machine learning(732A99) lab1

Anubhav Dikshit(anudi287)

26 November 2018

Assignment 1

Loading The Libraries

```
## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contril
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'numDeriv' successfully unpacked and MD5 sums checked
## package 'SQUAREM' successfully unpacked and MD5 sums checked
## package 'lava' successfully unpacked and MD5 sums checked
## package 'CVST' successfully unpacked and MD5 sums checked
## package 'magic' successfully unpacked and MD5 sums checked
## package 'prodlim' successfully unpacked and MD5 sums checked
## package 'DRR' successfully unpacked and MD5 sums checked
## package 'sfsmisc' successfully unpacked and MD5 sums checked
## package 'geometry' successfully unpacked and MD5 sums checked
## package 'ipred' successfully unpacked and MD5 sums checked
## package 'dimRed' successfully unpacked and MD5 sums checked
## package 'timeDate' successfully unpacked and MD5 sums checked
## package 'ddalpha' successfully unpacked and MD5 sums checked
## package 'gower' successfully unpacked and MD5 sums checked
## package 'RcppRoll' successfully unpacked and MD5 sums checked
## package 'pls' successfully unpacked and MD5 sums checked
## package 'ModelMetrics' successfully unpacked and MD5 sums checked
## package 'recipes' successfully unpacked and MD5 sums checked
## package 'caret' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\anubh\AppData\Local\Temp\RtmpuW9Nrq\downloaded_packages

## Warning: unable to access index for repository http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contril
##   cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.5/PACKAGES'

## package 'kknn' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\anubh\AppData\Local\Temp\RtmpuW9Nrq\downloaded_packages
```

Loading Input files

```
spam_data <- read.xlsx("spambase.xlsx", sheetName = "spambase_data")
spam_data$Spam <- as.factor(spam_data$Spam)

tecator_data <- read.xlsx("tecator.xlsx", sheetName = "data")
```

1.1 Import the data into R and divide it into training and test sets (50%/50%) by using the following code

```
set.seed(12345)

n =  NROW(spam_data)
id = sample(1:n, floor(n*0.5))
train = spam_data[id,]
test = spam_data[-id,]
```

1.2 Use logistic regression (functions glm(), predict()) to classify the training and test data by the classification principles

```
min.model = glm(Spam ~ 1, family=binomial, data=train)
biggest <- formula(glm(Spam ~., family=binomial, data=train))

full.model <- glm(Spam ~., family=binomial, data=train)
step.model <- step(min.model, direction='forward', scope=biggest)
summary(step.model)
```

Manual Feature Selection

```
best_model <- glm(formula = Spam ~ Word35 + Word46 + Word42 + Word44 + Word33 +
    Word45 + Word39 + Word48 + Word30 + Word43 + Word37 +
    Word36 + Word31, family = binomial, data = train)

#export_summs(step.model, best_model,
#model.names = c("Model using Step","Model Manually Tunned"))
```

