DS 805: Statistical Learning
Final Project: Classification of Home Loan Applications
- Shawn Bedard

Introduction

This analysis involves a partial data set from the Dream Financial Housing Company home loan application process. The company is involved in various types of home loans and has a presence across all urban, semi-Urban, and rural areas. The data intake process involves an applicant filling out their financial information through an online application. Some of the variables that are provided include the applicant's marital status, education status, gender, income level, credit history, and many others. The data was published by Rushkikesh Konapure and was designed as a data analytics project from Kaggle. The data set originally consisted of 614 unique observations and 13 variables with "Loan_Status" being our dependent or predicted variable. In the data set, there are 420 approved loans and 191 non-approved loans. The task for this analysis is to develop a classification model that efficiently and accurately classifies applicants based off whether their loan application would be approved or not approved. This model could serve as an automated screening process enabling the Dream Housing Financial company to save on employee labor time and save money by streamlining their overall loan process.

Data Preparation

In order for the analysis of the home loan approval data set to be accurate, the data needed to be cleaned and modified in order to get rid of any inconsistencies and to simplify the variables' values. The first step was to check and determine whether the data had any N/A values within any of the observations. In order to test this, the function "complete.cases(df)" was used to visually see if the total number of instances has changed. After checking the data frames, it was clear that there were many N/A values within our data that needed to be removed.

df	611 obs. of 13 variables
df_complete	526 obs. of 12 variables

```{r}
#1
#Checking for missing values
df_complete = df[complete.cases(df),]

#Getting Rid of N/A values
df = na.omit(df)
```

After removing the N/A values from the data, the next step in the cleaning process was to change all of the predictive variables to factors, as well as changing some of the variables to binary

values. In this process the index column was also removed in order to simplify the data set and remove any possible problems throughout the analysis.

```
df$Loan_Status <- ifelse(df$Loan_Status=="Y",1,0)
df$Married <- as.factor(ifelse(df$Married=="Yes",1,0))
df$Education <- as.factor(ifelse(df$Education=="Graduate",1,0))
df$Self_Employed <- as.factor(ifelse(df$Self_Employed=="Yes",1,0))
df$Gender <- as.factor(ifelse(df$Gender=="Male",1,0))
df$Dependents <- as.factor(df$Dependents)
df$Property_Area <- as.factor(df$Property_Area)
df$Credit_History <- as.factor(df$Credit_History)
```

Variables

In this classification analysis, the data incorporated twelve variables leaving eleven variables as predictors for the models. Within the eleven variables, three are continuous and eight are categorical. The list of all variables and their descriptions are the following:

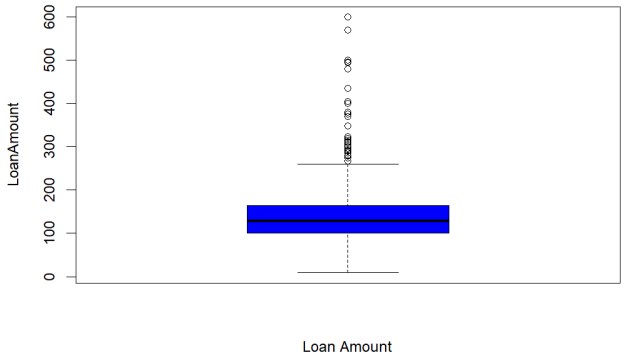
Variable	Values	Description
Loan ID	LP001392, LP002301	Unique Loan ID
Gender	Male, Female	Gender of applicant
Married	Yes / No	Marital status of applicant
Dependents	0,1,2,3+	Number of dependents of applicant
Education	Graduate, Not Graduate	If applicant graduated college or not
Self-Employed	Yes / No	If applicant if self-employed or not
Applicant Income	0-50,000	Monthly income of applicant
Coapplicant Income	0-10,000	Monthly income of coapplicant
Loan Amount	10-400	Loan Amount in thousands
Loan Amount Term	480,360,180,60	Length of loan term in months
Credit History	Yes/No	If applicant has good credit history or not
Property Area	Urban, Semi-Urban, Rural	Location of property
Loan Status	Y / N	Status if applicant was Approved or Not Approved

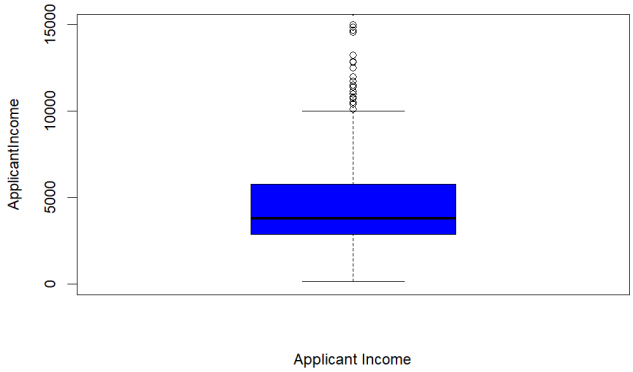
```
'data.frame':   526 obs. of  13 variables:
 $ Loan_ID          : chr  "LP001536" "LP001640" "LP002422" "LP001637" ...
 $ Gender           : Factor w/ 1 level "0": 1 1 1 1 1 1 1 1 1 1 ...
 $ Married          : Factor w/ 1 level "0": 1 1 1 1 1 1 1 1 1 1 ...
 $ Dependents       : Factor w/ 5 levels "","0","1","2",..: 5 2 3 3 5 2 2 3 3 2 ...
 $ Education        : Factor w/ 1 level "0": 1 1 1 1 1 1 1 1 1 1 ...
 $ Self_Employed    : Factor w/ 1 level "0": 1 1 1 1 1 1 1 1 1 1 ...
 $ ApplicantIncome  : int  39999 39147 37719 33846 23803 20233 19730 19484 18333 18165 ...
 $ CoapplicantIncome: num  0 4750 0 0 0 ...
 $ LoanAmount       : int  600 120 152 260 370 480 570 600 500 125 ...
 $ Loan_Amount_Term : Factor w/ 10 levels "12","36","60",..: 6 9 9 9 9 9 9 9 9 9 ...
 $ Credit_History   : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 2 ...
 $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 2 2 2 2 1 1 1 2 3 3 ...
 $ Loan_Status      : num  0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "na.action")= 'omit' Named int [1:85] 6 7 9 30 40 49 87 90 91 94 ...
  ..- attr(*, "names")= chr [1:85] "6" "7" "9" "30" ...
```

