Computer Lab 1 Block 1 - Machine Learning | 732A99 | Group A20

Simon Alsén |Aderinto Adesijibomi| Mohammed Ali

1/10/2022

Assignment 1. Handwritten digit recognition with K-means

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

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. 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.
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.
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.
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?

```
##       predicted
## target   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
```

```
##       predicted
## target  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
```

```
## [1] 0.04238619
```

```
## [1] 0.04916318
```

Comment The overall missclassification rates are 4.5% for training data and 4.9% for test data respectively. Digits that seemed hard to classify is 8 which is being missclassified as 1 and digits 5 which is being classified as 9. Model prediction quality is good as the model seems to classify with ~95% accuracy.

```
##              8 actual
## 1624 0.1000000      8
## 1663 0.1666667      8
## 229  0.2333333      8
```

```
##      8 actual
## 1864 1      8
## 1811 1      8
```

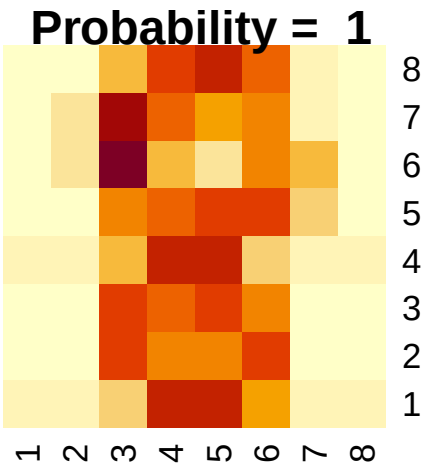


fig 1

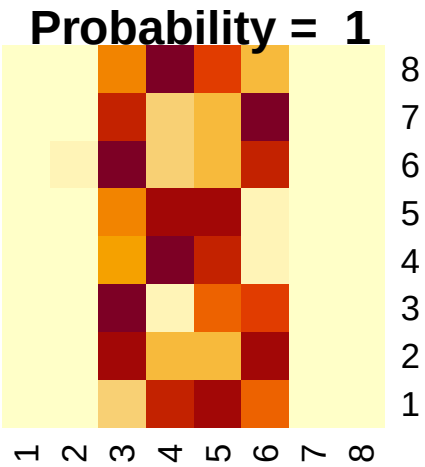


fig 2

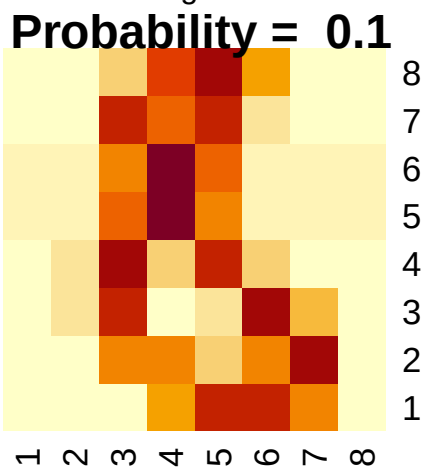


fig 3

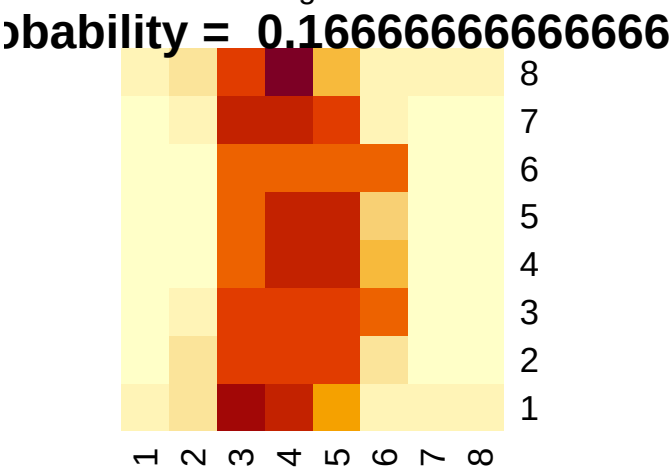


fig 4

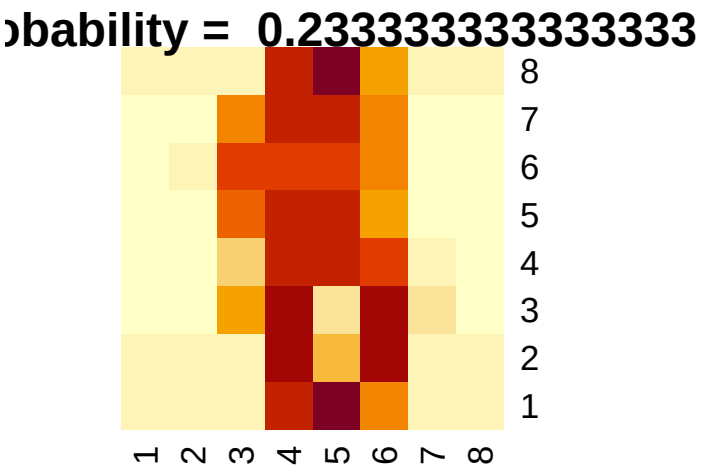
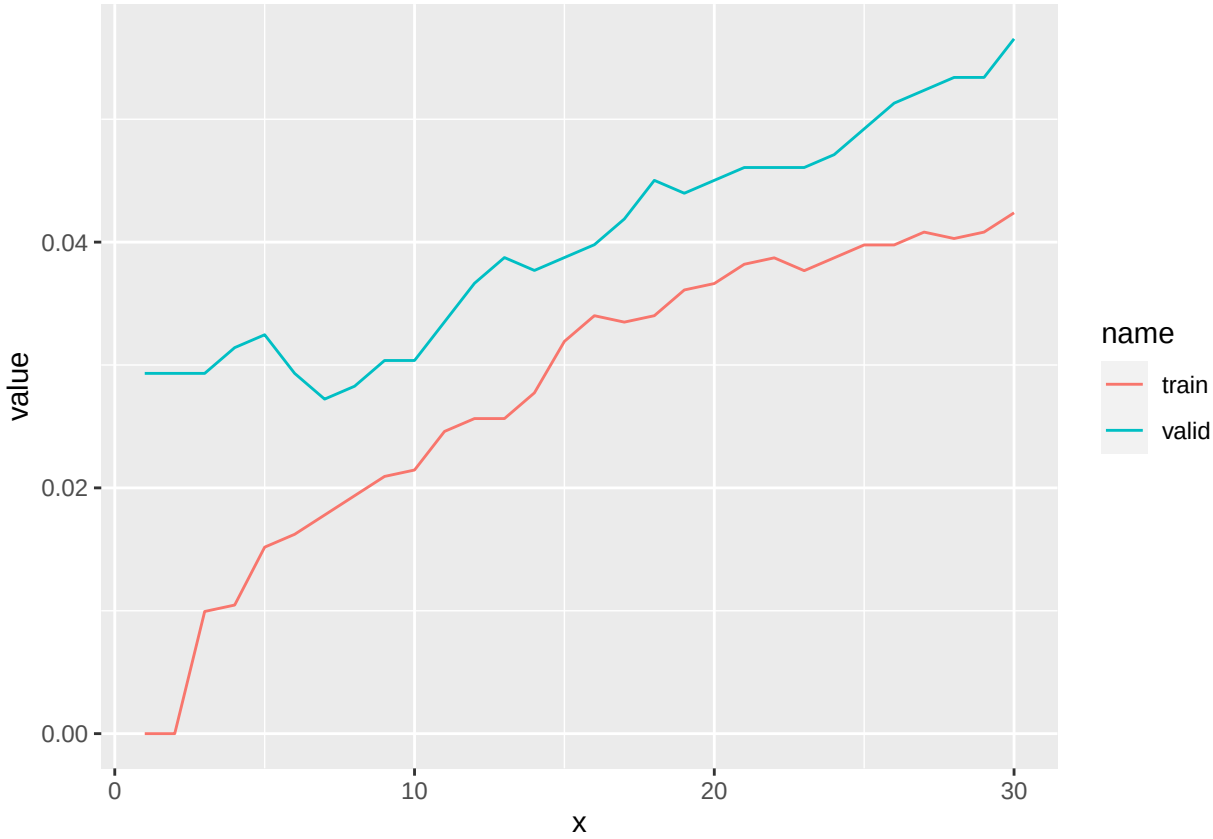