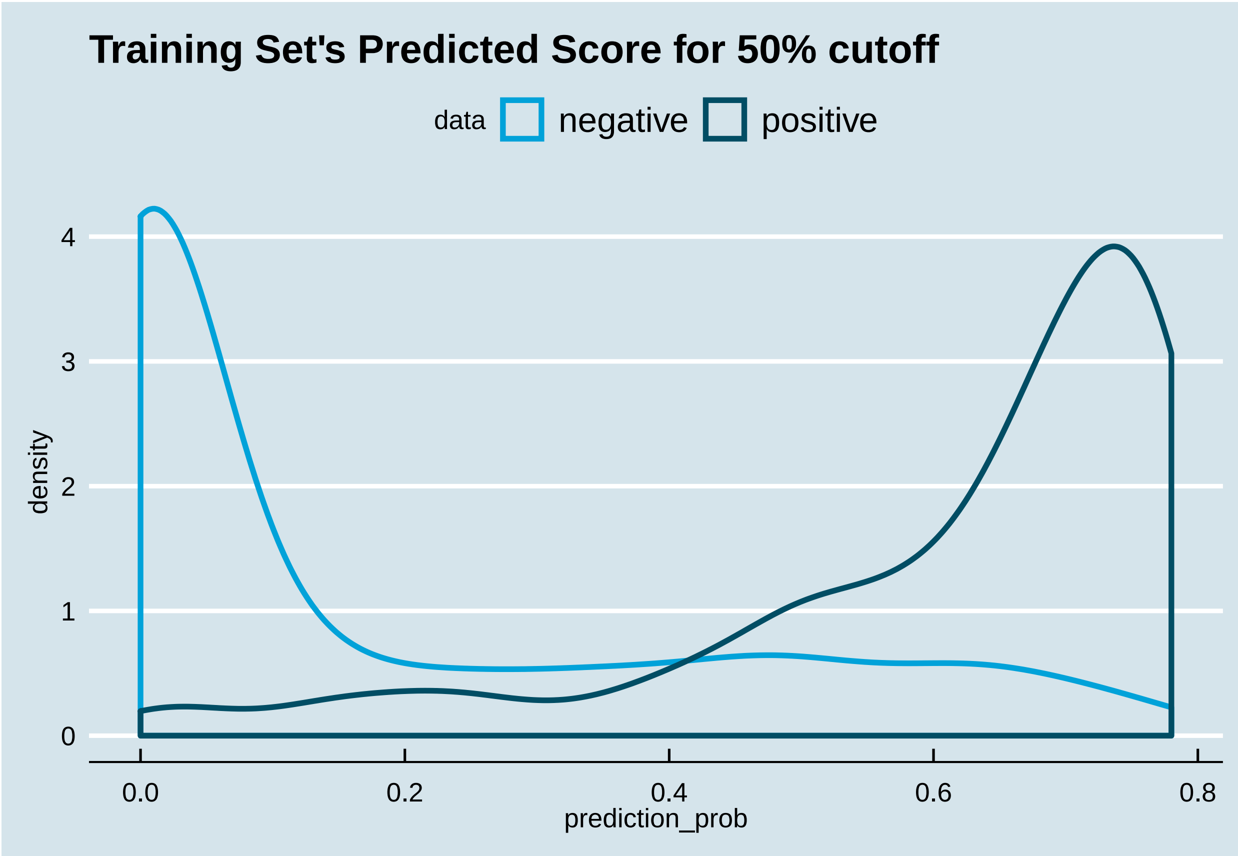
Prediction for probability greater than 50% and 90%

```
# prediction
train$prediction_prob <- predict(best_model, newdata = train, type = "response")
test$prediction_prob <- predict(best_model, newdata = test , type = "response")

train$prediction_class_50 <- ifelse(train$prediction_prob > 0.50, 1, 0)
test$prediction_class_50 <- ifelse(test$prediction_prob > 0.50, 1, 0)

train$prediction_class_90 <- ifelse(train$prediction_prob > 0.90, 1, 0)
test$prediction_class_90 <- ifelse(test$prediction_prob > 0.90, 1, 0)
```

