Group A 3 Project Report

Akshath(akssr921), Aswath(aswma317), Varun(varsi146)

2022-11-17

Statement of Contribution

For Lab1, we decided to split the three assignments equally, Akshath completed Assignment 1, Aswath completed Assignment 2 and Varun completed Assignment 3, after which, for verification sake, we completed each other's assignments as well and validated our findings. The report was also compiled by three of us, with each handling their respective assignments.

Assignment 1: Handwritten digit recognition with K-nearest neighbors

Question 1

Please see the code in appendix

Question 2

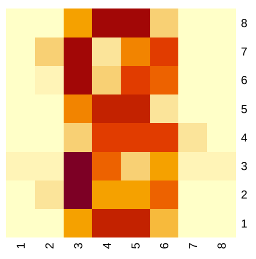
As we can see the confusion matrices for both test data and train data predictions in which rows are actual values and columns are predicted values. Diagonal values shows the correct predictions. Most incorrect predictions for test and train data is digit 9 and most correct prediction is zero. We can also see the misclassification error for train and test data below. Test data error(5.3%) is slightly higher than the training error(4.5). From this two factors we can draw the conclusion that overall prediction quality is pretty good.

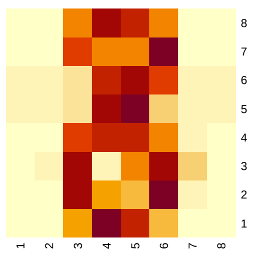
```
##    pred_test
##       0   1   2   3   4   5   6   7   8   9
##   0  77   0   0   0   1   0   0   0   0   0
##   1   0  81   2   0   0   0   0   0   0   3
##   2   0   0  98   0   0   0   0   0   3   0
##   3   0   0   0 107   0   2   0   0   1   1
##   4   0   0   0   0  94   0   2   6   2   5
##   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
```

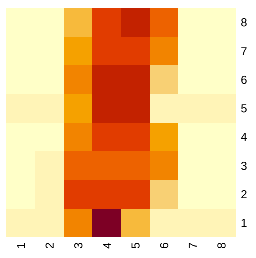
```
##    pred_train
##       0   1   2   3   4   5   6   7   8   9
##   0 202   0   0   0   0   0   0   0   0   0
##   1   0 179  11   0   0   0   0   1   1   3
##   2   0   1 190   0   0   0   0   1   0   0
##   3   0   0   0 185   0   1   0   1   0   1
##   4   1   3   0   0 159   0   0   7   1   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

## Misclassification error on test data : 0.05329154

## Misclassification error on train data : 0.04500262
```

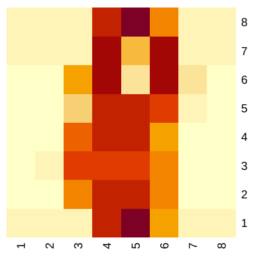
Question 3

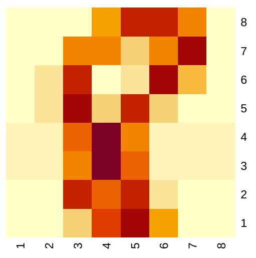
The heatmap for the easiest and hard cases to predict 8 where 8 is the correct class are shown below. First two cases are easiest to classify which we can see from the graph that looks exactly like 8. Next three cases are hardest to classify which has probabilities(0.1,0.13,0.16) very low. the heatmap looks nowhere near the digit 8. Two of the hard cases looks like 1 visually and one looks like alphabet c.







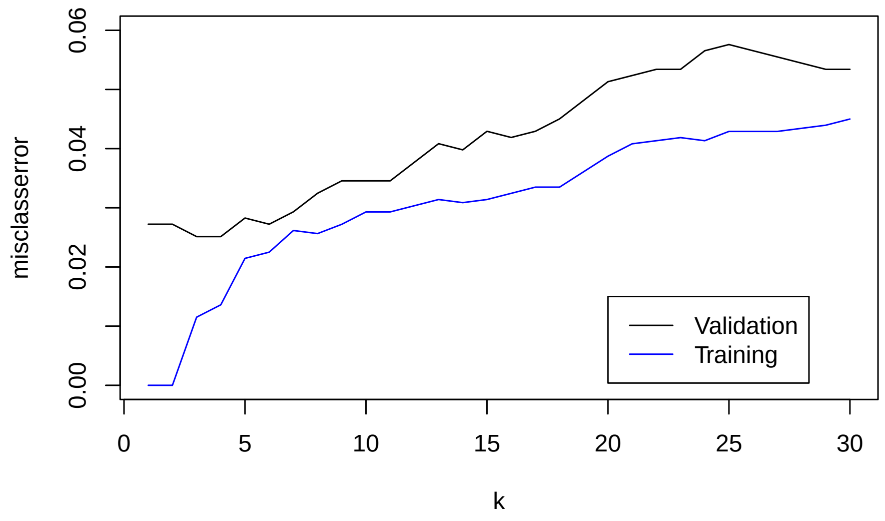




Question 4

For KNN model complexity is high when k is less which we mean the ability of a method to adapt to patterns in the training data and the complexity reduces as the K is increased.From the plot, we can observe that for very less K the generalisation gap is large but training error is less, as K increases we find the optimal area where validation error(E new) and generalisation gap are less . Further more, if we increase K beyond 7 the **miscalssification error** on training and validation data increases and also the generalisation gap increases.To conclude, KNN model complexity is high when K is less.

The optimal K is 4 as the validation error (E new) is less and also the generalisation gap is small.



After predicting the digits using the optimal K we got above we can see from the table the predictions on test data improved when compared to the predictions in subquestion 2. The incorrect predictions of 9 is reduced and also we can observe that misclassification error is reduced to 2.5% from 5.3%. The model quality is improved a lot as the misclassification error was reduced by 50%.

