Data622_Hwk3

Alexis Mekueko

2022-04-08

Github Link Web Link

Assignment:

Perform an analysis of the dataset used in Homework #2 using the SVM algorithm.Compare the results with the results from previous homework. Based on articles https://www.hindawi.com/journals/complexity/2021/5550344/ https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8137961/ Search for academic content (at least 3 articles) that compare the use of decision trees vs SVMs in your current area of expertise. Which algorithm is recommended to get more accurate results? Is it better for classification or regression scenarios? Do you agree with the recommendations? Why?

We will skip the Exploratory Data Analysis (EDA) that was done in the precedent assignment. we will bring the clean dataset and build the model on Support Vector Machine (SVM) Algorithm.

We imported the data from local drive. Another option could be to load the date from Github.

```
## 'data.frame':    614 obs. of  5 variables:
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : int  128 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
##  $ Loan_Status      : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 1 2 1 ...
```

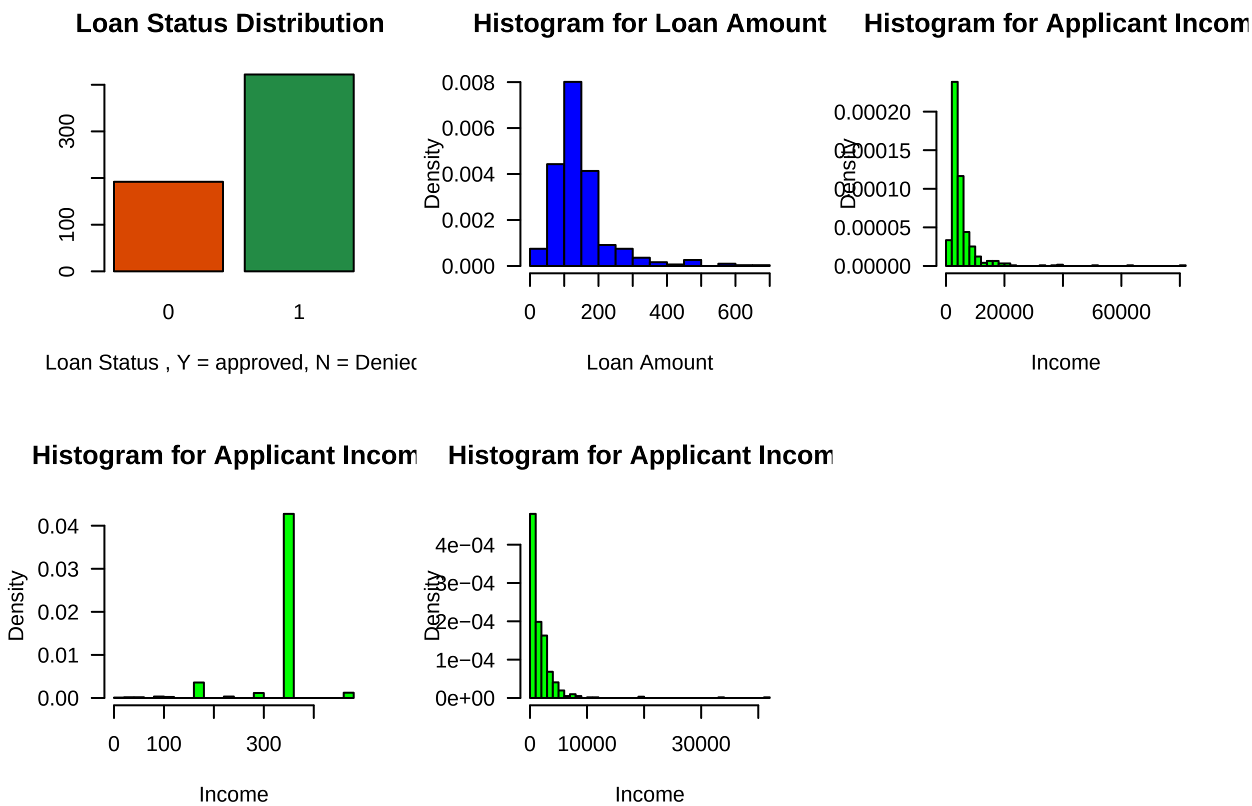
```
##   ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Loan_Status
## 1            5849                 0        128              360           1
## 2            4583              1508        128              360           0
## 3            3000                 0         66              360           1
## 4            2583              2358        120              360           1
## 5            6000                 0        141              360           1
## 6            5417              4196        267              360           1
```

```
##
##   0   1
## 192 422
```

```
##  ApplicantIncome CoapplicantIncome   LoanAmount    Loan_Amount_Term Loan_Status
##  Min.   :  150   Min.   :    0     Min.   :  9.0   Min.   : 12.0    0:192
##  1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0   1st Qu.:360.0    1:422
##  Median : 3812   Median : 1188     Median :128.0   Median :360.0
##  Mean   : 5403   Mean   : 1621     Mean   :146.8   Mean   :342.3
##  3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168.0   3rd Qu.:360.0
##  Max.   :81000   Max.   :41667     Max.   :700.0   Max.   :480.0
```