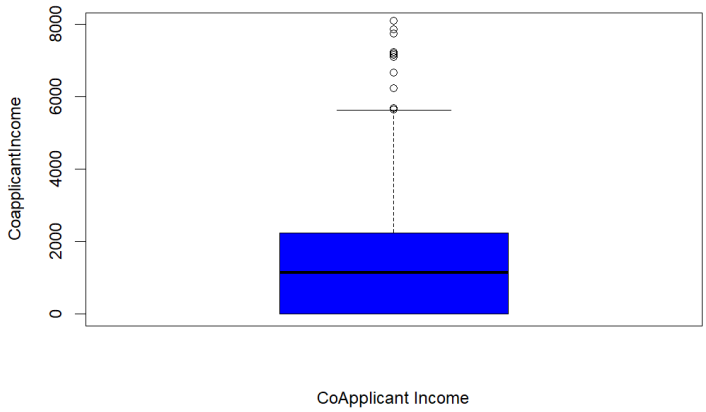
Exploratory Analysis

Starting off the analysis, the cleaned data was then plotted to better understand the distribution of the continuous variables as well as the count of each factor in the categorical variables. The continuous variables, ApplicantIncome, Coapplicant Income, and Loan_Amount were plotted using a box plot to visualize their distributions and the results are below.

Box plots:



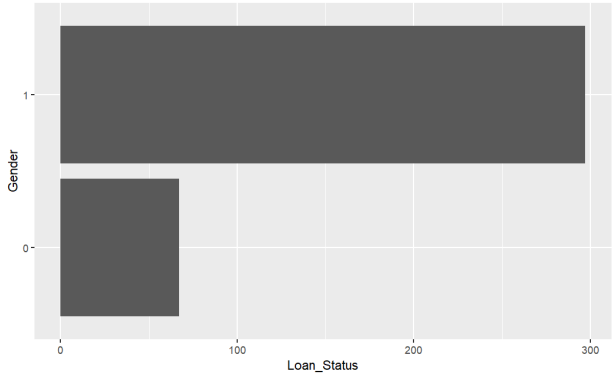


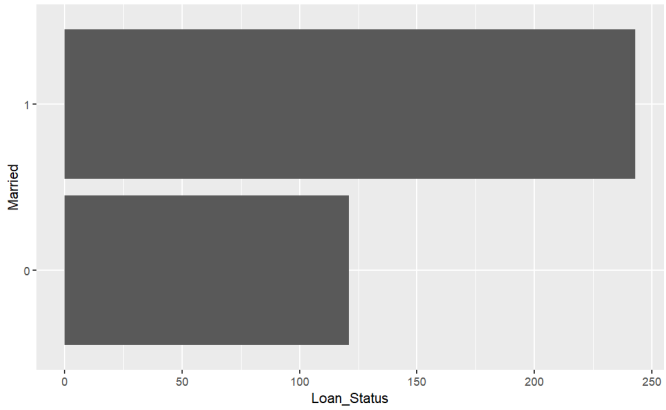


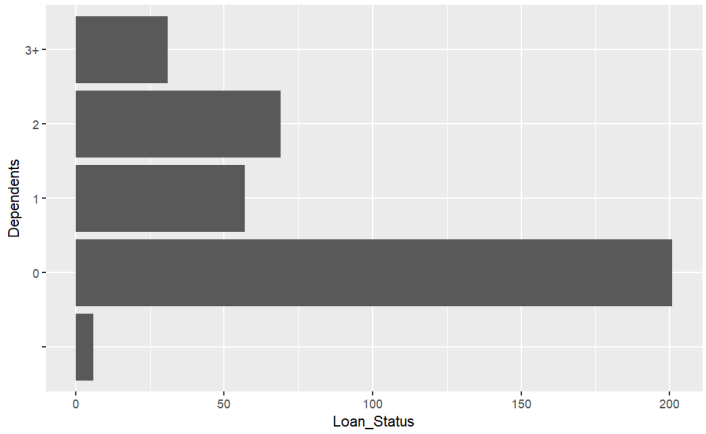
Based on these box plots, we can determine that the average loan amount is roughly $150,000 with its upper quartile reaching around the $250,000 mark. In this visualization, there are clear outliers in the data that are applying for much bigger loans than the average applicant. Continuing to the second box plot of the applicant income variable, we can see that the average monthly income falls just below the $5,000 mark and its upper quartile reaches around the $10,000 range with again more outliers that are making much more per month than the average applicant. Finally, the third box plot visualizes the co-applicant's income and based off the chart we can see the mean around the $1500 range and its upper quartile reaches just under the $6,000 mark.

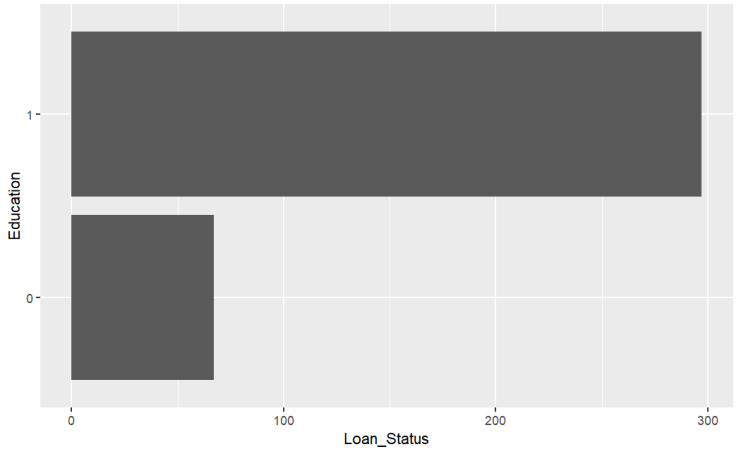
The same process was then applied to the categorical variables but plotted on a bar chart to visualize the count of each option within the variables. The categorical variables in this data set consist of, Gender, Married, Dependents, Education, Self Employed, Credit History, Property Area, and Loan Amount Term. The visualization of these variables follows below.