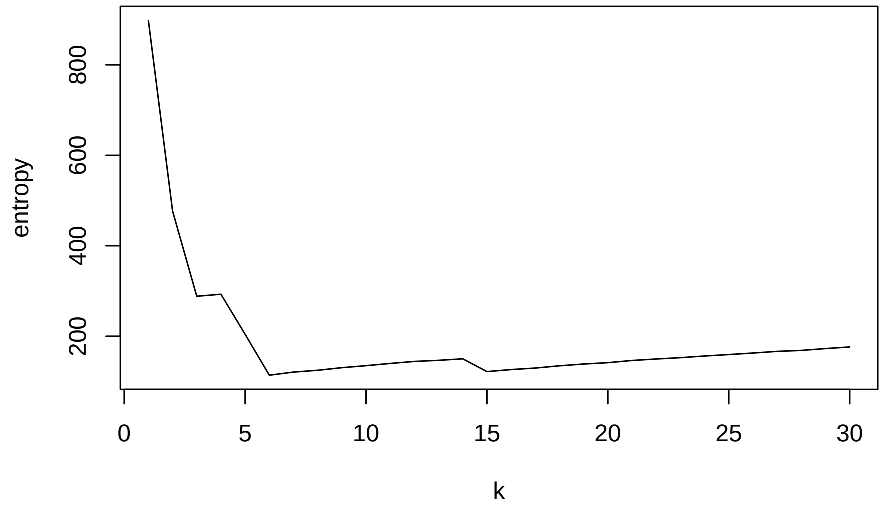
```
##    optim_pred
##       0   1   2   3   4   5   6   7   8   9
##   0  77   0   0   0   1   0   0   0   0   0
##   1   0  84   1   0   0   0   0   0   0   1
##   2   0   1 100   0   0   0   0   0   0   0
##   3   0   1   0 109   0   1   0   0   0   0
##   4   0   0   0   0 103   0   1   2   0   3
##   5   0   0   0   0   0 101   0   0   0   2
##   6   0   0   0   0   0   0  90   0   0   0
##   7   0   0   0   1   1   0   0 110   0   0
##   8   0   4   0   0   0   0   0   0  74   0
##   9   0   0   0   1   0   3   0   0   0  85
```

```
## Misclassification error on test data for optimal K : 0.02507837
```

Question 5

There are two reasons why misclassification error is not suitable. First, using a different loss function can result in a model that generalizes better from the training data. The second reason for not using misclassification loss as the training objective, which is also important, is that it would result in a cost function that is piece wise constant. From a numerical optimization perspective, this is a difficult objective since the gradient is zero everywhere, except where it is undefined.

Cross entropy loss is better because it is calculated by maximum likelihood estimation in a multinomial distribution. Another reason to use cross entropy is that this results in convex loss function, of which global minimum will be easy to find.



Assignment 2: Linear Regression and Ridge Regression

Question 2

a) Estimate the Training Error and Test Error

```
## Training Error: 0.8785431
```

```
## Test Error:  0.9354477
```

b) Which variables contribute significantly?

The following variables have a p-value less than the significance level of 0.05, hence they can be considered as statistically significant:

```
p_df <- data.frame(p_val = summary(mod)$coefficients[,4]) %>%
  filter(p_val < 0.05) %>%
  arrange(p_val)
print(p_df)
```

```
##                      p_val
## DFA           6.391767e-43
## PPE           6.704282e-12
## HNR           6.410881e-11
## Shimmer.APQ11 6.342651e-07
## Jitter.Abs.   3.308307e-05
## NHR           4.837795e-05
## Shimmer.APQ5  6.679006e-04
## Shimmer       4.049628e-03
```

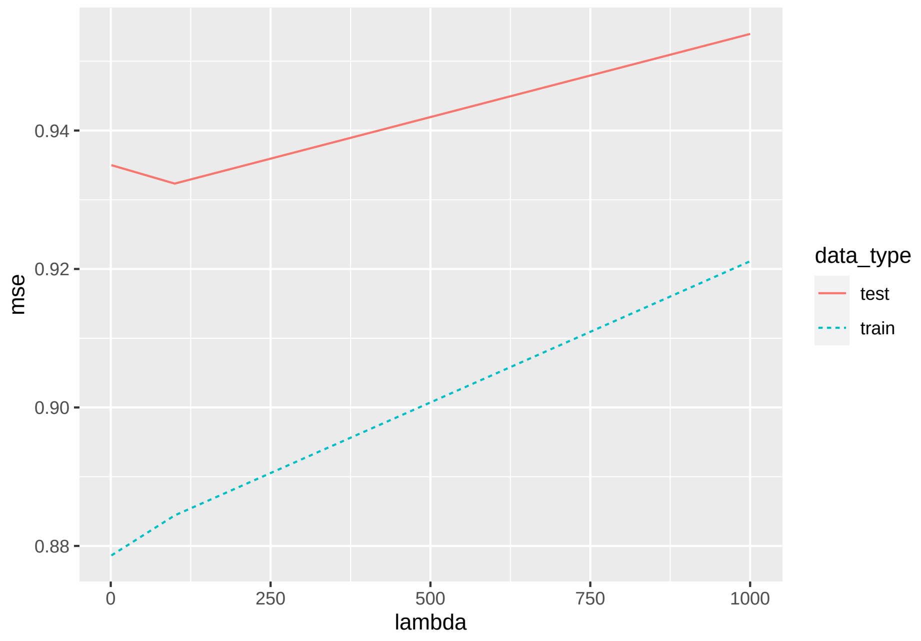
Question 4

a) Use the estimated parameters to predict the motor_UPDRS values for training and test data and report the training and test MSE values.

```
## MSE for Training and Testing data using ridge reg:
##   lambda data_type       mse       rsqr
## 1    100      test 0.9323316 0.08263808
## 2      1      test 0.9349969 0.08455688
## 3   1000      test 0.9539482 0.06524384
## 4      1     train 0.8786271 0.12112518
## 5    100     train 0.8844104 0.11713352
## 6   1000     train 0.9211216 0.09519874
```