Let's see Loan approval, applicant income and loan amount distributions

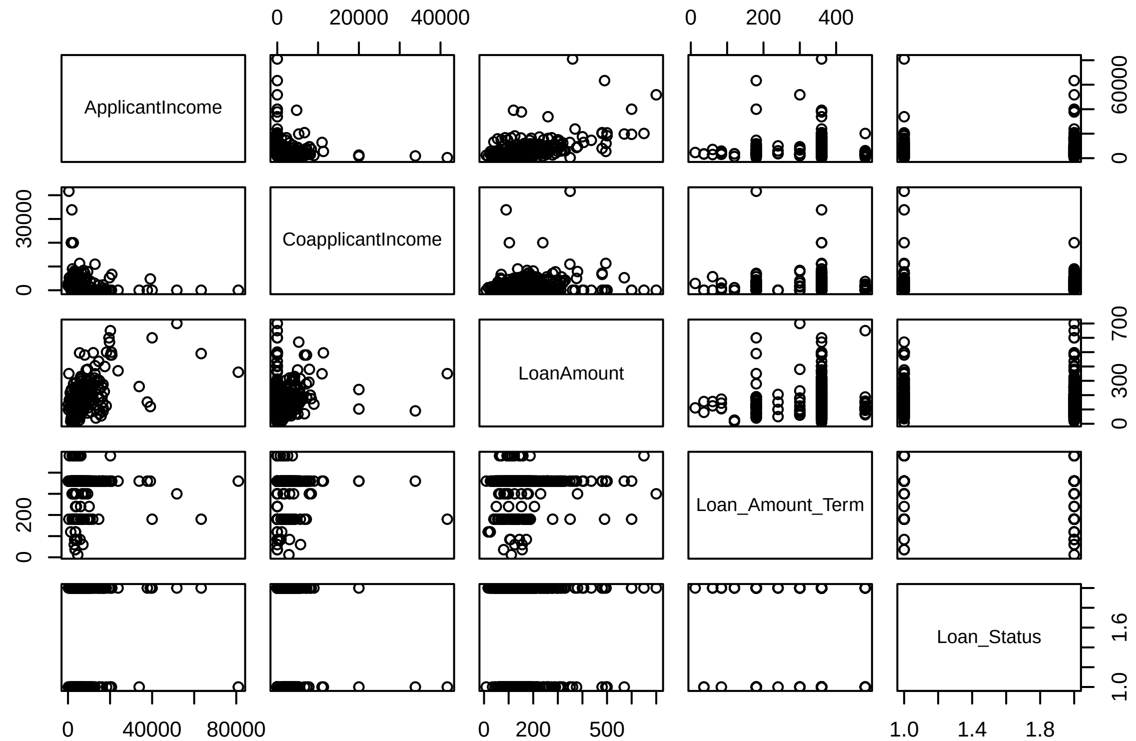
```
## Let's visualize the Loan Status Distribution

## [1] "\n"

## Let's visualize the loan amount distribution

## [1] "\n"

## Let's visualize the applicant income distribution
```



The features are mostly right skewed distribution. The income-variable shows an abnormal distribution. Despite the median being low, loan status shows more applicants get approved. If this approval was based only on credit score, we could say the bank probably considering low score or the bank just lower the requirements to meet to get more people.

