732A99/732A68/ TDDE01 Machine Learning

Computer lab 1 block 1

Johannes Hedström, Mikael Montén & Siddhesh Sreedar

STIMA
Department of Computer and Information Science
Linköpings universitet

2023-11-19

Contents

1 Contribution of work **1**

2 Assignment 1 **1**

2.1 1 . . . 1

2.2 2 . . . 1

2.3 3 . . . 2

2.3.1 Two eights with probability of 1 to be an eight . . . 2

2.3.2 The three eights with the lowest probability of being an eight . . . 4

2.4 4 . . . 6

2.5 5 . . . 7

3 Assignment 2 **8**

3.1 1 . . . 8

3.2 2 . . . 8

3.3 3 . . . 9

3.3.1 a . . . 9

3.3.2 b . . . 9

3.3.3 c . . . 10

3.3.4 d . . . 10

3.4 4 . . . 10

4 Assignment 3. Logistic regression and basis function expansion **11**

4.1 1. . . . 11

4.2 2. . . . 11

4.3 3. . . . 12

4.4 4. . . . 13

4.5 5. . . . 14

5 References **15**

6 Appendix **15**

1 Contribution of work

Johannes did the first assignment, Siddhesh did assignment two and Mikael did the third. Code and answers were first made individually, but we have all(Johannes, Siddhesh and Mikael) read and discussed our answers together afterwards and this lab report is the result of those discussions.

2 Assignment 1

The data file optdigits.csv contains information about normalized bitmaps of handwritten digits from a preprinted form from a total of 43 people. The data were first derived as 32x32 bitmaps which were then divided into nonoverlapping blocks of 4x4 and the number of on pixels are counted in each block. This has generated the resulting image of size 8x8 where each element is an integer in the range 0..16. Accordingly, each row in the data file is a sequence corresponding to 8x8 matrix, and the last element shows the actual digit from 0 to 9.

2.1 1

Import the data into R and divide it into training, validation and test sets (50%/25%/25%) by using the partitioning principle specified in the lecture slides.

2.2 2

Use training data to fit 30-nearest neighbor classifier with function kknn() and kernel="rectangular" from package kknn and estimate

- Confusion matrices for the training and test data (use table())
- Misclassification errors for the training and test data

Comment on the quality of predictions for different digits and on the overall prediction quality

Table 1: Confusion matrix for train data

	0	1	2	3	4	5	6	7	8	9
0	177	0	0	0	1	0	0	0	0	0
1	0	174	9	0	0	0	1	0	1	3
2	0	0	170	0	0	0	0	1	2	0
3	0	0	0	197	0	2	0	1	0	0
4	0	1	0	0	166	0	2	6	2	2
5	0	0	0	0	0	183	1	2	0	11
6	0	0	0	0	0	0	200	0	0	0
7	0	1	0	1	0	1	0	192	0	0
8	0	10	0	1	0	0	2	0	190	2
9	0	3	0	4	2	0	0	2	4	181

Table 2: Confusion matrix for test data

	0	1	2	3	4	5	6	7	8	9
0	97	0	0	0	0	0	1	0	0	0
1	0	91	3	0	0	0	0	0	0	3
2	0	0	93	1	0	0	0	0	1	0
3	0	0	0	95	0	0	0	2	1	0
4	1	0	0	0	89	0	1	5	1	3
5	0	1	0	1	0	79	1	0	0	5
6	0	0	0	0	0	0	94	0	0	0
7	0	2	0	0	0	1	0	91	1	0
8	0	3	0	1	0	0	1	0	86	0
9	0	0	0	4	0	0	0	2	1	94

Table 3: Misclassification errors for the training and test data

Train	Test
0.0423862	0.0491632

The model predicts the number quite good, but has some problems to differentiate between eights and ones, fives and nines and four and sevens which you can see in both the confusion matrices for the training and test data. The problem with eight and ones seem to mostly be a problem for the training data though, which might indicate that these numbers are written in an outlying way.

As the misclassification error is low, and the difference between the train and test misclassification error is small, therefore the quality of the model prediction is quite good.

2.3 3

Find any 2 cases of digit “8” in the training data which were easiest to classify and 3 cases that were hardest to classify (i.e. having highest and lowest probabilities of the correct class). Reshape features for each of these cases as matrix 8x8 and visualize the corresponding digits (by using e.g. heatmap() function with parameters Colv=NA and Rowv=NA) and comment on whether these cases seem to be hard or easy to recognize visually

2.3.1 Two eights with probability of 1 to be an eight

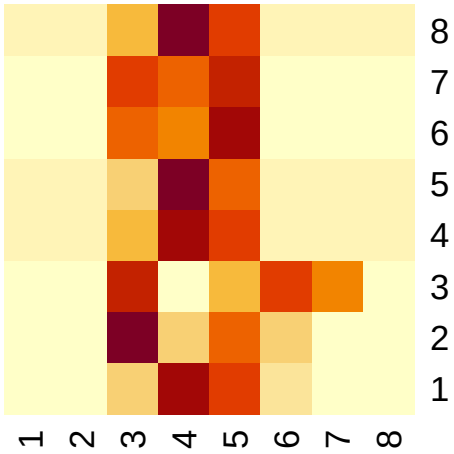
Table 4: Two with 1 as probability of being an eight