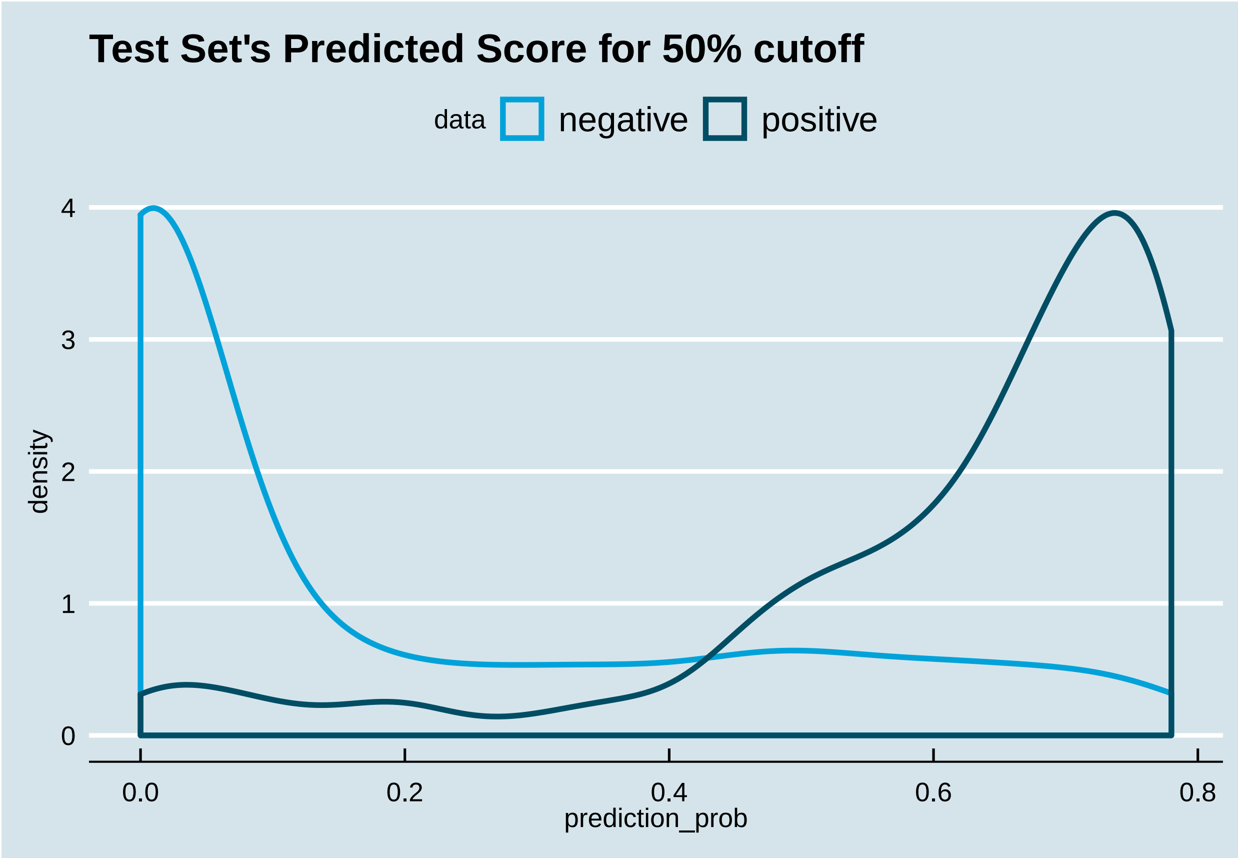
Assessing the Model

```
# plots
ggplot(train, aes(prediction_prob, color = Spam)) +
geom_density(size = 1) + ggtitle("Training Set's Predicted Score for 50% cutoff") +
```

```r
  scale_color_economist(name = "data", labels = c("negative", "positive")) +
  theme_economist()
```




```r
ggplot(test, aes(prediction_prob, color = Spam)) +
geom_density(size = 1) + ggtitle("Test Set's Predicted Score for 50% cutoff") +
  scale_color_economist(name = "data", labels = c("negative", "positive")) +
  theme_economist()
```



1.2 Assessing the Fit on train dataset for 50%

```r
#confusion table
conf_train <- table(train$Spam, train$prediction_class_50)
names(dimnames(conf_train)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train)
```

```
## Confusion Matrix and Statistics
##
##             Predicted Train
## Actual Train   0   1
##            0 799 146
##            1  88 337
##
##                Accuracy : 0.8292
##                  95% CI : (0.8082, 0.8488)
##     No Information Rate : 0.6474
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.6153
##  Mcnemar's Test P-Value : 0.0001944
##
##             Sensitivity : 0.9008
##             Specificity : 0.6977
##          Pos Pred Value : 0.8455
```

```
##          Neg Pred Value : 0.7929
##              Prevalence : 0.6474
##          Detection Rate : 0.5832
##    Detection Prevalence : 0.6898
##       Balanced Accuracy : 0.7993
##
##        'Positive' Class : 0
##
```

```
conf_test <- table(test$Spam, test$prediction_class_50)
names(dimnames(conf_test)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test)
```

```
## Confusion Matrix and Statistics
##
##            Predicted Test
## Actual Test   0   1
##           0 785 152
##           1  80 353
##
##                Accuracy : 0.8307
##                  95% CI : (0.8097, 0.8502)
##     No Information Rate : 0.6314
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.6251
##  Mcnemar's Test P-Value : 0.000003141
##
##             Sensitivity : 0.9075
##             Specificity : 0.6990
##          Pos Pred Value : 0.8378
##          Neg Pred Value : 0.8152
##              Prevalence : 0.6314
##          Detection Rate : 0.5730
##    Detection Prevalence : 0.6839
##       Balanced Accuracy : 0.8033
##
##        'Positive' Class : 0
##
```

Analysis: Distribution of the prediction score grouped by known outcome given that our model's final objective is to classify new instances into one of two categories (spam vs. non-spam). We will want the model to give high scores to positive instances (1: spam) and low scores (0 : not spam) otherwise.Ideally you want the distribution of scores to be separated, with the score of the negative instances to be on the left and the score of the positive instance to be on the right.