b) Which penalty parameter is most appropriate among the selected ones

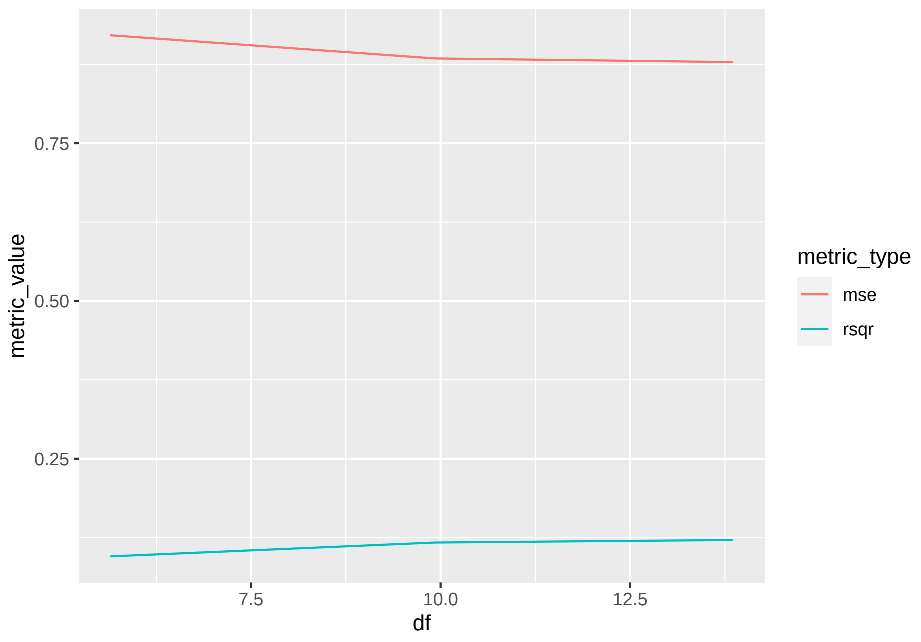
```
ggplot(final_list$mse_df, aes(x=lambda, y = mse)) +
  geom_line(aes(color = data_type, linetype = data_type))
```



As seen above, lambda = 100 gives the least test error and hence is relatively the best.

c) Compute and compare the degrees of freedom of these models and make appropriate conclusions

```
ggplot(data = df_final, aes(x = df, y = metric_value)) +
  geom_line(aes(color = metric_type))
```



```r
print(df_final)
```

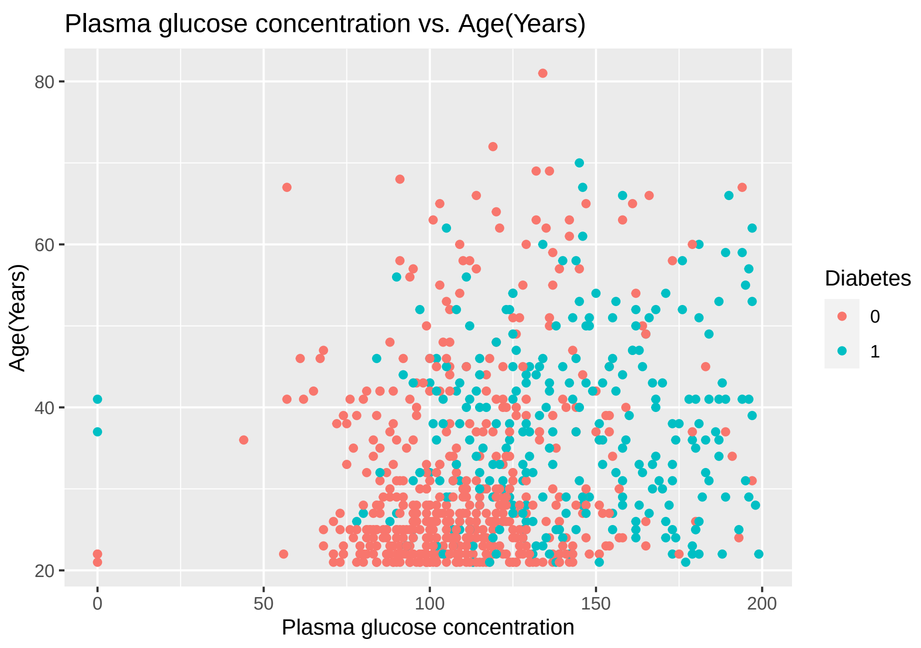
```
##   lambda        df data_type metric_type metric_value
## 1   1000  5.643925     train         mse   0.92112157
## 2    100  9.924887     train         mse   0.88441041
## 3      1 13.860736     train         mse   0.87862709
## 4   1000  5.643925     train        rsqr   0.09519874
## 5    100  9.924887     train        rsqr   0.11713352
## 6      1 13.860736     train        rsqr   0.12112518
```

As seen as degrees of freedom increases, the MSE decreases, because there are more features to explain. Also, the R^2 also increases. But at 9DF, we see the graph stabilizing without much decrease/increase. Hence, this could be a good point to select lambda, which in this case is 100.

Assignment 3: Logistic Regression

Question 1

The scatterplot is shown below for Plasma Glucose Concentration on Age where the observations are colored by Diabetes levels. Diabetes can be classified by a standard logistic regression model since the output of the classification is in a binary form, that is, 0 corresponding to a person doesn't have diabetes and 1 being, the person has diabetes and given that the two selected independent variables are discrete in nature.