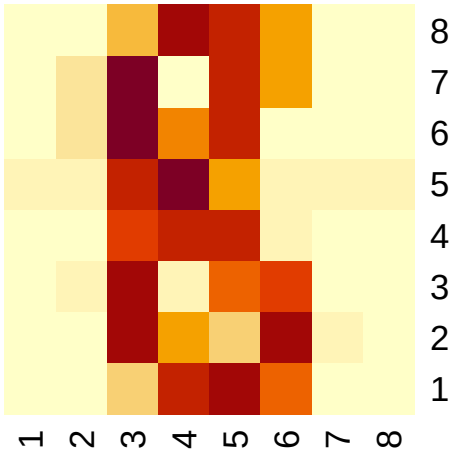
	True	Prob	Fitted
34	8	1	8
209	8	1	8

Table 5: The three lowest probabilities of being an eight

	True	Prob	Fitted
229	8	0.2333333	1
1663	8	0.1666667	1
1624	8	0.1000000	6

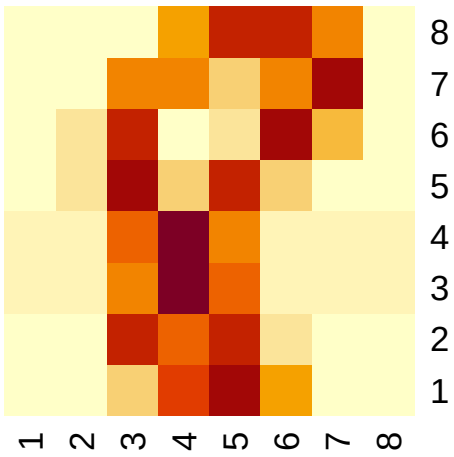
In these two tables we can see the highest and lowest probabilities and the rownames for each observation which we use as indices to get the right observations for the heatmaps.

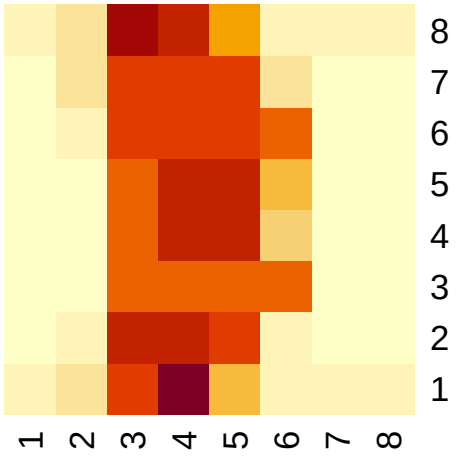


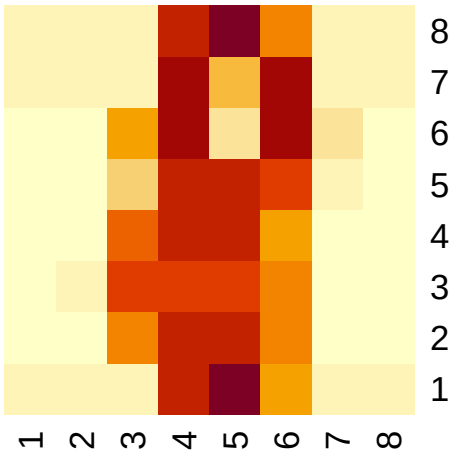


Its clear that these two numbers are written as an eight.

2.3.2 The three eights with the lowest probability of being an eight



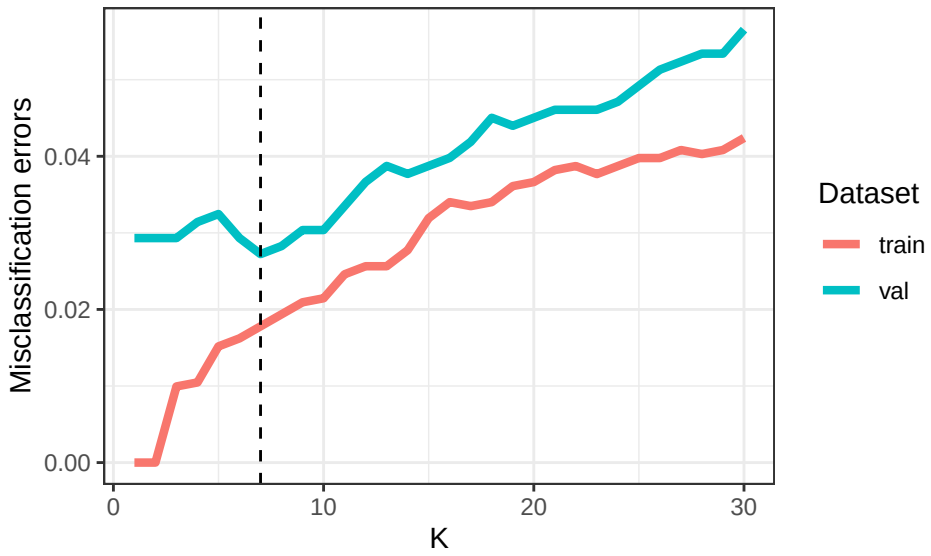

8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8


8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8

Its clearly harder visually to see an 8 in the last three heatmaps, its not that strange that the model is having problems to classify these 3 numbers as an 8(which could be seen in the confusing matrix for the training data).

2.4 4

Fit a K-nearest neighbor classifiers to the training data for different values of K = 1,2, ..., 30 and plot the dependence of the training and validation misclassification errors on the value of K (in the same plot). How does the model complexity change when K increases and how does it affect the training and validation errors? Report the optimal K according to this plot. Finally, estimate the test error for the model having the optimal K, compare it with the training and validation errors and make necessary conclusions about the model quality



The complexity of the model decreases as the k increases, as each cluster of neighbors is less detailed when k grows larger and the model is less likely to overfit(can be seen for low K values in the graph), but when k is getting too large the model is underfitting instead which can be seen as the error is increasing for both the training and validation data.

The misclassification error follow the 2 datasets very similarly, and is lowest for the validation-data at k = 7, which then should be the optimal K.