Visualizing Linearity Amount Features

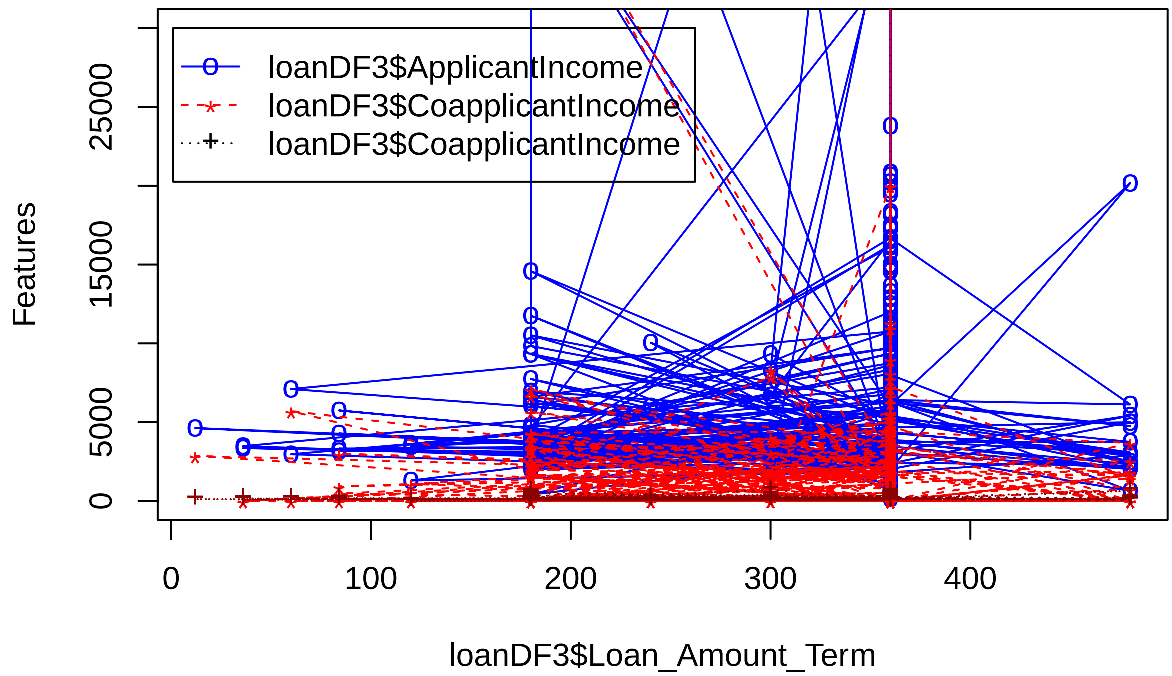
```
plot(loanDF3)
lines(lowess(loanDF3))
```



We barely see linearity amount these variables.

Another way to visualize these variables in a x-y plane.

```
# plot the first curve by calling plot() function
# First curve is plotted

plot(loanDF3$Loan_Amount_Term, loanDF3$ApplicantIncome, type="o", col="blue", pch="o", lty=1, ylim=c(0,
points(loanDF3$Loan_Amount_Term, loanDF3$CoapplicantIncome, col="red", pch="*")
lines(loanDF3$Loan_Amount_Term, loanDF3$CoapplicantIncome, col="red",lty=2)
points(loanDF3$Loan_Amount_Term, loanDF3$LoanAmount, col="dark red",pch="+")
lines(loanDF3$Loan_Amount_Term, loanDF3$LoanAmount, col="dark red", lty=3)
# Adding a legend inside box at the location (2,40) in graph coordinates.
legend(1,30000,legend=c("loanDF3$ApplicantIncome" ,"loanDF3$CoapplicantIncome","loanDF3$CoapplicantInco
```



