Predicting Loan Default for a Bank

Name: Thomas Sugg **G Number**: G********

```r
# Add all library you will need here
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------ t
```

```
## v ggplot2 3.1.0     v purrr   0.3.0
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts -------------------------------------------------------------------------------- tidyvers
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(rpart)
library(rpart.plot)
library(kknn)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(ISLR)

# This will read in the data frame
loan_data <- readRDS(file = "/cloud/project/Final Project/loan_data.rds")
```

```r
# Create training and test data
set.seed(314)
train_index <- sample(1:nrow(loan_data), floor(0.7*nrow(loan_data)))

# training
loan_training <- loan_data[train_index, ]

# test
loan_test <- loan_data[-train_index, ]

# Function for analyzing confusion matrices
cf_matrix <- function(actual_vec, pred_prob_vec, positive_val,
                      cut_prob = 0.5, search_cut = FALSE) {

  if (search_cut == FALSE) {
  actual <- actual_vec == positive_val; pred <- pred_prob_vec >= cut_prob
  P <- sum(actual); N <- length(actual) - P; TP <- sum(actual & pred)
  FN <- P - TP; TN <- sum(!(actual) & !(pred)); FP <- N - TN

  if (TP != 0) { Precision <- TP/(TP + FP); Recall <- TP/(TP + FN)
                 F1 <- 2*((Precision*Recall)/(Precision + Recall))}

  if(TP == 0) { Precision = 0; Recall = 0; F1 = 0 }

  model_results <- list(confusion_matrix =
    data.frame(metric = c("Correct", "Misclassified", "True Positive",
                           "True Negative","False Negative", "False Positive"),
               observations = c(TN + TP, FN + FP, TP, TN, FN, FP),
               rate = c((TN + TP)/(N + P), (FN + FP)/(N + P), TP/P, TN/N, FN/P, FP/N),
               pct_total_obs = c((TN + TP), (FN + FP), TP, TN, FN, FP)*(1/(N + P)),
               stringsAsFactors = FALSE),
    F1_summary =
    data.frame(metric = c("Precision", "Recall", "F1 Score"),
               value = c(Precision, Recall, F1),
               stringsAsFactors = FALSE))
return(model_results) }

  if (search_cut == TRUE) {
    optimal_cut = data.frame(cut_prob = seq(0,1, by = 0.05),
                             correct_rate = NA, F1_score = NA,
                             false_pos_rate = NA, false_neg_rate = NA)

    for (row in (1:nrow(optimal_cut))) {
      actual <- actual_vec == positive_val
      pred <- pred_prob_vec >= optimal_cut$cut_prob[row]
      P <- sum(actual); N <- length(actual) - P
      TP <- sum(actual & pred); FN <- P - TP
      TN <- sum(!(actual) & !(pred)); FP <- N - TN

      if (TP != 0) { Precision <- TP/(TP + FP); Recall <- TP/(TP + FN)
          F1 <- 2*((Precision*Recall)/(Precision + Recall))}

      if(TP == 0) { Precision = 0; Recall = 0; F1 = 0 }
```

```
      optimal_cut[row, 2:5] <- c((TN + TP)/(N + P), F1, FP/N, FN/P)
    }
return(optimal_cut)
  }
}
```

Loan Data

The loan_data data frame contains information on 3-year loans that were originated in 2013 by a local bank for customers residing in the United States. The company is looking to see if it can determine the factors that lead to loan default and whether it can predict if a customer will eventually default on their loan at time of loan origination. The goal is to become better at identifying customers at risk of defaulting on their loans to minimize the bank's financial losses.

The dataset contains a mixture of applicant demographics (gender, age, residence, etc..), financial information (income, debt ratios, FICO scores, etc..), and applicant behavior (number of open accounts, historical engagement with the bank's products, number of missed payments, etc. . .)

Specifically, the broad questions that the bank is trying to answer include:

1. What are the factors that contribute to customers defaulting on their loans?
2. Is it possible to predict whether a customer will default on their loan? If so, how accurate are the predictions?
3. How many costly errors does the predictive model produce (customers classified as not defaulting, but eventually do)?