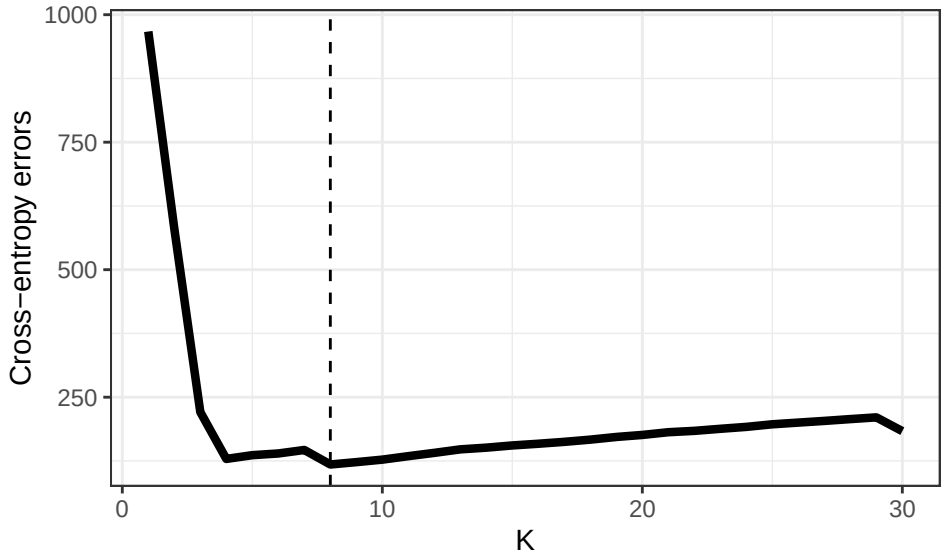
Table 6: Test misclassification error for optimal k

Test
0.0387

The test error is higher than both the training and validation error(both below 0.03 for K=7), this is often expected as we haven´t picked our model based on the results from this dataset, but it might indicate that K = 7 isnt the optimal K when using the model on new data because we might be overfitting the model slightly. K = 7 could produce the best model for our validation data but not generally when classifying written numbers for example.

2.5 5

Fit K-nearest neighbor classifiers to the training data for different values of K = 1,2, ..., 30, compute the error for the validation data as cross-entropy (when computing log of probabilities add a small constant within log, e.g. 1e-15, to avoid numerical problems) and plot the dependence of the validation error on the value of K. What is the optimal K value here? Assuming that response has multinomial distribution, why might the cross-entropy be a more suitable choice of the error function than the misclassification error for this problem?



Here we get K = 8 as the optimal K value for the model, so it slightly differs to the result in 1.4.

Cross-entropy is better for this problem as we have several classes and our previous results(for k=30) showed that our model had problems classifying some numbers more than other. It penalizes models more(than misclassification error) for predictions that are incorrect and provides a more detailed measure of the difference between predicted and true class probabilities, using this to chose K ultimately gives us a model with well calibrated probabilities for each digit.

3 Assignment 2

The data file parkinson.csv is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson's disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring. The purpose is to predict Parkinson's disease symptom score (motor UPDRS) from the following voice characteristics:

- Jitter(%),Jitter(Abs),Jitter:RAP,Jitter:PPQ5,Jitter:DDP - Several measures of variation in fundamental frequency
- Shimmer,Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,Shimmer:APQ11,Shimmer:DDA - Several measures of variation in amplitude
- NHR,HNR - Two measures of ratio of noise to tonal components in the voice
- RPDE - A nonlinear dynamical complexity measure
- DFA - Signal fractal scaling exponent
- PPE - A nonlinear measure of fundamental frequency variation

3.1 1

Divide it into training and test data (60/40) and scale it appropriately. In the coming steps, assume that motor_UPDRS is normally distributed and is a function of the voice characteristics, and since the data are scaled, no intercept is needed in the modelling.

3.2 2

Compute a linear regression model from the training data, estimate training and test MSE and comment on which variables contribute significantly to the model.

```
## [1] 0.8785431
```

```
## [1] 0.9354477
```

```
##
## Call:
## lm(formula = motor_UPDRS ~ . - 1, data = trainS)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.0255 -0.7363 -0.1087  0.7333  2.1960
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## Jitter...       0.186931   0.149561   1.250 0.211431
## Jitter.Abs.    -0.169609   0.040805  -4.157 3.31e-05 ***
## Jitter.RAP     -5.269544  18.834160  -0.280 0.779658
## Jitter.PPQ5    -0.074568   0.087766  -0.850 0.395592
## Jitter.DDP      5.249558  18.837525   0.279 0.780510
## Shimmer         0.592436   0.205981   2.876 0.004050 **
## Shimmer.dB.    -0.172655   0.139316  -1.239 0.215315
## Shimmer.APQ3   32.070932  77.159242   0.416 0.677694
## Shimmer.APQ5   -0.387507   0.113789  -3.405 0.000668 ***
## Shimmer.APQ11   0.305546   0.061236   4.990 6.34e-07 ***
## Shimmer.DDA   -32.387241  77.158814  -0.420 0.674695
## NHR            -0.185387   0.045567  -4.068 4.84e-05 ***
## HNR            -0.238543   0.036395  -6.554 6.41e-11 ***
## RPDE            0.004068   0.022664   0.179 0.857556
## DFA            -0.280318   0.020136 -13.921  < 2e-16 ***
## PPE             0.226467   0.032881   6.887 6.70e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9394 on 3509 degrees of freedom
## Multiple R-squared:  0.1212, Adjusted R-squared:  0.1172
## F-statistic: 30.25 on 16 and 3509 DF,  p-value: < 2.2e-16
```

Upon comparing the t-value and p-value, the variables that are staistically significant are: Jitter.Abs. ,Shimmer , Shimmer.APQ5, Shimmer.APQ11, NHR, HNR, DFA, PPE

3.3 3

Implement 4 following functions by using basic R commands only (no external packages):

3.3.1 a

Loglikelihood function that for a given parameter vector θ and dispersion σ computes the log-likelihood function $logP(T|\theta, \sigma)$ for the stated model and the training data

3.3.2 b

Ridge function that for given vector θ, scalar σ and scalar λ uses function from 3a and adds up a Ridge penalty $\lambda||\theta||^2$ 2 to the minus log-likelihood

3.3.3 c

RidgeOpt function that depends on scalar λ , uses function from 3b and function optim() with method="BFGS" to find the optimal θ and σ for the given λ.

3.3.4 d

DF function that for a given scalar λ computes the degrees of freedom of the Ridge model based on the training data.

