Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

Computer Lab 1, Block 1

Statement of Contribution

Discussions of results from all parts of the lab has been done together, and the report is mainly written based on these discussions.
Assignment 1: Report written by Unn Zachrison, code written together.
Assignment 2: Report written by Victoria Winqvist, code written together.
Assignment 3: Report written by Alice Paulsen, code written by Alice Paulsen and Victoria Winqvist.

Assignment 1

Confusion matrices and misclassification error

After using the training data to fit 30-nearest neighbor classifier with the kknn() function, the estimated confusion matrices and the respective misclassification errors for both train and test data were calculated and are shown below in figure 1 and figure 2.

```
> confusionMatrixTrain
   predTrain
      0   1   2   3   4   5   6   7   8   9
  0 202   0   0   0   0   0   0   0   0   0
  1   0 179  11   0   0   0   0   1   1   3
  2   0   1 190   0   0   0   0   1   0   0
  3   0   0   0 185   0   1   0   1   0   1
  4   1   3   0   0 159   0   0   7   1   4
  5   0   0   0   1   0 171   0   1   0   8
  6   0   2   0   0   0   0 190   0   0   0
  7   0   3   0   0   0   0   0 178   1   0
  8   0  10   0   2   0   0   2   0 188   2
  9   1   3   0   5   2   0   0   3   3 183
> 
```

Figure 1: Confusion matrix for train data, misclassification error = 0.04500262 (4.5%)

```
> confusionMatrixTest
   predTest
      0   1   2   3   4   5   6   7   8   9
  0  77   0   0   0   1   0   0   0   0   0
  1   0  81   2   0   0   0   0   0   0   3
  2   0   0  98   0   0   0   0   0   3   0
  3   0   0   0 107   0   2   0   0   1   1
  4   0   0   0   0  94   0   2   6   2   5
  5   0   1   1   0   0  93   2   1   0   5
  6   0   0   0   0   0   0  90   0   0   0
  7   0   0   0   1   0   0   0 111   0   0
  8   0   7   0   1   0   0   0   0  70   0
  9   0   1   1   1   0   0   0   1   0  85
> 
```

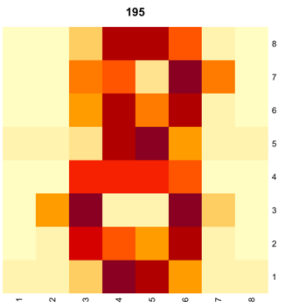
Figure 2: Confusion matrix for test data, misclassification error 0.03970742 (3.97%)

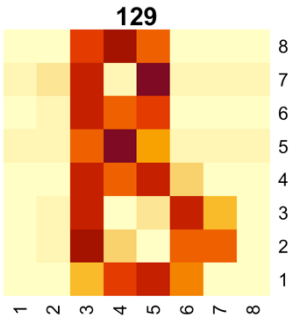
Overall, the misclassification rate of 3-5% for the training and test data seems to be a correct calculation based on the confusion matrices, where most predictions were correct. This indicates that the overall quality of predictions is good, at least compared to random guesses of the digits. The fact that the misclassification rates does not differ that much, indicates that the model based on the training date is not overfitted for the test data, which also an indication that the prediction quality is

good. In the confusion matrices we can see that there are three cases where the predictions deviate the most. These are 8 predicted as 1, 9 as 3 and 4 as 7 (marked in yellow). We can also see in the confusion matrix for test data that other wrongly predicted cases (Marked in green) are more prominent which seems correct since they were not as distinct in the confusion matrix from train data, and therefore the model is not as fitted to those predictions. The incorrect predictions also seem reasonable, when blurred the three different cases could absolutely be misinterpret as each other. This means that the predication for the different digits probably is depended on how alike the digits are, and therefore the quality of the predictions must also differ.

2 Easiest cases of digit 8 to classify

By comparing the different probabilities, two of the ones with the highest probability (therefore easiest to classify) were rows 195 and 129 in the matrix containing all probabilities from training data for the digit 8 (probability_8_train in the code). The heatmaps are shown blow in figure 3 and figure 4.



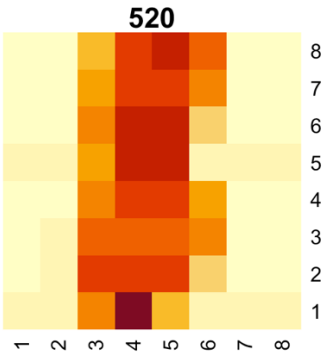


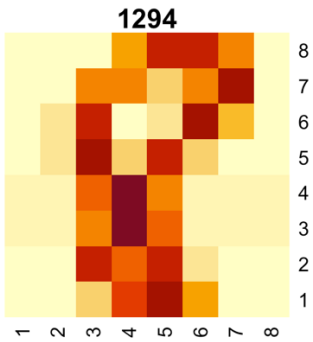
Visually, both heatmaps of rows 195 and 129 contain the typical characteristics of the digit 8, with an almost hourglass shape and "empty" pixels in the middle of the two circles. Both had a probability of 1, which seems reasonable when examine the heatmaps.

Figure 3: Heatmap of row 195 *Figure 4: Heatmap of row 129*

3 Hardest cases of digit 8 to classify

The three rows with the lowest probability of being an eight where rows 502, 1294, 431 in the matrix, shown in figure 5, 6 and 7.





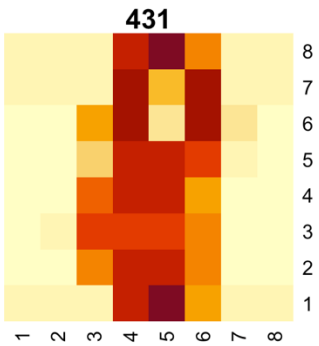


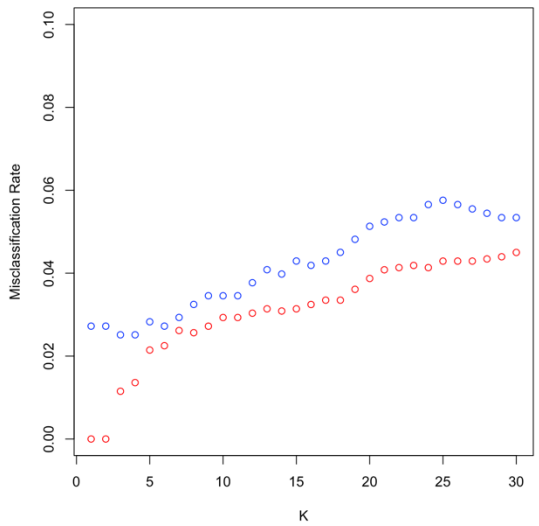
Figure 5: Heatmap of row 520 *Figure 6: Heatmap of row 1294* *Figure 7: Heatmap of row 431*

All three heatmaps above lack the distinct attributes of an eight. 520 had the probability of 10.0% and was therefore the case with the lowest probability. This could be an effect of the almost rectangular shape, which could be interpret as for example a thick 1. 1295 had a probability of 16,7% and 431 had 13.3% which also seems like a good estimate since both heatmaps could be interpret as other digits.