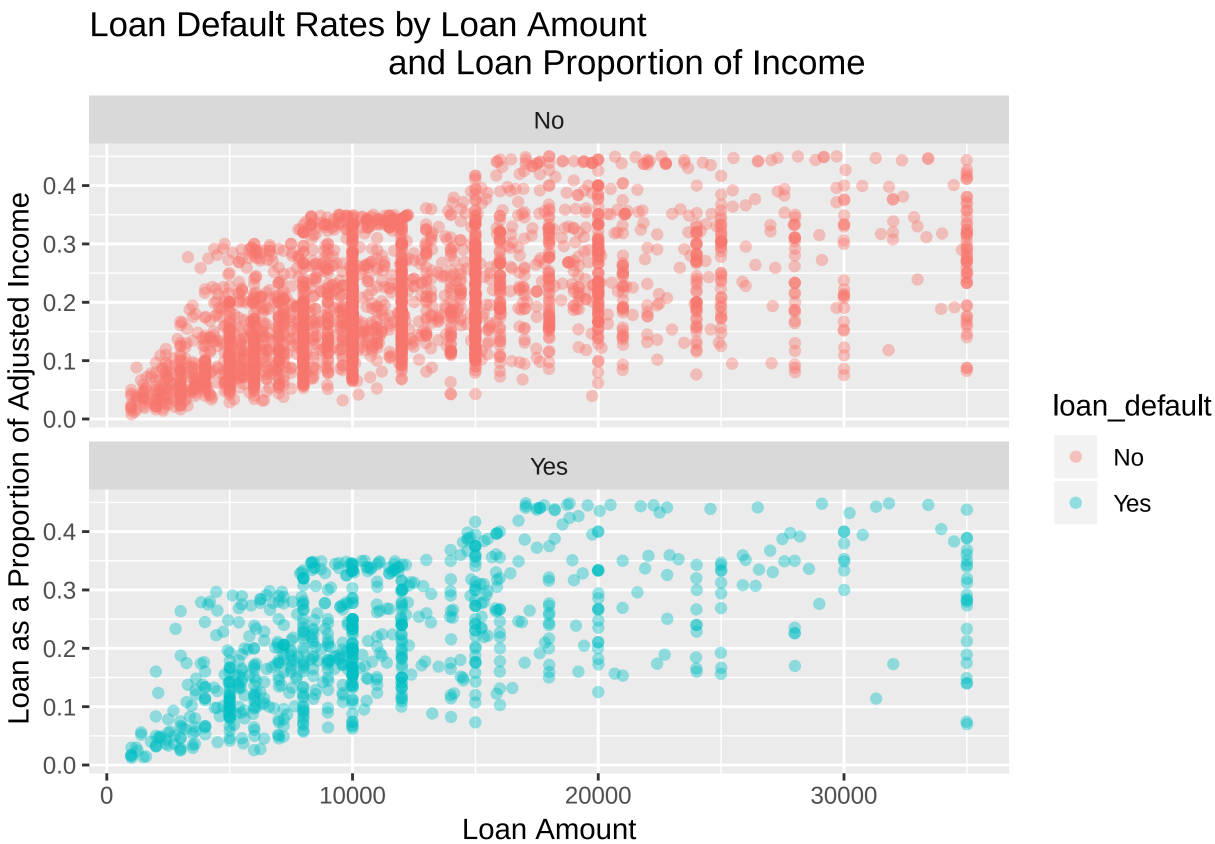
Exporatory Data Analysis Section

1. Does the loan amount and the loan proportion of income determine loan default?

Findings: No, loan amount and loan proportion of income do not appear to have an effect on loan default. For default "Yes" and "No", both scatter plots look the same.

```
loan_amnt_graph <- ggplot(data = loan_data, mapping = aes(x = loan_amnt, y = pct_loan_income, color = lo
                          geom_jitter(alpha = 0.4) +
                          facet_wrap(~loan_default, nrow = 2)+
                          labs(title = "Loan Default Rates by Loan Amount
                               and Loan Proportion of Income",
                            x = "Loan Amount",
                            y = "Loan as a Proportion of Adjusted Income")

loan_amnt_graph
```



2. Do loan default rates differ by location of residence and type of residence?

Findings: Yes, customers who rent and own in the Northeast and Midwest have higher default rates than other customers. Default rates in these areas are more than double the rates in the West, Mid-Atlantic, South, and Southwest.

```
default_by_residence <- loan_data %>% group_by(us_region_residence, residence_property) %>% summarise(to
                                                                                                       customers_who_defaulted = sum(lo
                                                                                                       default_rate = customers_who_def
arrange(default_by_residence, desc(default_rate))
```

```
## # A tibble: 12 x 5
## # Groups:   us_region_residence [6]
##    us_region_resid~ residence_prope~ total_customers customers_who_d~
##    <fct>            <fct>                      <int>            <int>
##  1 Northeast        Rent                         327              155
##  2 Midwest          Rent                         189               81
##  3 Midwest          Own                          283              100
##  4 Northeast        Own                          385              133
##  5 West             Rent                         396               71
##  6 Mid-Atlantic     Rent                         439               75
##  7 South            Rent                         126               20
##  8 West             Own                          494               77
##  9 Southwest        Rent                         141               21
## 10 Mid-Atlantic     Own                          603               79
## 11 Southwest        Own                          198               23
## 12 South            Own                          164               12
## # ... with 1 more variable: default_rate <dbl>
```