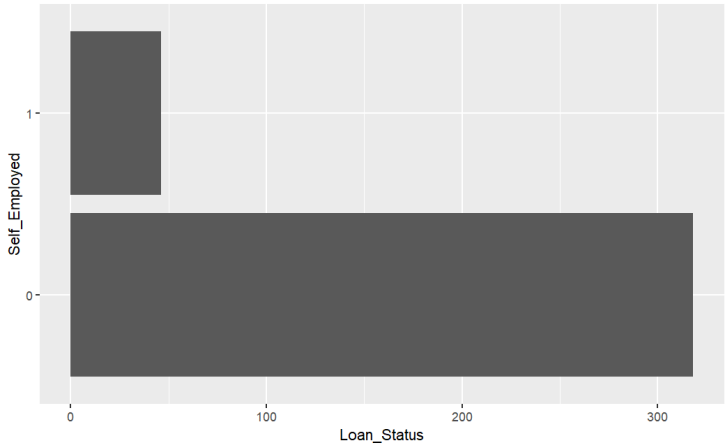
Bar Charts:

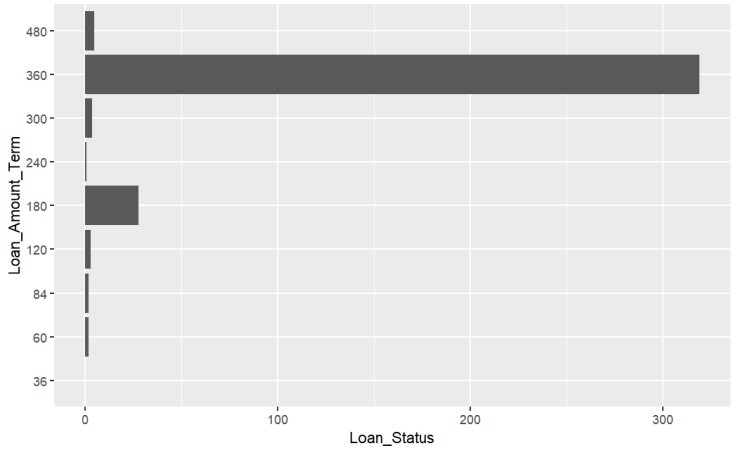


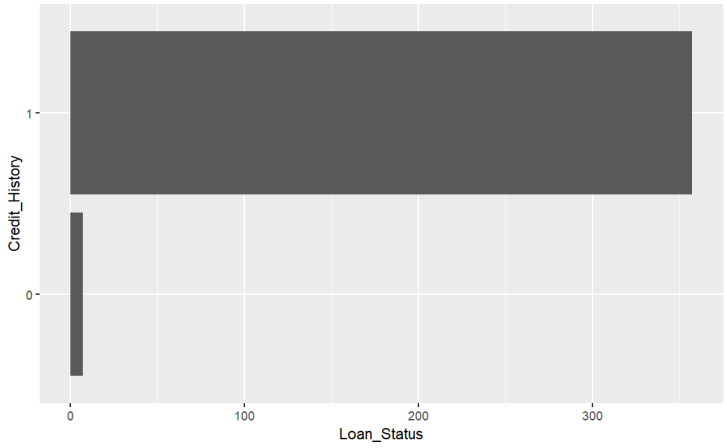

Married
1
0
0
50
100
150
200
250
Loan_Status

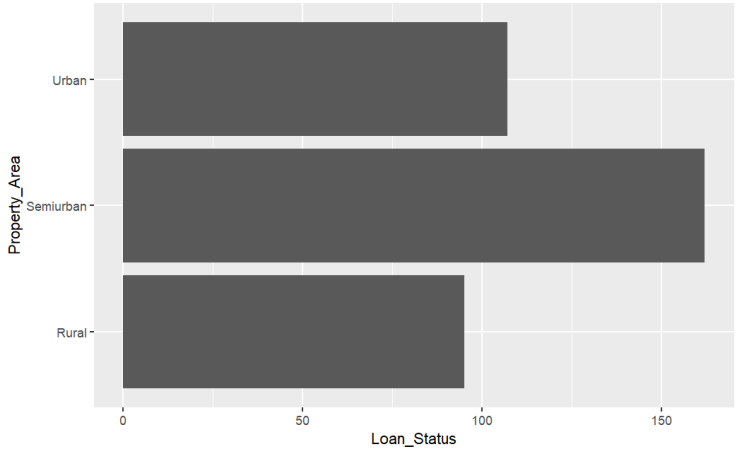

Dependents
3+
2
1
0
-
0
50
100
150
200
Loan_Status


Education
1
0
0
100
200
300
Loan_Status


Self_Employed
1
0
0
100
200
300
Loan_Status


Loan_Amount_Term
480
360
300
240
180
120
84
60
36
0
100
200
300
Loan_Status


Credit_History
1
0
0
100
200
300
Loan_Status



Based off the visualization of these categorical variables, we can visually determine what makes up each of these variables. For the first variable, gender, there is a clear division between both male and female applicants since a large majority of the applications were submitted by a male. The number of male applicants reaches just under 300 observations while the number of female applicants is below 100 observations. Moving on to the second bar chart, we see a similar pattern seeing the number of applicants who are married heavily outweigh the number of applicants who are not by over 100 observations. The dependents bar chart shows if an applicant has any dependents and the number of dependents they are claiming. From the chart, we can determine that large majority of the applicants have no dependents followed by an applicant having two dependents, then only one, and finally the least likely would be having three or more dependents.

