FML_Assignment_2

Karthik Badiganti

2022-10-02

Loading Packages

```
library(class)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(ISLR)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(fastDummies)
library(knitr)
library(gmodels)
library(ggplot2)
```

Importing & Cleaning Data

We are Importing Data from CSV file and cleaning

```
Universal_bank <- read.csv("C://Users//gbkar//Documents//R Scripts//UniversalBank.csv")


#Removing Variables that are not required
Universal_bank <-Universal_bank[,c(2,3,4,6,7,8,9,10,11,12,13,14)]
```

```r
#Making decision variable into factor as it is a classification model
Universal_bank$Personal.Loan<-as.factor(Universal_bank$Personal.Loan)

# Making categorical variable to factor and creating dummy variables
Universal_bank$Education<-as.factor(Universal_bank$Education)
Universal_bank <- dummy_columns(Universal_bank,select_columns = 'Education')

#Removing unnecessary variables and rearranging the variable as per test data
Universal_bank <-Universal_bank[,c("Personal.Loan",'Age','Experience','Income',"Family","CCAvg","Educat

head(Universal_bank)
```

```
##   Personal.Loan Age Experience Income Family CCAvg Education_1 Education_2
## 1             0  25          1     49      4   1.6           1           0
## 2             0  45         19     34      3   1.5           1           0
## 3             0  39         15     11      1   1.0           1           0
## 4             0  35          9    100      1   2.7           0           1
## 5             0  35          8     45      4   1.0           0           1
## 6             0  37         13     29      4   0.4           0           1
##   Education_3 Mortgage Securities.Account CD.Account Online CreditCard
## 1           0        0                  1          0      0          0
## 2           0        0                  1          0      0          0
## 3           0        0                  0          0      0          0
## 4           0        0                  0          0      0          0
## 5           0        0                  0          0      0          1
## 6           0      155                  0          0      1          0
```

Data Partition and Normalization

```r
set.seed(123)
#Partitioning Data into 60% Training and 40% Validation
Index_Train<-createDataPartition(Universal_bank$Personal.Loan, p=0.6, list=FALSE)

Universal_bank_Train <-Universal_bank[Index_Train,]
Universal_bank_Validation  <-Universal_bank[-Index_Train,]

#Creating Test data based on the question
Universal_bank_test<-Universal_bank[0,-1]
test_data<-c(40,10,84,2,2,0,1,0,0,0,0,1,1)
Universal_bank_test[nrow(Universal_bank_test) + 1, ] <- test_data

#Omitting all categorical variables for Normalization
norm_var <- c("Age","Experience","Income","Family","CCAvg","Mortgage")

#Creating Normalization model based on Training data
norm_model<-preProcess(Universal_bank_Train[,norm_var], method = c("center", "scale"))


#Applying Normalization model to all three data
Universal_bank_norm_Train <-predict(norm_model,Universal_bank_Train)
```

```
Universal_bank_norm_Validation  <-predict(norm_model,Universal_bank_Validation)
Universal_bank_norm_test<-predict(norm_model,Universal_bank_test)
```