3. Do loan default rates differ by education?

Findings: Yes, customers with the two lowest levels of education have the highest deault rates. Interestingly, customers with the highest level of education have the third highest default rate.

```r
default_by_education <- loan_data %>% group_by(highest_ed_level) %>% summarise(total_customers = n(),
                                                                                customers_who_defaulted = sum(lo
                                                                                default_rate = customers_who_def
arrange(default_by_education, desc(default_rate))
```

```
## # A tibble: 5 x 4
##   highest_ed_level total_customers customers_who_defaulted default_rate
##   <fct>                      <int>                   <int>        <dbl>
## 1 High School                  328                     202        0.616
## 2 < High School                298                     130        0.436
## 3 PhD or Doctorate             408                      94        0.230
## 4 Masters                      877                     138        0.157
## 5 Bachelors                   1834                     283        0.154
```

4. Do loan default rates differ by income bracket? Income levels determined by the Pew Research Center in 2017.

(lower = less than $39,500). (middle = between $39,500 and $118,000). (upper = more than $118,000).

Findings: Yes, default rates decrease as income increases.

```r
income_levels <-cut(x = loan_data$adjusted_annual_inc,
                    breaks = c(-Inf, 39500, 118000, Inf),
                    labels = c("lower","middle","upper"),
                    right = TRUE)

income_levels_data <- cbind(loan_data, income_levels)

income_levels_data %>% group_by(income_levels) %>% summarise(total_customers = n(),
                                                              customers_who_defaulted =
                                                              default_rate = customers_w
```

```
## # A tibble: 3 x 4
##   income_levels total_customers customers_who_defaulted default_rate
##   <fct>                   <int>                   <int>        <dbl>
## 1 lower                    1540                     435        0.282
## 2 middle                   1973                     370        0.188
## 3 upper                     232                      42        0.181
```