The next visualization is of the number of applicants who have graduated at the high school level. From the bar chart we can clearly see that a majority of the applicants have graduated, making up close to 300 of the observations. When looking at the self-employed bar chart, we can see that most applicants are not self employed but there are still roughly 50 observations with the applicant being self-employed. For the loan term length, we see that a that almost all the loans are for 360 months but we can still see that there are loans of all the different lengths of time. Looking at the credit history bar chart, we can see that there are only a few of the applicants that have declared bankruptcy in the past. Finally in the property area chart, we can see that the location with the most amount of loans is in semi-urban locations but both urban and rural locations still have a large portion of observations.

Classification Models

Logistic Model:

To start off the classification models, a logistic model was created to determine its accuracy and ability in classifying an applicant into whether they should be approved for their loan or not. The model was then converted into binary values based on whether the logistic probability is higher or lower than 50%. The logistic model was created using all the variables in the training set and summary of the model came out to be:

```r
#Logistic Model
logfit2<-glm(Loan_Status~., data=train)
summary(logfit2)

logprob2<-predict(logfit2, newdata=test, type="response")

#Any prediction higher than 50% accuracy will be presented as 1
logpred2=rep(0, nrow(test))
logpred2[logprob2>=.5]=1
logpred2=as.factor(logpred2)
```

```
Call:
glm(formula = Loan_Status ~ ., data = train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.96552  -0.04295   0.11369   0.22855   0.90384

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          -8.720e-01  3.003e-01  -2.903 0.003898 **
Gender1              -3.587e-03  5.000e-02  -0.072 0.942846
Married1              6.131e-02  4.549e-02   1.348 0.178475
Dependents0          -2.343e-02  1.224e-01  -0.191 0.848332
Dependents1          -4.077e-03  1.291e-01  -0.032 0.974822
Dependents2           4.871e-02  1.279e-01   0.381 0.703540
Dependents3+          3.981e-02  1.363e-01   0.292 0.770433
Education1            7.290e-02  4.627e-02   1.576 0.115932
Self_Employed1       -2.359e-02  5.627e-02  -0.419 0.675213
ApplicantIncome      -6.335e-06  5.784e-06  -1.095 0.274089
CoapplicantIncome    -6.452e-06  7.666e-06  -0.842 0.400526
LoanAmount           -1.882e-04  3.032e-04  -0.621 0.535160
Loan_Amount_Term60    1.108e+00  3.802e-01   2.914 0.003774 **
Loan_Amount_Term84    6.485e-01  3.444e-01   1.883 0.060412 .
Loan_Amount_Term120   9.760e-01  3.774e-01   2.586 0.010073 *
Loan_Amount_Term180   9.051e-01  2.773e-01   3.265 0.001191 **
Loan_Amount_Term240   4.486e-01  3.774e-01   1.189 0.235277
Loan_Amount_Term300   7.001e-01  2.951e-01   2.372 0.018156 *
Loan_Amount_Term360   8.486e-01  2.683e-01   3.163 0.001679 **
Loan_Amount_Term480   5.720e-01  2.856e-01   2.003 0.045859 *
Credit_History1       7.276e-01  5.139e-02  14.158  < 2e-16 ***
Property_AreaSemiurban 1.673e-01  4.509e-02   3.709 0.000237 ***
Property_AreaUrban    5.334e-02  4.893e-02   1.090 0.276294
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.138538)

    Null deviance: 91.349  on 420  degrees of freedom
Residual deviance: 55.138  on 398  degrees of freedom
AIC: 386.94
```

From the summary of the model, the significant variables for this model appear to be the loan term length at 60 months, 120 months, 180 months, 300 months, 360 months, and 480 months. Credit history with the value of “1” is very significant and so is the semi-urban property area variable. The model also has an AIC of 386.94.

To test the accuracy of the model, the model was tasked with classifying the testing data set to determine how well it can perform the task. The testing data set has 105 observations and the model’s confusion matrix is provided below.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 10  3
         1 18 74

               Accuracy : 0.8
                 95% CI : (0.7107, 0.8717)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.07271

                  Kappa : 0.3836

 Mcnemar's Test P-Value : 0.00225

            Sensitivity : 0.35714
            Specificity : 0.96104
         Pos Pred Value : 0.76923
         Neg Pred Value : 0.80435
             Prevalence : 0.26667
         Detection Rate : 0.09524
   Detection Prevalence : 0.12381
      Balanced Accuracy : 0.65909

       'Positive' Class : 0
```

Looking at the models confusion matrix we can visually see that the model successfully predicted 84 of the testing observations while incorrectly classifying 21 of the new data points. Overall the model had an accuracy of 80% with a 95% confidence interval between 71.07% and 87.17%. The model has a decent accuracy level but does suffer from a larger confidence interval which would not make this model the most reliable. Even with an 80% accuracy level, this model often would approve candidates that should be denied by the model. This is also a reason why this model is not the most effective and another model would fit the given scenario much better.

Logistic Model with Lasso:

In order to further improve the logistic model, a lasso cross validation technique was added in order to manipulate the coefficients in order to optimize the model. The lasso method is also a method of subset selection since it can lower variables to 0 making it have no impact on the model. The coefficients of the lasso cross validation logistic regression model can be seen below.

```
24 x 1 sparse Matrix of class "dgCMatrix"
                                      s0
(Intercept)                0.075059074
Gender0                    .
Gender1                    .
Married1                   0.032816592
Dependents0                .
Dependents1                .
Dependents2                0.001112693
Dependents3+               .
Education1                 0.005010008
Self_Employed1             .
ApplicantIncome            .
CoapplicantIncome          .
LoanAmount                 .
Loan_Amount_Term60         .
Loan_Amount_Term84         .
Loan_Amount_Term120        .
Loan_Amount_Term180        .
Loan_Amount_Term240        .
Loan_Amount_Term300        .
Loan_Amount_Term360        .
Loan_Amount_Term480       -0.106125086
Credit_History1            0.658463633
Property_AreaSemiurban     0.070485867
Property_AreaUrban         .
```

After creating the new model with the variables selected through the lasso cross validation method, I was then able to test the model against the testing data set. The model was tested using a confusion matrix as well as calculating the model's testing error. The model results are shown below.

