Lab2 - Naive Bayes Model

Group 1 - Laxman Panthi; Zubin Shah

PART 1:

Overview:

The first part focuses on the credit risk dataset for a bank. The dataset tells us whether the customer will be defaulted on the debt. Defaulting is considered if a customer is not able to pay the borrowed amount on time. Initially the focus would be to create a predictive model specifically Naïve Bayesian Classifier which to determine which customers might be on credit risks and later to optimize and improve it.

Credit Rating Dataset

First dataset was read in R using the read.csv("creditData.csv") function and we use summary(creditData) to explore the data.

Data Exploration

```
##  Creditability Account Balance Duration of Credit (month)
##  Min.   :0.0   Min.   :1.000   Min.   : 4.0
##  1st Qu.:0.0   1st Qu.:1.000   1st Qu.:12.0
##  Median :1.0   Median :2.000   Median :18.0
##  Mean   :0.7   Mean   :2.577   Mean   :20.9
##  3rd Qu.:1.0   3rd Qu.:4.000   3rd Qu.:24.0
##  Max.   :1.0   Max.   :4.000   Max.   :72.0
##  Payment Status of Previous Credit    Purpose       Credit Amount
##  Min.   :0.000                     Min.   : 0.000   Min.   :  250
##  1st Qu.:2.000                     1st Qu.: 1.000   1st Qu.: 1366
##  Median :2.000                     Median : 2.000   Median : 2320
##  Mean   :2.545                     Mean   : 2.828   Mean   : 3271
##  3rd Qu.:4.000                     3rd Qu.: 3.000   3rd Qu.: 3972
##  Max.   :4.000                     Max.   :10.000   Max.   :18424
##  Value Savings/Stocks Length of current employment Instalment per cent
##  Min.   :1.000        Min.   :1.000                Min.   :1.000
##  1st Qu.:1.000        1st Qu.:3.000                1st Qu.:2.000
##  Median :1.000        Median :3.000                Median :3.000
##  Mean   :2.105        Mean   :3.384                Mean   :2.973
##  3rd Qu.:3.000        3rd Qu.:5.000                3rd Qu.:4.000
##  Max.   :5.000        Max.   :5.000                Max.   :4.000
##  Sex & Marital Status   Guarantors    Duration in Current address
##  Min.   :1.000        Min.   :1.000   Min.   :1.000
##  1st Qu.:2.000        1st Qu.:1.000   1st Qu.:2.000
##  Median :3.000        Median :1.000   Median :3.000
##  Mean   :2.682        Mean   :1.145   Mean   :2.845
##  3rd Qu.:3.000        3rd Qu.:1.000   3rd Qu.:4.000
```

```
##  Max.   :4.000        Max.   :3.000   Max.   :4.000
##  Most valuable available asset  Age (years)    Concurrent Credits
##  Min.   :1.000                 Min.   :19.00   Min.   :1.000
##  1st Qu.:1.000                 1st Qu.:27.00   1st Qu.:3.000
##  Median :2.000                 Median :33.00   Median :3.000
##  Mean   :2.358                 Mean   :35.54   Mean   :2.675
##  3rd Qu.:3.000                 3rd Qu.:42.00   3rd Qu.:3.000
##  Max.   :4.000                 Max.   :75.00   Max.   :3.000
##  Type of apartment No of Credits at this Bank   Occupation
##  Min.   :1.000     Min.   :1.000              Min.   :1.000
##  1st Qu.:2.000     1st Qu.:1.000              1st Qu.:3.000
##  Median :2.000     Median :1.000              Median :3.000
##  Mean   :1.928     Mean   :1.407              Mean   :2.904
##  3rd Qu.:2.000     3rd Qu.:2.000              3rd Qu.:3.000
##  Max.   :3.000     Max.   :4.000              Max.   :4.000
##  No of dependents   Telephone     Foreign Worker
##  Min.   :1.000    Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000    1st Qu.:1.000   1st Qu.:1.000
##  Median :1.000    Median :1.000   Median :1.000
##  Mean   :1.155    Mean   :1.404   Mean   :1.037
##  3rd Qu.:1.000    3rd Qu.:2.000   3rd Qu.:1.000
##  Max.   :2.000    Max.   :2.000   Max.   :2.000
```

The dataset consists of 1000 observations and 21 variables.