5. Is there an interaction between customer credit history and default rate? FICO score levels replicate those of Experian.

(very poor = between 300 and 579). (fair = between 580 and 669). (good = between 670 and 739). (very good = between 740 and 799). (exceptional = between 800 and 850).

Findings: Yes, lower FICO scores appear to have a higher default rate. The same can be said about credit inquiries.

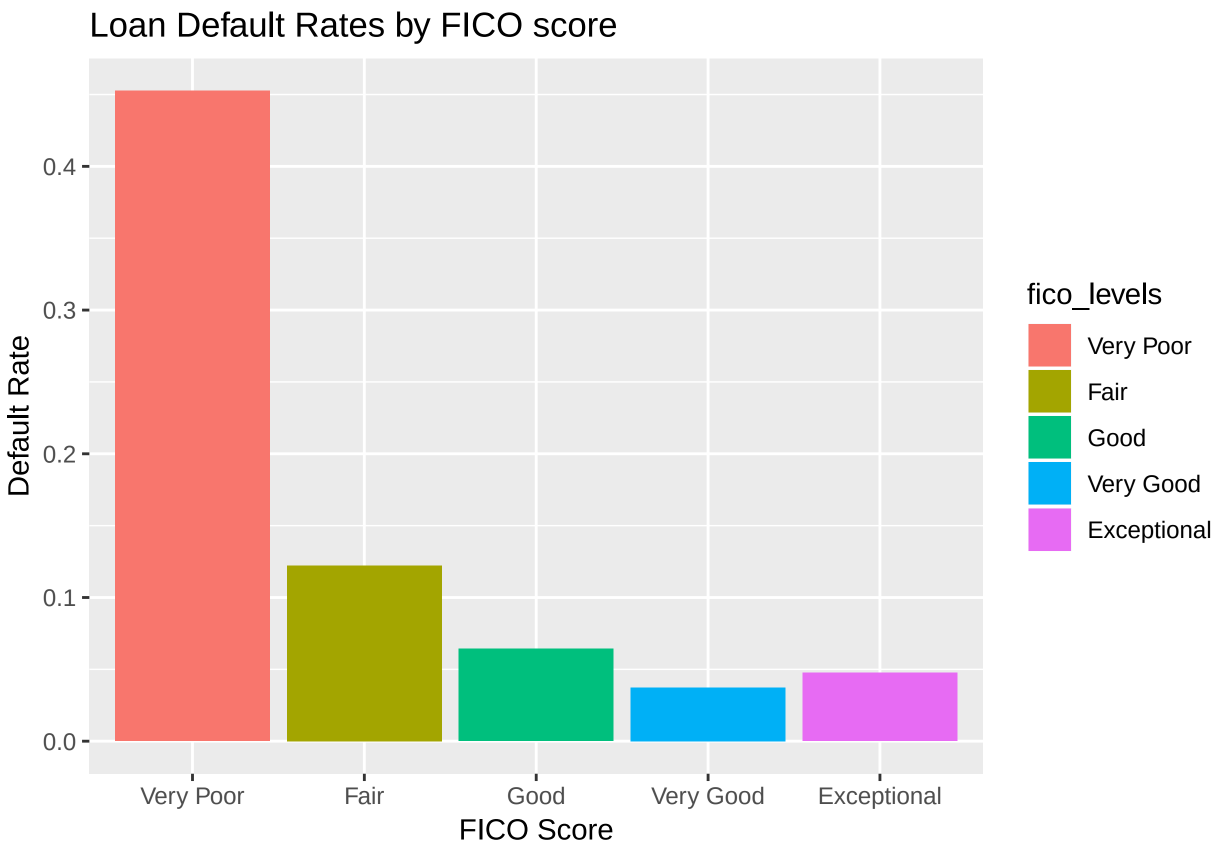
```r
fico_levels <-cut(x = loan_data$fico_score,
                  breaks = c(-Inf, 580, 670, 740, 800, 850),
                  labels = c("Very Poor","Fair","Good","Very Good","Exceptional"),
                  right = TRUE)

loan_fico_data <- cbind(loan_data, fico_levels)
```

```
default_by_fico <- loan_fico_data %>% group_by(fico_levels) %>% summarise(total_customers = n(),
                                                                          number_of_credit_inquirie
                                                                          customers_who_defaulted =
                                                                          default_rate = customers_
default_by_fico
```

```
## # A tibble: 5 x 5
##   fico_levels total_customers number_of_credi~ customers_who_d~
##   <fct>                 <int>            <int>            <int>
## 1 Very Poor              1339             1176              606
## 2 Fair                   1571             1175              192
## 3 Good                    653              497               42
## 4 Very Good               161              126                6
## 5 Exceptional              21               16                1
## # ... with 1 more variable: default_rate <dbl>
```

```
fico_default_graph <- ggplot(data = default_by_fico, mapping = aes(x = fico_levels, y = default_rate, f
                          geom_bar(stat = "identity") +
                          labs(title = "Loan Default Rates by FICO score",
                               x = "FICO Score",
                               y = "Default Rate")
fico_default_graph
```