3.4 4

By using function RidgeOpt, compute optimal θ parameters for $\lambda = 1$, $\lambda = 100$ and $\lambda = 1000$. Use the estimated parameters to predict the motor_UPDRS values for training and test data and report the training and test MSE values. Which penalty parameter is most appropriate among the selected ones? Compute and compare the degrees of freedom of these models and make appropriate conclusions.

```
## [1] 0.8786272
```

```
## [1] 0.8844056
```

```
## [1] 0.9211175
```

```
## [1] 0.934998
```

```
## [1] 0.9323296
```

```
## [1] 0.9539457
```

```
## [1] 13.86074
```

```
## [1] 9.924887
```

```
## [1] 5.643925
```

For the training data, we can see that for = 1 , the MSE is the least while for the test data, = 1 and = 100 are almost the same but lower than = 1000. So the most appropropiate one is = 1.

The DoF for = 1 is the highest while for = 1000, it is the lowest. This implies that there has been more regularization for =1000 which makes the model more robust and less sensitive to individual data points while =1 has lesser regularization which makes the model more flexible and fits the training data better.

So based on the above analysis of DoF and the MSE, =1 would be ideal.

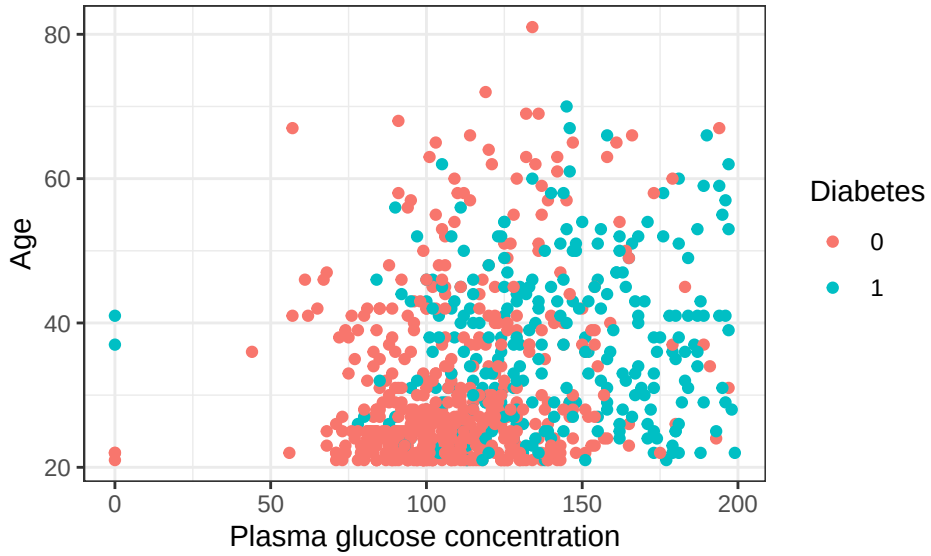
4 Assignment 3. Logistic regression and basis function expansion

This assignment involves a data file about the onset of diabetes within 5 years in Pima Indians given medical details, the variables are (in order);

1. Number of times pregnant.
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test.
3. Diastolic blood pressure (mm Hg).
4. Triceps skinfold thickness (mm).
5. 2-Hour serum insulin (mu U/ml).
6. Body mass index (weight in kg/(height in m)^2).
7. Diabetes pedigree function.
8. Age (years).
9. Diabetes (0=no or 1=yes).

4.1 1.

Make a scatterplot showing a Plasma glucose concentration on Age where observations are colored by Diabetes levels. Do you think that Diabetes is easy to classify by a standard logistic regression model that uses these two variables as features? Motivate your answer.



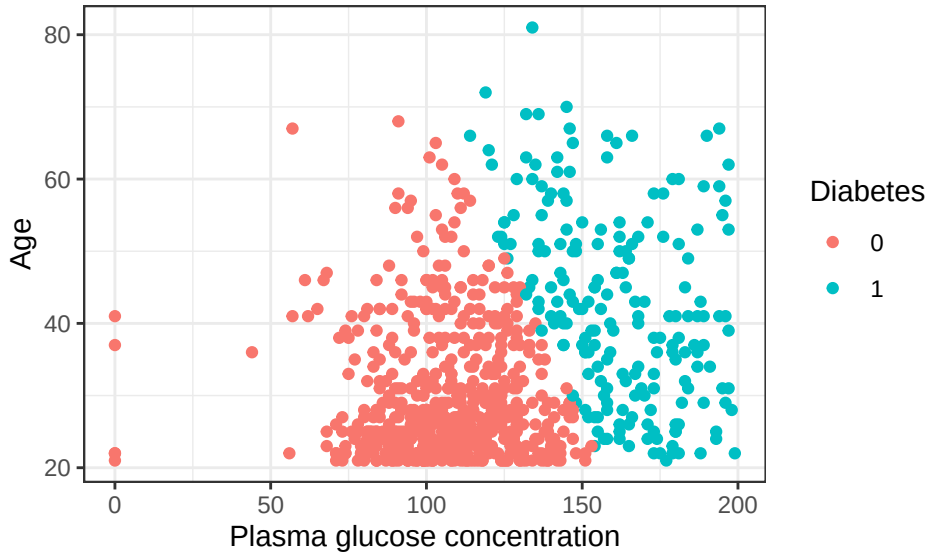
A standard logistic regression model would not classify diabetes well using these two variables. Reason being that the two groups are very mixed, and observations from both groups are scattered at all levels. This would result in the r = 0.5 classifying wrong for several observations.

4.2 2.