Building Model Support Vector Machines(SVMs)

```
library(caTools)
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

##
## Attaching package: 'modeltools'

## The following object is masked from 'package:arules':
##
##     info

## The following object is masked from 'package:plyr':
##
##     empty
```

```
## The following object is masked from 'package:BayesFactor':
##
##     posterior

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following object is masked from 'package:tsibble':
##
##     index

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
library(e1071)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:modeltools':
##
##     prior

## The following object is masked from 'package:arules':
##
##     size

## The following object is masked from 'package:ggplot2':
##
##     alpha

## The following object is masked from 'package:coda':
##
##     nvar
```

```
library(ROCR)

set.seed(21532)


#loanDF3 <- loanDF1 %>%
#                    dplyr::select(ApplicantIncome, CoapplicantIncome,LoanAmount,Loan_Amount_Term, Loan_
```

```r
#loanDF3 <- loanDF3 %>%
#                   dplyr::select(ApplicantIncome, LoanAmount, Loan_Status)

#View(loanDF2)
#loanDF2$Loan_Status <- ifelse(loanDF2$Loan_Status == "Y", 1 , 0)
#glimpse(loanDF3)

#loanDF2$Married <- ifelse(loanDF2$Married == "Yes", 1 , 0)
#loanDF2$Loan_Status <- ifelse(loanDF2$Loan_Status == "Y", 1 , 0)

#View(loanDF3)

data1 <- createDataPartition(y =loanDF3$Loan_Status, p= 0.7, list = FALSE)

train1 <- loanDF3[data1,]
test1 <- loanDF3[-data1,]

#train1 <- as.data.frame(train1)
#test1 <- as.data.frame(test1)

#is.data.frame(data1)
#is.data.frame(train1)
#is.data.frame(loanDF3)



dim(train1)
```

```
## [1] 431   5
```

```r
dim(test1)
```

```
## [1] 183   5
```

```r
anyNA(loanDF3)
```

```
## [1] FALSE
```

```r
# Cross validation
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3)

#ctrl <- trainControl(method="cv",
#                      number = 2,
#                      summaryFunction=twoClassSummary,
#                      classProbs=TRUE)

# Grid search to fine tune SVM
grid <- expand.grid(C = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.1, 1.25,
                         1.5, 1.75, 2, 2.25)
```

```
                    )

svm_linear <- train(Loan_Status ~.,
                    data = train1,
                    method = "svmLinear",
                    trControl = ctrl,
                    preProcess = c("center", "scale"),
                    #metric = "ROC",
                    tuneGrid = grid,
                    tuneLength = 10
                    )

svm_linear
```

```
## Support Vector Machines with Linear Kernel
##
## 431 samples
##   4 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (4), scaled (4)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 388, 387, 388, 387, 389, 388, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.01  0.6868267  0
##   0.05  0.6868267  0
##   0.10  0.6868267  0
##   0.25  0.6868267  0
##   0.50  0.6868267  0
##   0.75  0.6868267  0
##   1.00  0.6868267  0
##   1.10  0.6868267  0
##   1.25  0.6868267  0
##   1.50  0.6868267  0
##   1.75  0.6868267  0
##   2.00  0.6868267  0
##   2.25  0.6868267  0
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

```
#data1 = sample.split(loanDF3, SplitRatio = 0.80)
#train1 <- subset(loanDF3, data1 == TRUE)
#test1 <- subset(loanDF3, data1 == FALSE)