6. Is there a relationship between gender and age that may predict default rates?

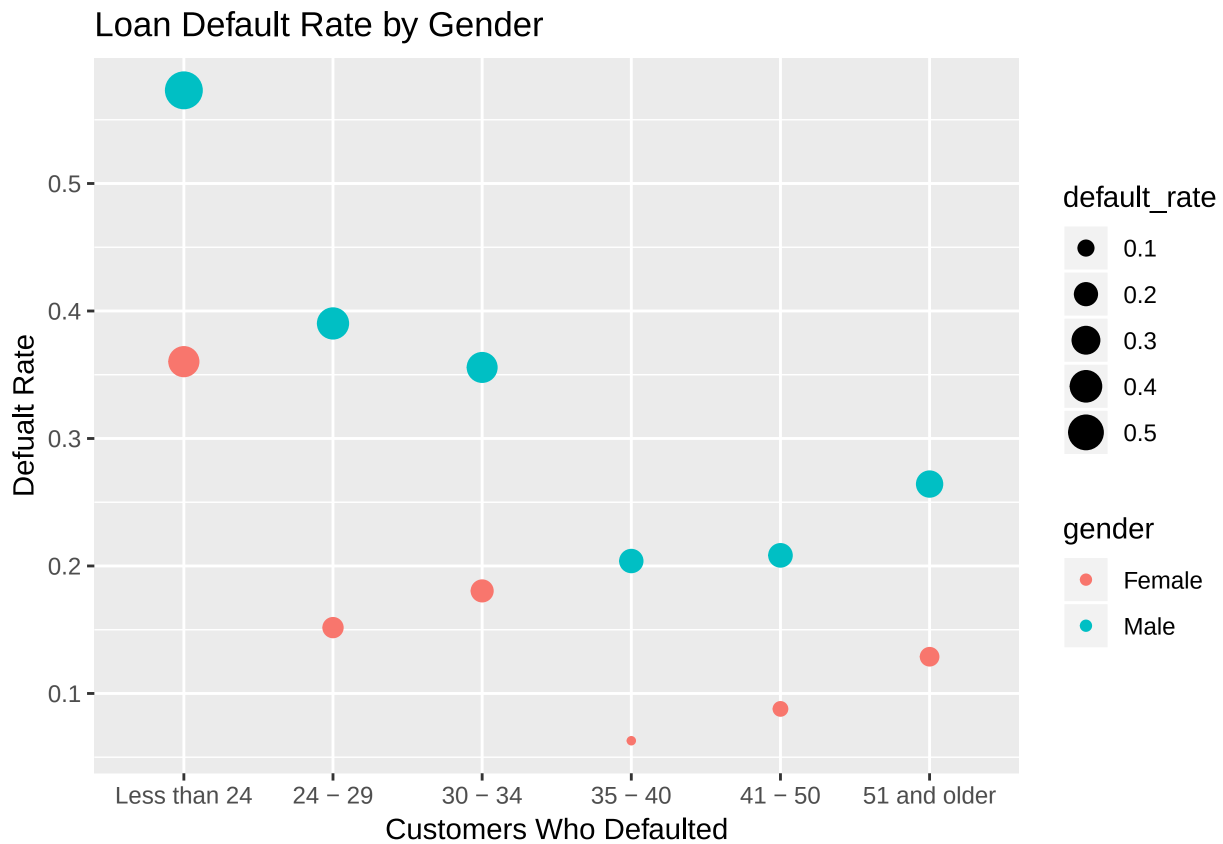
Findings: Yes, men consistently have higher default rates than women. As both genders become older, their default rates tend to decrease; however, default rates increase at the last two age categories.

```r
default_by_gender <- loan_data %>% group_by(gender,age_category) %>%  summarise(total_customers = n(),
                                                                           customers_who_defaulted = sum(loan_de
                                                                           default_rate = customers_who_defaulte
arrange(default_by_gender, desc(default_rate))
```

```
## # A tibble: 12 x 5
## # Groups:   gender [2]
##    gender age_category total_customers customers_who_defaulted default_rate
##    <fct>  <fct>                  <int>                   <int>        <dbl>
##  1 Male   Less than 24             260                     149       0.573
##  2 Male   24 - 29                  287                     112       0.390
##  3 Female Less than 24             297                     107       0.360
##  4 Male   30 - 34                  253                      90       0.356
##  5 Male   51 and older             193                      51       0.264
##  6 Male   41 - 50                  264                      55       0.208
##  7 Male   35 - 40                  309                      63       0.204
##  8 Female 30 - 34                  266                      48       0.180
##  9 Female 24 - 29                  455                      69       0.152
## 10 Female 51 and older             295                      38       0.129
## 11 Female 41 - 50                  421                      37       0.0879
## 12 Female 35 - 40                  445                      28       0.0629
```

```r
default_gender_plot <- ggplot(default_by_gender, mapping = aes(x = age_category, y = default_rate, colo
                       geom_point(mapping = aes(size = default_rate)) +
                       labs(title = "Loan Default Rate by Gender",
                            x = "Customers Who Defaulted",
                            y = "Defualt Rate")


default_gender_plot
```