Question 2

After training the model with response variable as Diabetes, the features as Plasma Glucose Concentration and Age and setting the classification threshold at 0.5, the following is the confusion matrix:

```
##    diab_pred
##       0   1
##   0 436  64
##   1 138 130
```

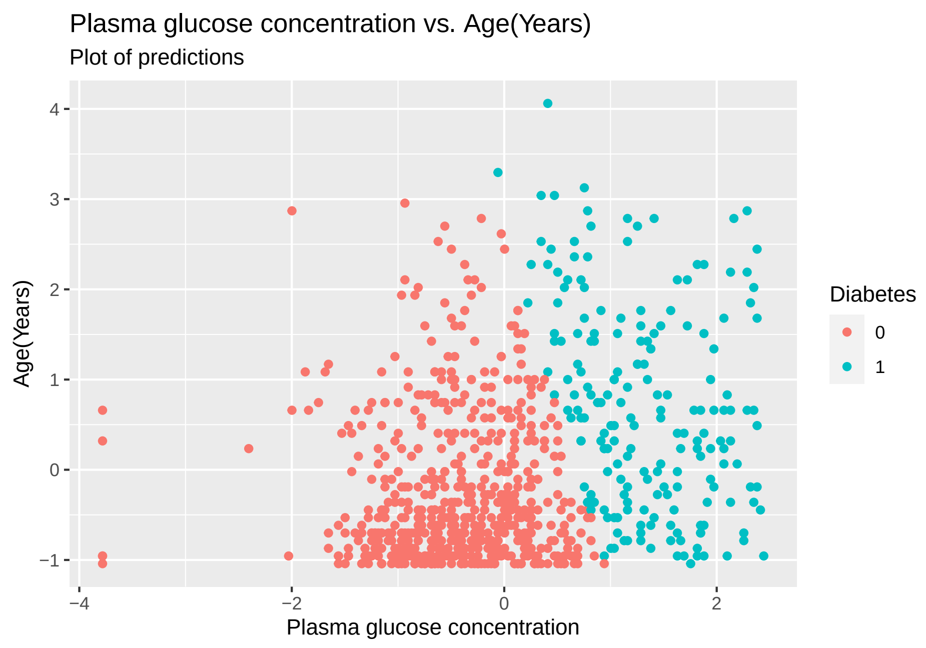
The probabilistic model is reported as:

$$P(Diabetes = 0) = \frac{1}{1 + \exp\left(-0.77962 + (1.13963x_1) + (0.29140x_2)\right)}$$

The training misclassification error is computed to be:

```
## [1] 0.2630208
```

The scatterplot below is showing the predicted values of Diabetes after logistic regression.



As it is visible from the scatterplot above, the logistic regression model is able to classify Diabetes with a clear demarcation, but however, the rate of missclassification error is slightly high at 26.3%

Question 3

3 a). The equation of the decision boundary between the two classes is reported as follows:

In logistic regression, we define the mapping of independent variable, x and parameter θ as z, which is called a logit, i.e.,

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Since this is a binary classification, with classification threshold, r = 0.5, the above equation is true when:

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0$$

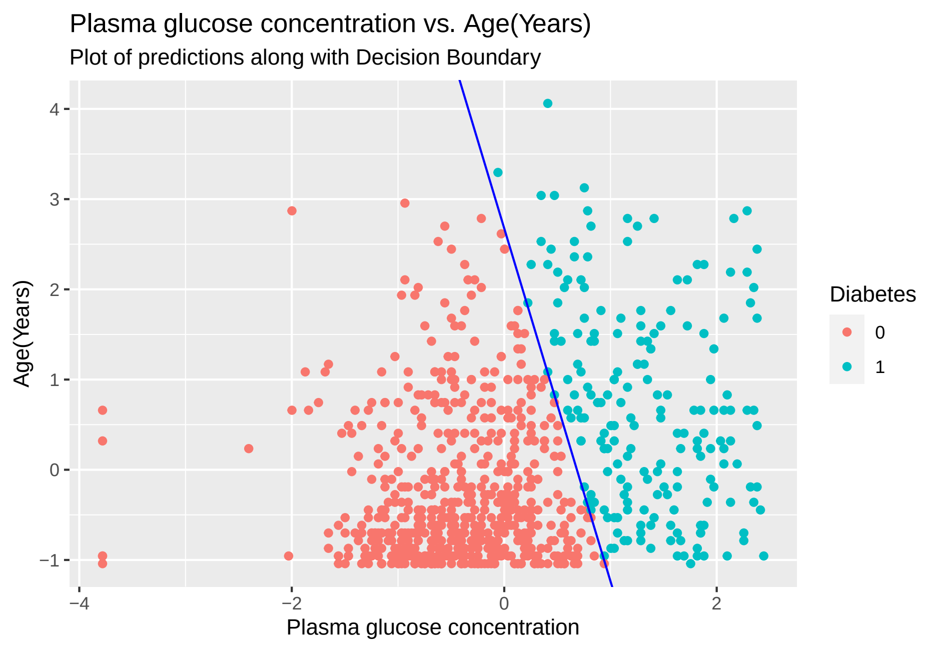
This is the equation for decision boundary and can be rearranged as:

$$x_2 \geq \frac{-\theta_0}{\theta_2} + \frac{-\theta_1}{\theta_2} x_1$$

This is in the form of y = mx+b, and for the decision boundary the slope 'm' and intercept 'b' is calculated in R by doing the following:

```
# Computing slope and intercept of the decision boundary.

slope <- coef(log_model)[2]/(-coef(log_model)[3])
intercept <- coef(log_model)[1]/(-coef(log_model)[3])
```

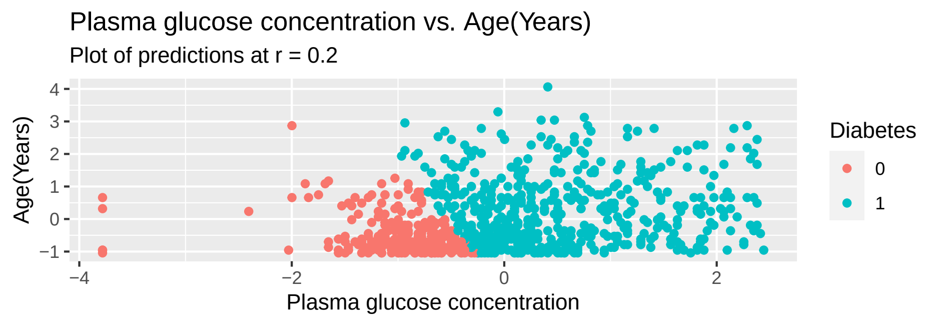
3 b). Now that we have computed the slope and intercept of the decision boundary of the classes, the scatterplot below contains the decision boundary.