KNN Classification

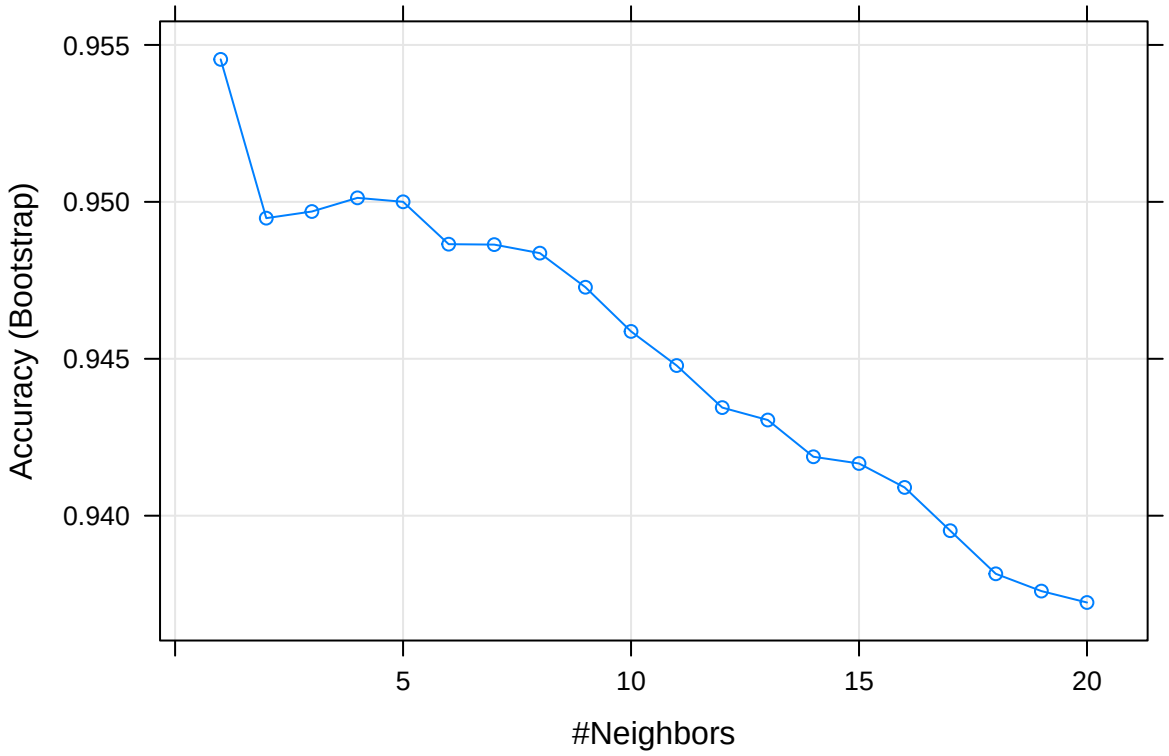
Problem 1 with K=1

```
set.seed(3333)

#Creating Train label and train predictor
train_label<- Universal_bank_Train[,1]
train_predictor<- Universal_bank_norm_Train[-1]

#Applying KNN for K=1 for test data
Loan_predicted_1 <-knn(train_predictor,Universal_bank_norm_test, cl=train_label,
               k=1)

if (Loan_predicted_1==1){
  print("Loan Granted for given test Dataset for K=1")
}else
  print("Loan Not Granted for given test Dataset for K=1")
```

```
## [1] "Loan Not Granted for given test Dataset for K=1"
```

Problem 2 Getting Best Optimal K

```
set.seed(3333)

#Training model to get best K value based on "repeated cross validation" method
Search_grid <- expand.grid(k=c(1:20))
trctrl <- trainControl(method = "boot")
model<-train(Personal.Loan~.,data=Universal_bank_norm_Train,trControl=trctrl,
             method="knn", tuneGrid=Search_grid
            )

#Plotting and printing Best K value
plot(model)
```




```
print(paste("Value of K based on accuracy is",model$bestTune[[1]],"and its corresponding accuracy is",ma
```

```
## [1] "Value of K based on accuracy is 1 and its corresponding accuracy is 0.95454012360381"
```

Problem 3 Applying Optimal K to Validation Data Set

```
set.seed(3333)
#Predicting Validation Data set Loan Output Based on Best Optimal K value
Loan_predicted_2 <-knn(train_predictor,Universal_bank_norm_Validation[-1], cl=train_label,
               k=model$bestTune[[1]])


#Creating Validation Label
validation_label<- Universal_bank_norm_Validation[,1]

confusionMatrix(Loan_predicted_2,Universal_bank_norm_Validation$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1796   53
##          1   12  139
```

```
##
##                Accuracy : 0.9675
##                  95% CI : (0.9588, 0.9748)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.793
##
##  Mcnemar's Test P-Value : 6.999e-07
##
##             Sensitivity : 0.9934
##             Specificity : 0.7240
##          Pos Pred Value : 0.9713
##          Neg Pred Value : 0.9205
##              Prevalence : 0.9040
##          Detection Rate : 0.8980
##    Detection Prevalence : 0.9245
##       Balanced Accuracy : 0.8587
##
##        'Positive' Class : 0
##
```

Problem 4 Applying Optimal K to Test Data Set

```
set.seed(3333)
#Applying KNN for Best optimal K to Test Data Set
Loan_predicted_3 <-knn(train_predictor,Universal_bank_norm_test, cl=train_label,
               k=model$bestTune[[1]])


if (Loan_predicted_3==1){
  print("Loan Granted for given test Dataset for Best optimal K")
}else
  print("Loan Not Granted for given test Dataset  for Best optimal K")
```

```
## [1] "Loan Not Granted for given test Dataset  for Best optimal K"
```