Train a logistic regression model with y =Diabetes as target x_1 =Plasma glucose concentration and x_2 =Age as features and make a prediction for all observations by using r = 0.5 as the classification threshold. Report

the probabilistic equation of the estimated model (i.e., how the target depends on the features and the estimated model parameters probabilistically). Compute also the training misclassification error and make a scatter plot of the same kind as in step 1 but showing the predicted values of Diabetes as a color instead. Comment on the quality of the classification by using these results.

```
## (Intercept)      plasma         age
## -6.01839056  0.03368988  0.03633943
```



Intercept	Beta1	Beta2	Missclassification error	Total sum	Offdiag sum
-6.018391	0.0336899	0.0363394	0.2591146	768	199

The probabilistic equation of the estimated model: $p(x) = P(Y = 1 \mid x) = \frac{1}{1+\exp(-\beta_0-\beta_i x)} = 1/(1 + exp(-(-6.01839056) - 0.03368988 \cdot plasma - 0.03633943 \cdot age)))$

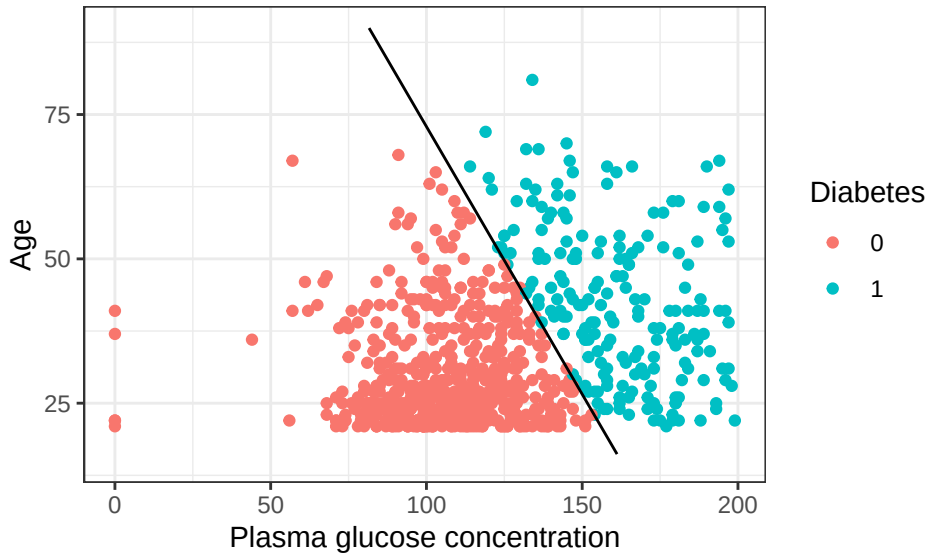
It is very easy to see how the binary classifier has worked. The threshold of r = 0.5 is portrayed as the hard line in where the predictions change from no predicted diabetes to predicted diabetes.

This simple classifying would not be good practice to use for classifying patients due to the misclassification error being 25.91%, meaning nearly 1/4th of all cases are either false positives or false negatives.

4.3 3.

Use the model estimated in step 2 to a) report the equation of the decision boundary between the two classes b) add a curve showing this boundary to the scatter plot in step 2. Comment whether the decision boundary seems to catch the data distribution well.

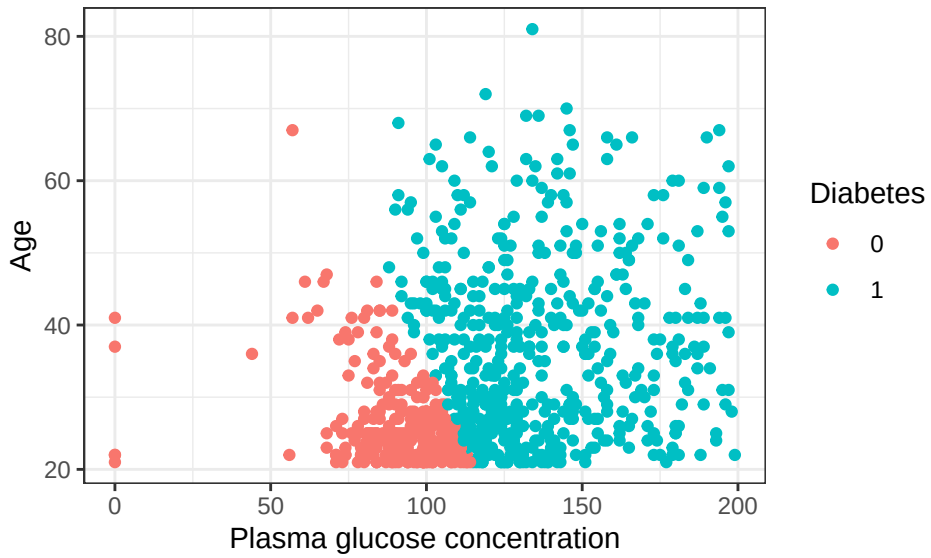
The equation of the decision boundary, alas when the two classes are equally probable is where $g(x) = 1 - g(x)$ which for logistic regression means where $\theta^T x = 0$ (Lindholm et al. 2022), or for our case where $(-6.01839056) - 0.03368988 \cdot plasma - 0.03633943 \cdot age = 0$.

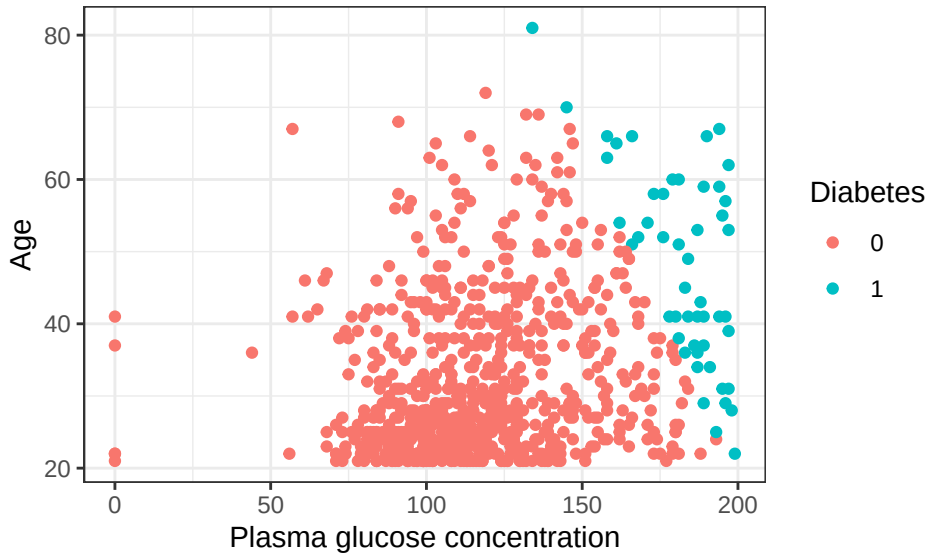


The plotted decision boundary (Sahu 2016) runs right between the classes, as expected.