From the confusion matrix it is apparent that Accuracy on train and test dataset when cutoff=50% is about 83%.

1.3 Assessing the Fit on train dataset for 90%

```
#confusion table
conf_train1 <- table(train$Spam, train$prediction_class_90)
names(dimnames(conf_train1)) <- c("Actual Train", "Predicted Train")
```

```r
conf_train1
```

```
##              Predicted Train
## Actual Train   0
##             0 945
##             1 425
```

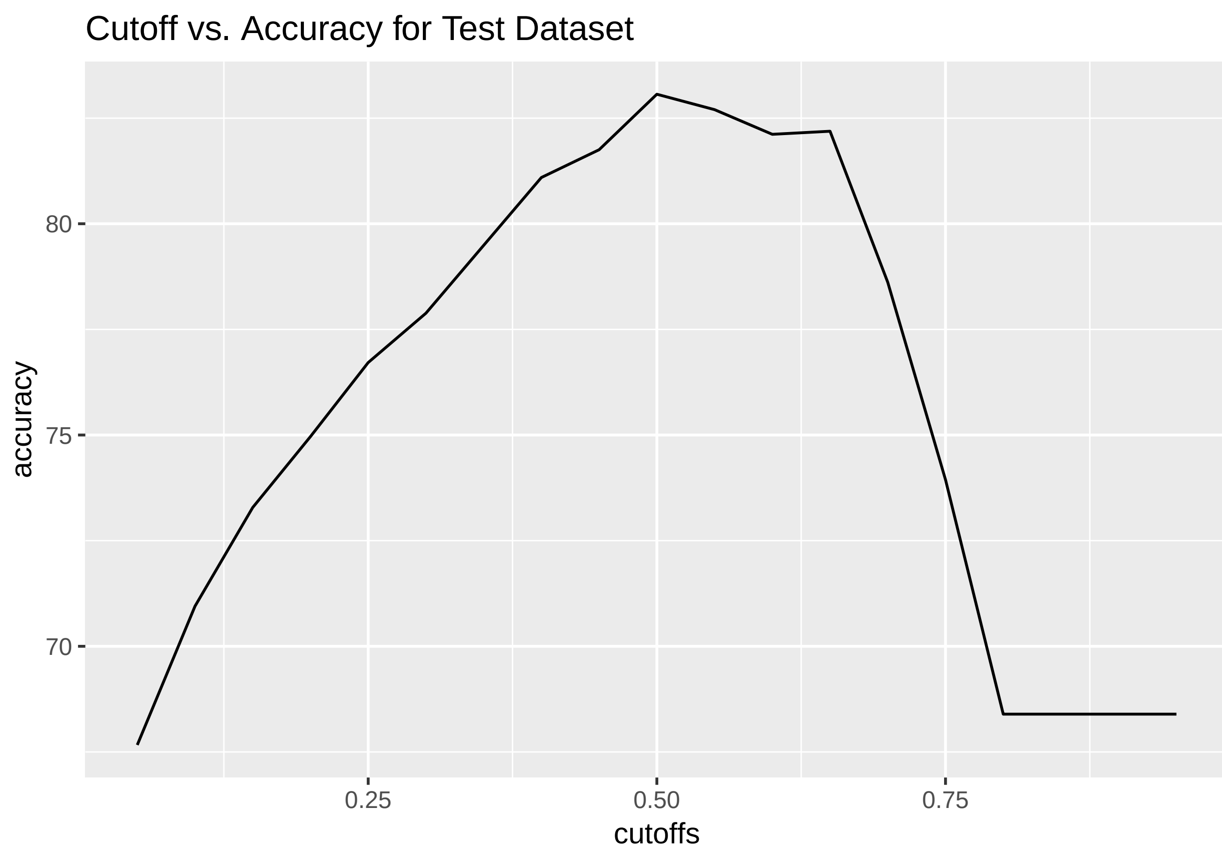
```r
conf_test1 <- table(test$Spam, test$prediction_class_90)
names(dimnames(conf_test1)) <- c("Actual Test", "Predicted Test")
conf_test1
```

```
##             Predicted Test
## Actual Test   0
##            0 937
##            1 433
```

Analysis: Strange, the model only predicts one class!! We know that the prediction of a logistic regression model is a probability, thus in order to use it as a classifier, we'll have to choose a cutoff value, or threshold (cutoff). Where scores above this value will classified as positive, those below as negative. Lets us find this optimum value.