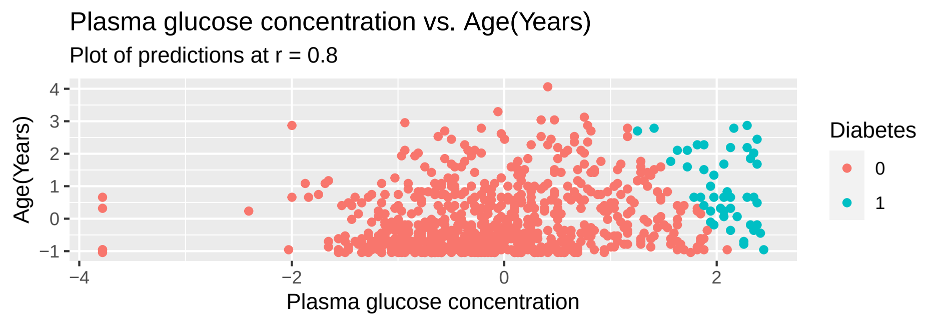


As it is visible from the plot above, the decision boundary seems to catch the distribution of the predicted values quite well.

Question 4

For classification threshold, r = 0.2 and r = 0.8, the plots are as follows:





As it is evident from the plots above, when the classification threshold set to 0.2 a lot of values are misclassified as diabetic and at 0.8, the converse happens that is a lot of values are misclassified as non diabetic.

Misclassification Rate at r = 0.2 :

```
## [1] 0.3723958
```

Misclassification Rate at r = 0.8:

```
## [1] 0.3151042
```

While the misclassification rate at r = 0.5 is 0.263 as already shown above in the report.

Question 5

After performing the basis function expansion trick by computing new features z_1, z_2, z_3, z_4, z_5, adding them to dataset and computing the logistic regression model below is the result:

The confusion matrix :

```
table(dataS_2[,3], diab_pred_4)
```

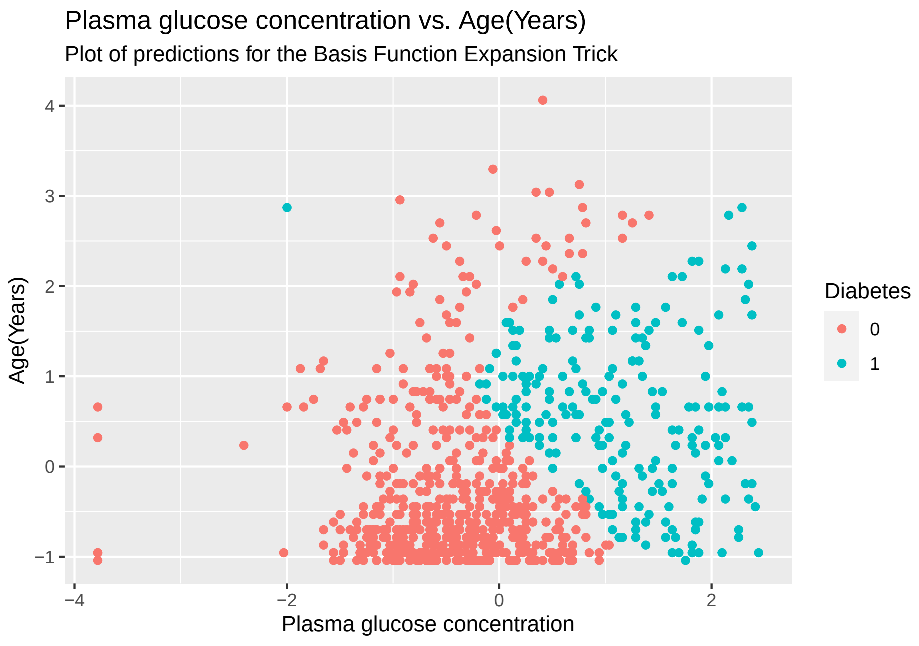
```
##    diab_pred_4
##       0   1
##   0 433  67
##   1 121 147
```

The misclassification error:

```
# Misclassification rate of Basis function expansion trick

missclass4=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

missclass4(dataS_2[,3], diab_pred_4)
```

```
## [1] 0.2447917
```

The scatterplot:



The misclassification error has gone down from 26.3% to 24.4%, so we can say that the basis function expansion trick has improved the classification prediction and that the model quality has increased.

Based on the reduction of the misclassification error we can come to the conclusion that the model accuracy has gone up albeit only by a slight margin.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
library(kknn)
library(dplyr)
library(ggplot2)
library(tidyr)
library(caret)
library(gridExtra)

##############Code for KNN##############
#loading data
opt<-read.csv('optdigits.csv',header=FALSE)

#splitting test, train and validation datasets
n=dim(opt)[1]
set.seed(12345)
id<- sample(1:n,floor(n*0.5))
train<-opt[id,]

id1<-setdiff(1:n,id)
set.seed(12345)
id2<-sample(id1,floor(n*0.25))
valid<-opt[id2,]

id3<-setdiff(id1,id2)
test<-opt[id3,]
m_test<-kknn::kknn(as.factor(V65)~., train,test, k=30,kernel="rectangular")
pred_test<-m_test$fitted.values

m_train<-kknn(as.factor(V65)~., train,train, k=30,kernel="rectangular")
pred_train<-m_train$fitted.values