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

How does the model complexity change when K increases and how does it affect the training and validation errors?

After fitting a K-nearest neighbor classifiers to the training data for different values of K (1-30), a plot showing the dependencies of training (red) and validation (blue) misclassification error for the different values for K is created below in figure 8.



Training (Red): When K increases, the model needs to adapt to more data, and will therefore generate a less complex model. This will in return increase the misclassification error.

Validation (Blue): In the same way the training data behaves when K increases, the model will become less complex, and the misclassification error will increase. The main difference is that the validation data is validating the model, and a larger error will be present at the starting values of K.

The optimal K for this plot would be K = 3, since it is the K with the smallest misclassification error for validation data.

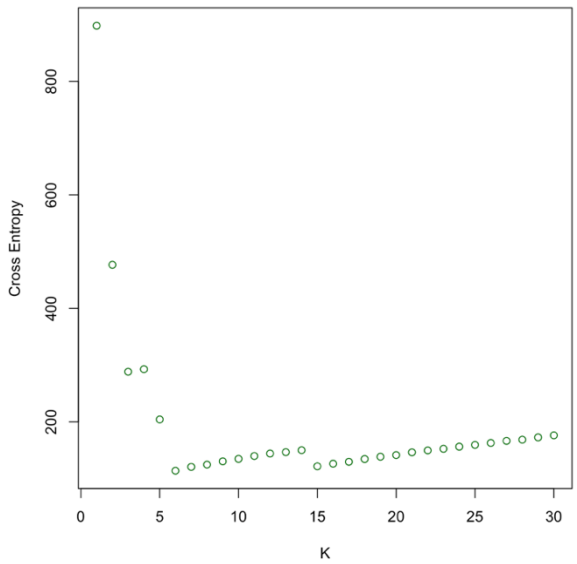
Figure 8: Dependencies of misclassification errors for different values of k

Estimate the test error for the model having the optimal *K=3*

The test misclassification error when K =3 is 2,4% which is lower than both train (2.6%) and validation misclassification error (2,3%). This indicates that the model quality is good.

Plot the dependence of the validation error on the value of K (1-30)

After fitting K-nearest neighbor classifiers to the training data for different values of K, the error for the validation data is computes as cross-entropy and is shown below in figure 9.



The optimal K value according to this plot is K = 6, where the cross-entropy value is the smallest (113.8). If the response has multinomial distribution, the cross entropy might be a more suitable choice than misclassification error since cross entropy assumes that all misclassification costs are of equal importance which would be consistent with multinomial distribution. This is because the ln( ) function in cross-entropy takes the closeness of a prediction into account.