fig 5

Comment Clearly from the first two images with high probabilities are ones which is easy to classify, while the the last three images are difficult to classify as digit 8

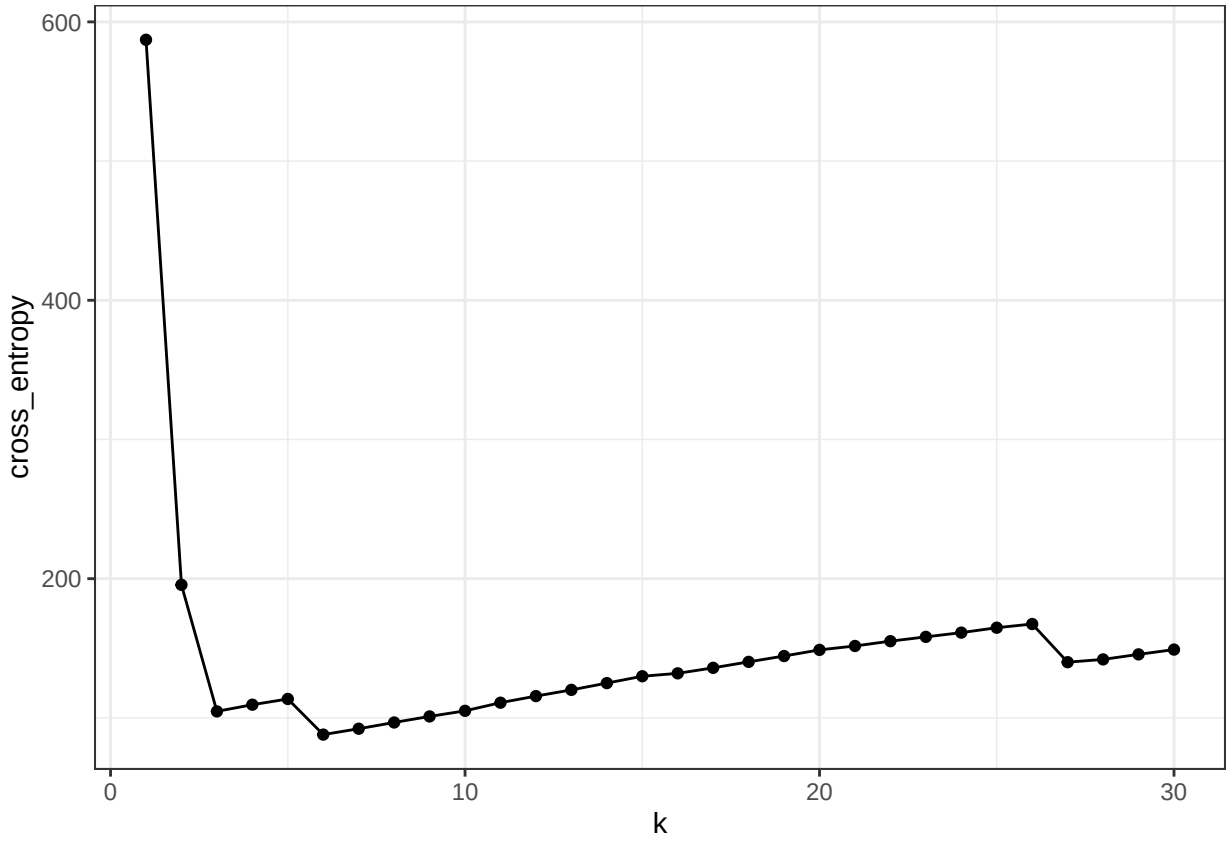



```
## [1] 0.03870293
```

Comment The misclassification rate is minimized when k=7 for the validation dataset. Hence k=7 is the optimal value of k. Generally the misclassifcation rate tend to increase as k increases in both cases. The test

error for the model with k=7 is 0.03870293 which indicates that the model quality is high. The test error is slightly higher than the training and validation errors.

```
## [1] 6
```



Comment The optimal value of K in this case can be determined to be K = 6, as this is the point where the cross-entropy error is at its lowest.

Assignment 2. Linear regression and ridge regression

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

1. Scale the data and divide it into training and test data (60/40). 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.

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.

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

a. likelihood function that for a given parameter vector θ and dispersion σ computes the log-likelihood function $\log P(\mathcal{T}\sigma)$ for the stated model and the training data
b. R function that for given vector θ, scalar σ and scalar λ uses function from 3a and adds up a Ridge penalty $|\lambda \mid \theta^2|$ to the minus log-likelihood
c. R 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.
d. D function that for a given scalar λ computes the degrees of freedom of the Ridge model based on the training data.