# confusion matrices for train and test data
table(test$V65,pred_test)
table(train$V65,pred_train)


#missclassification error for train and test data
missclass<-function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}
cat("Misclassification error on test data :", missclass(test$V65,pred_test),"\n")
cat("Misclassification error on train data :", missclass(train$V65,pred_train),"\n")
#getting probability of prediction of train data
prob<-m_train$prob
```

```
colnames(prob)<-c('level_0','level_1','level_2','level_3','level_4','level_5'
                  ,'level_6','level_7','level_8','level_9')

#copying train data set
get_prob<-train

#adding predicted column to the data set
get_prob$predicted<-pred_train

#adding digit 8 probability form prob matrix to data frame
get_prob$prob<-prob[,9]

#filtering and ordering based on correct class 8
fil<-dplyr::filter(get_prob,get_prob$V65==8)
fil<-fil[order(fil$prob),]

#easy case 1
new_df_ez_1 <- fil[nrow(fil)-2,]
as.list.data.frame(new_df_ez_1)
new_df_ez_long <- matrix(as.numeric(new_df_ez_1[,c(1:64)]), ncol = 8, byrow = TRUE)

#easy case 2
new_df_ez_2 <- fil[nrow(fil)-1,]
as.list.data.frame(new_df_ez_2)
new_df_ez_long_2 <- matrix(as.numeric(new_df_ez_2[,c(1:64)]), ncol = 8, byrow = TRUE)

#hard case 3
new_df_hard_1 <- fil[1,]
as.list.data.frame(new_df_hard_1)
new_df_hard_long_1 <- matrix(as.numeric(new_df_hard_1[,c(1:64)]), ncol = 8, byrow = TRUE)

#hard case 2
new_df_hard_2 <- fil[2,]
as.list.data.frame(new_df_hard_2)
new_df_hard_long_2 <- matrix(as.numeric(new_df_hard_2[,c(1:64)]), ncol = 8, byrow = TRUE)

#hard case 3
new_df_hard_3 <- fil[3,]
as.list.data.frame(new_df_hard_3)
new_df_hard_long_3 <- matrix(as.numeric(new_df_hard_3[,c(1:64)]), ncol = 8, byrow = TRUE)

#easy cases heat map
heatmap(x = new_df_ez_long, Rowv = NA, Colv = NA)
heatmap(x = new_df_ez_long_2, Rowv = NA, Colv = NA)

#hard cases heatmap
heatmap(x = new_df_hard_long_1, Rowv = NA, Colv = NA)
heatmap(x = new_df_hard_long_2, Rowv = NA, Colv = NA)
heatmap(x = new_df_hard_long_3, Rowv = NA, Colv = NA)

#calculating missclasification error for training and valid data and plotting
#in same plot to find optimal K
k<-1:30
```

```
misclasserror<-c()
misclasserror_train<-c()
  for(i in 1:length(k)){
    m_valid<-kknn(as.factor(V65)~., train,valid, k=i,kernel="rectangular")
    pred_valid<-m_valid$fitted.values
    m_train_1<-kknn(as.factor(V65)~., train,train, k=i,kernel="rectangular")
    pred_train<-m_train_1$fitted.values
    n1=length(valid$V65)
    v1<-(1-sum(diag(table(valid$V65,pred_valid)))/n1)
    misclasserror<-c(misclasserror,v1)
    n2=length(train$V65)
    v2<-(1-sum(diag(table(train$V65,pred_train)))/n2)
    misclasserror_train<-c(misclasserror_train,v2)
}
plot(k,misclasserror,type='l',ylim=c(0,0.06))
lines(k,misclasserror_train,type='l',col='blue')
legend(x = 20, y = 0.015,
       legend = c('Validation', 'Training'),
       col = c('black', 'blue'), lty = c(1,1))

#after getting optimal K predicting on test data
m_optim<-kknn(as.factor(V65)~., train,test, k=4,kernel="rectangular")
optim_pred<-m_optim$fitted.values
table(test$V65,optim_pred)
cat("Misclassification error on test data for optimal K :", missclass(test$V65,optim_pred),"\n")


#cross entropy error
#creating one hot encoding matrix using model.matrix
y_1 <- model.matrix(~0+as.factor(valid$V65), data = valid)
k<-1:30
entropy<-c()
for(i in 1:length(k)){
  m_e<-kknn(as.factor(train$V65)~., train,valid, k=i,kernel="rectangular")
  prob<-m_e$prob
  log_prb <- log(prob+10^-15)
  entropy <-c(entropy,-sum(y_1 * log_prb))
}
plot(k,entropy,type='l')
##############Code for Linear regression and ridge regression#############
set.seed(12345)
#Read csv
data <- read.csv(file = 'parkinsons.csv', header = T)
data <- data %>%
  dplyr::select(motor_UPDRS,starts_with("Jitter"),starts_with("Shimmer"),
                NHR, HNR, RPDE, DFA, PPE)
#Divide training and test data in 60-40 ratio
n <- nrow(data)
train_index <- sample(1:n, size=floor(n*0.6))
train <- data[train_index,]
test <- data[-train_index,]
#Scaling the data
params <- caret::preProcess(train)
```

```r
Stest <- predict(params, test)
Strain <- predict(params, train)

mod <- lm(formula = motor_UPDRS~-1+., data = Strain)#model without intercept
test_predict <- predict(mod,newdata = Stest)
train_predict <- predict(mod,newdata = Strain)
#MSE - train
mse_train <- mean((train_predict-Strain$motor_UPDRS)^2) #0.9016564
cat('Training Error:', mse_train)