# Feature Scaling
#train1[-3] = scale(train1[-3])
#test1[-3] = scale(test1[-3])

# Fitting SVM to the Training set
#install.packages('e1071')
```

```r
#
# classifier = svm(formula = Loan_Status ~ .,
#                  data = train1,
#                  type = 'C-classification',
#                  kernel = 'linear')
#
# classifier
```

Prediction of model SVM

```r
# Predicting the Test set results
#pred1svm = predict(classifier, newdata = test1[-3])
test_predi <- predict(svm_linear, newdata = test1)
test_predi
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## Levels: 0 1
```

Making Confusion Matrix , Accuracy of Model 1

```r
# Making the Confusion Matrix
#cm = table(test1[, 3], pred1svm)
confusionMatrix(table(test_predi, test1$Loan_Status))
```

```
## Confusion Matrix and Statistics
##
##
## test_predi   0   1
##          0   0   0
##          1  57 126
##
##                Accuracy : 0.6885
##                  95% CI : (0.616, 0.7548)
##     No Information Rate : 0.6885
##     P-Value [Acc > NIR] : 0.5358
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.195e-13
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.6885
##              Prevalence : 0.3115
```

```
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```
# Another way of looking at model performance
#Let see misclassification error
predicted_table <- table(Predicted = test_predi, Actual = test1$Loan_Status)
predicted_table
```

```
##          Actual
## Predicted   0   1
##         0   0   0
##         1  57 126
```

```
#misclassification error rate
1 - sum(diag(predicted_table))/sum(predicted_table)
```

```
## [1] 0.3114754
```

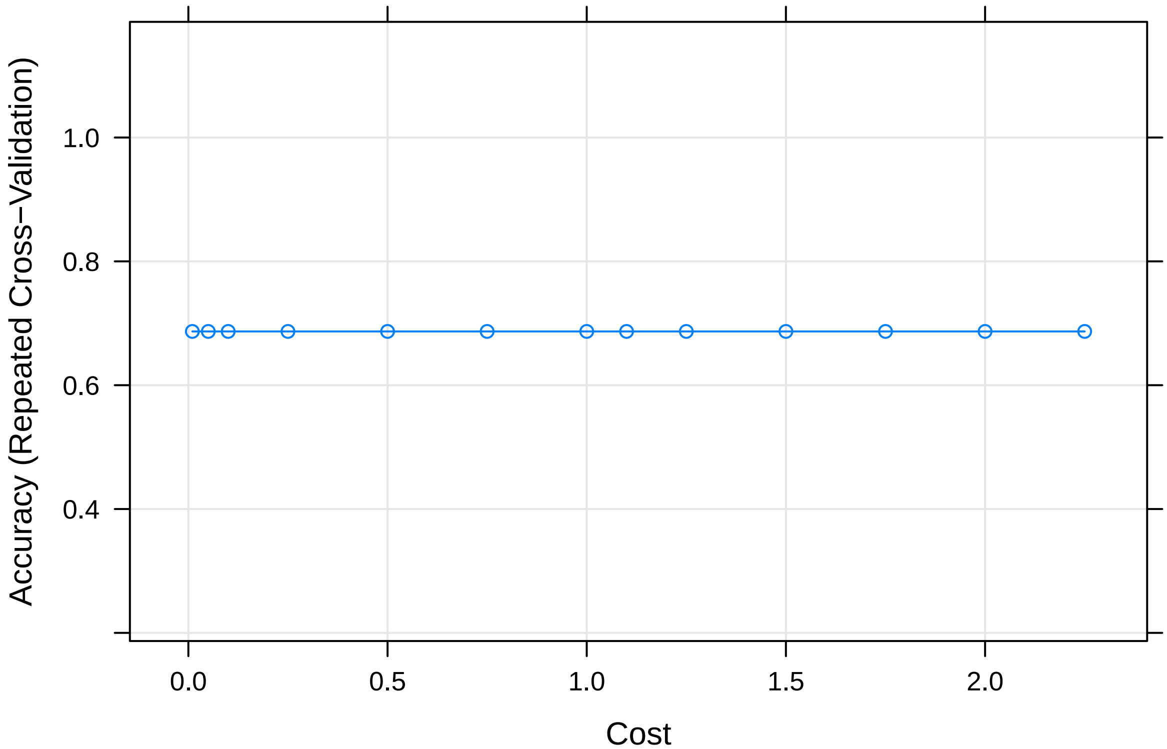
```
#Accuracy
sum(diag(predicted_table))/sum(predicted_table)
```