```
#Lasso

#confusion matrix
confusionMatrix(data=logpred_lasso, reference=as.factor(test$Loan_Status))

#Test Error and Accuracy
round( mean(logpred_lasso!=test[,"Loan_Status"]),4)
1-round( mean(logpred_lasso!=test[,"Loan_Status"]),4)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 10  3
         1 18 74

               Accuracy : 0.8
                 95% CI : (0.7107, 0.8717)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.07271

                  Kappa : 0.3836

 Mcnemar's Test P-Value : 0.00225

            Sensitivity : 0.35714
            Specificity : 0.96104
         Pos Pred Value : 0.76923
         Neg Pred Value : 0.80435
             Prevalence : 0.26667
         Detection Rate : 0.09524
   Detection Prevalence : 0.12381
      Balanced Accuracy : 0.65909

       'Positive' Class : 0
```

After adding the lasso cross validation method to the original logistic regression model, we can see that there is no improvements in the accuracy or the 95% confidence interval of the model. Like the logistic regression model, this model performed decently well but does seem to allow a large amount of non-approved applicants through the screening model and still has a large confidence interval range.

LDA Model:

Similar to the logistic model, the LDA model was created as a way to determine how good of a logistic model we have created in the previous step. The LDA model was created using all the variables from the training data and resulted in very similar results to the logistic model which was expected. The LDA model and its results are shown below.

```
Call:
lda(Loan_Status ~ ., data = train, family = binomial)

Prior probabilities of groups:
        0         1
0.3182898 0.6817102

Group means:
    Gender1  Married1 Dependents0 Dependents1 Dependents2 Dependents3+ Education1 Self_Employed1 ApplicantIncome
0 0.7761194 0.5671642   0.5746269   0.1716418   0.1417910   0.07462687  0.7089552      0.1492537        5069.366
1 0.8153310 0.6759582   0.5261324   0.1602787   0.2055749   0.09059233  0.8153310      0.1219512        5056.282
  CoapplicantIncome LoanAmount Loan_Amount_Term60 Loan_Amount_Term84 Loan_Amount_Term120 Loan_Amount_Term180
0          1633.649   146.9478        0.000000000        0.007462687         0.000000000          0.06716418
1          1520.620   141.7875        0.006968641        0.006968641         0.006968641          0.08013937
  Loan_Amount_Term240 Loan_Amount_Term300 Loan_Amount_Term360 Loan_Amount_Term480 Credit_History1
0         0.007462687          0.03731343           0.8059701          0.05970149       0.5447761
1         0.003484321          0.01393728           0.8641115          0.01742160       0.9860627
  Property_AreaSemiurban Property_AreaUrban
0              0.3059701          0.3208955
1              0.4390244          0.2926829

Coefficients of linear discriminants:
                                LD1
Gender1                -1.570647e-02
Married1                2.684562e-01
Dependents0            -1.025662e-01
Dependents1            -1.785045e-02
Dependents2             2.132794e-01
Dependents3+            1.742841e-01
Education1              3.191714e-01
Self_Employed1         -1.033002e-01
ApplicantIncome        -2.773674e-05
CoapplicantIncome      -2.824877e-05
LoanAmount             -8.240674e-04
Loan_Amount_Term60      4.850204e+00
Loan_Amount_Term84      2.839307e+00
Loan_Amount_Term120     4.273108e+00
Loan_Amount_Term180     3.962958e+00
Loan_Amount_Term240     1.964047e+00
Loan_Amount_Term300     3.065095e+00
Loan_Amount_Term360     3.715643e+00
Loan_Amount_Term480     2.504545e+00
Credit_History1         3.185546e+00
Property_AreaSemiurban  7.323588e-01
Property_AreaUrban      2.335393e-01
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 10  3
         1 18 74

               Accuracy : 0.8
                 95% CI : (0.7107, 0.8717)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.07271

                  Kappa : 0.3836

 Mcnemar's Test P-Value : 0.00225

            Sensitivity : 0.35714
            Specificity : 0.96104
         Pos Pred Value : 0.76923
         Neg Pred Value : 0.80435
             Prevalence : 0.26667
         Detection Rate : 0.09524
   Detection Prevalence : 0.12381
      Balanced Accuracy : 0.65909

       'Positive' Class : 0
```

The results of the LDA model appear to be identical to the Logistic model and the logistic model with lasso having an accuracy of 80% and 95% confidence interval between 71.07% and 87.17%.

KNN Model:

The KNN model was utilized in this analysis since it is a model that does not rely on the variables included in the data set. Creating the KNN model involved testing to determine which level of "K" should be used in order to best fit our data. After trial and error, the best "K" level was determined to be 17 since it best fit the training data and created the model with the highest accuracy scores. Moving forward with determining the testing error of the model, the model was then tested using the testing data set. The model and the results of the confusion matrix are shown below.