Figure 9: Cross entropy and different values of K

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

Assignment 2

1. -
2. The MSE was computed to 0.8731931 for training data and to 0.9097431 for test data. The misclassification is slightly larger for test due to overfitting in the model of train data. To determine the variables that contribute most to the model, the p-value is inspected. The variable with the smallest p-value is the one that contributes the most. The coefficients in the training model are displayed in figure 1.

```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
Jitter...       0.181065   0.144249   1.255 0.209481
Jitter.Abs.    -0.169830   0.040851  -4.157 3.30e-05 ***
Jitter.RAP     -5.098809  18.184783  -0.280 0.779196
Jitter.PPQ5    -0.071777   0.084701  -0.847 0.396816
Jitter.DDP      5.079056  18.188164   0.279 0.780069
Shimmer         0.590992   0.205286   2.879 0.004015 **
Shimmer.dB.    -0.172860   0.139380  -1.240 0.214983
Shimmer.APQ3   32.213852  77.012847   0.418 0.675759
Shimmer.APQ5   -0.386846   0.113713  -3.402 0.000677 ***
Shimmer.APQ11   0.310256   0.062270   4.982 6.58e-07 ***
Shimmer.DDA   -32.529915  77.012630  -0.422 0.672761
NHR            -0.186755   0.045741  -4.083 4.55e-05 ***
HNR            -0.239777   0.036565  -6.558 6.27e-11 ***
RPDE            0.003958   0.022611   0.175 0.861052
DFA            -0.277038   0.019888 -13.930  < 2e-16 ***
PPE             0.229006   0.033264   6.885 6.84e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 10 – Output from summary(parkinsons_train_LM)

The variables with the with the smallest p-value (< 0.05) and therefore contribute the most to the model are Shimmer.APQ5, Shimmer.APQ11, DFA, HNR, PPE, NHR, Jitter.Abs and Shimmer. These variables have a p-value close to zero which indicates strong evidence against the null hypothesis.

3.
   a. The following function was used to calculate the loglikelihood

$$l\big(p(Y;X,\theta)\big) = -\frac{n}{2}(\log(2\pi\sigma^2)) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - x_i\theta)^2$$

   b. For the ridge function, the following function was used:

$$-\Big(l\big(p(Y;X,\theta)\big)\Big) + \lambda\sum_{j=1}^{p}{\theta_j}^2$$

   c. –
4. To calculate the degrees of freedom the sum of the diagonal of the hat matrix was computed. The diagonal describes the how changes in y influences $\hat{y}$.

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

5.

- Lambda = 1:
  - MSE:
    - Train: 0.873277
    - Test: 0.9290358
  - Degrees of freedom: 13.86281

- Lambda = 100:
  - MSE:
    - Train: 0.8790601
    - Test: 0.9263091
  - Degrees of freedom: 9.939085

- Lambda = 1000:
  - MSE:
    - Train: 0.9156279
    - Test: 0.9479198
  - Degrees of freedom: 5.643351

Comments:

MSE: To select the most appropriate penalty parameter lambda, the MSE for test data is inspected. Since the test data "tests" the model and is less likely to overfit compared to train, it is more interesting to inspect the MSE for test. The lambda with the smallest MSE for test data is lambda = 100.

Degrees of freedom: Lambda = 1 has the largest number of degrees of freedom which indicates that the model is the most complex when lambda is 1. According to the computed data, the complexity decreases when lambda increases, the reason is that the penalizing factor, that penalizes complexity in the ridge function, becomes larger with a larger lambda.