4. By using function RidgeOpt, compute optimal θ parameters for λ=1 λ=100 and λ=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.

```
##
## Call:
## lm(formula = motor_UPDRS ~ . - 1, data = train_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0119 -0.7270 -0.1018  0.7384  2.1959
##
## 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
##
## Residual standard error: 0.9366 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
```

```
## [1] 0.9294911
```

```
## [1] 0.8731931
```

lambda_levels	MSE_Train	DF_Test	MSE_Test
1	0.873277	13.862811	0.9290357
100	0.873277	9.939085	0.9290357
1000	0.873277	5.643351	0.9290357

Comment The above result from lm model shows the p-value significant level for model coefficients, we can see that:- Out of all model coefficients we have Jitter.Abs., Shimmer,Shimmer.APQ5,Shimmer.APQ11,NHR,HNR,DFA and PPE with a significant P-value. Which mean that those parameters have the most contribution in the model result. The test MSE is obtained by using the predict method of the fitted model and using the test data as input,MSE for test is 92% and MSE for train is 87%

Comment Among the selected ones, a penalty parameter of 100 should be the most appropriate as it decreases the test MSE the most of the three, but all are close to each other. Also as λ increases the degrees of freedom decreases because the number of coefficients get smaller.

Assignment 3. Logistic regression and basis function expansion

The data file pima-indians-diabetes.csv contains information about the onset of diabetes within 5 years in Pima Indians given medical details. The variables are (in the same order as in the dataset): 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).

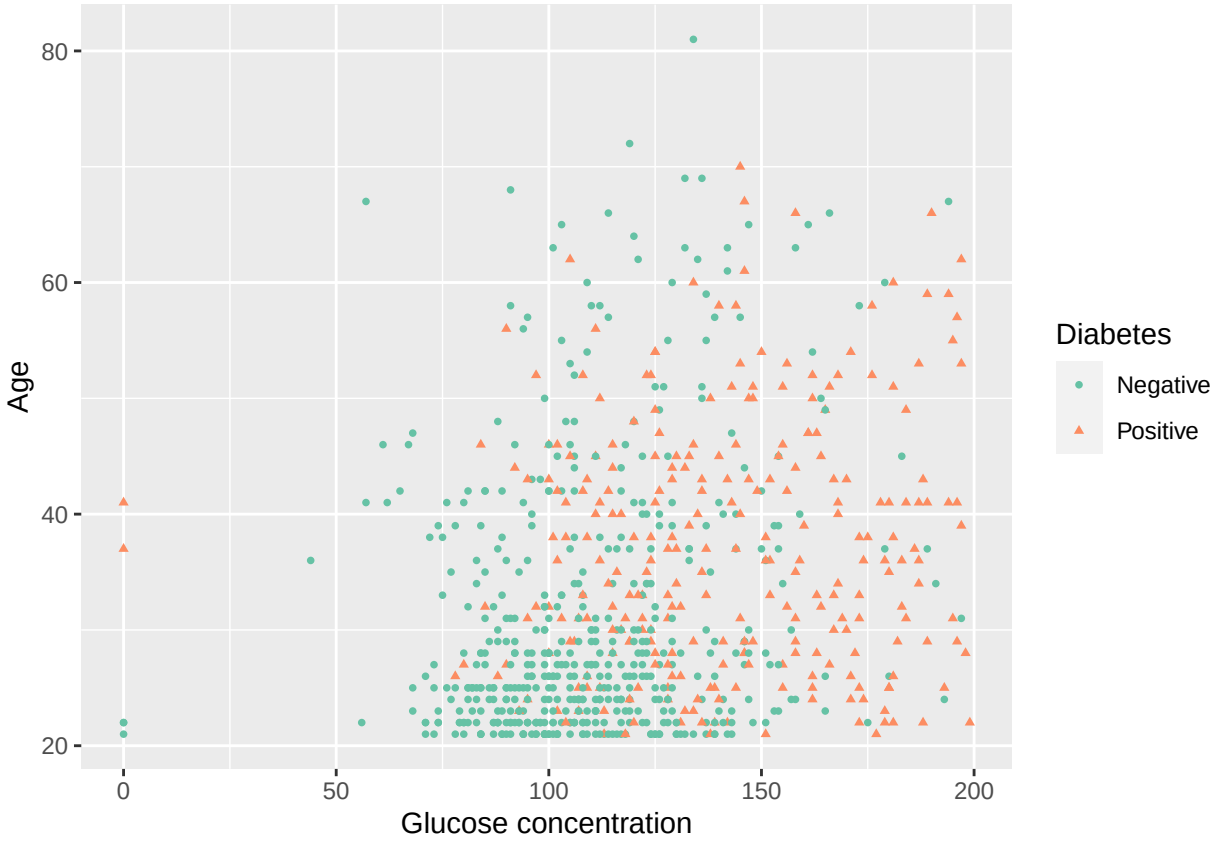
1. Make a scatterplot showing a Plasma glucose concentration on Age where observations are colored by Diabetes levels. Do you think that Diabetes is easy to classify by a standard logistic regression model that uses these two variables as features? Motivate your answer.

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.

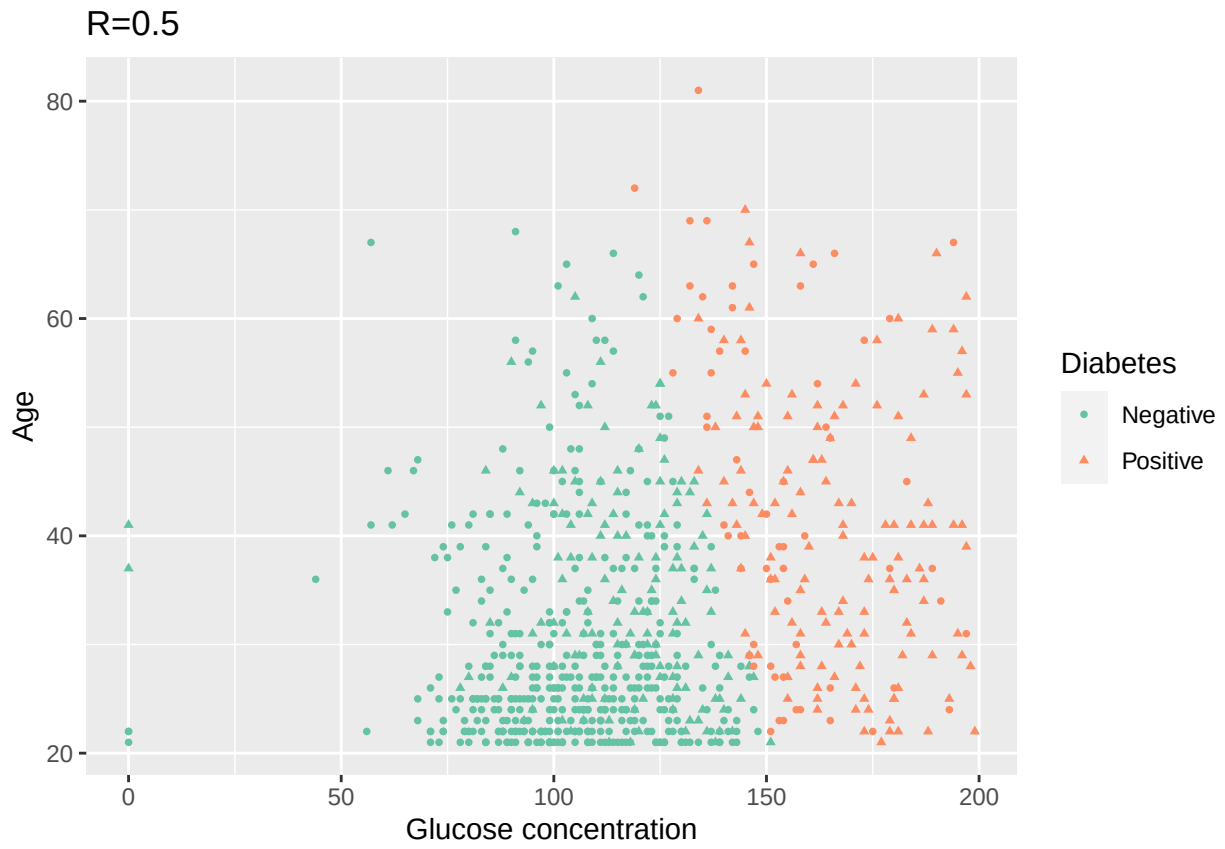
3. Use the model estimated in step 2 to a) report the equation of the decision boundary between the two classes b) add a curve showing this boundary to the scatter plot in step 2. Comment whether the decision boundary seems to catch the data distribution well.