We know that our variable for focus (dependent variable) should be first converted to factor and we also check for any missing values.

Data Preprocessing

```
creditData$Creditability <- as.factor(creditData$Creditability)
sum(is.na(creditData))

## [1] 0
```

No NA values.

We now randomize the data and create a training (75%) and the testing set (25%) and evaluate how much percent credibility is within each set.

```
# 75% means 750 for training and the rest for testing
set.seed(12345)
credit_rand <- creditData[order(runif(1000)), ]
credit_train <- credit_rand[1:750, ]
credit_test <- credit_rand[751:1000, ]

prop.table(table(credit_train$Creditability))

##
##         0         1
## 0.3146667 0.6853333
```

```
prop.table(table(credit_test$Creditability))

##
##     0     1
## 0.256 0.744
```

The datasets look well distributed.

Now we build the predict model using naïve bayes technique.

Full Model

```
naive_model <- naive_bayes(Creditability ~ ., data= credit_train)
naive_model

## ================================ Naive Bayes
=================================
## Call:
## naive_bayes.formula(formula = Creditability ~ ., data = credit_train)
##
## A priori probabilities:
##
##         0         1
## 0.3146667 0.6853333
##
## Tables:
##
## Account Balance        0        1
##            mean 1.923729 2.793774
##            sd   1.036826 1.252008
##
##
## Duration of Credit (month)        0        1
##                       mean 24.46610 19.20039
##                       sd   13.82208 11.13433
##
##
## Payment Status of Previous Credit        0        1
##                              mean 2.161017 2.665370
##                              sd   1.071649 1.045219
##
##
## Purpose        0        1
##    mean 2.927966 2.803502
##    sd   2.944722 2.633253
##
##
## Credit Amount        0        1
##          mean 3964.195 2984.177
##          sd   3597.093 2379.685
```

```
##
## # ... and 15 more tables
```

We create a confusion matrix to evaluate how many values were predicted correctly compared to the actual.

Confusion Matrix

```
conf_nat <- table(predict(naive_model, credit_test),
credit_test$Creditability)
conf_nat

##
##       0   1
##   0  42  35
##   1  22 151
```

The false negative percentage is higher than the false positive.

```
Accuracy <- sum(diag(conf_nat))/sum(conf_nat)*100
Accuracy

## [1] 77.2
```

This is an okay accuracy.

Now we move for improving the model by using feature selection approach. We find the highly corrected variable and remove it from the model and the evaluate the performance of the model.

For the same, first we do scaling for the data. Scaling uses the principle of:

(value-avg(var))/(max(var)-min(var))

Optimization

```
creditDataScaled <- scale(credit_rand[,2:ncol(credit_rand)], center=TRUE,
scale = TRUE)
m <- cor(creditDataScaled)
highlycor <- findCorrelation(m, 0.30)
highlycor

## [1]  5 12 19 15  3

#check how the above variables are correlated with the dependent variable
check <- credit_rand%>%select(highlycor,1)
check$Creditability<-as.numeric(check$Creditability)
cor(check)

##                               Purpose Duration in Current address
## Purpose                    1.00000000                -0.038221345
## Duration in Current address -0.03822134                 1.000000000
```

```
## No of dependents           -0.03257687                0.042643426
## Concurrent Credits         -0.10023039                0.022654074
## Duration of Credit (month)  0.14749187                0.034067202
## Creditability              -0.01797887               -0.002967159
##                            No of dependents Concurrent Credits
## Purpose                        -0.032576874        -0.10023039
## Duration in Current address     0.042643426         0.02265407
## No of dependents                1.000000000        -0.07689064
## Concurrent Credits             -0.076890642         1.00000000
## Duration of Credit (month)     -0.023834475        -0.06288379
## Creditability                   0.003014853         0.10984410
##                            Duration of Credit (month) Creditability
## Purpose                                    0.14749187  -0.017978870
## Duration in Current address                0.03406720  -0.002967159
## No of dependents                          -0.02383448   0.003014853
## Concurrent Credits                        -0.06288379   0.109844099
## Duration of Credit (month)                 1.00000000  -0.214926665
## Creditability                             -0.21492667   1.000000000

filteredData <- credit_rand[, -(c(6,13,20,16))]
filteredTraining <- filteredData[1:750, ]
filteredTest <- filteredData[751:1000, ]
```