In conclusion, the most appropriate model is the one with the smallest MSE test, where lambda = 100, this model is more complex than lambda = 1000 but less complex than lambda = 1.

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

Assignment 3

Step 1
The first question was to make a scatterplot showing a Plasma glucose concentration on Age where observations are colored by diabetes levels.

As can be seen in the scatterplot in Figure 11 below, it is hard to differentiate between the two sets where pink represents diabetes, and blue represents no diabetes. Therefore, diabetes might be hard to classify with a standard logistic regression model using only these two variables, because a lot of points could end up being misclassified.

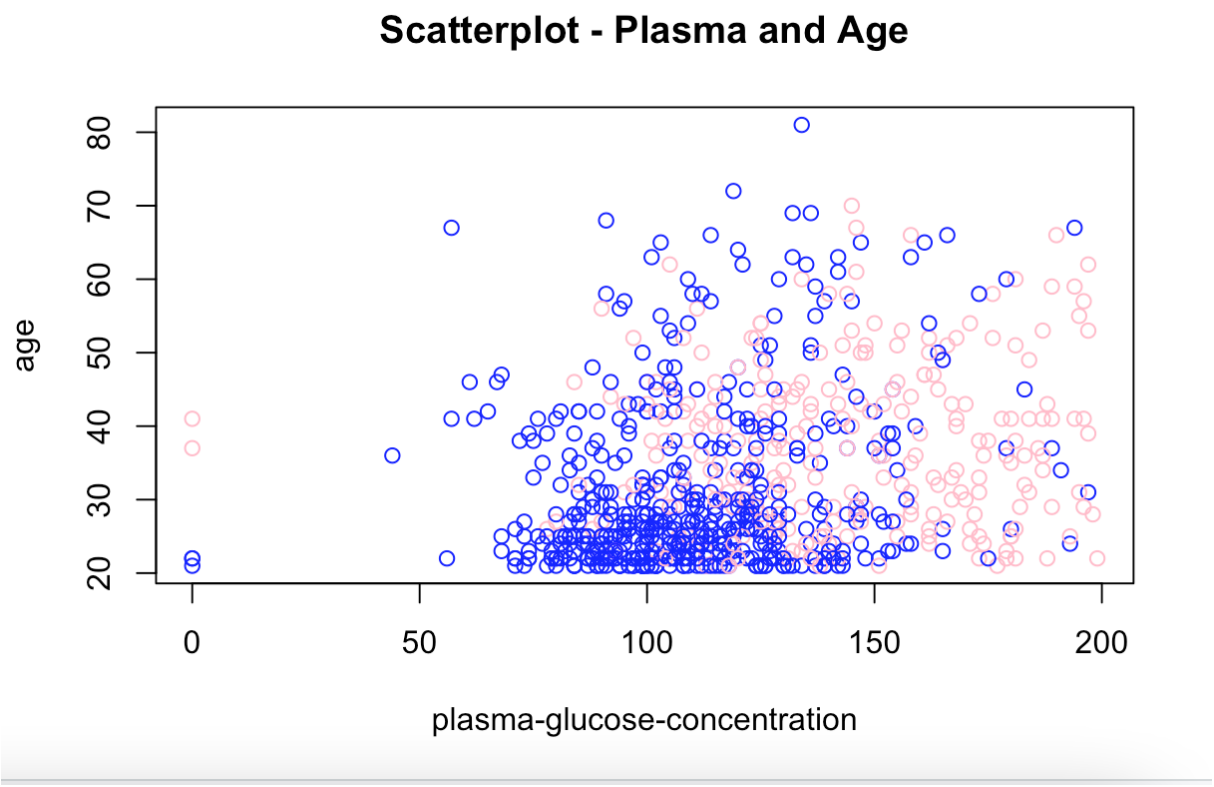


Figure 11 - A scatterplot depending on Plasma glucose concentration and Age

Step 2
The second question was to train a logistic regression model with $y = Diabetes$ as target, $x_1 = Plasma\ glocose\ concentration$ and $x_2 = Age$ as features and make a prediction for all observations using $r = 0.5$ as the classification threshold.

We did the logistic regression with family=" binomial" since we had two classes. From the coeff() command we got that $\theta^T = (0.03564404, 0.02477835, 5.91244906)$ and $X = (x1, x2, 1)$
We then got the following equation for the model: $y = \ 0.03564404 * x_1 + 0.02477835 * x_2 - 5.91244906$

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

By using the formula $g(x) = \frac{1}{1+e^{-\theta^T x}}$, we got the following probabilistic equation of the estimated model:

$$g(x) = \frac{1}{1 + e^{-(0.03564,x_1 + 0.02478 * x_2 - 5.91245)}}$$

$$p(y = C|x, \theta) = \begin{cases} g(x), & if\ C = 1 \\ 1 - g(x), & if\ C = 0 \end{cases}$$

Then the training misclassification error was calculated to $MSE = 0.2630208$

A scatterplot was made, see Figure 12, showing the predicted values of Diabetes as color, where blue represents no diabetes and orange represents diabetes.

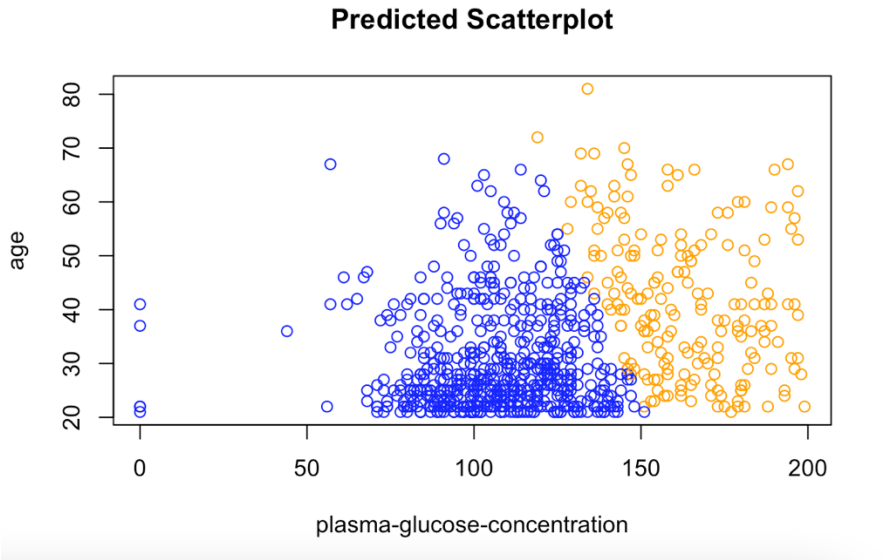


Figure 12 - A scatterplot with predicted values depending on Plasma glucose concentration and Age

By comparing the predicted scatterplot in Figure 12 with the scatterplot in Figure 11, the quality of the classification seems to be good, but far from perfect. MSE = 26.3% confirms that assumption.

Step 3

The third question was to use the model estimated in step 2 to report the equation of the decision boundary between the two classes, and then add a curve showing the boundary to the scatterplot from step 2. We know that (1) is true because of (2), we resolved x_2 and got (3), which could be used to calculate k and m for $y = kx + m$.

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0 \ \ where\ x_1 = plasma\ glucose\ concentration\ , x_2 = age \ (1)$$

$$\theta^T = 0 \ (2)$$

$$\Rightarrow\ x_2 = -\frac{\theta_1}{\theta_2} X_1 - \frac{\theta_0}{\theta_2} \ (3)$$

Below, both the scatterplot with predicted values as well as the scatterplot with actual values can be seen, with the curve representing the decision boundary between the two classes. The decision boundary seems to catch the data distribution in the predicted scatterplot, see Figure 13, very