4.4 4.

Make same kind of plots as in step 2 but use thresholds $r = 0.2$ *and* $r = 0.8$*. By using these plots, comment on what happens with the prediction when value* r *value changes.*

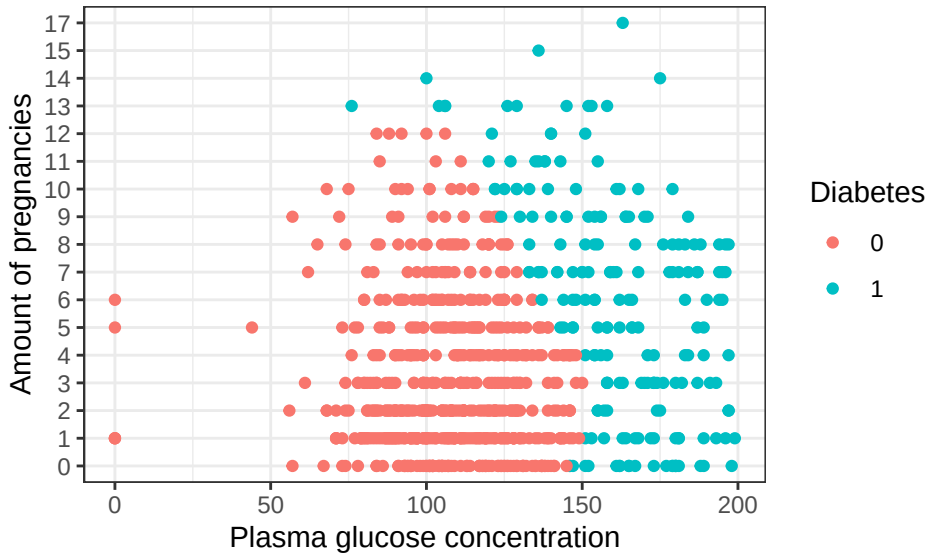




When r changes, the subsequent classification either increases or decreases the weight for the classes. A lower threshold for r gives a plot that classifies more observations as having diabetes, and a higher threshold classifies more observations as non-diabetes.

4.5 5.

- Perform a basis function expansion trick by computing new features $z_1 = x_1^4, z_2 = x_1^3 x_2, z_3 = x_1^2 x_2^2, z_4 = x_1 x_2^3, z_5 = x_2^4$, adding them to the data set and then computing a logistic regression model with y as target and $x_1, x_2, z_1, ...z_5$ as features. Create a scatterplot of the same kind as in step 2 for this model and compute the training misclassification rate. What can you say about the quality of this model compared to the previous logistic regression model? How have the basis expansion trick affected the shape of the decision boundary and the prediction accuracy?*



Intercept	Beta1	Beta2	Missclassification error	Total sum	Offdiag sum
-6.018391	0.0336899	0.0363394	0.2591146	768	199

This model is worse than the first one. Reason being that the training misclassification error is pretty much the same, however the model is severely more complicated than the first one and therefore complicates the relationship between dependent and response variables without any true benefit, only making it harder to interpret the coefficients. Due to the non-linear nature of the transformations, the decision boundary appears to be non-linear.

5 References

Lindholm, A., N. Wahlström, F. Lindsten, and T. B. Schön. 2022. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press.

Sahu, Abhishek. 2016. "Why Not Stat Function?" 2016. https://stackoverflow.com/a/37896739.

6 Appendix

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning=FALSE, fig.width = 5, fig.height = 3, fig.a
set.seed(12345)
# packages
library(kknn)
library(knitr)
library(ggplot2)
library(cowplot)
```

```r
library(dplyr)
library(caret)

## 1.1
optdigits <- read.csv("optdigits.csv")

set.seed(12345)

# number of observations
n <- nrow(optdigits)
set.seed(12345)
# sampling out train, val and testdata
id <- sample(1:n, floor(n*0.5))

train_1 <- optdigits[id,] # traindata

id1 <- setdiff(1:n, id)

set.seed(12345)

id2 = sample(id1, floor(n*0.25))

val_1 = optdigits[id2,] # validation data

id3 <- setdiff(id1, id2)

test_1 <- optdigits[id3,] # testdata




# training the model with k =30
knn_mod <- train.kknn(as.factor(X0.26)~. , train_1, ks=30, kernel = "rectangular")



# confusion matrix for train data

knitr::kable(table(train_1$X0.26,predict(knn_mod, train_1)), caption = 'Confusion matrix for train data'

# confusion matrix for test data
knitr::kable(table(test_1$X0.26,predict(knn_mod, test_1)), caption = 'Confusion matrix for test data')
# missclass train
m1 <- 1 - ( sum(diag(table(predict(knn_mod, train_1), train_1$X0.26)))/sum(table(predict(knn_mod, train_



# Missclass test
m2 <- 1 - ( sum(diag(table(predict(knn_mod, test_1), test_1$X0.26)))/sum(table(predict(knn_mod, test_1),
```

```r
knitr::kable(data.frame('Train'=m1, 'Test'=m2), caption = 'Misclassification errors for the training and


knn_mod <- kknn(as.factor(X0.26)~. , train=train_1,test=train_1, k=30, kernel = "rectangular")
index <- which(train_1$X0.26==8)

df <- data.frame("True"=train_1$X0.26[index],"Prob"=knn_mod$prob[index,9],"Fitted"=knn_mod$fitted.values
rownames(df) <- index
# two 8s with prob = 1

map1 <- train_1[34,-ncol(train_1)]
map2 <- train_1[209,-ncol(train_1)]

# the three lowest
map3 <- train_1[1624,-ncol(train_1)]
map4 <- train_1[1663,-ncol(train_1)]
map5 <- train_1[229,-ncol(train_1)]

mat1 <- matrix(as.numeric(map1), ncol=8)
mat2 <- matrix(as.numeric(map2), ncol=8)

mat3 <- matrix(as.numeric(map3), ncol=8)
mat4 <- matrix(as.numeric(map4), ncol=8)
mat5 <- matrix(as.numeric(map5), ncol=8)