Optimized Model

```
nb_model <- naive_bayes(Creditability ~ ., data=filteredTraining)
nb_model

## ================================ Naive Bayes
=================================
## Call:
## naive_bayes.formula(formula = Creditability ~ ., data = filteredTraining)
##
## A priori probabilities:
##
##         0         1
## 0.3146667 0.6853333
##
## Tables:
##
## Account Balance        0        1
##            mean 1.923729 2.793774
##            sd   1.036826 1.252008
##
##
## Duration of Credit (month)        0        1
##                       mean 24.46610 19.20039
##                       sd   13.82208 11.13433
##
##
## Payment Status of Previous Credit        0        1
```

```
##                              mean 2.161017 2.665370
##                              sd   1.071649 1.045219
##
##
## Purpose        0        1
##    mean 2.927966 2.803502
##    sd   2.944722 2.633253
##
##
## Value Savings/Stocks        0        1
##                 mean 1.711864 2.334630
##                 sd   1.340700 1.674510
##
## # ... and 11 more tables
```

```r
filteredTestPred <- predict(nb_model, newdata = filteredTest)
table(filteredTestPred, filteredTest$Creditability)
```

```
##
## filteredTestPred   0   1
##                0  43  37
##                1  21 149
```

```r
conf_nat <- table(filteredTestPred, filteredTest$Creditability)
conf_nat
```

```
##
## filteredTestPred   0   1
##                0  43  37
##                1  21 149
```

```r
Accuracy <- sum(diag(conf_nat))/sum(conf_nat)*100
Accuracy
```

```
## [1] 76.8
```

Summary:

We are trying to predict the creditability of the data by using the Naïve Bayes Model. For this assignment, we created a full model with all the variables and the accuracy was only 77.2.

To further better that model, we selected variables based on the correlation between them. Selection was made by taking out highly correlated variables that did not affect the dependent variable. A 76.8 accuracy was achieved with that approach.

PART 2:

Overview:

The second part focuses on the another data for online news popularity. The dataset tells us whether an online news is popular or not. For the same, we try to create an indicator variable based on the mean value of shares and decide whether the news is popular or not. For the same we would first create a a predictive model specifically Naïve Bayesian Classifier which to determine and later optimize and improve it.

News Popularity Dataset

We load the dataset and select the required variables only.

```
newsShort <- read_csv("OnlineNewsPopularity.csv")%>%
  select("n_tokens_title", "n_tokens_content", "n_unique_tokens",
"n_non_stop_words", "num_hrefs", "num_imgs", "num_videos",
"average_token_length", "num_keywords", "kw_max_max",
"global_sentiment_polarity", "avg_positive_polarity", "title_subjectivity",
"title_sentiment_polarity", "abs_title_subjectivity",
"abs_title_sentiment_polarity", "shares")
```