```
## [1] 0.6885246
```

```
#str(loanDF2)

# # load package
# #install.packages("ggstatsplot")
# library(ggstatsplot)
#
# # correlogram
# ggstatsplot::ggcorrmat(
#   data = data1000R1,
#   type = "parametric", # parametric for Pearson, nonparametric for Spearman's correlation
#   colors = c("darkred", "white", "steelblue") # change default colors
# )
```

Visualizing the Training set results

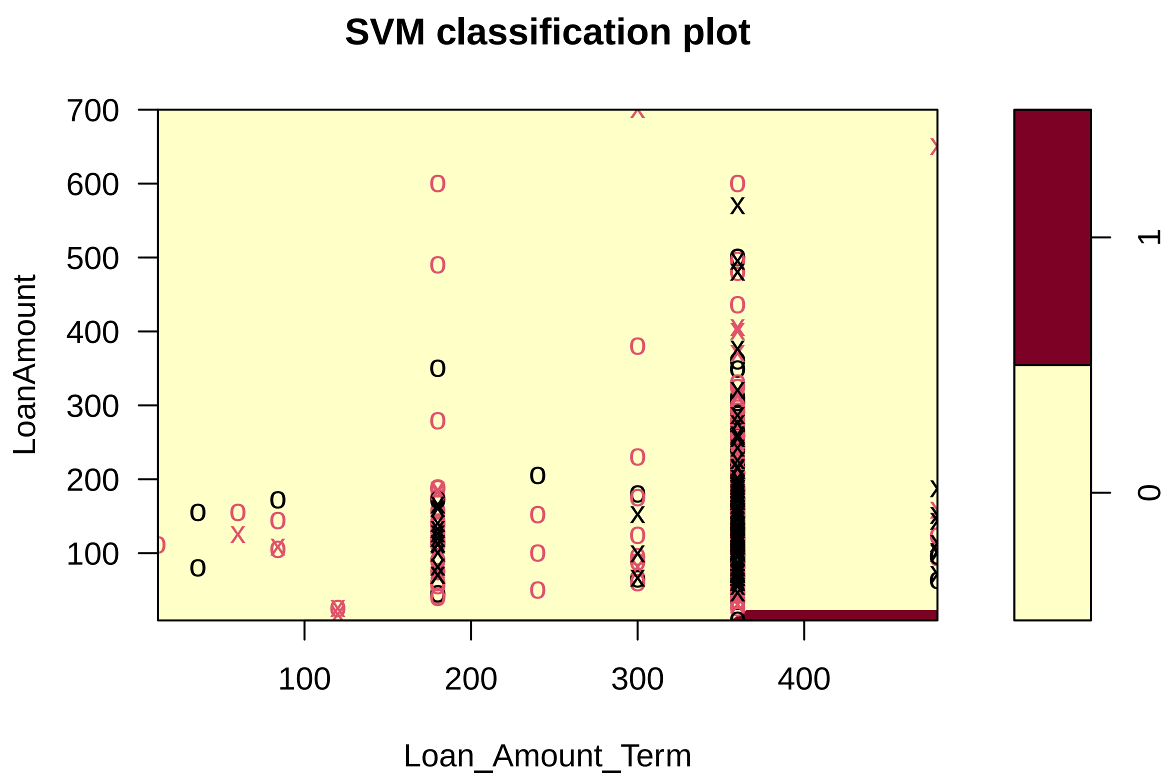
```
library (e1071)

plot(svm_linear)
```




```
#plot 2 dimensions with 5 variable
svm_model <- svm(Loan_Status ~ ., data = loanDF3, kernel = "linear", cost = 10, scale = FALSE)
summary(svm_model)
```