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 r value changes.

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 x1,x2,x3,x4,x5 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?



Comment Observations who do not have diabetes generally appear to be younger and have lower plasma glucose concentration than those who have diabetes. However, many observations seem to deviate from this pattern,and as a consequence a standard logistic regression model that uses these two variables as features will probably not be able to classify all the observations correctly. In order for standard logistic regression to perform well the two classes should be more easy to separate from one another.




```
## [1] "Missclassification error on train is "
```

The probabilistic equation of the estimated model is:

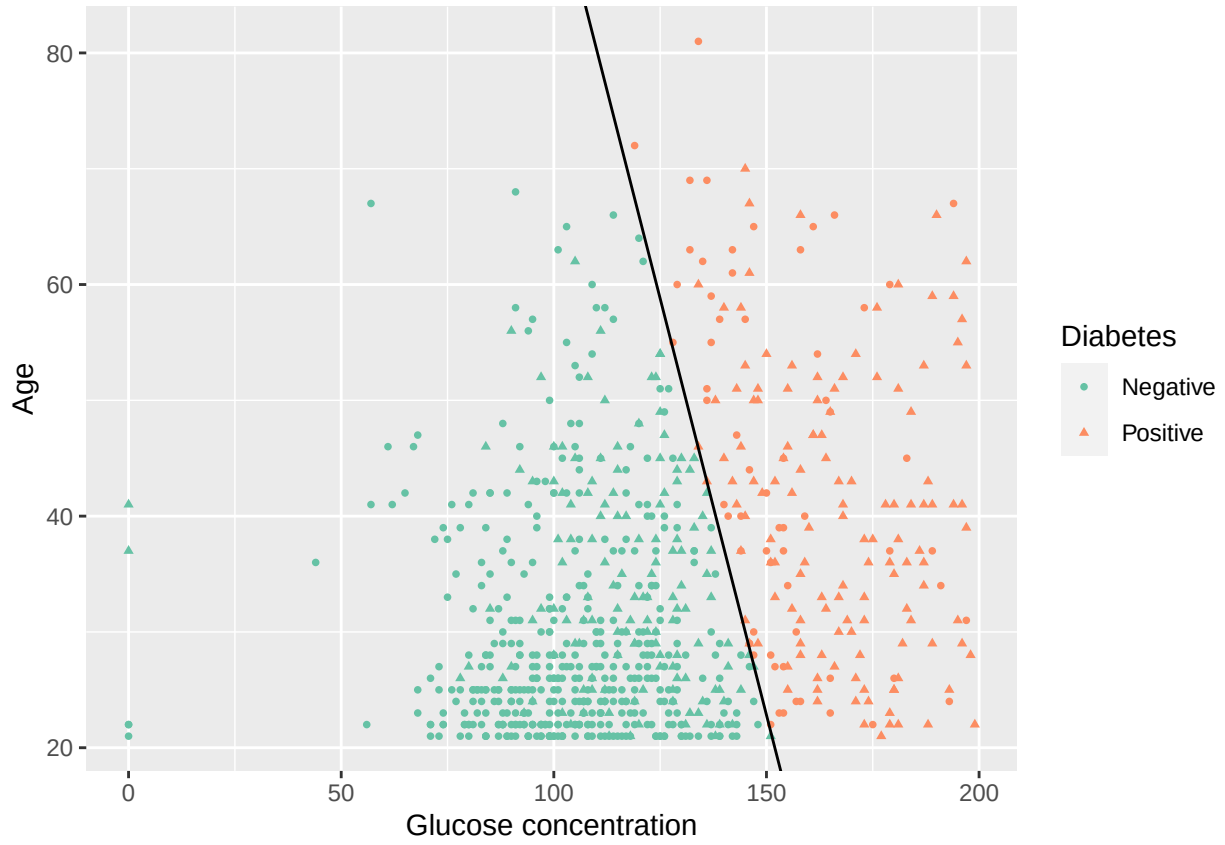
$$E(Y_i) = \frac{\exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2})}{1 + \exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2})}$$

$$= \frac{\exp(-5.91245 + 0.03564(\text{age}) + 0.02478(\text{plasma glucose concentration})))}{1 + \exp(-5.91245 + 0.03564(\text{age}) + 0.02478(\text{plasma glucose concentration})))}$$

Comment For every unit increase in age (a year) the log odds of a person having diabetes increases by 0.03564, and for every unit increase in plasma glucose concentration the log odds of a person having diabetes increases by 0.02478.

The training misclassification error is 0.2630, which means that the model manage to classify about 73.7% of the observations correctly.