# the two best 8s
#heatmap(t(mat1),Colv=NA, Rowv=NA)

df <- df[order(df$Prob, decreasing = TRUE),]

knitr::kable(head(df,2), caption = "Two with 1 as probability of being an eight")
knitr::kable(tail(df,3), caption = "The three lowest probabilities of being an eight")

plot_grid(heatmap(t(mat1),Colv=NA, Rowv=NA),heatmap(t(mat2),Colv=NA, Rowv=NA))
plot_grid(heatmap(t(mat3),Colv=NA, Rowv=NA),heatmap(t(mat4),Colv=NA, Rowv=NA),heatmap(t(mat5),Colv=NA, R
#heatmap(t(mat3),Colv=NA, Rowv=NA)

#heatmap(t(mat4),Colv=NA, Rowv=NA)
#heatmap(t(mat5),Colv=NA, Rowv=NA)

miss_t <- c()
miss_v <- c()
i=1
# looping from 1 to 30 and fitting a new model for each i, with k=1
for (i  in 1:30) {
```

```
  knn_loop <- train.kknn(as.factor(X0.26)~. , train_1, ks=i, kernel = "rectangular")
  miss_t[i] <- 1 - ( sum(diag(table(predict(knn_loop, train_1), train_1$X0.26)))/sum(table(predict(knn_l
  miss_v[i] <- 1 - ( sum(diag(table(predict(knn_loop, val_1), val_1$X0.26)))/sum(table(predict(knn_loop,

}

      # creating a df to make the plot in ggplot
df <- data.frame('K'=rep(1:30,2), 'Dataset'=c(rep('train', 30), rep('val', 30)), 'MissC' = c(miss_t, mis

# creating the plot
ggplot(df, aes(x=K,y=MissC, color=Dataset)) + geom_line(size=1.5) + theme_bw() +
            ylab('Misclassification errors') + geom_vline(xintercept=which.min(miss_v),linetype='dashed'

best_mod <- train.kknn(as.factor(X0.26)~. , train_1, ks=7, kernel = "rectangular")

m3 <- 1 - ( sum(diag(table(predict(best_mod, test_1), test_1$X0.26)))/sum(table(predict(best_mod, test_1

knitr::kable(data.frame('Test'= m3), caption = 'Test misclassification error for optimal k', digits=4)


cross_i <- c()

# looping from 1 to 30 and fitting a new model for each i, with k=1
for (i  in 1:30) {

  knn_loop <- kknn(as.factor(X0.26)~. , train_1,test = val_1, k=i, kernel = "rectangular")
  x <- 0
  t <- 1

  while(t <= length(val_1$X0.26)){
      for(j in val_1$X0.26){

  x <-  (x + log(knn_loop$prob[t,colnames(knn_loop$prob) == as.character(j)] + 1e-15))

  t <- t + 1
      }

   }

  cross_i[i] <- -x

}




# creating a df to make the plot in ggplot
df <- data.frame('K'=c(1:30), 'Cross-entropy' = cross_i)
```

```r
# creating the plot
ggplot(df, aes(x=K,y=cross_i)) + geom_line(linewidth=1.5) + theme_bw() +
  ylab('Cross-entropy errors') + geom_vline(xintercept=which.min(cross_i),linetype='dashed')


data<-read.csv("parkinsons.csv")

#dependent variable: motor_UPDRS
#removing variables that are not required
data1<-data[,-c(1:4)]
data1<-data1[,-2]

#Splitting the dataset into test and train
set.seed(12345)
n<- nrow(data1)
id<-sample(1:n,floor(n*0.6))
train<-data1[id,]
test<-data1[-id,]

#Scaling the data
library(caret)

scaler<- preProcess(train)
trainS<-predict(scaler,train)
testS<-predict(scaler,test)
#creating the linear regression model

model<-lm(motor_UPDRS ~ . -1 , data = trainS)


# MSE of the model
MSE_model<-mean(model$residuals^2)  #0.8785431

#train data
prediction<- predict(model, newdata = trainS)

mean((trainS$motor_UPDRS-prediction)^2)   #0.8785431


#test data
prediction2<- predict(model, newdata = testS)

mean((testS$motor_UPDRS-prediction2)^2)   #0.9354477


#The variables that contribute significantly to the model:
summary(model)
x<-as.matrix(trainS[,-1])
n<-nrow(trainS)
```

```r
y <- as.matrix(trainS$motor_UPDRS)

loglikelihood <- function(theta , dispersion){
  #theta<-t(as.matrix(theta))

  x<-as.matrix(trainS[,-1])
  n<-nrow(trainS)
  y <- as.matrix(trainS$motor_UPDRS)

  #values<-c()
  #for (i in 1:nrow(trainS)){
  #  b<-as.matrix(a[i,])
  #  values[i]<- (trainS$motor_UPDRS[i]- theta%*%(t(b)))^2
  #}

  final<- (-n / 2 * log(2 * pi * dispersion^2) - (1 / (2 * dispersion^2)) * sum((y - x %*% theta)^2))

  #value<-sum(as.matrix(trainS$motor_UPDRS)-theta%*%t(b))^2


  #c<-(value)/(2*(dispersion^2))

  #d<- n * log(sqrt(2*pi)*dispersion)

  return(final)

}

ridge <- function(theta, lambda){

  sigma <- tail(theta,1)
  theta <- theta[-17]

  value<- - loglikelihood(theta,sigma) #What is the meaning of minus log-likelihood? When to use what?

  a<- lambda * sum(theta^2)

  return(value + a)
}
RidgeOpt<- function(lambda){

  #intial_theta<-rep(0,16)
  #intial_dispersion<-0.5

  #opt<- function(param){
  #  ridge(param[1:length(param) - 1],param[length(param)],lambda)
  #}

  return(optim(rep(1,17),fn = ridge,lambda = lambda, method = "BFGS"))
```

```
}
I <- as.matrix(diag(16))

DF <- function(lambda){
  r<-x%*%solve((t(x)%*%x + lambda*I),t(x))

  sum(diag(r))