```
##
## Call:
## svm(formula = Loan_Status ~ ., data = loanDF3, kernel = "linear",
##     cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  255
##
##  ( 132 123 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

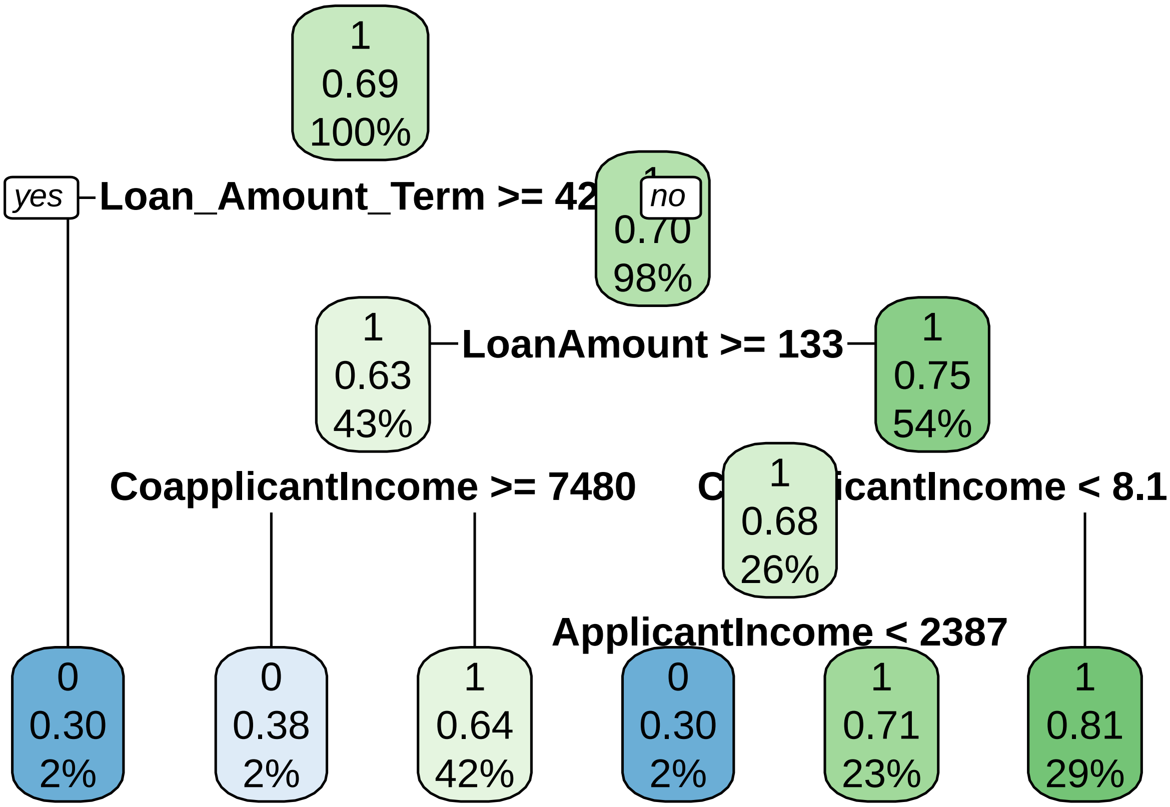
```r
plot(svm_model, formula = LoanAmount ~ Loan_Amount_Term, data=loanDF3)
```




```r
# plot(svm_linear, loanDF3, Loan_Status ~ LoanAmount,
#      slice = list(ApplicantIncome = 3, CoapplicantIncome= 4, Loan_Amount_Term = 6))
```