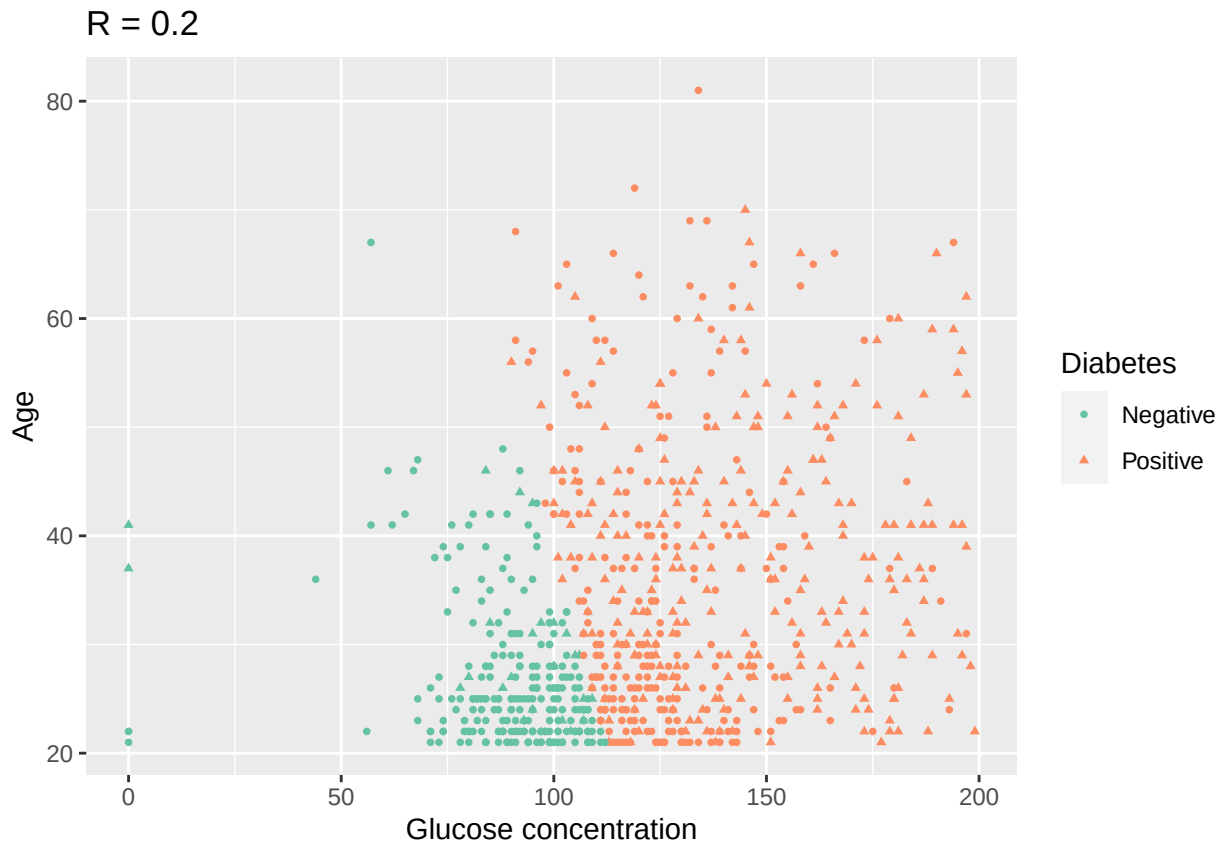
For observations who actually have diabetes the misclassification rate is 0.5149, while the misclassfication rate for observation who do not have diabetes is only 0.128.



The equation of the decision boundary between the two classes are:

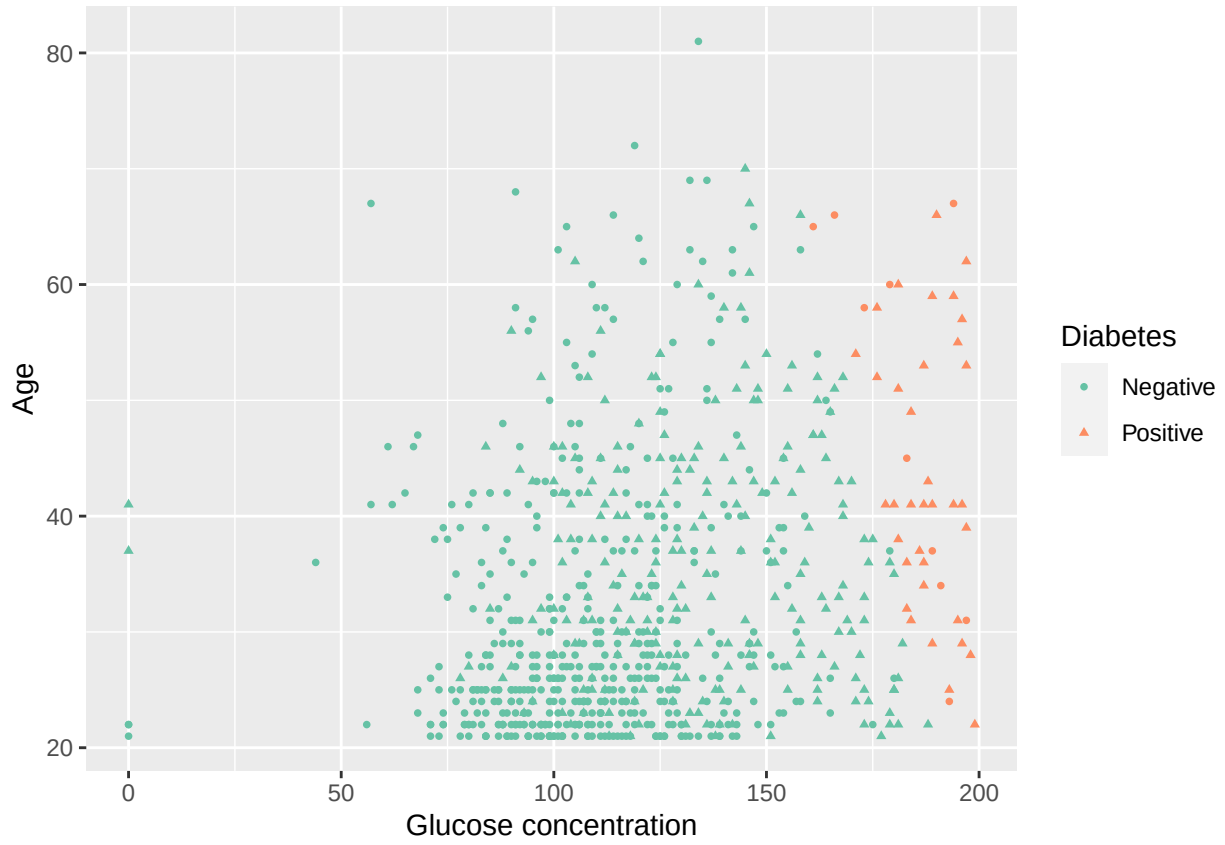
$$d = \frac{\beta_0}{|\beta_2|} + \frac{\beta_1}{|\beta_2|}x_1 + \frac{\beta_2}{|\beta_2|}x_2$$

The decision boundary appear to catch most of the observations without diabetes, but it seem to perform worse in separating the observations with diabetes.