7. Does the number of accounts 120 days overdue and public bankruptcies influence default rates?

Findings: The proportion of customers who were 120 days past due and the proportion of customers who had publicly filled bankruptcy were higher for customers who defaulted. However, the difference is very small and may prove unimportant in variable selection.

```
default_by_bankruptcies <- loan_data %>% group_by(loan_default) %>% summarise(number_of_customers = n()
                                                                     num_accounts_open = sum(r
                                                                     num_bankruptcies = sum(pu
                                                                  prop_acct_overdue = num_acco
                                                                  prop_bankruptcies = num_bank
default_by_bankruptcies
```

```
## # A tibble: 2 x 6
##   loan_default number_of_custo~ num_accounts_op~ num_bankruptcies
##   <fct>                   <int>            <int>            <int>
## 1 No                       2898              915              247
## 2 Yes                       847              279               89
## # ... with 2 more variables: prop_acct_overdue <dbl>,
## #   prop_bankruptcies <dbl>
```

8. Does a customer's debt to income ratio influence default rates?

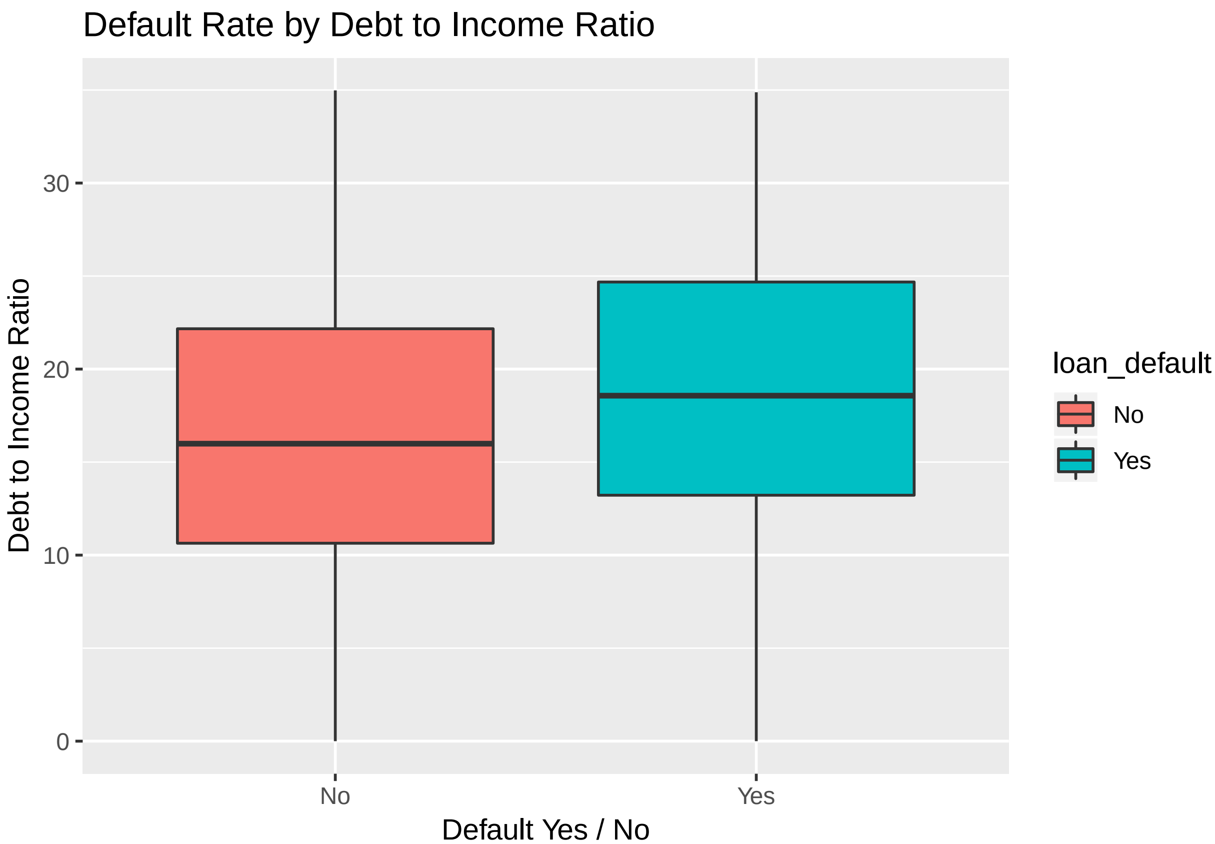
Findings: Customers who defaulted have a higher median debt to income ratio. However, it is only a small difference compared to those who did not default.

```
default_by_dti <- loan_data %>% group_by(loan_default)

default_by_dti
```

```
## # A tibble: 3,745 x 16
## # Groups:   loan_default [2]
##    loan_default residence_prope~ gender age_category highest_ed_level
##    <fct>        <fct>            <fct>  <fct>        <fct>
##  1 No           Rent             Female 41 - 50      Bachelors
##  2 No           Own              Female 30 - 34      Bachelors
##  3 No           Own              Female 35 - 40      Bachelors
##  4 Yes          Rent             Female Less than 24 Bachelors
##  5 No           Own              Female 35 - 40      Masters
##  6 No           Own              Male   41 - 50      Bachelors
##  7 No           Own              Female 24 - 29      Bachelors
##  8 No           Own              Female 24 - 29      PhD or Doctorate
##  9 No           Own              Female Less than 24 Bachelors
## 10 No           Rent             Female 35 - 40      PhD or Doctorate
## # ... with 3,735 more rows, and 11 more variables:
## #   us_region_residence <fct>, loan_amnt <int>, adjusted_annual_inc <dbl>,
## #   pct_loan_income <dbl>, fico_score <dbl>, dti <dbl>,
## #   inq_last_6mths <int>, open_acc <int>, bc_util <dbl>,
## #   num_accts_ever_120_pd <int>, pub_rec_bankruptcies <int>
```