#MSE - test
mse_test <- mean((test_predict-Stest$motor_UPDRS)^2) #0.8996531 - Without intercept
cat('Test Error: ',mse_test)
p_df <- data.frame(p_val = summary(mod)$coefficients[,4]) %>%
  filter(p_val < 0.05) %>%
  arrange(p_val)
print(p_df)
Loglikelihood <- function(theta, sigma){
  n <- nrow(Strain)
  x <- as.matrix(Strain[,-1])
  return(-n*log(sigma*sqrt(2*pi))-
    (1/(2*sigma^2))*sum((Strain$motor_UPDRS-(x%*%(theta)))^2))
}
Ridge <- function(x, lambda){
  theta <- x[1:16]
  sigma <- x[17]
  # lambda <- x[18]
  return(-Loglikelihood(theta,sigma) + (lambda * sum(theta * theta)))
  # return(-Loglikelihood(theta,sigma))
}

RidgeOpt <- function(lambdas){
  # parameter <- c(rep(1,17),lambda)
  parameter <- c(rep(0,16))
  parameter <- c(parameter,1)
  mse_df <- as.data.frame(matrix(nrow = 0, ncol = 4))
  colnames(mse_df) <- c('lambda', 'data_type', 'mse', 'rsqr')
  coef_df <- as.data.frame(matrix(nrow=0, ncol=3))
  colnames(coef_df) <- c('lambda','coef', 'val')
  for (i in lambdas) {
    res <- optim(parameter,
                 fn=Ridge,
                 lambda = i,
                 method = "BFGS") #Get the optimal theta & sigma
    theta <- res$par[1:16]
    #MSE for training data
    train_x <- as.matrix(Strain[,-1])
    train_y_hat <- train_x %*% theta
    train_mse = mean((Strain$motor_UPDRS-train_y_hat)^2)
    train_rsq <- cor(Strain$motor_UPDRS, train_y_hat)^2

    #MSE for training data
    test_x <- as.matrix(Stest[,-1])
    test_y_hat <- test_x %*% theta
    test_mse <- mean((Stest$motor_UPDRS-test_y_hat)^2)
    test_rsq <- cor(Stest$motor_UPDRS, test_y_hat)^2

    mse_df <- rbind(mse_df,
                    data.frame(lambda = i,
                               data_type = 'train',
                               mse = train_mse,
                               rsqr = train_rsq
                    )
    )
    mse_df <- rbind(mse_df,
                    data.frame(lambda = i,
                               data_type = 'test',
                               mse = test_mse,
                               rsqr = test_rsq
                    ))
    coef_df <- rbind(
      coef_df,
      data.frame(lambda = i, coef = 'DFA', val = theta[15]),
      data.frame(lambda = i, coef = 'PPE', val = theta[16]),
      data.frame(lambda = i, coef = 'HNR', val = theta[13]),
      data.frame(lambda = i, coef = 'NHR', val = theta[12]),
      data.frame(lambda = i, coef = 'Shimmer.APQ11', val = theta[10]),
      data.frame(lambda = i, coef = 'Jitter.Abs.', val = theta[2]),
      data.frame(lambda = i, coef = 'Shimmer.APQ5', val = theta[9]),
      data.frame(lambda = i, coef = 'Shimmer', val = theta[6])
      )
  }
  mse_df <- mse_df %>% arrange(data_type, mse)
  cat('MSE for Training and Testing data using ridge reg:\n')
  print(mse_df)
  return(list(mse_df = mse_df, coef_df = coef_df))
}
lambdas = c(1, 100, 1000)
final_list <- RidgeOpt(lambdas)
ggplot(final_list$mse_df, aes(x=lambda, y = mse)) +
  geom_line(aes(color = data_type, linetype = data_type))
DF <- function(lambdas){
  #Slide 22 - has the hat matrix for Ridge Regression
  #To compute the df based on training data
  x <- as.matrix(Strain[-1])
  dim(x)
  degfree_df <- data.frame(matrix(ncol=2, nrow = 0))
  colnames(degfree_df) <- c('lambda', 'df')
  for (i in lambdas) {
    hat_mat <- x %*% solve( t(x)%*%x + i*diag(ncol(x)) ) %*% t(x)
    degfree_df <- rbind(degfree_df,
                        data.frame(lambda = i, df = sum(diag(hat_mat)))
    )
  }
  return(degfree_df)
}
```

```r
df <- DF(lambdas)
df_final <- final_list$mse_df %>%
  filter(data_type == 'train') %>%
  inner_join(x=df, y=., by = 'lambda') %>%
  gather(.,key = "metric_type",
         value = "metric_value",
         -lambda, -df, -data_type) %>%
  arrange(metric_type, df)
ggplot(data = df_final, aes(x = df, y = metric_value)) +
  geom_line(aes(color = metric_type))
print(df_final)
##############Code for Logistic Regression#############
set.seed(12345)
# Question 1
# reading the data from csv file

diabetes_dataset <- read.csv('pima-indians-diabetes.csv', header = FALSE)
names(diabetes_dataset) <- c('Number of times pregnant',
                             'Plasma glucose concentration a 2 hours in an oral
                             glucose tolerance test',
                             'Diastolic blood pressure (mm Hg)',
                             'Triceps skinfold thickness (mm)',
                             '2-Hour serum insulin (mu U/ml)',
                             'Body mass index (weight in kg/(height in m)^2)',
                             'Diabetes pedigree function',
                             'Age (years)',
                             'Diabetes (0=no or 1=yes)')
# Scatterplot for Plasma Glucose Conc. vs. Age, observations as diabetes.
p_scatter <- ggplot(data = diabetes_dataset,
            aes(x = diabetes_dataset[,2],
                y = diabetes_dataset[,8]))+
  geom_point(aes(colour = as.factor(diabetes_dataset[,9])))+
  ggtitle(label = 'Plasma glucose concentration vs. Age(Years)')+
  xlab('Plasma glucose concentration')+
  ylab('Age(Years)')+
  guides(color = guide_legend(title = 'Diabetes'))
p_scatter
# Scaling the data, computing the Logistic Regression Model and the confusion matrix

n <- ncol(diabetes_dataset)
scale_data <- preProcess(diabetes_dataset[,1:(n-1)])
dataS <- predict(scale_data, diabetes_dataset)
log_model <- glm(formula = dataS[,9]~dataS[,2]+
                   dataS[,8],
                 family = binomial,data = dataS)
diab_prob <- predict(log_model, type = 'response')
r <- 0.5
diab_pred <- ifelse(diab_prob > r, 1, 0)

table(dataS[,9], diab_pred)