```
#KNN model with k=17
knn17 <- knn(train = knn.train, test = knn.test, cl = knn.trainLabels, k=17)
summary(knn17)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0  5  6
         1 23 71

               Accuracy : 0.7238
                 95% CI : (0.628, 0.8066)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.635364

                  Kappa : 0.1247

 Mcnemar's Test P-Value : 0.002967

            Sensitivity : 0.17857
            Specificity : 0.92208
         Pos Pred Value : 0.45455
         Neg Pred Value : 0.75532
             Prevalence : 0.26667
         Detection Rate : 0.04762
   Detection Prevalence : 0.10476
      Balanced Accuracy : 0.55032

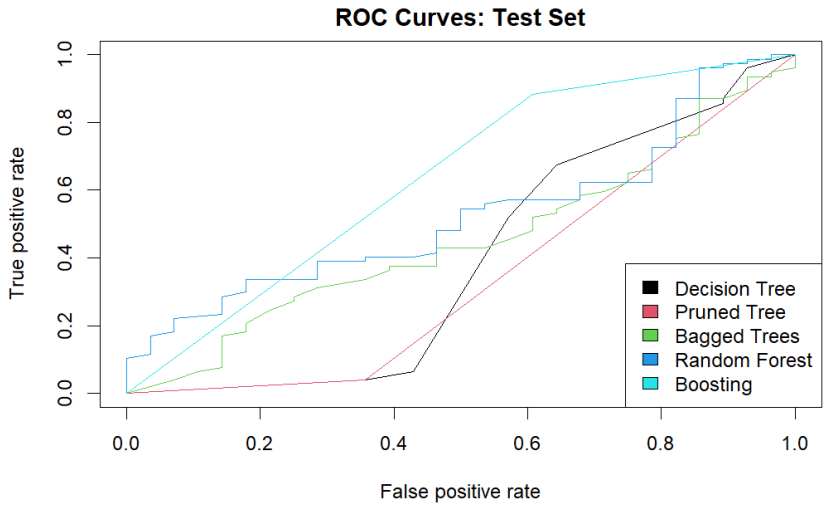
       'Positive' Class : 0
```

From these results, the accuracy of the model being around 71% shows that this model is not a good fit for the given data set. The model has a 95% confidence interval that is in between 61.79% and 79.82% which is much lower than the logistic and LDA models that have already been tested. One of the main issues with this dataset is the fact that it is incorrectly predicting candidates that should not be approved to the category of approved applicants. This would only cost the company more employee labor time since it is filling the applications that reach an employee with candidates that should not be approved.

Tree Based Decision Models:

To determine the best possible tree classification model for this data set, five different tree-based models and test all of their accuracies. The five different models are a decision tree, a pruned

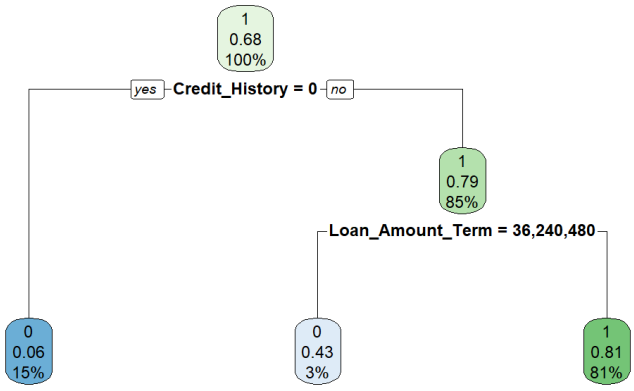
decision tree, bagging model, a random forest model, and boosting. To determine which model will most likely perform the best when introducing the data set, the models were plotted on a ROC curve plot to see their positive versus false rate. Based on the chart below, it is clear that the boosting model performed the best followed by the random forest model and the worst model appears to be the pruned tree.



In order to investigate further, the models were then tested against the testing data in order to determine their accuracies as well as their confidence intervals. The creation of the models and their respective results are shown below.

Classification Tree:

```
#Decision Tree
tree.mod = rpart(as.factor(Loan_Status)~., train, method="class", parms=list(split="gini"))
tree.pred=apply(predict(tree.mod, newdata=test), 1, max)
```



```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 14  8
         1 14 69

               Accuracy : 0.7905
                 95% CI : (0.7001, 0.8638)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.1106

                  Kappa : 0.4251

 Mcnemar's Test P-Value : 0.2864

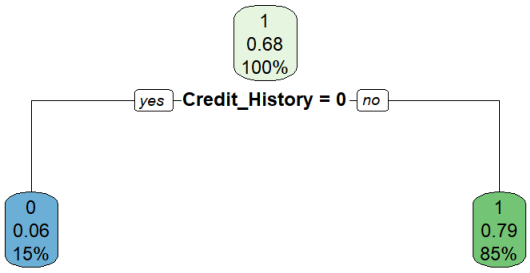
            Sensitivity : 0.5000
            Specificity : 0.8961
         Pos Pred Value : 0.6364
         Neg Pred Value : 0.8313
             Prevalence : 0.2667
         Detection Rate : 0.1333
   Detection Prevalence : 0.2095
      Balanced Accuracy : 0.6981

       'Positive' Class : 0
```

In this model, we can see that there are only two decisions nodes out of all of the variables in the data set which is somewhat concerning. The two variables in the model are credit history and if the credit history of an applicant is equal to 0 then the second decision uses the loan amount term variable and splits the data based on if the length is equal to 36, 240, or 480 months. The accuracy of this model falls just short of 80% and has a 95% confidence interval between 70% to 86.37%. Based on the make up of the model being very limited in decision nodes as well as the model's accuracy, I would claim that this model is not the best fit for this data.

Pruning Model:

```
# Pruned Tree
cp_opt=tree.mod$cptable[which.min(tree.mod$cptable[, "xerror"]), "CP"]
pruned.mod= prune(tree = tree.mod, cp = cp_opt)
pruned.pred=apply(predict(pruned.mod, newdata=test), 1, max)
```



```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 10  3
         1 18 74

               Accuracy : 0.8
                 95% CI : (0.7107, 0.8717)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.07271

                  Kappa : 0.3836

 Mcnemar's Test P-Value : 0.00225

            Sensitivity : 0.35714
            Specificity : 0.96104
         Pos Pred Value : 0.76923
         Neg Pred Value : 0.80435
             Prevalence : 0.26667
         Detection Rate : 0.09524
   Detection Prevalence : 0.12381
      Balanced Accuracy : 0.65909