Problem 5 Repartition the data into Training, Validation, and Test sets (50% : 30% : 20%)

```
set.seed(123)
#Partitioning Data into 50% Training and 30% Validation and 20% Test
Index_Train_2<-createDataPartition(Universal_bank$Personal.Loan, p=0.5, list=FALSE)

Universal_bank_Train_2 <-Universal_bank[Index_Train_2,]
Universal_bank_Validation_x  <-Universal_bank[-Index_Train_2,]

Index_Train_3<-createDataPartition(Universal_bank_Validation_x$Personal.Loan, p=0.6, list=FALSE)
```

```
Universal_bank_Validation_2 <-Universal_bank_Validation_x[Index_Train_3,]
Universal_bank_test_2  <-Universal_bank_Validation_x[-Index_Train_3,]
```

Normalize the data.

```
set.seed(123)
#Creating Normalization model based on Training data
norm_model_2<-preProcess(Universal_bank_Train_2[,norm_var], method = c("center", "scale"))


#Applying Normalization model to all three data
Universal_bank_norm_Train_2 <-predict(norm_model_2,Universal_bank_Train_2)
Universal_bank_norm_Validation_2  <-predict(norm_model_2,Universal_bank_Validation_2)
Universal_bank_norm_test_2<-predict(norm_model_2,Universal_bank_test_2)
```

Applying KNN for Partitioned data

```
# Applying KNN for Validation data set and building confusion matrix
Loan_predicted_4 <- knn(Universal_bank_norm_Train_2[-1], Universal_bank_norm_Validation_2[-1],cl=Univers
s2<-confusionMatrix(Loan_predicted_4,Universal_bank_norm_Validation_2$Personal.Loan)
s2$table
```

```
##            Reference
## Prediction    0    1
##          0 1340   38
##          1   16  106
```

```
# Applying KNN for Test data set and building confusion matrix
Loan_predicted_5 <- knn(Universal_bank_norm_Train_2[-1], Universal_bank_norm_test_2[-1],cl=Universal_ban
s1<-confusionMatrix(Loan_predicted_5,Universal_bank_norm_test_2$Personal.Loan)
s1$table
```

```
##            Reference
## Prediction   0   1
##          0 900  28
##          1   4  68
```

```
# Applying KNN for Train data set and building confusion matrix
Loan_predicted_6 <- knn(Universal_bank_norm_Train_2[-1], Universal_bank_norm_Train_2[-1],cl=Universal_ba
s3<-confusionMatrix(Loan_predicted_6,Universal_bank_norm_Train_2$Personal.Loan)
s3$table
```

```
##            Reference
## Prediction    0    1
##          0 2260    0
##          1    0  240
```

```
print(paste("Miscalculations happened for Test data is",sum(s1$table[[2]],s1$table[[3]])))
```

```
## [1] "Miscalculations happened for Test data is 32"
```

```
print(paste("Miscalculations happened for Validation data is",sum(s2$table[[2]],s2$table[[3]])))
```

```
## [1] "Miscalculations happened for Validation data is 54"
```

