dk_vir_x[i, 1] > Dk_ver_x[i, 1]) {
    my_data[i, ]$Species <- "virginica"
  }
}

plot(my_data$Sepal.Width, my_data$Sepal.Length, pch = c(1,2,3)[my_data$Species],
     col=c("brown1","dodgerblue1","limegreen")[my_data$Species],
     main = "Discriminant Plot",
     xlab = "Sepal Width", ylab = "Sepal Length", xlim = c(1.8, 5), ylim = c(4, 8.5))
legend('topright', legend = c("Setosa", "Versicolor", "Virginica"),
       fill = c("brown1","dodgerblue1","limegreen"))

#Confusion matrix
table(iris$Species, my_data$Species, dnn = c('Actual class', 'Predicted class'))
```

```r
#Misclassification rate of prediction
n_errors = 1+15+14
error_rate = n_errors/150

#LDA
iris_lda = lda(Species ~ Sepal.Length + Sepal.Width, data = iris)

lda_pred = predict(iris_lda)$class

table(iris$Species, lda_pred, dnn = c('Actual class', 'Predicted class'))

lda_n_errors = 1+15+14
lda_error_rate = lda_n_errors/150

## Task 4 - Generate with Discriminant functions¨
my_new_data = iris

#Sample species
test = sample(unique(iris$Species), 150, replace = TRUE)

#Generate new values
for(i in 1:150) {
  if (test[i] == "setosa") {
    generate = rmvnorm(1, mean = mu_set, sigma = Pooled_covariance)
    my_new_data[i, ]$Sepal.Length <- generate[1,1]
    my_new_data[i, ]$Sepal.Width <- generate[1,2]
    my_new_data[i, ]$Species <- "setosa"
  } else if (test[i] == "versicolor") {
    generate1 = rmvnorm(1, mean = mu_ver, sigma = Pooled_covariance)
    my_new_data[i, ]$Sepal.Length <- generate1[1,1]
    my_new_data[i, ]$Sepal.Width <- generate1[1,2]
    my_new_data[i, ]$Species <- "versicolor"
  } else if (test[i] == "virginica") {
    generate2 = rmvnorm(1, mean = mu_vir, sigma = Pooled_covariance)
    my_new_data[i, ]$Sepal.Length <- generate2[1,1]
    my_new_data[i, ]$Sepal.Width <- generate2[1,2]
    my_new_data[i, ]$Species <- "virginica"
  }
}

plot(my_new_data$Sepal.Width, my_new_data$Sepal.Length,
     pch = c(1,2,3)[my_new_data$Species],
     col=c("brown1","dodgerblue1","limegreen")[my_new_data$Species],
     main = "Generated Plot",
     xlab = "Sepal Width", ylab = "Sepal Length", xlim = c(1.8, 5.5), ylim = c(4, 8.5))
legend('topright', legend = c("Setosa", "Versicolor", "Virginica"),
       fill = c("brown1","dodgerblue1","limegreen"))

## Task 5 - Logistic regression

iris_multinom = multinom(Species ~ Sepal.Length + Sepal.Width, data = iris)

multinom_pred = predict(iris_multinom, type = "class")
```

```r
for(i in 1:150) {
  my_data[i, ]$Species = multinom_pred[i]
}

plot(my_data$Sepal.Width, my_data$Sepal.Length, pch = c(1,2,3)[my_data$Species],
     col=c("brown1","dodgerblue1","limegreen")[my_data$Species],
     main = "Log Regression Plot",
     xlab = "Sepal Width", ylab = "Sepal Length", xlim = c(1.8, 5), ylim = c(4, 8.5))
legend('topright', legend = c("Setosa", "Versicolor", "Virginica"),
       fill = c("brown1","dodgerblue1","limegreen"))

table(iris$Species, multinom_pred, dnn = c('Actual class', 'Predicted class'))

multinom_n_errors = 12+13
multinom_error_rate = multinom_n_errors/150
```

4.2 Code for Assignment 2

```
colClasses = c("numeric", rep("factor",4), "numeric", rep("factor",3),
               "numeric", "factor", rep("numeric",4), rep("factor", 2))
dataPure = read.csv('bank-full.csv', sep=';', header=TRUE, colClasses=colClasses)

# TASK 1
# Split data 40/30/30
data = subset(dataPure, select = -duration)
n = dim(data)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.4))
id1 = setdiff(1:n, id)
set.seed(12345)
id2 = sample(id1, floor(n*0.3))
id3 = setdiff(id1,id2)
train = data[id, ]
valid = data[id2,]
test = data[id3,]