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

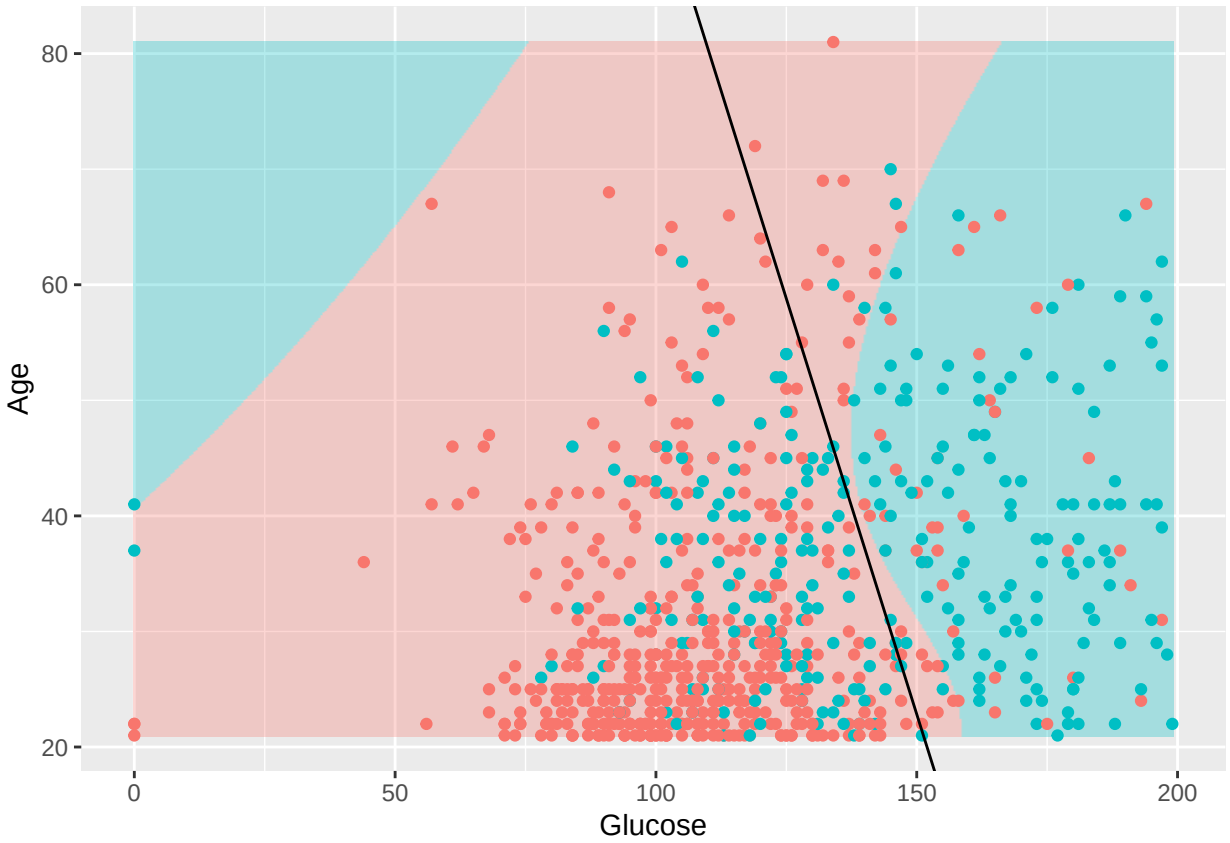
```
## [1] 0.3151042
```




```
## $title
## [1] "R=0.8"
##
## attr(,"class")
## [1] "labels"
```

The misclassifcation rate for $r = 0.2$ is 0.3723958. The misclassifcation rate for $r = 0.8$ is 0.3151042. When $r = 0.2$ more observations are classified as having diabetes, while fewer observations are classified as having diabetes when $r = 0.8$.

```
## [1] 0.2408854
```



Comment The blue and red background shows the new boundaries, the color of the points shows the actual classes, and the black line shows the original boundary. The basis function expansion trick makes the boundaries more flexible, which improves the prediction accuracy.

Statement of Contribution:

Simon, Mohamed and Adesijibomi devised the whole assignment together, the main conceptual ideas and codes outline. Adesijibomi worked out Assignment 1 (Handwritten digit recognition with K-means), Mohammed worked out Assignment 2 (Linear regression and ridge regression), Simon worked out Assignment 3 (Logistic regression and basis function expansion) and the report creation using r markdown.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
knitr::opts_chunk$set(echo = FALSE)

library(kknn)
library(ggplot2)
library(dplyr)
library(graphics)
library(tidyr)
data <- read.csv("optdigits.csv")

n <- dim(data)[1]
```

```
set.seed(12345)
id <- sample(1:n,floor(n*0.5))
train <- data[id,]
id1 <- setdiff(1:n,id)
set.seed(12345)
id2 <- sample(id1,floor(n*0.25))
valid <- data[id2,]
id3 <- setdiff(id1,id2)
test <- data[id3,]

knitr::opts_chunk$set(echo = FALSE)

names(train)[65] <- "target"
names(test)[65]<- "target"
names(valid)[65]<- "target"

train$target <- as.factor(train$target)
test$target <- as.factor(test$target)
valid$target <- as.factor(valid$target)

data_kknn_test <- kknn(formula=target~.,train = train,test = test,
                       kernel = "rectangular",k = 30)

data_kknn_train <- kknn(formula = target~.,train = train, test = train,
                        kernel = "rectangular", k=30)

CM_train <- table(train$target,data_kknn_train$fitted.values, dnn = c("target","predicted"))
CM_test <- table(test$target, data_kknn_test$fitted.values, dnn = c("target","predicted"))

CM_train
CM_test

MCE_train <- 1-sum(diag(CM_train))/sum(CM_train)
MCE_test <- 1-sum(diag(CM_test))/sum(CM_test)
MCE_train
MCE_test
knitr::opts_chunk$set(echo = FALSE)

result <- as.data.frame(predict(data_kknn_train,type = "prob",train))
result$actual <- train$target
result_8_digits <- as.data.frame(result[which(result$actual==8),c(9,11)])
result_8_digits_hard <- result_8_digits[order(result_8_digits$`8`),]%>% head(3)
result_8_digits_hard

result_8_digits_easy <- result_8_digits[rev(order(result_8_digits$`8`)),]%>% head(2)
result_8_digits_easy

temp1 <- matrix(as.numeric(train[as.numeric(rownames(result_8_digits_easy)[1])
                                 ,-ncol(train)]),nrow=8, ncol= 8, byrow =T)
temp1 <- temp1[nrow(temp1):1,]
heatmap(temp1, Colv = NA, Rowv = NA,xlab = "fig 1", main =
          paste (" Probability = ",result_8_digits_easy$`8`[1]))
```