well, but only fairly well in the scatterplot with the actual values, see Figure X, since it is hard to differentiate the pink and blue points from each other.

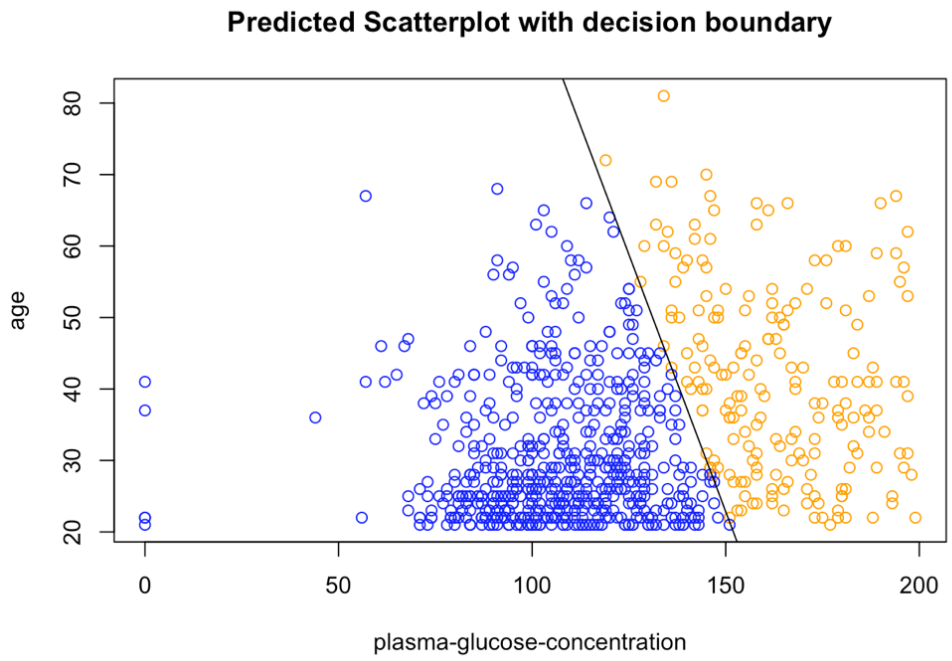


Figure 13 - A scatterplot of predicted values and decision boundary

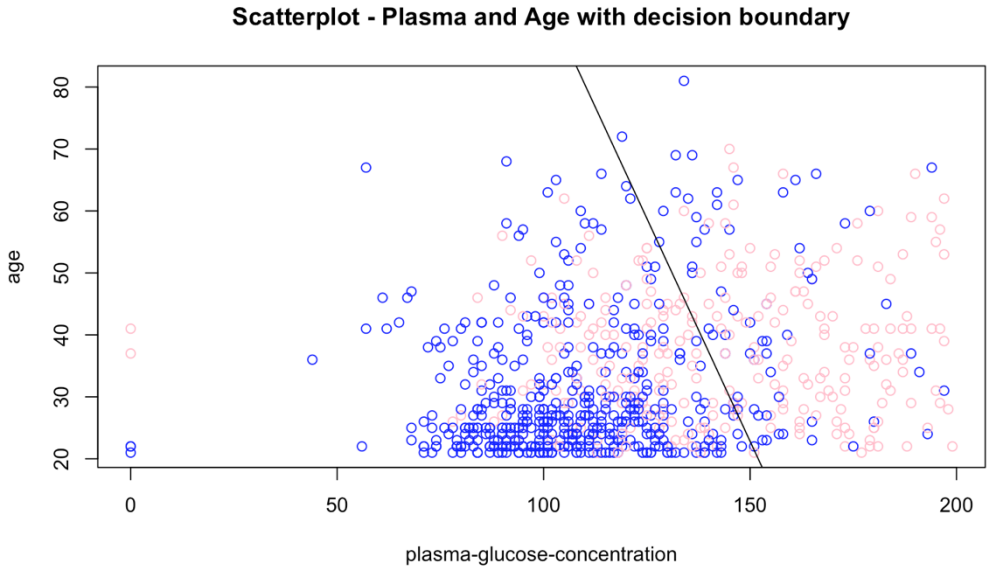


Figure 14 - A scatterplot with decision boundary

Step 4

The fourth question was to make the same kind of plots as in step 3, but use thresholds $r = 0.2$ and $r = 0.8$. The scatterplot for $r = 0.2$ can be seen in Figure 15, and the scatterplot for $r = 0.8$ can be seen in Figure 16. We can by looking at the plots se visually that when the r-value is lower, a larger part of the predicted values will be classified as "No diabetes", while when the r-value is higher, a larger part of the predicted values will be classified as "Diabetes".

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

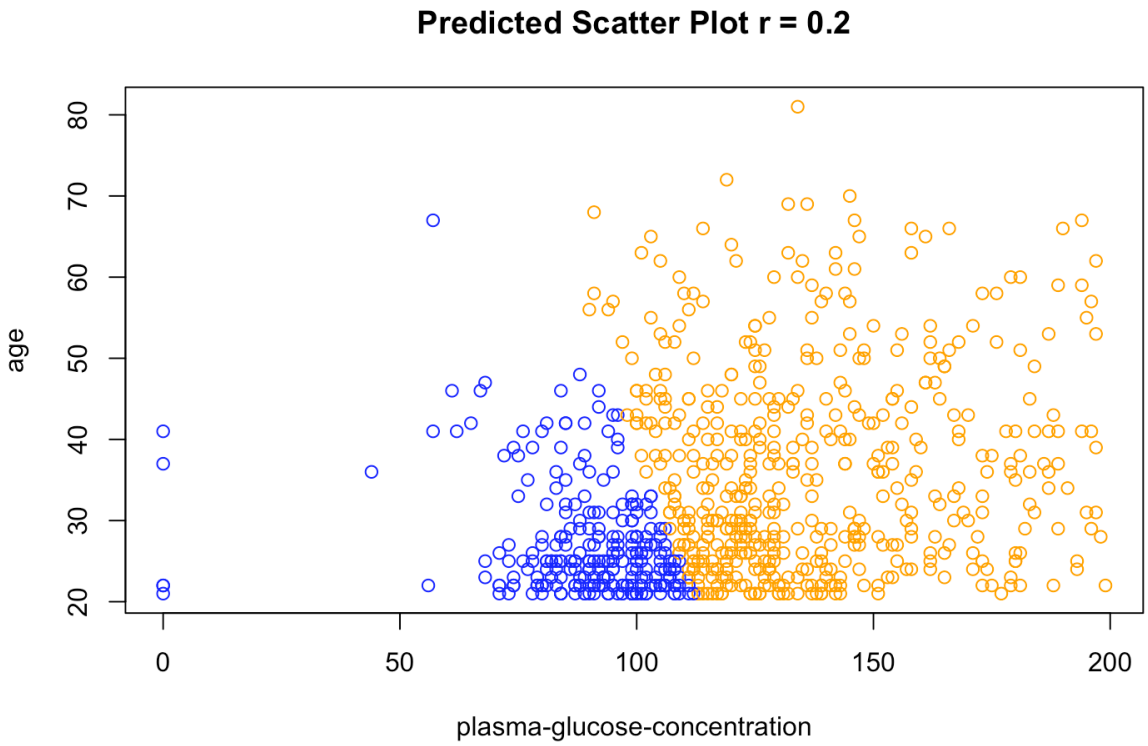


Figure 15 - A predicted scatterplot with r=0.2

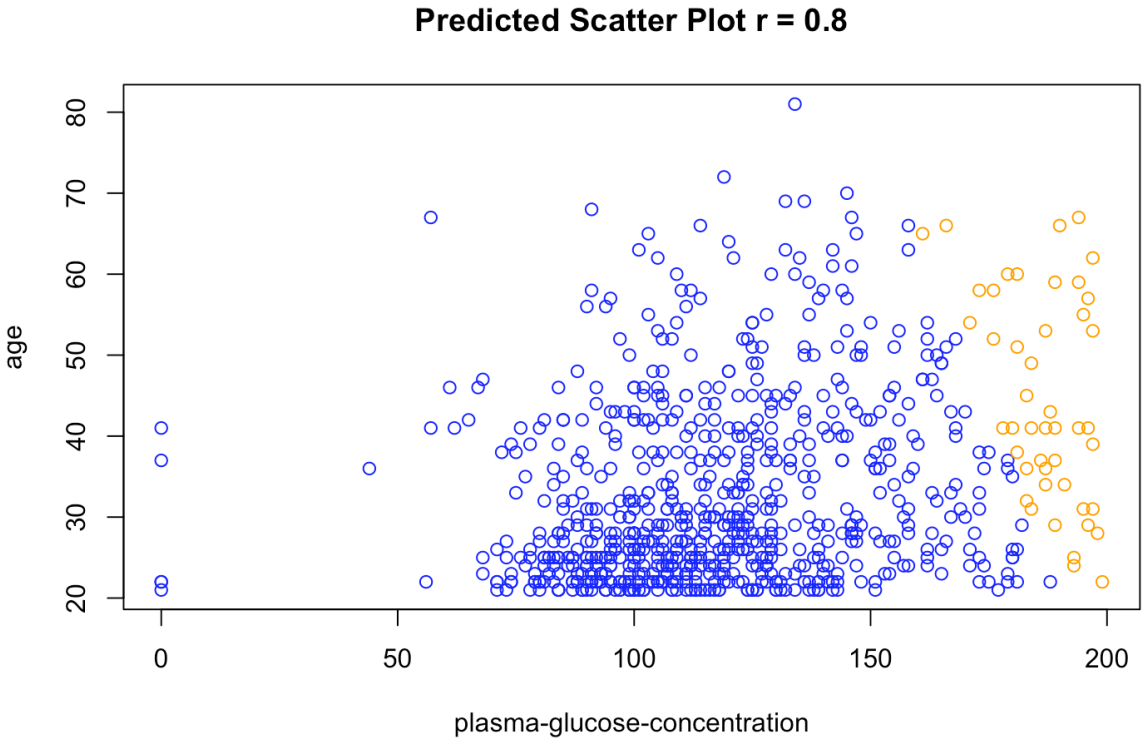


Figure 16 - A predicted scatterplot with r=0.8

Step 5

For this question, the task was to perform a basis function expansion trick by computing new features $z_1 = x_1^4, z_2 = x_1^3 x_2, z_3 = x_1^2 x_2^2, z_4 = x_1 x_2^3, z_5 = x_2^4$. The features were then added to

the data set, and a logistic regression model was computed with y as target and previously used features x_1 and x_2 as well as the new features $z_1, \dots, z_5$. A scatterplot of the same kind as in step 2, i.e. with prediction for all observations using $r = 0.5$, was plotted, see Figure 17. Further, the training misclassification rate was calculated to $MSE = 0.2447917$. Based on the misclassification rate, the basis function expansion trick helped slightly in terms of quality of the model, since the misclassification rate has decreased from 26.30% to 24.48%.

Further, by comparing the scatterplot in Figure 17, with the basis function expansion trick, to Figure 12, the predicted scatterplot in step 2, we can see that the decision boundary is no longer linear as it was in step 2. The model has increased in complexity, and the summary of the model, Figure X, indicates that the model may not be significant based on at $z_1, \dots, z_5$ in Figure 18, where the $p - value > 0.05$. This model is not necessarily better than the model used in step 2.

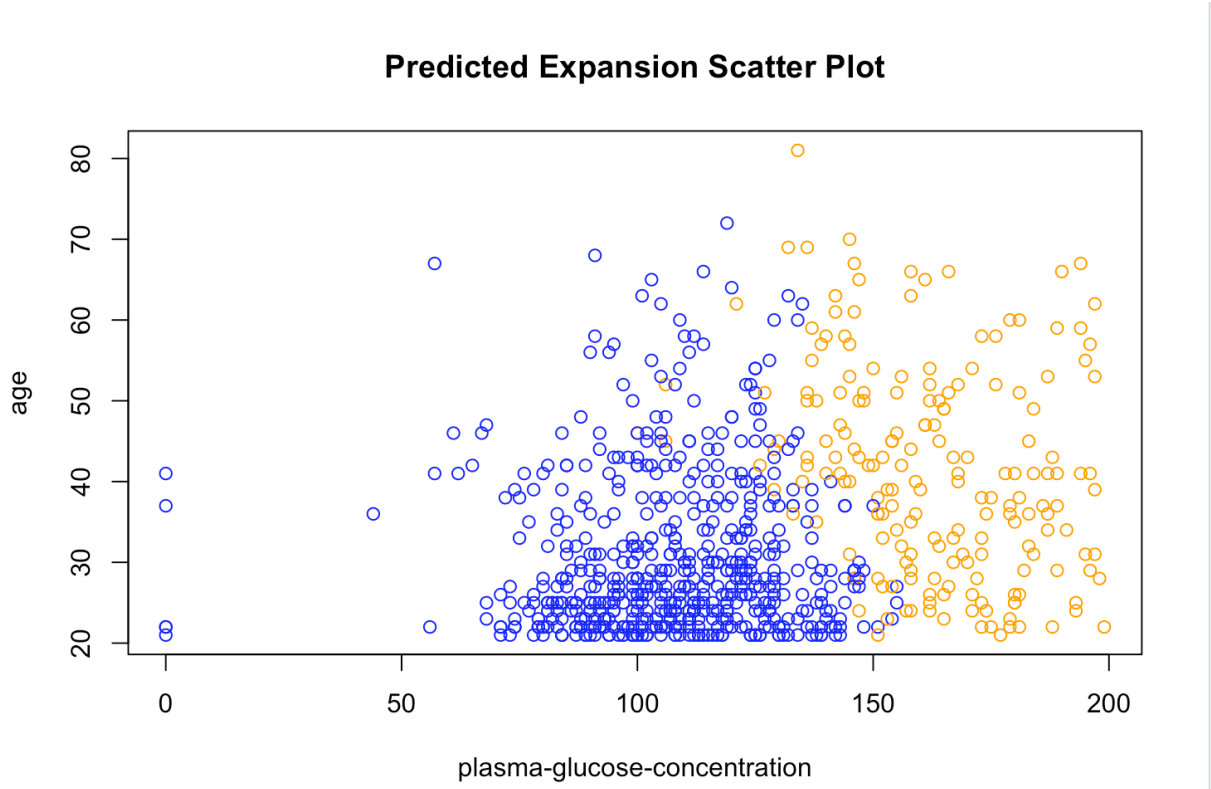


Figure 17 - Predicted expansion scatterplot

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5113  -0.7475  -0.5188   0.8237   2.9229

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)   -5.002e+00  9.608e-01  -5.206 1.93e-07 ***
diabetes[, 2]  2.784e-02  9.598e-03   2.900  0.00373 **
diabetes[, 8]  1.499e-02  8.357e-03   1.794  0.07287 .
diabetes$z1    3.035e-04  2.114e-04   1.436  0.15104
diabetes$z2   -6.561e-05  4.593e-05  -1.428  0.15318
diabetes$z3    5.180e-06  3.156e-06   1.641  0.10075
diabetes$z4   -1.274e-07  7.478e-08  -1.704  0.08847 .
diabetes$z5    1.372e-09  1.032e-09   1.330  0.18341
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 993.48  on 767  degrees of freedom
Residual deviance: 783.29  on 760  degrees of freedom
AIC: 799.29

Number of Fisher Scoring iterations: 4
```