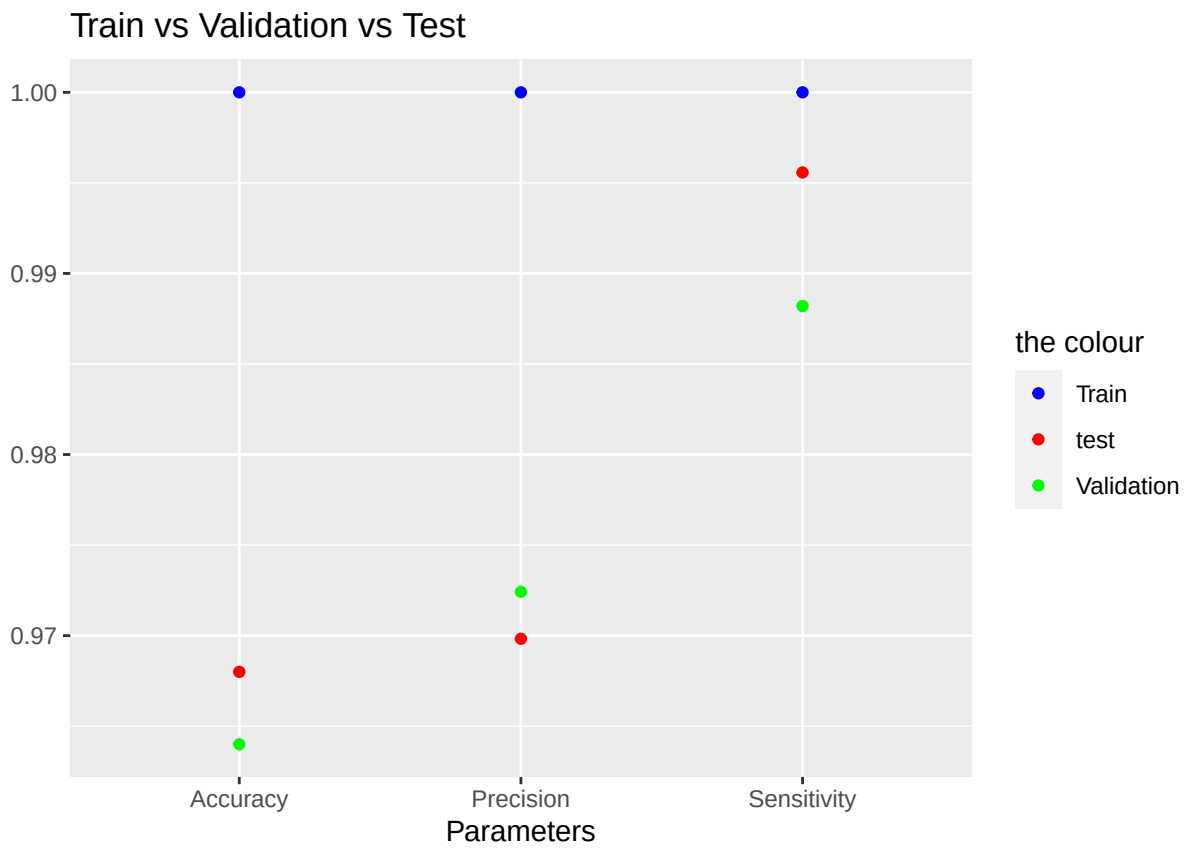
```
print(paste("Miscalculations happened for Train data is",sum(s3$table[[2]],s3$table[[3]])))
```

```
## [1] "Miscalculations happened for Train data is 0"
```

```
Test_2<-c(s1$byClass[[1]],s1$byClass[[5]],s1$overall[[1]])
Valid_2<-c(s2$byClass[[1]],s2$byClass[[5]],s2$overall[[1]])
Train_2<-c(s3$byClass[[1]],s3$byClass[[5]],s3$overall[[1]])
Parameters<-c("Sensitivity","Precision","Accuracy")




New_plot<-data.frame(Parameters,Test_2,Valid_2,Train_2)
#Printing Error Percentages for different parameters
print(New_plot)
```

```
##    Parameters    Test_2   Valid_2 Train_2
## 1 Sensitivity 0.9955752 0.9882006       1
## 2   Precision 0.9698276 0.9724238       1
## 3    Accuracy 0.9680000 0.9640000       1
```

```
# Plotting Accuracy, Precision, Sensitivity
ggplot(New_plot, aes(Parameters))+
  geom_point(aes(y=Test_2,color="Test")) +
  geom_point(aes(y=Valid_2), colour="green")+
  geom_point(aes(y=Train_2), colour="blue")+
    scale_colour_manual(name = 'the colour',
         values =c('blue'='blue','Test'='red','green'='green'), labels = c('Train','test','Validation'))
  xlab("Parameters") +ylab(" ")+

  labs(title="Train vs Validation vs Test")
```



Based on the results and the plots above,

1. Since the training data has been known accuracy,precision and sensitivity are at 100%

2. Validation data has an accuracy of 96.4% with precision of 98.8%, the accuracy decreased because the data has been re partitioned but the difference is almost minimum

3. Test data has an accuracy of 96.8% with precision of 97.24%, the parameters has didn't show much of a difference when compared to validation