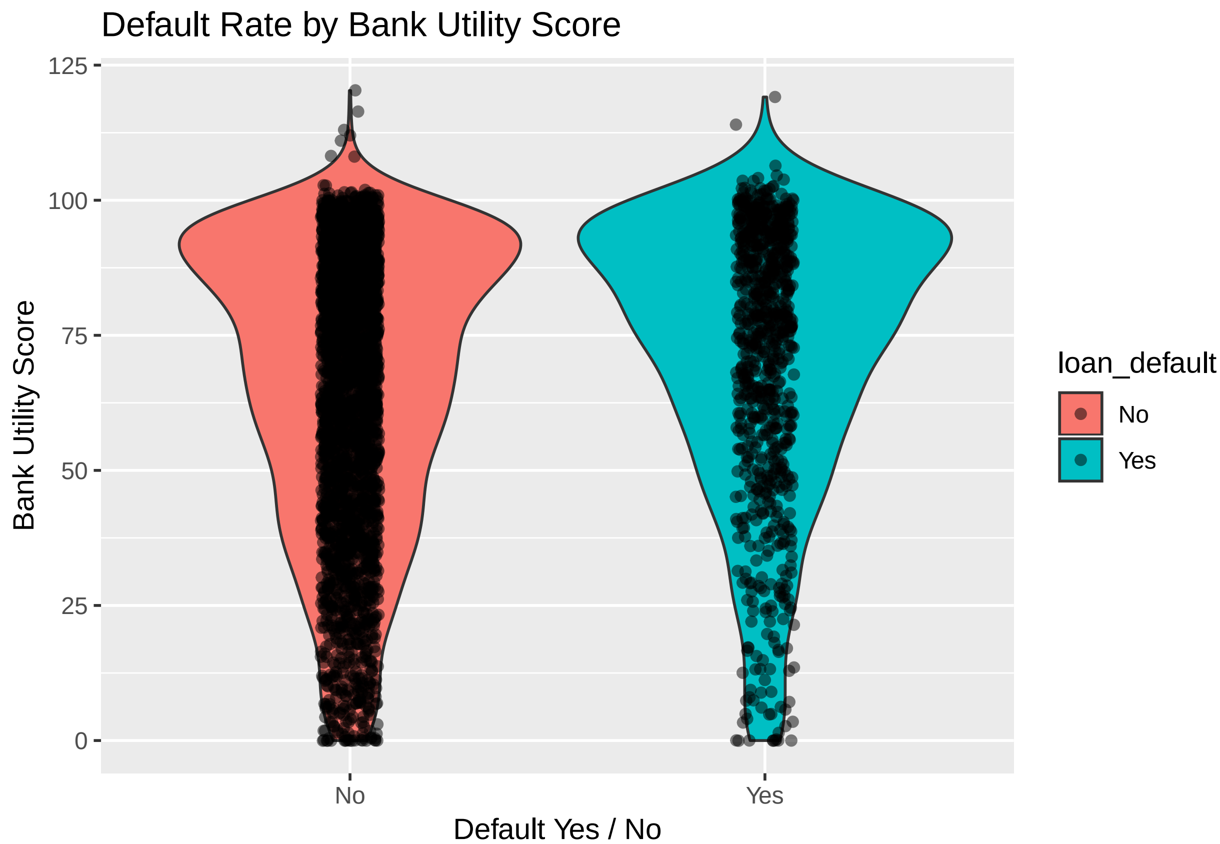
```
default_dti_plot <- ggplot(data = default_by_dti, mapping = aes(x = reorder(loan_default, dti, FUN =
                                                                                   median), y = dti, fill = loa
                                  geom_boxplot() +
                                  labs(title = "Default Rate by Debt to Income Ratio",
                                       x = "Default Yes / No",
                                       y = "Debt to Income Ratio")
default_dti_plot
```



9. Is there a relationship between bank utility score and loan default?

Findings: No, bank utility scores appears to be spread similarly for default "Yes" and defualt "No".

```
default_by_bc <- ggplot(data = loan_data, mapping = aes(x = reorder(loan_default, bc_util, FUN =
                                                                        median), y = bc_util, fill =
                                 geom_violin() +
                                 geom_jitter(width = 0.07, alpha = 0.5) +
                                 labs(title = "Default Rate by Bank Utility Score",
                                      x = "Default Yes / No",
                                      y = "Bank Utility Score")
default_by_bc
```