```
temp2 <- matrix(as.numeric(train[as.numeric(rownames(result_8_digits_easy)[2])
                                 ,-ncol(train)]),nrow=8, ncol= 8, byrow =T)
temp2 <- temp2[nrow(temp2):1,]
heatmap(temp2, Colv = NA, Rowv = NA,xlab = "fig 2", main  =
          paste (" Probability = ", result_8_digits_easy$`8`[2]))

temp3 <- matrix(as.numeric(train[as.numeric(rownames(result_8_digits_hard)[1])
                                 ,-ncol(train)]),nrow=8, ncol= 8, byrow =T)
temp3 <- temp3[nrow(temp3):1,]
heatmap(temp3, Colv = NA, Rowv = NA, xlab = "fig 3",main =
          paste (" Probability = ", result_8_digits_hard$`8`[1]))

temp4 <- matrix(as.numeric(train[as.numeric(rownames(result_8_digits_hard)[2])
                                 ,-ncol(train)]),nrow=8, ncol= 8, byrow =T)
temp4 <- temp4[nrow(temp4):1,]
heatmap(temp4, Colv = NA, Rowv = NA,xlab ="fig 4",main =
          paste (" Probability = ", result_8_digits_hard$`8`[2]))

temp5 <- matrix(as.numeric(train[as.numeric(rownames(result_8_digits_hard)[3])
                                 ,-ncol(train)]),nrow=8, ncol= 8, byrow =T)
temp5 <- temp5[nrow(temp5):1,]
heatmap(temp5, Colv = NA, Rowv = NA, xlab = "fig 5" ,main =
          paste (" Probability = ", result_8_digits_hard$`8`[3]))

knitr::opts_chunk$set(echo = FALSE)

x <- 1:30
MSCE <- as.data.frame(matrix(NA, nrow=30, ncol=2))
colnames(MSCE) <- c("train","valid")

for (i in 1:length(x)){
  kknn_train_4 <- kknn(formula = target~., train = train,test = train,k = i,
                       kernel = "rectangular")
  CM_train_4 <- table(train$target,kknn_train_4$fitted.values)
  MSCE[i,1] <- 1-sum(diag(CM_train_4))/sum(CM_train_4)

  kknn_valid_4 <- kknn(formula = target~., train = train,test = valid, k = i,
                       kernel = "rectangular")
  CM_valid_4 <- table(valid$target, kknn_valid_4$fitted.values)
  MSCE[i,2] <- 1-sum(diag(CM_valid_4))/sum(CM_valid_4)
  }

plot_data <- data.frame(x,MSCE)
plot_data <- pivot_longer(plot_data, cols = c(2,3))
ggplot(plot_data, aes(x = x, y = value, col = name)) +
  geom_line()

fit_model <- kknn(target~., train = train, test = test, k = 7, kernel = "rectangular")
fit_model_MCE <- table(test$target, fit_model$fitted.values)
MSC_fit <- 1-sum(diag(fit_model_MCE))/sum(fit_model_MCE)
MSC_fit

knitr::opts_chunk$set(echo = FALSE)
```

```
cross_entropy_fcn <- function(model, valid ){
  log_prob <- 0
  for (i in 1:dim(valid)[1]){
    prob <- model$prob[i,valid[i,65]]
    log_prob <- log_prob + log(prob + 1e-15)
  }
  return(-log_prob)
}
log_prob <- 0
set.seed(12345)
for (i in 1:30){
  log_prob[i]<- cross_entropy_fcn(kknn(train[,65]~.,train = train,test = valid,k=i,
                                       kernel = "rectangular"),valid)
}

CE_model <- data.frame(log_prob,k=c(1:30))
which.min(CE_model$log_prob)

ggplot(CE_model,aes(x=k, y=log_prob))+
  geom_point()+
  geom_line()+
  labs(x="k", y="cross_entropy")+
  theme_bw()
library(kableExtra)
data_1 <- read.csv("parkinsons.csv")

data_1 <- subset(data_1, select = -c(subject.,age,sex,test_time,total_UPDRS))

scaled_data <- as.data.frame(scale(data_1))

n_1 <- dim(scaled_data)[1]
set.seed(12345)
id_1 <- sample(1:n_1,floor(n_1*0.6))
train_1 <- scaled_data[id_1,]
test_1 <-scaled_data[-id_1,]

knitr::opts_chunk$set(echo = FALSE)

fit_model_2 <- lm(motor_UPDRS~.-1, data = train_1)###########3 to remove intercepts
summary(fit_model_2)

MSE_1_test <- mean((test_1$motor_UPDRS - predict.lm(fit_model_2,test_1))^2)
MSE_1_test
MSE_1_train <- mean(fit_model_2$residuals^2)
MSE_1_train

llk <- function(theta,sigmA){
  x <- as.matrix(train_1[,-1])
  y <- as.matrix(train_1[,1])
  n <- nrow(y)
    log_llk <- -0.5*n*log(2*pi*sigmA^2)-(1/(2*sigmA^2))*sum((t(theta)%*%t(x)-t(y))^2)
  return(log_llk)
}
```

```r
ridge_fn <- function(thetheta, lambda){
  theta <- as.matrix(thetheta[1:16])
  sigmA <- thetheta[17]
  log_llk <- llk(theta,sigmA)
    ridgelogp <- -log_llk + (lambda*norm(theta, "F")^2)
  return(ridgelogp)
}

ridgeOpt <- function(lambda){
  r_opt <- optim(par=rep(0.1,17),fn = ridge_fn,method = "BFGS",lambda =lambda)
  return(r_opt)
}

Df <- function(lambda){
  x <- as.matrix(train_1[,-1])
  I <- dim(x)[2]
  df <- sum(diag(x %*%solve(t(x)%*%x+lambda*diag(dim(train_1[,-1])[2]))%*%t(x)))
}

theta_1 <- ridgeOpt(1)$par[1:16]
Y_training_1 <- t(as.matrix(theta_1))%*% t(as.matrix(train_1[,-1]))
Y_test_1 <- t(as.matrix(theta_1))%*% t(as.matrix(test_1[,-1]))
Mse_train_1 <- mean((train_1$motor_UPDRS-t(Y_training_1))^2)
Mse_test_1 <- mean((test_1$motor_UPDRS -t(Y_test_1))^2)
DF1 <- Df(1)


theta_100 <- ridgeOpt(100)$par[1:16]
Y_training_100 <- t(as.matrix(theta_1))%*% t(as.matrix(train_1[,-1]))
Y_test_100 <- t(as.matrix(theta_1))%*% t(as.matrix(test_1[,-1]))
Mse_train_100 <- mean((train_1$motor_UPDRS-t(Y_training_100))^2)
Mse_test_100 <- mean((test_1$motor_UPDRS -t(Y_test_100))^2)
DF100 <- Df(100)