Figure 18 - Summary of the model

Appendix A. Code

```
#####################################
#        LAB 1 Assignment 1     #
#####################################

##########
#   1   #
##########
optdigits = read.csv("optdigits.csv",header = FALSE)
n = dim(optdigits)[1]
set.seed(12345)
#split data
id = sample (1:n, floor(n*0.5)) #id = 50% of optdigits
train = optdigits[id,] #train 50% of optdigits
id2 = setdiff(1:n, id) #difference: n-id (100%-50%)
set.seed(12345)
id3 = sample(id2, floor(n*0.25))
validation = optdigits[id3,]
id4 = setdiff(id2,id3) #difference id2-id3 (50%-25%)
test = optdigits[id4,]

##########
#   2   #
##########
#train-kkn:
kknnTrain=kknn(as.factor(train$V65)~., train, train, k=30,kernel="rectangular")
#compute misclass-error:
```

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

```
predTrain= kknnTrain$fitted.values #predictions
confusionMatrixTrain = table(train$V65,predTrain)
misclassTrain = (1-sum(diag(confusionMatrixTrain))/length(train$V65))

#test-kkn:
kknnTest=kknn(as.factor(train$V65)~., train, test, k=30,kernel="rectangular")
#compute misclass-error:
predTest= kknnTest$fitted.values #predictions
confusionMatrixTest = table(test$V65,predTest)
misclassTest = (1-sum(diag(confusionMatrixTest))/length(test$V65))

#########
#   3   #
#########
#probabilities for digit 8:
probabilityTrain = kknnTrain$prob
probability_8_Train = probabilityTrain[,9] #probabilities from column "8"
#easiest to classify (2 cases):
high_prob_8 = matrix(0, nrow=2, ncol=2)#matrix with the probability of the 8's with highest
prob and its row.
for (i in 1:2){
  for(j in 1:length(probability_8_Train)){
    if(train[j,65] == 8 && probability_8_Train[j] > high_prob_8[i,2]){#inspecting the last column
- find digit 8
      if (j != high_prob_8[1,1]){
        high_prob_8[i,1] = j
        high_prob_8[i,2] = probability_8_Train[j]
      }
    }
  }
}
#hardest to classify (3 cases):
low_prob_8 = matrix(1, nrow=3, ncol=2)#matrix with the probability of the 8's with lowest
prob and its row.
for (i in 1:3){
  for(j in 1:length(probability_8_Train)){
    if(train[j,65] == 8 && probability_8_Train[j] < low_prob_8[i,2]){#inspecting the last column -
find digit 8
      if (j != low_prob_8[1,1] && j!= low_prob_8[2,1]){
        low_prob_8[i,1] = j
        low_prob_8[i,2] = probability_8_Train[j]
      }
    }
  }
}
#heatmaps:
heatmapPlot <- function(row_number){
  plotData = as.numeric(train[row_number,1:64])
  plotMatrix = matrix(plotData, nrow = 8, byrow = TRUE)
  heatmap(plotMatrix, Rowv = NA, Colv = NA, main = row_number)
}
```

```
heatmapPlot(195)#easy
heatmapPlot(129)#easy
heatmapPlot(520)#hard
heatmapPlot(1294)#hard
heatmapPlot(431)#hard

#########
#   4   #
#########
#training:
misclass_errorsTrain =c(0,30)
for (i in 1:30){
kknnTrain2=kknn(as.factor(train$V65)~., train = train, test = train, k=i,kernel="rectangular")
predTrain2= kknnTrain2$fitted.values #predictions
confusionMatrixTrain2 = table(train$V65,predTrain2)
misclassTrain2 = (1-sum(diag(confusionMatrixTrain2))/length(train$V65))
misclass_errorsTrain[i] = misclassTrain2
}
#validation:
misclass_errorsVal = c(0,30)
for (i in 1:30){
  kknnValidation=kknn(as.factor(train$V65)~., train = train, test = validation,
k=i,kernel="rectangular")
  predValidation= kknnValidation$fitted.values #predictions
  confusionMatrixValidation = table(validation$V65,predValidation)
  misclassValidation = (1-sum(diag(confusionMatrixValidation))/length(validation$V65))
  misclass_errorsVal[i] = misclassValidation
}
#plot misclassification rate and k:
plot(misclass_errorsTrain, xlab = "K", ylab = "Misclassification Rate", col ="red", ylim = c(0,
0.1))
points(misclass_errorsVal, col = "blue") #when k increases the error increases since the model
becomes less complex (less accurate)
which.min(misclass_errorsVal) #receive minimal error for validation data
#test data model for optimal k = 3
kknnTest2=kknn(as.factor(train$V65)~., train, test, k=3,kernel="rectangular")
predTest2= kknnTest2$fitted.values #predictions
confusionMatrixTest2 = table(test$V65,predTest2)
misclassTest2 = (1-sum(diag(confusionMatrixTest2))/length(test$V65))

#########
#   5   #
#########
crossEntropy_k = c()
for (k in 1:30){
  kknnValidation_CE=kknn(as.factor(train$V65)~., train, validation, k = k,kernel="rectangular")
  probabilityValidation_CE = kknnValidation_CE$prob #probabilities for each k

  sum_crossEntropy = 0 #for each k - create new variable
  for (i in 1:nrow(validation)){#for each row: check last column- what is the "correct" number?
    for (j in 0:9){
```

```
    if (validation[i,65] == j) { #find "correct" number of the row
      sum_crossEntropy <- sum_crossEntropy + (log(as.numeric(probabilityValidation_CE[i,
j+1])+ 1e-15))
    }
   }
  }
  crossEntropy_k = c(crossEntropy_k,-sum_crossEntropy)
}
plot(crossEntropy_k,  xlab = "K", ylab = "Cross Entropy",col ="darkgreen")
```

```
####################################
#        LAB 1 Assignment 2     #
####################################

# 1
library(dplyr)
library(tidyr)
parkinsons = read.csv("parkinsons.csv")
#scale data:
parkinsons_scale = scale(parkinsons)

n = dim(parkinsons_scale)[1]
set.seed(12345)

#id = 60% of parkinsons
id = sample (1:n, floor(n*0.6))

#train 60% of parkinsons
train = parkinsons_scale[id,]

#test 40% of parkinsons
test = parkinsons_scale[-id,]

##########
```