# Computing the misclassification error
```

```r
missclass=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

missclass(dataS[,9], diab_pred)

# Scatterplot for Plasma Glucose Conc. vs. Age, observations as predicted
# classification of diabetes.

upd_dataset <- cbind(dataS, diab_pred)
p2 <- ggplot(data = upd_dataset,
            aes(x = upd_dataset[,2],
                y = upd_dataset[,8]))+
  geom_point(aes(colour = as.factor(upd_dataset[,10])))+
  ggtitle(label = 'Plasma glucose concentration vs. Age(Years) ',
          subtitle = 'Plot of predictions')+
  xlab('Plasma glucose concentration')+
  ylab('Age(Years)')+
  guides(color = guide_legend(title = 'Diabetes'))
p2
# Computing slope and intercept of the decision boundary.

slope <- coef(log_model)[2]/(-coef(log_model)[3])
intercept <- coef(log_model)[1]/(-coef(log_model)[3])
# Scatterplot for Plasma Glucose Conc. vs. Age, observations as predicted
# classification of diabetes with the decision boundary.
p3 <- ggplot(data = upd_dataset,
             aes(x = upd_dataset[,2],
                 y = upd_dataset[,8]))+
  geom_point(aes(colour = as.factor(upd_dataset[,10])))+
  geom_abline(slope = slope, intercept = intercept,col = 'blue')+
  ggtitle(label = 'Plasma glucose concentration vs. Age(Years) ',
          subtitle = 'Plot of predictions along with Decision Boundary')+
  xlab('Plasma glucose concentration')+
  ylab('Age(Years)')+
  guides(color = guide_legend(title = 'Diabetes'))
p3
# Computing classification for thresholds r = 0.2  and r = 0.8
r <- 0.2
diab_pred_2 <- ifelse(diab_prob > r, 1, 0)
upd_dataset_2 <- cbind(dataS, diab_pred_2)

# Scatterplot of predicted classification at r = 0.2

p4 <- ggplot(data = upd_dataset_2,
             aes(x = upd_dataset_2[,2],
                 y = upd_dataset_2[,8]))+
  geom_point(aes(colour = as.factor(upd_dataset_2[,10])))+
  ggtitle(label = 'Plasma glucose concentration vs. Age(Years) ',
          subtitle = 'Plot of predictions at r = 0.2')+
  xlab('Plasma glucose concentration')+
  ylab('Age(Years)')+
  guides(color = guide_legend(title = 'Diabetes'))

r <- 0.8
diab_pred_3 <- ifelse(diab_prob > r, 1, 0)
upd_dataset_3 <- cbind(dataS, diab_pred_3)

# Scatterplot of predicted classification at r = 0.8
p5 <- ggplot(data = upd_dataset_3,
             aes(x = upd_dataset_3[,2],
                 y = upd_dataset_3[,8]))+
  geom_point(aes(colour = as.factor(upd_dataset_3[,10])))+
  ggtitle(label = 'Plasma glucose concentration vs. Age(Years) ',
          subtitle = 'Plot of predictions at r = 0.8')+
  xlab('Plasma glucose concentration')+
  ylab('Age(Years)')+
  guides(color = guide_legend(title = 'Diabetes'))


grid.arrange(p4,p5)
# Computing misclassification error at r = 0.2

missclass2=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}
missclass2(dataS[,9], diab_pred_2)
# Computing misclassification error at r = 0.8

missclass3=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}
missclass3(dataS[,9], diab_pred_3)
# Computing the new features for Basis function expansion.

z1 <- diabetes_dataset[,2]^4
z2 <- (diabetes_dataset[,2]^3)*(diabetes_dataset[,8])
z3 <- (diabetes_dataset[,2]^2)*(diabetes_dataset[,8]^2)
z4 <- (diabetes_dataset[,2])*(diabetes_dataset[,8]^3)
z5 <- (diabetes_dataset[,8])^4

new_df <- diabetes_dataset[,c(2,8,9)]
new_df$z1 <- z1
new_df$z2 <- z2
new_df$z3 <- z3
new_df$z4 <- z4
new_df$z5 <- z5
updated_diab_dataset <- cbind(diabetes_dataset, z1, z2, z3, z4, z5)

updated_scale <- preProcess(new_df[,-3])
dataS_2 <- predict(updated_scale, new_df)
colnames(dataS_2)[3] <- 'diabetes'
```

```r
# Computing the model by using transformed features.

log_model_2 <- glm(formula = as.factor(diabetes)~.,
                       data = dataS_2, family = binomial)
diab_prob_2 <- predict(log_model_2, type = 'response')
r_1 <- 0.5
diab_pred_4 <- ifelse(diab_prob_2 > r_1, 1, 0)

upd_dataset_4 <- cbind(dataS_2, diab_pred_4)

table(dataS_2[,3], diab_pred_4)
# Misclassification rate of Basis function expansion trick

missclass4=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

missclass4(dataS_2[,3], diab_pred_4)
#Scatterplot for Plasma Glucose Conc. vs. Age, observations as predicted classification
# of diabetes performed by Basis Function Expansion Trick
p6 <- ggplot(data = upd_dataset_4,
             aes(x = upd_dataset_4[,1],
                 y = upd_dataset_4[,2]))+
  geom_point(aes(colour = as.factor(upd_dataset_4[,9])))+
  ggtitle(label = 'Plasma glucose concentration vs. Age(Years) ',
          subtitle = 'Plot of predictions for the Basis Function Expansion Trick')+
  xlab('Plasma glucose concentration')+
  ylab('Age(Years)')+
  guides(color = guide_legend(title = 'Diabetes'))

p6
```