  }
opt_1<-RidgeOpt(1)
opt_2<-RidgeOpt(100)
opt_3<-RidgeOpt(1000)


pred_1_train<-as.matrix(trainS[,-1])%*%as.matrix(opt_1$par[-17])
pred_1_test<-as.matrix(testS[,-1])%*%as.matrix(opt_1$par[-17])
pred_2_train<-as.matrix(trainS[,-1])%*%as.matrix(opt_2$par[-17])
pred_2_test<-as.matrix(testS[,-1])%*%as.matrix(opt_2$par[-17])
pred_3_train<-as.matrix(trainS[,-1])%*%as.matrix(opt_3$par[-17])
pred_3_test<-as.matrix(testS[,-1])%*%as.matrix(opt_3$par[-17])



mean((trainS$motor_UPDRS-pred_1_train)^2)
mean((trainS$motor_UPDRS-pred_2_train)^2)
mean((trainS$motor_UPDRS-pred_3_train)^2)

mean((testS$motor_UPDRS-pred_1_test)^2)
mean((testS$motor_UPDRS-pred_2_test)^2)
mean((testS$motor_UPDRS-pred_3_test)^2)

DF(1)
DF(100)
DF(1000)

set.seed(12345)
df3 <- read.csv("pima-indians-diabetes.csv", header = FALSE)
colnames(df3) <- c("preg", "plasma", "bloodp", "triceps", "serum", "bmi", "pedigree", "age", "diabetes")
p31 <- ggplot(df3, aes(x = plasma, y = age, color = as.factor(diabetes))) + geom_point() + theme_bw() +
  labs(y = "Age", x = "Plasma glucose concentration", color = "Diabetes")
p31
set.seed(12345)
# split data into train/test
n <- nrow(df3)
id <- sample(1:n, floor(n*0.70))
train <- df3[id,]
test <- df3[-id,]

# Training logistic regression model
```

```r
logmod <- glm(diabetes ~ plasma + age, data = train, family = "binomial")

# Probabilistic equation of the estimated model
coef(logmod)

# Prediction for all observations
diabetesPred <- predict(logmod, newdata = df3, type = "response")
diabetesPred05 <- ifelse(diabetesPred >= 0.5,1,0)
df3$diabetesPred <- diabetesPred05

# Training misclassification error
# misclassification rate = sum(off diag) / sum(all)
log_confmat <- table(df3$diabetes, df3$diabetesPred)
totsum <- sum(log_confmat) # sum all
offdiagsum <- log_confmat[1,2] + log_confmat[2,1] # sum off diag

log_misclass_error <- (offdiagsum) / totsum # misclassification error

# Scatter plot of same kind as in step 1 but predicted values instead
p32 <- ggplot(df3, aes(x = plasma, y = age, color = as.factor(diabetesPred))) + geom_point() + theme_bw(
  labs(y = "Age", x = "Plasma glucose concentration", color = "Diabetes")
p32

result3 <- as.data.frame(cbind(coef(logmod)[1], coef(logmod)[2], coef(logmod)[3], log_misclass_error, to
colnames(result3) <- c("Intercept", "Beta1", "Beta2", "Missclassification error", "Total sum", "Offdiag
knitr::kable(result3, row.names = FALSE)
# Plot the decision boundary
p33 <- p32 + stat_function(fun = function(x){((-coef(logmod)[1]-coef(logmod)[2]*x)) / coef(logmod)[3]},
p33
# Prediction for all observations using r = 0.2 and r = 0.8
diabetesPred02 <- ifelse(diabetesPred >= 0.2,1,0)
df3$diabetesPred02 <- diabetesPred02

diabetesPred08 <- ifelse(diabetesPred >= 0.8,1,0)
df3$diabetesPred08 <- diabetesPred08


p341 <- ggplot(df3, aes(x = plasma, y = age, color = as.factor(diabetesPred02))) + geom_point() + theme_
  labs(y = "Age", x = "Plasma glucose concentration", color = "Diabetes")
p341

p342 <- ggplot(df3, aes(x = plasma, y = age, color = as.factor(diabetesPred08))) + geom_point() + theme_
  labs(y = "Age", x = "Plasma glucose concentration", color = "Diabetes")
p342
# Computing new features
df3$z1 <- df3[,1]^4
df3$z2 <- df3[,1]^3 * df3[,2]
df3$z3 <- df3[,1]^2 * df3[,2]^2
df3$z4 <- df3[,1] * df3[,2]^3
```

```r
df3$z5 <- df3[,2]^4

logmod_exp <- glm(diabetes ~ preg + plasma + z1 + z2 + z3 + z4 + z5, data = df3, family = "binomial")


# Prediction for all observations for expansion trick
diabetesPred_exp <- predict(logmod_exp, newdata = df3, type = "response")
diabetesPred_exp <- ifelse(diabetesPred_exp >= 0.5,1,0)
df3$diabetesPred_exp <- diabetesPred_exp

# Training misclassification error for expansion trick
# misclassification rate = sum(off diag) / sum(all)
log_confmat_exp <- table(df3$diabetes, df3$diabetesPred_exp)
totsum_exp <- sum(log_confmat_exp) # sum all
offdiagsum_exp <- log_confmat_exp[1,2] + log_confmat_exp[2,1] # sum off diag

log_misclass_error_exp <- (log_confmat_exp[1,2] + log_confmat_exp[2,1]) / sum(log_confmat_exp)

# Scatter plot of same kind as in step 1 but predicted values instead
p35 <- ggplot(df3, aes(x = plasma, y = as.factor(preg), color = as.factor(diabetesPred_exp))) + geom_poi
  labs(y = "Amount of pregnancies", x = "Plasma glucose concentration", color = "Diabetes")
p35

result3_exp <- as.data.frame(cbind(coef(logmod_exp)[1], coef(logmod_exp)[2], coef(logmod_exp)[3], log_mi
colnames(result3_exp) <- c("Intercept", "Beta1", "Beta2", "Missclassification error", "Total sum", "Offd
knitr::kable(result3, row.names = FALSE)
```