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

```
#  2  #
#########

#linear regression model of motor_UPDRS as a function of voice characteristics
#exclude age, sex, subject, test_time from model (not releated to voice):

#train:
train_df = as.data.frame(train)
parkinsons_train_LM = lm(motor_UPDRS~. -age -sex -subject. -test_time -motor_UPDRS -
total_UPDRS -1,data=train_df)
pred_train=predict(parkinsons_train_LM) #predicted values from LM
MSE_parkinsons_train = mean((train_df$motor_UPDRS-pred_train)^2) #mean square error
between predicted values and data values

#test
test_df = as.data.frame(test)
parkinsons_test_LM = lm(motor_UPDRS~. -age -sex -subject. -test_time -motor_UPDRS -
total_UPDRS -1,data=test_df)
pred_test=predict(parkinsons_test_LM)#predicted values from LM
MSE_parkinsons_test = mean((test_df$motor_UPDRS-pred_test)^2)#mean square error
between predicted values and data values

#significantly contributing variables
summary(parkinsons_train_LM)

#########
#  3  #
#########

#3a) loglikelihood - calculates the probablity of the parameters fitting the model, higher values
indicates a better fit
loglikelihood_function <- function(data_in,theta,sigma) {
  #divide data into X and Y:
  data_X = data_in[,7:22] #column 7 to 22 (columns related to voice)
  data_Y = data_in[,5] #column 5, motor_UPDRS which is a function of voice
  n = nrow(data_X)
  loglikelihood = (-n*1/2*log(2*pi*sigma^2))-(1/(2*sigma^2))*sum((data_Y - ((data_X) %*%
theta))^2)
  return(loglikelihood)
}

#3b) ridge function - adds a factor to the loglikelihood_function to penalize complexity
ridge_function <- function(data_in,theta,sigma,lambda){
  n = nrow(data_in)
  data_X = data_in[,7:22]
  ridge_penalty = lambda * sum(theta^2) #penalizes complexity
  ridge = -loglikelihood_function(data_in,theta,sigma) + ridge_penalty
  return(ridge)
}

#3c) ridgeOpt - optimizes the results from the ridge function
```

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

```
#help function - create an "array" of the parameters theta and sigma that is used in the ridge
function
help_ridge_function <- function(data_in,empty_array, lambda) {
  theta = empty_array[1:16]
  sigma = empty_array[17]
  return(ridge_function(data_in,theta, sigma, lambda))
}
#ridgeoptfunction - uses help function
ridgeopt_function <- function(data_in,lambda){
  optim(rep(1:17), fn=help_ridge_function, lambda=lambda, data_in = data_in, method =
"BFGS") #rep 17 times to optimize all parameters
}

#3d) degrees of freedom - calculates the number of independent variables that can vary
degrees_of_freedom_function <- function(data_in,lambda){
  data_X = data_in[,7:22]
  I = diag(ncol(data_X))
  #use hat matrix = X * inv( X' * X + lambda * I ) * X' to calculate the degrees of freedom
  hat_matrix = (data_X)%*%solve((t(data_X)%*%data_X + lambda*I))%*%t(data_X)
  degrees_of_freedom = (sum(diag(hat_matrix)))
  return (degrees_of_freedom)
}

##########
#   4   #
##########
train_data_X = train[,7:22]
test_data_X = test[, 7:22]

#compute optimal parameters for lambda = 1, 100 and 1000:
ridge_1 = ridgeopt_function(data=train,lambda=1)
ridge_100 = ridgeopt_function(data=train,lambda=100)
ridge_1000 = ridgeopt_function(data=train,lambda=1000)

#compute predicted values of y lambda = 1, 100 and 1000 using train data
#lambda_train = 1
y1_train = (train_data_X %*%(as.matrix(ridge_1$par))[1:16,])
MSE_train1 = mean((train[,5] - y1_train)^2)
MSE_train1
#lambda_train = 100
y100_train = (train_data_X %*%(as.matrix(ridge_100$par))[1:16,])
MSE_train100 = mean((train[,5] - y100_train)^2)
MSE_train100
#lambda_train = 1000
y1000_train = (train_data_X %*%(as.matrix(ridge_1000$par))[1:16,])
MSE_train1000 = mean((train[,5] - y1000_train)^2)
MSE_train1000

#compute predicted values of y lambda = 1, 100 and 1000 using test data
#lambda_test = 1
y1_test = (test_data_X %*%(as.matrix(ridge_1$par))[1:16,])
```

```
MSE_test1 = mean((test[,5] - y1_test)^2)
MSE_test1
#lambda_test = 100
y100_test = (test_data_X %*%(as.matrix(ridge_100$par))[1:16,])
MSE_test100 = mean((test[,5] - y100_test)^2)
MSE_test100
#lambda_test = 1000
y1000_test = (test_data_X %*%(as.matrix(ridge_1000$par))[1:16,])
MSE_test1000 = mean((test[,5] - y1000_test)^2)
MSE_test1000

#compute degrees of freedom for lambda = 1, 100 and 1000
df_1 = degrees_of_freedom_function(lambda = 1, data = train)
df_100 = degrees_of_freedom_function(lambda = 100, data = train)
df_1000 = degrees_of_freedom_function(lambda = 1000, data = train)
```