Data Pre-Processing

Create a new variable called popular if the share is higher than 1400, 1400 is the median of the shares.

```
newsShort <- newsShort%>%
  mutate(popular=if_else((shares >= 1400),1,0))%>%
  select(-shares)
newsShort$popular <- as.factor(newsShort$popular)
glimpse(newsShort)

## Observations: 39,644
## Variables: 17
## $ n_tokens_title               <dbl> 12, 9, 9, 9, 13, 10, 8, 12, 11, 1...
## $ n_tokens_content             <dbl> 219, 255, 211, 531, 1072, 370, 96...
## $ n_unique_tokens              <dbl> 0.6635945, 0.6047431, 0.5751295, ...
## $ n_non_stop_words             <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ num_hrefs                    <dbl> 4, 3, 3, 9, 19, 2, 21, 20, 2, 4, ...
## $ num_imgs                     <dbl> 1, 1, 1, 1, 20, 0, 20, 20, 0, 1, ...
## $ num_videos                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ...
## $ average_token_length         <dbl> 4.680365, 4.913725, 4.393365, 4.4...
## $ num_keywords                 <dbl> 5, 4, 6, 7, 7, 9, 10, 9, 7, 5, 8,...
## $ kw_max_max                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ global_sentiment_polarity    <dbl> 0.09256198, 0.14894781, 0.3233333...
## $ avg_positive_polarity        <dbl> 0.3786364, 0.2869146, 0.4958333, ...
## $ title_subjectivity           <dbl> 0.5000000, 0.0000000, 0.0000000, ...
```

```
## $ title_sentiment_polarity     <dbl> -0.1875000, 0.0000000, 0.0000000,...
## $ abs_title_subjectivity       <dbl> 0.00000000, 0.50000000, 0.5000000...
## $ abs_title_sentiment_polarity <dbl> 0.1875000, 0.0000000, 0.0000000, ...
## $ popular                      <fct> 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, ...
```

```
news_rand <- newsShort[order(runif(10000)), ]
set.seed(12345)
```

```
#Split the data into training and test datasets
news_train <- news_rand[1:9000, ]
news_test <- news_rand[9001:10000, ]
```

Full Model

```
nb_model <- naive_bayes(popular ~ ., data=news_train)
nb_model

## ================================ Naive Bayes
=================================
## Call:
## naive_bayes.formula(formula = popular ~ ., data = news_train)
##
## A priori probabilities:
##
##         0         1
## 0.4291111 0.5708889
##
## Tables:
##
## n_tokens_title        0        1
##          mean 9.820559 9.695991
##          sd   1.929249 1.987754
##
##
## n_tokens_content        0        1
##            mean 452.2315 515.1051
##            sd   347.1779 450.0206
##
##
## n_unique_tokens         0         1
##           mean 0.5702437 0.5542023
##           sd   0.1127776 0.1232687
##
##
## n_non_stop_words          0          1
##            mean 0.99404453 0.99124172
##            sd   0.07695147 0.09318398
##
```

```
##
## num_hrefs         0         1
##      mean  9.147851 10.570650
##      sd    8.644083 11.540711
##
## # ... and 11 more tables
```

Create Prediction

```
news_Pred <- predict(nb_model, newdata = news_test)
conf_nat <- table(news_Pred, news_test$popular)
conf_nat

##
## news_Pred   0   1
##         0 329 400
##         1 101 170

Accuracy <- sum(diag(conf_nat))/sum(conf_nat)*100
Accuracy

## [1] 49.9
```

Not great.

Optimization

To optimize the model, we will look how we can remove variables which are correlated with each other and remove the highly correlated ones withoput affecting the model.

```
newsDataScaled <- scale(news_rand[,0:(ncol(news_rand)-1)], center=TRUE, scale
= TRUE)
m <- cor(newsDataScaled)
highlycor <- findCorrelation(m, 0.30)
highlycor

## [1]  3 16  2  4 12 13
```

These are the indices of the variables that are highly correlated with each other. Below, we run a correlation of these variables with the dependent variables.

```
#check how the above variables are correlated with the dependent variable
check <- news_rand%>%select(3,16,2,4,12,13,17)
check$popular<-as.numeric(check$popular)
cor(check)

##                              n_unique_tokens abs_title_sentiment_polarity
## n_unique_tokens                  1.000000000                 -0.004251095
## abs_title_sentiment_polarity    -0.004251095                  1.000000000
```

```
## n_tokens_content                 -0.626662198                  0.011187585
## n_non_stop_words                  0.417037438                 -0.023983298
## avg_positive_polarity             0.154500778                  0.140369050
## title_subjectivity                0.025234677                  0.725455997
## popular                          -0.062992819                  0.026636966
##                                n_tokens_content n_non_stop_words
## n_unique_tokens                    -0.626662198       0.41703744
## abs_title_sentiment_polarity        0.011187585      -0.02398330
## n_tokens_content                    1.000000000       0.10552159
## n_non_stop_words                    0.105521587       1.00000000
## avg_positive_polarity               0.078679331       0.34982582
## title_subjectivity                 -0.009127765      -0.03605004
## popular                             0.067211377      -0.01946947
##                                avg_positive_polarity title_subjectivity
## n_unique_tokens                          0.154500778        0.025234677
## abs_title_sentiment_polarity             0.140369050        0.725455997
## n_tokens_content                         0.078679331       -0.009127765
## n_non_stop_words                         0.349825816       -0.036050041
## avg_positive_polarity                    1.000000000        0.081716910
## title_subjectivity                       0.081716910        1.000000000
## popular                                  0.008526717        0.018061939
##                                     popular
## n_unique_tokens                -0.062992819
## abs_title_sentiment_polarity    0.026636966
## n_tokens_content                0.067211377
## n_non_stop_words               -0.019469471
## avg_positive_polarity           0.008526717
## title_subjectivity              0.018061939
## popular                         1.000000000

findCorrelation(m,0.6)

## [1]  3 16  4
```