theta_1000 <- ridgeOpt(1000)$par[1:16]
Y_training_1000 <- t(as.matrix(theta_1))%*% t(as.matrix(train_1[,-1]))
Y_test_1000 <- t(as.matrix(theta_1))%*% t(as.matrix(test_1[,-1]))
Mse_train_1000 <- mean((train_1$motor_UPDRS-t(Y_training_1000))^2)
Mse_test_1000 <- mean((test_1$motor_UPDRS -t(Y_test_1000))^2)
DF1000 <- Df(1000)


res<- data.frame(lambda_levels=c(1,100,1000),
                 MSE_Train = c(Mse_train_1, Mse_train_100,Mse_train_1000),
                 DF_Test =c(DF1,DF100,DF1000),
                 MSE_Test = c(Mse_test_1, Mse_test_100, Mse_test_1000))

res %>%
  kbl()%>%
  kable_styling()

data_2 <- read.csv("pima-indians-diabetes.csv", header = F)
```

```r
colnames(data_2)<- c("pregnant","Glucose","blood","skin","insulin","bmi","dpf",
                     "Age","Diabetes")

data_2 %>%
  ggplot(aes(x=Glucose,y=Age,color=as.factor(Diabetes)))+
  geom_point(size=1,aes(shape=as.factor(Diabetes)))+
  scale_color_brewer(palette = "Set2",name="Diabetes",labels=c("Negative","Positive"))+
  scale_shape(name="Diabetes",labels=c("Negative","Positive"))+
  ylab("Age")+
  xlab("Glucose concentration")

data_2$Diabetes <- as.factor(data_2$Diabetes)
data_2_fit <- glm(formula = Diabetes ~ Glucose +Age,family = "binomial",data = data_2 )
data_2_pred <- predict(data_2_fit, type ="response" )
data_2$V10 <- ifelse(data_2_pred > 0.5,1,0)
data_2$V10 <- as.factor(data_2$V10)
data_2 %>%
  ggplot(aes(x=Glucose, y=Age, color = as.factor(V10)))+
  geom_point(size=1,aes(shape=as.factor(Diabetes)))+
  scale_color_brewer(palette = "Set2",name="Diabetes",labels=c("Negative","Positive"))+
  scale_shape(name="Diabetes",labels=c("Negative","Positive"))+
  ylab("Age")+
  xlab("Glucose concentration")+
  ggtitle("R=0.5")
data_2_tb <- table(data_2$Diabetes, data_2$V10)
MSE_data_2 <- 1- sum(diag(data_2_tb))/sum(data_2_tb)
print("Missclassification error on train is " ,MSE_data_2)

data_2 %>%
  ggplot(aes(x=Glucose, y=Age, color=as.factor(V10)))+
  geom_point(size=1,aes(shape=as.factor(Diabetes)))+
  scale_color_brewer(palette = "Set2",name="Diabetes",labels=c("Negative","Positive"))+
  scale_shape(name="Diabetes",labels=c("Negative","Positive"))+
  ylab("Age")+
  xlab("Glucose concentration")+
  geom_abline(intercept = -coef(data_2_fit)[1]/coef(data_2_fit)[3],
              slope = -coef(data_2_fit)[2]/coef(data_2_fit)[3])

data_2$V11 <- ifelse(data_2_pred > 0.2,1,0)
data_2 %>%
  ggplot(mapping=aes(x=Glucose, y=Age, color=as.factor(V11)))+
  geom_point(size=1,aes(shape=as.factor(Diabetes)))+
  scale_color_brewer(palette = "Set2",name="Diabetes",labels=c("Negative","Positive"))+
  scale_shape(name="Diabetes",labels=c("Negative","Positive"))+
  ylab("Age")+
  xlab("Glucose concentration")+
  ggtitle("R = 0.2")
data_2_tb_11 <- table(data_2$Diabetes, data_2$V11)
MSE_data_2_11 <- 1- sum(diag(data_2_tb_11))/sum(data_2_tb_11)
MSE_data_2_11


data_2$V12 <- ifelse(data_2_pred > 0.8,1,0)
```

```r
data_2_tb_12 <- table(data_2$Diabetes, data_2$V12)
MSE_data_2_12 <- 1- sum(diag(data_2_tb_12))/sum(data_2_tb_12)
MSE_data_2_12
data_2 %>%
  ggplot(aes(x=Glucose, y=Age, color=as.factor(V12)))+
  geom_point(size=1,aes(shape=as.factor(Diabetes)))+
  scale_color_brewer(palette = "Set2",name="Diabetes",labels=c("Negative","Positive"))+
  scale_shape(name="Diabetes",labels=c("Negative","Positive"))+
  ylab("Age")+
  xlab("Glucose concentration")
  ggtitle("R=0.8")

data_2$Z1 <- data_2$Glucose^4
data_2$Z2 <- data_2$Glucose^3*data_2$Age
data_2$Z3 <- data_2$Glucose^2*data_2$Age^2
data_2$Z4 <- data_2$Glucose*data_2$Age^3
data_2$Z5 <- data_2$Age^4

data_2_fit_1 <- glm(formula = Diabetes ~ Z1+Z2+Z3+Z4+Z5, data = data_2, family = "binomial")
data_2_pred_1 <- predict(data_2_fit_1, type ="response")
data_2$V13 <- ifelse(data_2_pred_1>=0.5,1,0)
data_2_tb_13 <- table(data_2$Diabetes,data_2$V13)
MSE_data_2_1 <- 1-sum(diag(data_2_tb_13))/sum(data_2_tb_13)
MSE_data_2_1

data_2_grid <- expand.grid(seq(min(data_2$Age),max(data_2$Age),length.out=500),
                           seq(min(data_2$Glucose),max(data_2$Glucose),length.out=500))
colnames(data_2_grid)<- c("Age","Glucose")
data_2_grid$Z1 <- data_2_grid$Glucose^4
data_2_grid$Z2 <- data_2_grid$Glucose^3*data_2_grid$Age
data_2_grid$Z3 <- data_2_grid$Glucose^2*data_2_grid$Age^2
data_2_grid$Z4 <- data_2_grid$Glucose*data_2_grid$Age^3
data_2_grid$Z5 <- data_2_grid$Age^4
data_2_pred_1 <- predict(data_2_fit_1, type ="response", newdata = data_2_grid )
data_2_grid$V13 <- ifelse(data_2_pred_1 >= 0.5,1,0)
ggplot(data_2_grid, aes(x=Glucose, y=Age,fill=as.factor(V13)))+
  geom_raster(alpha = 0.3, interpolate = TRUE)+
  geom_point(data=data_2,aes(x=Glucose, y=Age, color=as.factor(Diabetes)))+
  geom_abline(intercept = -coef(data_2_fit)[1]/coef(data_2_fit)[3],
              slope = -coef(data_2_fit)[2]/coef(data_2_fit)[3])+
  theme(legend.position = "none")
```