Decision Tree Model

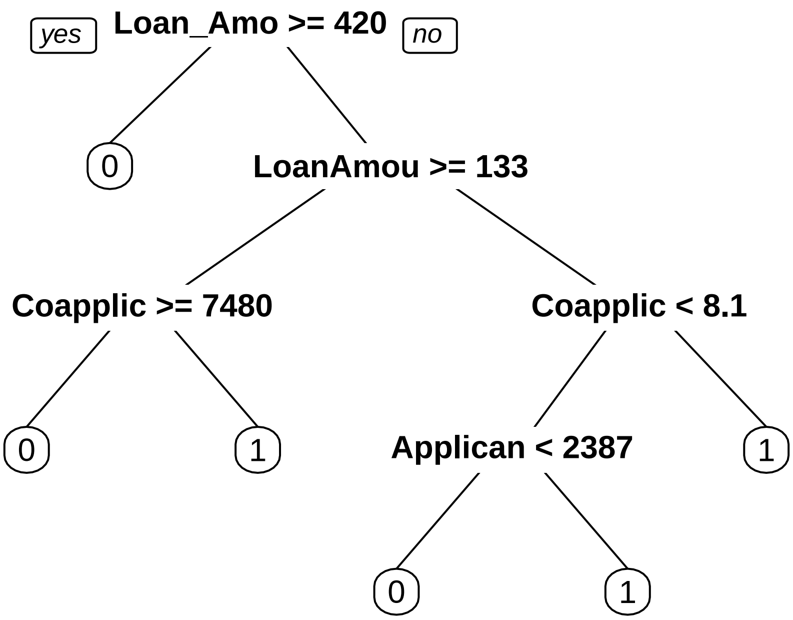
```r
library(rpart)
library(rpart.plot)
library(caret)

model2 <- rpart(Loan_Status ~.,method="class", data=train1)

rpart.plot(model2, tweak =1.6)
```




```r
cat("Visualizing the model")
```

```
## Visualizing the model
```

```r
prp(model2)
```




```
model2.pred <- predict(model2, test1, type="class")
model2.accuracy <- table(test1$Loan_Status, model2.pred, dnn=c("Actual", "Predicted"))
model2.accuracy
```

```
##        Predicted
## Actual   0   1
##       0   5  52
##       1   4 122
```

```
confusionMatrix(predict(model2, type = "class"), train1$Loan_Status)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  19   9
##          1 116 287
##
##                Accuracy : 0.71
##                  95% CI : (0.6646, 0.7524)
##     No Information Rate : 0.6868
##     P-Value [Acc > NIR] : 0.1619
##
##                   Kappa : 0.1407
```

```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.14074
##             Specificity : 0.96959
##          Pos Pred Value : 0.67857
##          Neg Pred Value : 0.71216
##              Prevalence : 0.31323
##          Detection Rate : 0.04408
##    Detection Prevalence : 0.06497
##       Balanced Accuracy : 0.55517
##
##        'Positive' Class : 0
##
```

Notes

We selected numerical variables to build the two models (SVM and Decision Tree). We also transformed the target variables from character to factor('1', '0'). We encountered numerous issues trying to visualize the hyper-plane to see the boundary with SVM. Based on the referenced articles 2 and 3, SVM seems to be good at model accuracy. On this assignment, the decision tree appears to output the SVM. We are actually surprised by the misclassification error rate (31.15%) generated by SVM model. We know SVM uses the kernel to trick data with non-linearity to perform the learning algorithm on the model. Our dataset appears to be non-linear. Looking at the results for both models (decision tree and SVM), we can say decision tree performs better for classification model with 02 classes. We wonder how the two models will perform if we have more than 02 classes. In addition, we discover a data mining application called 'orange' for machine learning models. We will try and see if we get a different outcome.

References

1- https://medium.com/@jackmaughan_50251/machine-learning-with-orange-8bc1a541a1d7

2- https://www.hindawi.com/journals/complexity/2021/5550344/

3- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8137961/

4- https://www.youtube.com/watch?v=RKZoJVMr6CU

5- https://hastie.su.domains/ISLR2/ISLRv2_website.pdf