# TASK 2
# Fit decision trees to training data
library(tree)
n = dim(train)[1]
fita = tree(y ~ ., data=train)
fitb = tree(y ~ ., data=train, control=tree.control(nobs= n, minsize=7000))
fitc = tree(y ~ ., data=train, control=tree.control(nobs= n, mindev=0.0005))

print("Training data")
summary(fita)
summary(fitb)
summary(fitc)
print("Validation data")
predicta = predict(fita, newdata=valid, type='class')
1-sum(diag(table(valid$y,predicta))/sum(table(valid$y,predicta)))

predictb = predict(fitb, newdata=valid, type='class')
1-sum(diag(table(valid$y,predictb))/sum(table(valid$y,predictb)))

predictc = predict(fitc, newdata=valid, type='class')
1-sum(diag(table(valid$y,predictc))/sum(table(valid$y,predictc)))

# TASK 3
# Finding optimal tree depth
library(ggplot2)
fit = tree(y ~ ., data=train, control=tree.control(nobs= n, mindev=0.0005))
trainScore=rep(0,50)
validScore=rep(0,50)
n_train = dim(train)[1]
n_valid = dim(valid)[1]

for(i in 2:50) {
  prunedTree=prune.tree(fit,best=i)
```

```
  pred=predict(prunedTree, newdata=valid, type="tree")
  trainScore[i]=deviance(prunedTree)/n_train
  validScore[i]=deviance(pred)/n_valid
}
data_frame = data.frame(y1=trainScore[2:50], y2=validScore[2:50], x=2:50)

ggplot(data_frame, aes(x=x)) +
  geom_point(aes(y=y1, color='Training score')) +
  geom_point(aes(y=y2, color='Validation score')) + labs(color="") +
  xlab('Number of leaves') + ylab('Deviance')


bestTree=prune.tree(fit,best=22)
summary(bestTree)

Yfit=predict(bestTree, newdata=test, type="class")
print('Task 3 missclass')
table(test$y,Yfit)
1-(sum(diag(table(test$y,Yfit)))/sum(table(test$y,Yfit)))

# TASK 4
library(rpart)
library(rpart.plot)
l = matrix(c(0, 5, 1, 0), ncol = 2)
fit = rpart(y ~ ., data=test, parms = list(loss = l))
pred <- predict(fit, type = "class")
table(test$y, pred)
1-(sum(diag(table(test$y,pred)))/sum(table(test$y,pred)))

# TASK 5
library(e1071)
library(ggplot2)

fit_bayes = naiveBayes(y ~ ., data=train)
y_bayes = predict(fit_bayes, newdata=test, type='raw')[,2]
y_tree = predict(bestTree, test, type='vector')[,2]

tpr_bayes=vector()
fpr_bayes=vector()
tpr_tree=vector()
fpr_tree=vector()
i=1
for (thr in seq(from=0.00, to=1, by=0.05)) {
  y_hat_bayes = ifelse(y_bayes>thr, 'yes', 'no')
  y_hat_tree = ifelse(y_tree>thr, 'yes', 'no')

  conf_m_bayes = table(test$y, y_hat_bayes)
  conf_m_tree = table(test$y, y_hat_tree)
  if (is.na(table(y_hat_tree)[2])) {
    if (colnames(conf_m_tree)[1] == 'yes') {
      conf_m_tree = cbind(c(0,0), conf_m_tree)
    } else {
      conf_m_tree = cbind(conf_m_tree, c(0,0))
```

```
    }
  }
  if (is.na(table(y_hat_bayes)[2])) {
      if (colnames(conf_m_bayes)[1] == 'yes') {
        conf_m_bayes = cbind(c(0,0), conf_m_bayes)
      } else {
        conf_m_bayes = cbind(conf_m_bayes, c(0,0))
      }
  }

  tpr_bayes[i] = conf_m_bayes[2,2]/sum(conf_m_bayes[2,])
  fpr_bayes[i] = conf_m_bayes[1,2]/sum(conf_m_bayes[1,])

  tpr_tree[i] = conf_m_tree[2,2]/sum(conf_m_tree[2,])
  fpr_tree[i] = conf_m_tree[1,2]/sum(conf_m_tree[1,])

  i = i + 1
}
df <- data.frame(tpr=c(tpr_tree,tpr_bayes), fpr=c(fpr_tree, fpr_bayes),
                 Method=c(rep(paste("Optimal Tree"), each=length(tpr_tree)),
                          rep(paste("Naïve Bayes"), each=length(tpr_bayes))) )

ggplot(aes(x=fpr, y=tpr, color=Method), data=df) +
  geom_line() + xlim(0,1) + ylim(0,1) +
  xlab('fpr') + ylab('tpr')
```

4.3 Code for Assignment 3