Choosing the best cutoff for test

```r
cutoffs <- seq(from = 0.05, to = 0.95, by = 0.05)
accuracy <- NULL

for (i in seq_along(cutoffs)){
    prediction <- ifelse(test$prediction_prob >= cutoffs[i], 1, 0) #Predicting for cut-off

    accuracy <- c(accuracy,length(which(test$Spam == prediction))/length(prediction)*100)}

cutoff_data <- as.data.frame(cbind(cutoffs, accuracy))

ggplot(data = cutoff_data, aes(x = cutoffs, y = accuracy)) +
  geom_line() +
  ggtitle("Cutoff vs. Accuracy for Test Dataset")
```



Analysis: Our small detour suggests that the cutoff value of 50% was the best for our purpose and going higher than this leads to worse results, at 0.8 and above the accuracy drastically reduces which is what we see when we make cutoff as 0.9.

From the confusion matrix it is evident that the model becomes a trivial model(predicts all cases as one class) and thus the prediction is no better than tossing a coin. This should be the absoutely the worst case that we should avoid.

1.4 Use standard classifier kknn() with K=30 from package kknn, report the the misclassification rates for the training and test data and compare the results with step 1.2.

```
knn_model30 <- train.kknn(Spam ~ Word35 + Word46 + Word42 + Word44 + Word33 +
    Word45 + Word39 + Word48 + Word30 + Word43 + Word37 +
    Word36 + Word31, data = train, kmax = 30)

train$knn_prediction_class <- predict(knn_model30, train)
test$knn_prediction_class <- predict(knn_model30, test)

conf_train2 <- table(train$Spam, train$knn_prediction_class)
names(dimnames(conf_train2)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train2)
```

```
## Confusion Matrix and Statistics
##
```

```
##              Predicted Train
## Actual Train   0   1
##            0 869  76
##            1  48 377
##
##                Accuracy : 0.9095
##                  95% CI : (0.893, 0.9242)
##     No Information Rate : 0.6693
##     P-Value [Acc > NIR] : < 0.0000000000000002
##
##                   Kappa : 0.7923
##  Mcnemar's Test P-Value : 0.01532
##
##             Sensitivity : 0.9477
##             Specificity : 0.8322
##          Pos Pred Value : 0.9196
##          Neg Pred Value : 0.8871
##              Prevalence : 0.6693
##          Detection Rate : 0.6343
##    Detection Prevalence : 0.6898
##       Balanced Accuracy : 0.8899
##
##        'Positive' Class : 0
##
```

```r
conf_test2 <- table(test$Spam, test$knn_prediction_class)
names(dimnames(conf_test2)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test2)
```

```
## Confusion Matrix and Statistics
##
##            Predicted Test
## Actual Test   0   1
##           0 800 137
##           1  67 366
##
##                Accuracy : 0.8511
##                  95% CI : (0.8311, 0.8695)
##     No Information Rate : 0.6328
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.6699
##  Mcnemar's Test P-Value : 0.000001359
##
##             Sensitivity : 0.9227
##             Specificity : 0.7276
##          Pos Pred Value : 0.8538
##          Neg Pred Value : 0.8453
##              Prevalence : 0.6328
##          Detection Rate : 0.5839
##    Detection Prevalence : 0.6839
##       Balanced Accuracy : 0.8252
##
##        'Positive' Class : 0
##
```

Analysis: Using KKNN with K = 30, increased our trainning accuracy to 90%, however using trainning error/accuracyis a bad 83%