Below, we create a filtered dataset by disselecting the varaibles that are highly likely to create high pairwise correlation, applied trial & error basis.

```
filteredData <- news_rand%>%select(-n_unique_tokens,-n_non_stop_words,-
abs_title_sentiment_polarity,-num_keywords)
filteredTraining <- filteredData[1:750, ]
filteredTest <- filteredData[751:1000, ]
```

Optimized Model

Training the Model:

```r
nb_model <- naive_bayes(popular ~ ., data=filteredTraining)
nb_model
```

```
## ================================ Naive Bayes
=================================
## Call:
## naive_bayes.formula(formula = popular ~ ., data = filteredTraining)
##
## A priori probabilities:
##
##         0         1
## 0.4173333 0.5826667
##
## Tables:
##
## n_tokens_title        0        1
##           mean 9.616613 9.704805
##           sd   1.903096 2.013099
##
##
## n_tokens_content        0        1
##             mean 475.8658 524.9130
##             sd   359.3243 447.9918
##
##
## num_hrefs         0         1
##      mean  9.092652 11.226545
##      sd    8.875260 11.738411
##
##
## num_imgs        0        1
##     mean 3.539936 3.951945
##     sd   6.977471 8.279242
##
##
## num_videos        0        1
##       mean 1.300319 1.212815
##       sd   5.538261 4.883789
##
## # ... and 7 more tables
```

Evaluating the Model:

```r
filteredTestPred <- predict(nb_model, newdata = filteredTest)
table(filteredTestPred, filteredTest$popular)
```

```
##
## filteredTestPred  0  1
```

```
##                 0 86 83
##                 1 26 55
```

Creating Confusion Matrix:

```
tab <- table(filteredTestPred, filteredTest$popular)
caret::confusionMatrix(tab)
```

```
## Confusion Matrix and Statistics
##
##
## filteredTestPred  0  1
##                 0 86 83
##                 1 26 55
##
##                Accuracy : 0.564
##                  95% CI : (0.5001, 0.6264)
##     No Information Rate : 0.552
##     P-Value [Acc > NIR] : 0.3761
##
##                   Kappa : 0.1588
##  Mcnemar's Test P-Value : 8.148e-08
##
##             Sensitivity : 0.7679
##             Specificity : 0.3986
##          Pos Pred Value : 0.5089
##          Neg Pred Value : 0.6790
##              Prevalence : 0.4480
##          Detection Rate : 0.3440
##    Detection Prevalence : 0.6760
##       Balanced Accuracy : 0.5832
##
##        'Positive' Class : 0
##
```

Calculating Model Accuracy:

```
conf_nat <- table(filteredTestPred, filteredTest$popular)
conf_nat
```

```
##
## filteredTestPred  0  1
##                 0 86 83
##                 1 26 55
```

```
Accuracy <- sum(diag(conf_nat))/sum(conf_nat)*100
Accuracy
```

```
## [1] 56.4
```

Accuracy is better.

Results & Discussion

We are trying to predict the popularity of the news by using the Naïve Bayes Model. For this assignment, we created a full model with all the variables and the accuracy was only 49.9.

To further better that model, we selected variables based on the correlation between them. Selection was made by taking out highly correlated variables that did not affect the dependent variable. A 56.4 accuracy was achieved with that approach.