```
library(boot)
library(ggplot2)
setwd("~/Desktop/TDDE01/tdde01/lab2/ass3")
origin_data = read.csv("communities.csv")
data_without_crimes = origin_data[, 1:100]
n = dim(data_without_crimes)[1]
data_without_crimes= scale(data_without_crimes)
data_without_crimes = as.matrix(data_without_crimes)

#Calculate covariance matrix
S = 1/n * t(data_without_crimes)%*%data_without_crimes

#calculate eigenvalues from the covariance matrix
eigen_values_vectors = eigen(S, only.values = FALSE)
eigenvalues = eigen_values_vectors$values

# variance covered by principal components as Strings and as numbers
variance = sprintf("%2.3f",eigenvalues/sum(eigenvalues)*100)
variance1 = eigenvalues/sum(eigenvalues)*100

#Calculate how many principal components is needed to cover 95% of variance
var = sum(variance1[1:34])
var_correct = sum(variance1[1:35])
var_2 = sum(variance1[1:2])
var_PC1=sum(variance1[1])
var_PC2=sum(variance1[2])



#TASK 2 ----------------------------

#plot the variance covered by principal components
#princomp function does the PCA on our data
#The elements of an eigenvector, that is, the values within a particular row of matrix A,
#are the weights aij. These values are called the loadings, and they describe how much
#each variable contributes to a particular principal component.
res=princomp(data_without_crimes)
pdf('ScreePlot_PCAs.pdf')
screeplot(res)
dev.off()
U= res$loadings
U1 = U[,1]
U1_vec = as.vector(U1)
pdf('Traceplot_PC1.pdf')
plot(U1, main="Traceplot, PC1")
dev.off()
#Get the 5 varaibles that covers the largest variation in the target variable
tail(sort(abs(U1)),5)

#
x = data.frame(pc1=res$scores[,1], pc2=res$scores[,2], crimes=origin_data$ViolentCrimesPerPop)
```

```
data = data.frame(pc1=res$scores[,1], crimes=origin_data$ViolentCrimesPerPop)
pdf('ScorePlot_PC1_PC2.pdf')
ggplot(x, aes(x=pc1, y=pc2, color=crimes)) + xlab("PC1") + ylab("PC2") +
  geom_point() + labs(color="Violent Crimes Per Pop")
dev.off()

polynomial <- lm(crimes ~ poly(pc1, 2), data)  #Doing second degree polynomial regression
#Fitting a line to the points

summary(polynomial)

pdf('Scatterplot_nobands.pdf')
plot(data$pc1, data$crimes)   #plotting the data points
mycol <- rgb(0, 120, 185, max = 255, alpha = 30, names = "blue50")
points(x=data$pc1,    #Plotting the fitted line to the data points
       y=fitted(polynomial),
       pch=16,
       col=mycol)
dev.off()

#How random values are to be generated
rng=function(data1, polynomial){
  data2=data.frame(Crime=data1$crimes, pc1=data1$pc1)
  n=length(data$crimes)
  #Generate new Crime level
  data2$crimes=rnorm(n=n,predict(polynomial, newdata=data2),sd(polynomial$residuals))
  return(data2)
}

f1=function(data2){
  result <- lm(crimes~poly(pc1,2), data=data2) #Fit polynomial model
  #Predict valies for all Crime values from the original data
  newCrime=predict(result, newdata = data)
  return(newCrime)
}

res_conf=boot(data, statistic=f1, R=1000, mle=polynomial, ran.gen=rng, sim="parametric")


f2=function(data3){
  result <- lm(crimes~poly(pc1,2), data=data3) #fit polynomial model
  #predict values for all Area values from the original data
  Newcrimes=predict(result,newdata=data)
  n=length(data3$crimes)
  predictedCrimes=rnorm(n,Newcrimes, sd(polynomial$residuals))
  return(predictedCrimes)
}

res_pred=boot(data, statistic=f2, R=10000, mle=polynomial, ran.gen=rng, sim="parametric")

env_conf=envelope(res_conf) #compute confidence bands
env_pred=envelope(res_pred) #Compute prediction bands
```

```
pdf('Scatterplot.pdf')
ggplot(data=data) +
  geom_point(mapping = aes(x=pc1, y=crimes), shape=1) +
  geom_line(mapping= aes(x=pc1, y=fitted(polynomial), color="red")) +
  geom_line(mapping =  aes(x = pc1, y = env_conf$point[2,]), color="#00AFBB") +
  geom_line(mapping =  aes(x = pc1, y = env_conf$point[1,]), color="#00AFBB") +
  geom_line(mapping =  aes(x = pc1, y = env_pred$point[1,]), color="#52854C") +
  geom_line(mapping =  aes(x = pc1, y = env_pred$point[2,]), color="#52854C") +
  xlab("PC1") +
  ylab("crimes")
dev.off()
```