1.5 Repeat step 4 for K=1 and compare the results with step 4. What effect does the decrease of K lead to and why?

```r
knn_model1 <- train.kknn(Spam ~ Word35 + Word46 + Word42 + Word44 + Word33 +
    Word45 + Word39 + Word48 + Word30 + Word43 + Word37 +
    Word36 + Word31, data = train, kmax = 1)

train$knn_prediction_class <- predict(knn_model1, train)
test$knn_prediction_class <- predict(knn_model1, test)

conf_train2 <- table(train$Spam, train$knn_prediction_class)
names(dimnames(conf_train2)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train2)
```

```
## Confusion Matrix and Statistics
##
##               Predicted Train
## Actual Train    0   1
##             0 912  33
##             1  18 407
##
##                Accuracy : 0.9628
##                  95% CI : (0.9513, 0.9722)
##     No Information Rate : 0.6788
##     P-Value [Acc > NIR] : < 0.0000000000000002
##
##                   Kappa : 0.9139
##  Mcnemar's Test P-Value : 0.04995
##
##             Sensitivity : 0.9806
##             Specificity : 0.9250
##          Pos Pred Value : 0.9651
##          Neg Pred Value : 0.9576
##              Prevalence : 0.6788
##          Detection Rate : 0.6657
##    Detection Prevalence : 0.6898
##       Balanced Accuracy : 0.9528
##
##        'Positive' Class : 0
##
```

```r
conf_test2 <- table(test$Spam, test$knn_prediction_class)
names(dimnames(conf_test2)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test2)
```

```
## Confusion Matrix and Statistics
##
##              Predicted Test
## Actual Test    0   1
##            0 782 155
```

```
##           1  77 356
##
##                Accuracy : 0.8307
##                  95% CI : (0.8097, 0.8502)
##     No Information Rate : 0.627
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.6264
##  Mcnemar's Test P-Value : 0.0000004297
##
##             Sensitivity : 0.9104
##             Specificity : 0.6967
##          Pos Pred Value : 0.8346
##          Neg Pred Value : 0.8222
##              Prevalence : 0.6270
##          Detection Rate : 0.5708
##    Detection Prevalence : 0.6839
##       Balanced Accuracy : 0.8035
##
##        'Positive' Class : 0
##
```

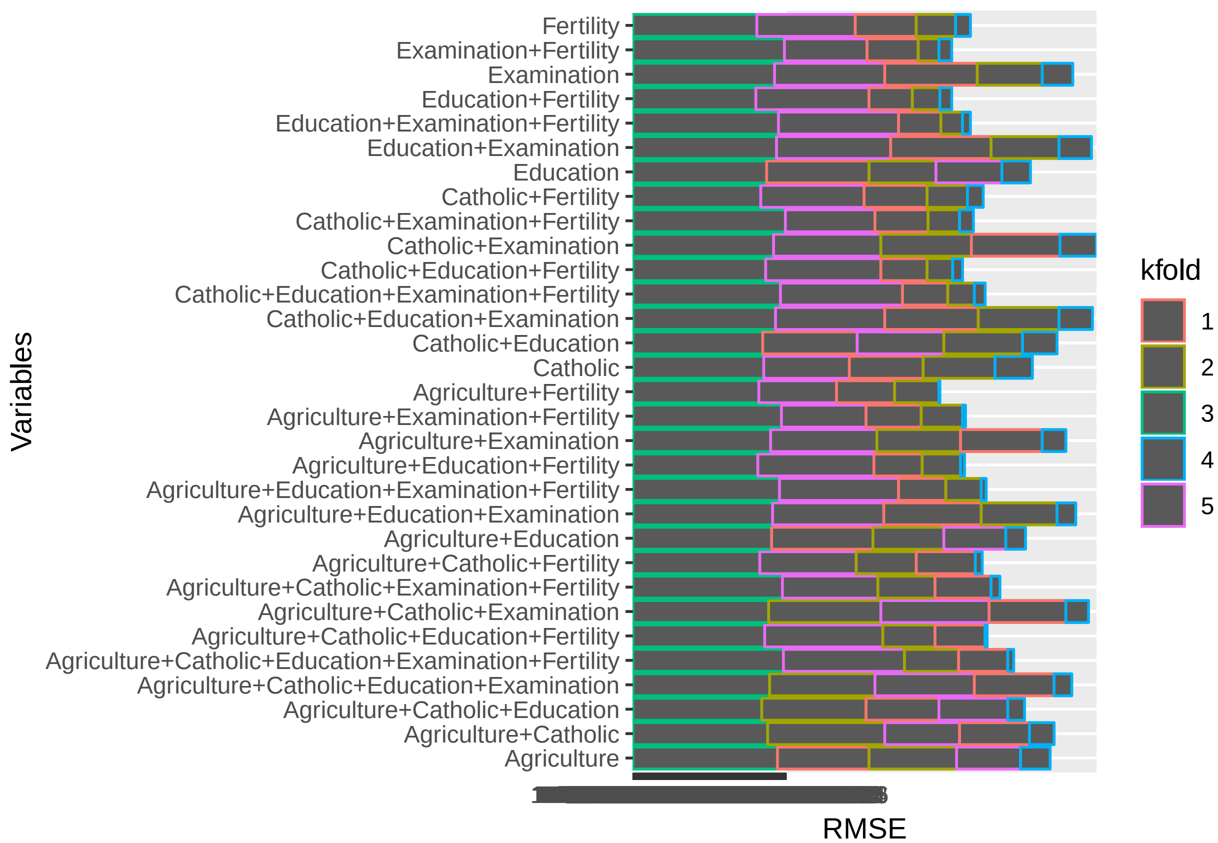
Analysis:

Assignment 2 Feature selection by cross-validation in a linear model

2.1 Implement an R function that performs feature selection (best subset selection) in linear regression by using k-fold cross-validation without using any specialized function like lm() (use only basic R functions)

```r
subset_function <- function(X,Y,N){

# X = swiss[,1:5]
# Y = swiss[,6:6]
# N = 5
df <- cbind(X,Y)

 temp <- NULL
for(i in 1:NCOL(X)){
combs <- as.data.frame(gtools::combinations(NCOL(X), r=i, v=colnames(X), repeats.allowed=FALSE))
combs <- tidyr::unite(combs, "formula", sep = "+")
temp <- rbind(combs, temp)
}

set.seed(12345)
df2 <- df[sample(nrow(df)),]
df2$k_fold <- sample(N, size = nrow(df), replace = TRUE)

result <- NULL


for (j in 1:NROW(temp))
```

```
{
  for(i in 1:N){

train = df2[df2$k_fold != i,]
test = df2[df2$k_fold == i,]

model_forumla = paste("Y ~ ", temp[j,],sep = "")

model <- lm(formula = model_forumla, data = train)
predicted <- predict(model, newdata = test)

RMSE <- sqrt(mean((predicted - test$Y)^2))
data <- cbind(i,temp[j,], RMSE)
result <- rbind(data, result)

}
}

colnames(result) <- c("kfold", "Variables", "RMSE")
result <- as.data.frame(result)
return(result)
}

swiss_subset <- subset_function(X = swiss[,1:5], Y = swiss[,6], N = 5)
ggplot(data = swiss_subset, aes(x = Variables, y = RMSE, color=kfold)) + geom_bar(stat="identity") + co
```



Apendix

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

if (!require("pacman")) install.packages("pacman")
pacman::p_load(xlsx, glmnet, MASS, jtools, huxtable, ggplot2,
               ggthemes, gridExtra, ROCR, broom, caret, e1071,
               kknn, tidyr)

options("jtools-digits" = 2, scipen = 999)

spam_data <- read.xlsx("spambase.xlsx", sheetName = "spambase_data")
spam_data$Spam <- as.factor(spam_data$Spam)

tecator_data <- read.xlsx("tecator.xlsx", sheetName = "data")

set.seed(12345)

n =  NROW(spam_data)
id = sample(1:n, floor(n*0.5))
train = spam_data[id,]
test = spam_data[-id,]

min.model = glm(Spam ~ 1, family=binomial, data=train)
biggest <- formula(glm(Spam ~., family=binomial, data=train))

full.model <- glm(Spam ~., family=binomial, data=train)
step.model <- step(min.model, direction='forward', scope=biggest)
summary(step.model)
best_model <- glm(formula = Spam ~ Word35 + Word46 + Word42 + Word44 + Word33 +
    Word45 + Word39 + Word48 + Word30 + Word43 + Word37 +
    Word36 + Word31, family = binomial, data = train)

#export_summs(step.model, best_model,
#model.names = c("Model using Step","Model Manually Tunned"))
# prediction
train$prediction_prob <- predict(best_model, newdata = train, type = "response")
test$prediction_prob <- predict(best_model, newdata = test , type = "response")

train$prediction_class_50 <- ifelse(train$prediction_prob > 0.50, 1, 0)
test$prediction_class_50 <- ifelse(test$prediction_prob > 0.50, 1, 0)

train$prediction_class_90 <- ifelse(train$prediction_prob > 0.90, 1, 0)
test$prediction_class_90 <- ifelse(test$prediction_prob > 0.90, 1, 0)