       'Positive' Class : 0
```

The pruned decision tree cuts down the number of decision nodes from two to only one. This model is very simple and only relies on the credit history variable and whether it is equal to 1 or 0. The accuracy of this model did slightly improve up to 80% but the model performed worse based on the number of would be rejected applicants being classified as an approved application. This model more accurately predicts applicants that would be approved but is worse when determining applicants that should be rejected.

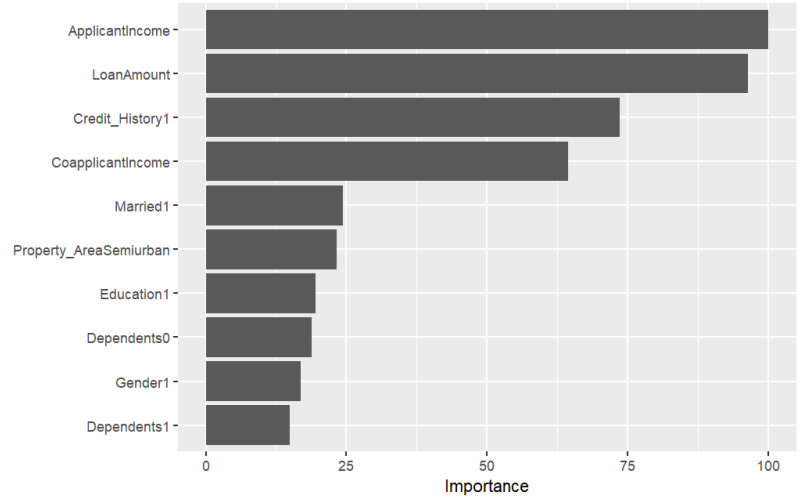
Bagging Model:

```
```{r}
#Bagging Model
ctrl <- trainControl(method = "cv",     # Cross-validation
                     number = 5)      # 5 folds

# Cross validate the credit model using "treebag" method;
bag_model <- train(as.factor(Loan_Status)~ .,
                           data = train,
                           method = "treebag",
                           trControl = ctrl)
bag_model

pred_caret <- predict(bag_model, newdata = test, type = "raw")

#most important variables
vip(bag_model)
```
```




```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 15  6
         1 13 71

               Accuracy : 0.819
                 95% CI : (0.7319, 0.8874)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.02689

                  Kappa : 0.4974

 Mcnemar's Test P-Value : 0.16867

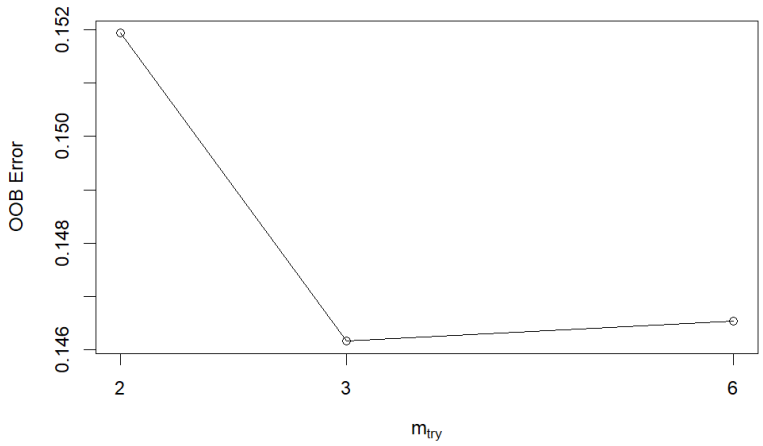
            Sensitivity : 0.5357
            Specificity : 0.9221
         Pos Pred Value : 0.7143
         Neg Pred Value : 0.8452
             Prevalence : 0.2667
         Detection Rate : 0.1429
   Detection Prevalence : 0.2000
      Balanced Accuracy : 0.7289

       'Positive' Class : 0
```

Looking at the bagging model, we achieved an accuracy level of 81.9% making this model the highest accuracy of all the models that have previously been tested. The confidence interval for this model ranges from 73% to a high of 88.7%. This model performed the best in all categories overall and had the most accurate rejection classification of all the models tested so far which is one of the most important elements in this type of application screening model.

Random Forest:

```
mtry.rf = tuneRF(x = subset(train, select = -Loan_Status),
                 y = train$Loan_Status,
                 ntreeTry = 500)
```




```
#Random Forrest
model_randfr= predict(rf.mod, newdata = test, type = "class",ntree = 3)
```

```
          Reference
Prediction  0  1
         0 12  3
         1 16 74

               Accuracy : 0.819
                 95% CI : (0.7319, 0.8874)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : 0.026887

                  Kappa : 0.4571

 Mcnemar's Test P-Value : 0.005905

            Sensitivity : 0.4286
            Specificity : 0.9610
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.8222
             Prevalence : 0.2667
         Detection Rate : 0.1143
   Detection Prevalence : 0.1429
      Balanced Accuracy : 0.6948

       'Positive' Class : 0
```

To determine the optimal ntree value that has the lowest error rate for our random forest model. I used the tuneRF function and plotted the results in order to visually see the lowest error rate which came out to be equal to 3. After determining the value of ntree for our model, the random forest model which was created using all of the training variables was then tested against the testing data. The accuracy of this model resulted in 81.9% with a confidence interval from 0.7319 to 0.8874. The model overall performed well but was only able to classify 12 out of the 28 rejected applications whereas the bagging model classified 15 out of 28 applications.

Boosting Model:

```
#Boosting
model.boos <- gbm(formula = Loan_Status~ ., data=train, n.trees = 1000)

pred.boost=predict(model.boos, newdata=test,n.trees=1000, type="response")
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 11 17
         1  9 68

               Accuracy : 0.7524
                 95% CI : (0.6586, 0.8314)
    No Information Rate : 0.8095
    P-Value [Acc > NIR] : 0.9431