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

 # Lab 1, Assignment 3

```
####################################
#        LAB 1 Assignment 3     #
####################################

diabetes = read.csv("pima-indians-diabetes.csv", header = FALSE)

#1: Make a scatterplot showing a plasma glucose concentration on age where obs. are colored by diabetes levels

#x = age (column 8) and y = plasma (column 2), diabetes boolean (column 9)

plot(diabetes[,2], diabetes[,8],xlab = "plasma-glucose-concentration", ylab = "age",col = ifelse(diabetes[,9]==1, "pink", "blue"))
title(main = "Scatterplot - Plasma and Age")

#2 Train a logistic regression model with y = diabetes as target, x1 = plasma glucose conc., x2 = age as features.
#Make a prediction for all obs by using r = 0.5 as classification threshold.

#family bionmial since it is two possible classes
diabetes_glm = glm(diabetes[,9]~ diabetes[,2] + diabetes[,8], family = "binomial")
prob = predict (diabetes_glm, type = "response")
pred = ifelse(prob>0.5, 1, 0) #1 is diabetes and 0 is not
table(diabetes[,9])
coef(diabetes_glm)#gives coeff to the equation
confusionMatrix_diabetes = table(diabetes[,9],pred)
missclass_error_diabetes = (1 - sum(diag(confusionMatrix_diabetes))/nrow(diabetes))

missclass_error_diabetes

plot(diabetes[,2], diabetes[,8], xlab = "plasma-glucose-concentration",ylab = "age", col = ifelse(pred==1, "orange", "blue"))
title(main = "Predicted Scatterplot")
summary(diabetes_glm)

#3 Use model in 2 to report the boundary between the two classes
#add a curve showing this boundary to the scatter plots

#We want to calculate a linear boundary y = kx+m
#We know that theta0 + theta1*plasma + theta2*age  = 0 since t(theta) * X = 0
#Because of this age = (-theta1*plasma-theta0)/theta2
k_diabetes = -coef(diabetes_glm)[2]/coef(diabetes_glm)[3] #slope
m_diabetes = -coef(diabetes_glm)[1]/coef(diabetes_glm)[3] #intercept
abline(m_diabetes,k_diabetes)


#4 Make same kind of plots as in step 2 but use r = 0.2 and r = 0.8
#r = 0.2
```

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

```
diabetes_glm = glm(diabetes[,9]~ diabetes[,2] + diabetes[,8], family = "binomial", data =
diabetes[,c(2,8,9)])
prob = predict (diabetes_glm, type = "response")
pred = ifelse(prob>0.2, 1, 0) #1 is diabetes and 0 is not
table(diabetes[,9])
pred
coef(diabetes_glm)#gives coeff to the equation
confusionMatrix_diabetes = table(diabetes[,9],pred)
missclass_error_diabetes = (1 - sum(diag(confusionMatrix_diabetes))/nrow(diabetes))

missclass_error_diabetes

plot(diabetes[,2], diabetes[,8],xlab = "plasma-glucose-concentration",ylab = "age",  col =
ifelse(pred==1, "orange", "blue"))
title(main = "Predicted Scatter Plot r = 0.2")

#r = 0.8
diabetes_glm = glm(diabetes[,9]~ diabetes[,2] + diabetes[,8], family = "binomial", data =
diabetes[,c(2,8,9)])
prob = predict (diabetes_glm, type = "response")
pred = ifelse(prob>0.8, 1, 0) #1 is diabetes and 0 is not
table(diabetes[,9])
pred
coef(diabetes_glm)#gives coeff to the equation
confusionMatrix_diabetes = table(diabetes[,9],pred)
missclass_error_diabetes = (1 - sum(diag(confusionMatrix_diabetes))/nrow(diabetes))

missclass_error_diabetes

plot(diabetes[,2], diabetes[,8],xlab = "plasma-glucose-concentration",ylab = "age", col =
ifelse(pred==1, "orange", "blue"))
title(main = "Predicted Scatter Plot r = 0.8")

#5 perform basis function expansion trick by computing new features
#z1 = x1^4, z2 = x1^3 *x2, z3 = x1^2 * x2^2, z4 = x1*x2^3, z5 = x2^4

diabetes = diabetes %>%
  mutate(z1 = diabetes$V2^4, z2 = diabetes$V2^3*diabetes$V8, z3 =
diabetes$V2^2*diabetes$V8^2, z4 = diabetes$V2*diabetes$V8^3, z5 = diabetes$V8^4) #adding
columns to dataframe
diabetes_glm_expansion = glm(diabetes[,9]~ diabetes[,2] + diabetes[,8] + diabetes$z1 +
diabetes$z2 + diabetes$z3 + diabetes$z4 + diabetes$z5, family = "binomial")
prob_expansion = predict(diabetes_glm_expansion, type = "response")
pred_expansion = ifelse(prob_expansion>0.5, 1, 0) #1 is diabetes and 0 is not
table(diabetes[,9])
pred_expansion

confusionMatrix_diabetes_expansion = table(diabetes[,9],pred_expansion)
missclass_error_diabetes_expansion = (1 -
sum(diag(confusionMatrix_diabetes_expansion))/nrow(diabetes))
```

Alice Paulsen Alipa232
Unn Zachrison Unnza234
Victora Winqvist Vicwi560

```
confusionMatrix_diabetes_expansion
missclass_error_diabetes_expansion

plot(diabetes[,2], diabetes[,8], xlab = "plasma-glucose-concentration", ylab = "age", col =
ifelse(pred_expansion==1, "orange", "blue"))
title(main = "Predicted Expansion Scatter Plot")

summary(diabetes_glm_expansion)
```