# plots
ggplot(train, aes(prediction_prob, color = Spam)) +
geom_density(size = 1) + ggtitle("Training Set's Predicted Score for 50% cutoff") +
  scale_color_economist(name = "data", labels = c("negative", "positive")) +
  theme_economist()


ggplot(test, aes(prediction_prob, color = Spam)) +
```

```
geom_density(size = 1) + ggtitle("Test Set's Predicted Score for 50% cutoff") +
  scale_color_economist(name = "data", labels = c("negative", "positive")) +
  theme_economist()

#confusion table
conf_train <- table(train$Spam, train$prediction_class_50)
names(dimnames(conf_train)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train)

conf_test <- table(test$Spam, test$prediction_class_50)
names(dimnames(conf_test)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test)

#confusion table
conf_train1 <- table(train$Spam, train$prediction_class_90)
names(dimnames(conf_train1)) <- c("Actual Train", "Predicted Train")
conf_train1

conf_test1 <- table(test$Spam, test$prediction_class_90)
names(dimnames(conf_test1)) <- c("Actual Test", "Predicted Test")
conf_test1


cutoffs <- seq(from = 0.05, to = 0.95, by = 0.05)
accuracy <- NULL

for (i in seq_along(cutoffs)){
    prediction <- ifelse(test$prediction_prob >= cutoffs[i], 1, 0) #Predicting for cut-off

    accuracy <- c(accuracy,length(which(test$Spam == prediction))/length(prediction)*100)}

cutoff_data <- as.data.frame(cbind(cutoffs, accuracy))

ggplot(data = cutoff_data, aes(x = cutoffs, y = accuracy)) +
  geom_line() +
  ggtitle("Cutoff vs. Accuracy for Test Dataset")

knn_model30 <- train.kknn(Spam ~ Word35 + Word46 + Word42 + Word44 + Word33 +
    Word45 + Word39 + Word48 + Word30 + Word43 + Word37 +
    Word36 + Word31, data = train, kmax = 30)

train$knn_prediction_class <- predict(knn_model30, train)
test$knn_prediction_class <- predict(knn_model30, test)

conf_train2 <- table(train$Spam, train$knn_prediction_class)
names(dimnames(conf_train2)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train2)

conf_test2 <- table(test$Spam, test$knn_prediction_class)
names(dimnames(conf_test2)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test2)
knn_model1 <- train.kknn(Spam ~ Word35 + Word46 + Word42 + Word44 + Word33 +
    Word45 + Word39 + Word48 + Word30 + Word43 + Word37 +
```

```
    Word36 + Word31, data = train, kmax = 1)

train$knn_prediction_class <- predict(knn_model1, train)
test$knn_prediction_class <- predict(knn_model1, test)

conf_train2 <- table(train$Spam, train$knn_prediction_class)
names(dimnames(conf_train2)) <- c("Actual Train", "Predicted Train")
confusionMatrix(conf_train2)

conf_test2 <- table(test$Spam, test$knn_prediction_class)
names(dimnames(conf_test2)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test2)

subset_function <- function(X,Y,N){

# X = swiss[,1:5]
# Y = swiss[,6:6]
# N = 5
df <- cbind(X,Y)

 temp <- NULL
for(i in 1:NCOL(X)){
combs <- as.data.frame(gtools::combinations(NCOL(X), r=i, v=colnames(X), repeats.allowed=FALSE))
combs <- tidyr::unite(combs, "formula", sep = "+")
temp <- rbind(combs, temp)
}

set.seed(12345)
df2 <- df[sample(nrow(df)),]
df2$k_fold <- sample(N, size = nrow(df), replace = TRUE)

result <- NULL


for (j in 1:NROW(temp))
{
  for(i in 1:N){

train = df2[df2$k_fold != i,]
test = df2[df2$k_fold == i,]

model_forumla = paste("Y ~ ", temp[j,],sep = "")

model <- lm(formula = model_forumla, data = train)
predicted <- predict(model, newdata = test)

RMSE <- sqrt(mean((predicted - test$Y)^2))
data <- cbind(i,temp[j,], RMSE)
result <- rbind(data, result)

}
}
```

```r
colnames(result) <- c("kfold", "Variables", "RMSE")
result <- as.data.frame(result)
return(result)
}

swiss_subset <- subset_function(X = swiss[,1:5], Y = swiss[,6], N = 5)
ggplot(data = swiss_subset, aes(x = Variables, y = RMSE, color=kfold)) + geom_bar(stat="identity") + coo
```