                  Kappa : 0.3036

 Mcnemar's Test P-Value : 0.1698

            Sensitivity : 0.5500
            Specificity : 0.8000
         Pos Pred Value : 0.3929
         Neg Pred Value : 0.8831
             Prevalence : 0.1905
         Detection Rate : 0.1048
   Detection Prevalence : 0.2667
      Balanced Accuracy : 0.6750

       'Positive' Class : 0
```

The boosting model was tested and had high expectations based on the decision tree ROC plot. The model was trained using all of the variables available with n.trees set to 1000. It was then tested using the testing data and the results are poor especially when compared to other models. The accuracy of the model was 0.7524 and had a confidence interval from 0.6586 and 0.8314. Based off the confusion matrix of the model, the model performed the worst out of all the models since it has the highest number of approved applications that are being classified as rejected. This model would simply cost the company more in revenue than saving in employee costs.

Based on the results from the tree based model's confusion matrices, we can infer that the best model based of its accuracy percentage are the bagging and random forest models with an accuracy percentage of 81.9% and confidence intervals from 73.19% to 88.74%. From both models, the bagging model was stricter at classifying rejected applications which led to more approved applications being rejected and the random forest model was able to less strict but had more rejected applications that made it through the screening model.

SVM Model:

The final model that was created in this analysis was the SVM model. This is a supervised machine learning model that helps classify a set of data. To determine the best cost level to assign to the SVM model, the model was tuned at cost levels of 0.001, 0.01, 0.1, 1,5,10, and 50. Doing this allowed

the creation of the best possible SVM model with the lowest error rate. The error rates for each cost rate are shown below.

Description: df [7 × 3]

cost <dbl>	error <dbl>	dispersion <dbl>
1e-03	0.3183832	0.05097406
1e-02	0.3183832	0.05097406
1e-01	0.1828350	0.03855153
1e+00	0.1971207	0.03159329
5e+00	0.2018826	0.03397912
1e+01	0.1971761	0.03198598
5e+01	0.1971761	0.03198598

7 rows

The optimal cost level was determined to be 0.01. Using this cost level, the most optimal SVM model was created and then tested using the testing data set. After doing this we received the model's accuracy and confusion matrix that are shown below.

```
Call:
best.tune(METHOD = svm, train.x = as.factor(Loan_Status) ~ ., data = train, ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 50)), type = "C-classification", kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1

Number of Support Vectors:  231


Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 10 18
         1  3 74

               Accuracy : 0.8
                 95% CI : (0.7107, 0.8717)
    No Information Rate : 0.8762
    P-Value [Acc > NIR] : 0.99101

                  Kappa : 0.3836

 Mcnemar's Test P-Value : 0.00225

            Sensitivity : 0.76923
            Specificity : 0.80435
         Pos Pred Value : 0.35714
         Neg Pred Value : 0.96104
             Prevalence : 0.12381
         Detection Rate : 0.09524
   Detection Prevalence : 0.26667
      Balanced Accuracy : 0.78679

       'Positive' Class : 0
```

This model has an accuracy of 80% and has a 95% confidence interval from 71.07% to 87.17%. Even though this model does not have the highest accuracy of all the models, it does have a higher strictness level that is preventing applicants that would not be approved from continuing through the application process. The model does also reject potential applicants that would be eligible for a loan higher than other models but depending on the goal of the Dream Housing Financial company this model may be a very good fit.

Closing Remarks:

Model	Accuracy	Confidence Interval	False Positive %	False Negative %
Logistic Model	80%	(0.7107, 0.8717)	64.29%	3.90%
Logistic + Lasso	80%	(0.7107, 0.8717)	64.29%	3.90%
LDA	80%	(0.7107, 0.8717)	64.29%	3.90%
KNN	72.38%	(0.6280, 0.8066)	82.14%	7.79%
Decision Tree	79.05%	(0.7001, 0.8638)	50%	10.39%
Pruned Tree	80%	(0.7107, 0.8717)	64.29%	3.90%
Bagging	81.9%	(0.7319, 0.8874)	46.43%	7.79%
Random Forest	81.9%	(0.7319, 0.8874)	57.14%	3.90%
Boosting	75.4%	(0.6586, 0.8314)	45%	20%
SVM	80%	(0.7107, 0.8717)	23%	19.57%

Overall, the models that have been created throughout this project to classify potential loans based on if the application would be approved or not. Based on the results of each of the models we can infer that both the bagging and random forest models have the highest accuracy percentage of all models and had identical confidence levels that were also the highest of all the other models. Looking at the false positive % of both these models, the bagging model had a percentage of 46% while the random forest model had a percentage of 57%. The random forest does have a lower false negative percentage at 3.90% and the bagging model has 7.79%. Depending on the goal of the Dream Housing Financial company, I would recommend utilizing the bagging model if they would rather a stricter model that is better at rejecting non approved applications at the cost of potential more approved applications being rejected. If the company would rather have a less strict model that better predicts the approved applications but at the cost of have more non approved applications making it through the screening process and cost employees more time to review the applications again.

To further improve this analysis, one of the methods would be to include much more confrontation to allow the models to have more prediction variables to properly classify the data. Since in this analysis the data set was only a part of the entire data set that the Dream Housing Financial company claims to have, I believe with the addition of those other variables the models could possibly achieve higher accuracy scores. Another method that would allow us to further improve the analysis of this data would be to have more observations that include a customer being denied for a loan since in our testing data set there were only 28 loans that were not approved which made it difficult to determine the not approved loans or at least made it difficult for our model to determine which variables were statistically significant in estimating the dependent variable "Loan_Status".

My overall learning from the assignment must be my statistical learning as well as my R skills since I have had previous experience working with models and was able to carry my skills well into learning more about them in this program. Through the project, it became clear that I was never able to determine if a model was doing good without having other models to compare it to. I am positive that there may be better models to fit the data that I worked with and would make me reconsider the model I consider the best for this scenario. Throughout my undergrad, I was very capable in the modeling and interpretation aspect of data analysis but was always lacking in the statistical side of the role which is why I am happy to be taking this program to help me be a more well-rounded data scientist.