Variable Selection

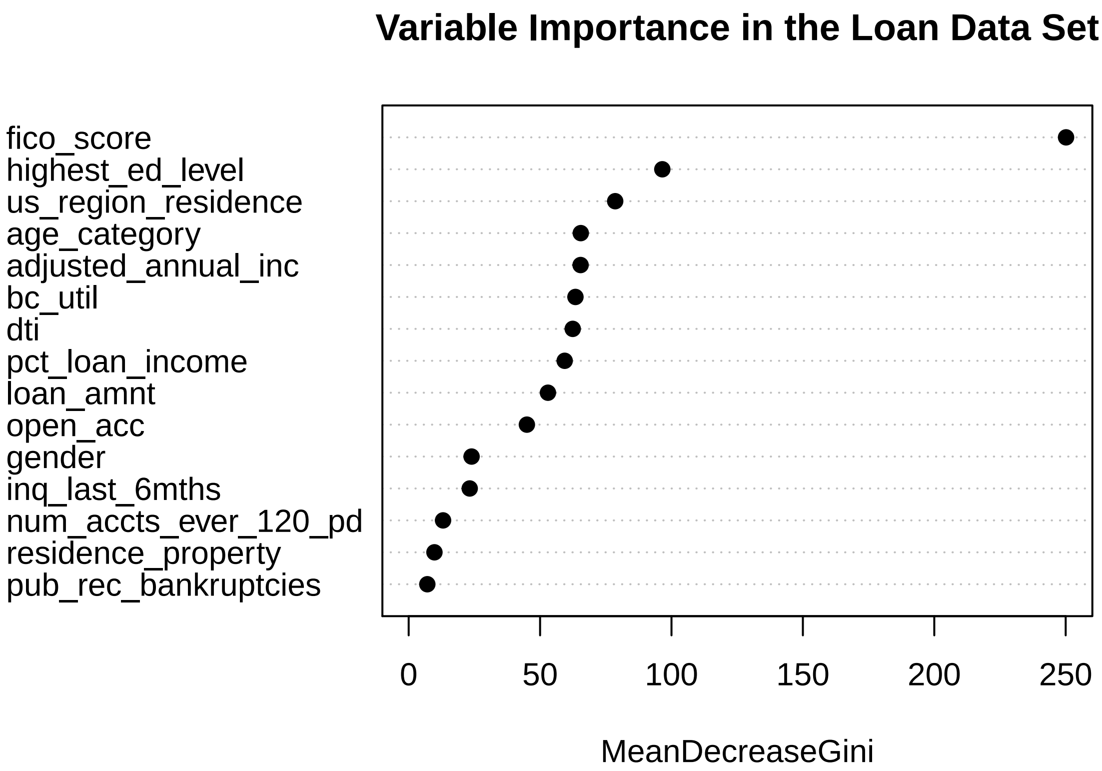
Random Forest Variable Importance

By using the varImpPlot function in Random Forests, the most important variables in this data set are determined. The "elbow" method can help eliminate variables that are the least important on the Gini index. After us_region_residence, the low variable importance becomes very similar throughout the rest of the list. By eliminating the variables below us_region_residence, the model will be simpler and over fitting will be avoided. The variables fico_score, highest_ed_level, and us_region_residence will be used in the predictive models.

```
set.seed(314)

loan_rf <- randomForest(loan_default ~., data = loan_training, importance = TRUE)

varImpPlot(loan_rf, type = 2, pch = 19, main = "Variable Importance in the Loan Data Set")
```



Predictive Modeling

Random Forests Classification: Predicting loan_default

```
#First, the model is fit using randomForest() on the training data.
set.seed(314)

loan_rf_training <- randomForest(loan_default ~ fico_score + highest_ed_level + us_region_residence,
                                 data = loan_training, importance = TRUE)
```

```
#Second, a results table is made.
loan_rf_training_results <- data.frame(loan_training,
                                       rforest_pred_0.5 = predict(loan_rf_training,
                                                                  newdata = loan_training,
                                                                  type = "response"), predict(loan_rf_t
                                                                                             newdata =
                                                                                             type = "p

loan_rf_training_results  %>% dplyr::select(loan_default, rforest_pred_0.5, Yes, No) %>% slice(1:10)
```

```
##    loan_default rforest_pred_0.5   Yes    No
## 1            No               No 0.000 1.000
## 2            No               No 0.000 1.000
## 3            No               No 0.010 0.990
## 4            No               No 0.002 0.998
## 5            No               No 0.000 1.000
## 6           Yes              Yes 0.724 0.276
## 7           Yes              Yes 0.894 0.106
## 8            No               No 0.002 0.998
## 9            No               No 0.016 0.984
## 10           No               No 0.000 1.000
```

The training results table is passed through the confusion matrix function and the optimal cut-off is determined. In this case, the optimal cut is 0.1 with an F1 score of 0.738.

```
cf_matrix(actual_vec = loan_rf_training_results$loan_default,
          pred_prob_vec = loan_rf_training_results$Yes,
          positive_val = "Yes", search_cut = TRUE)
```

```
##    cut_prob correct_rate  F1_score false_pos_rate false_neg_rate
## 1      0.00    0.2266311 0.3695179   1.0000000000     0.00000000
## 2      0.05    0.8298359 0.7081152   0.1938825851     0.08922559
## 3      0.10    0.8637924 0.7384615   0.1317217563     0.15151515
## 4      0.15    0.8733308 0.7381703   0.1016280217     0.21212121
## 5      0.20    0.8790538 0.7315834   0.0764676862     0.27272727
## 6      0.25    0.8855399 0.7368421   0.0621608288     0.29292929
## 7      0.30    0.8866845 0.7282708   0.0498273310     0.32996633
## 8      0.35    0.8840137 0.7126654   0.0429205723     0.36531987
## 9      0.40    0.8855399 0.7093023   0.0355204736     0.38383838
## 10     0.45    0.8863029 0.7061144   0.0305870745     0.39730640
## 11     0.50    0.8824876 0.6863544   0.0251603355     0.43265993
## 12     0.55    0.8801984 0.6694737   0.0187469166     0.46464646
## 13     0.60    0.8786723 0.6580645   0.0148001973     0.48484848
## 14     0.65    0.8779092 0.6491228   0.0108534780     0.50168350
## 15     0.70    0.8718047 0.6173121   0.0064134188     0.54377104
## 16     0.75    0.8641740 0.5771971   0.0024666996     0.59090909
## 17     0.80    0.8496757 0.5087282   0.0019733596     0.65656566
## 18     0.85    0.8149561 0.3120567   0.0004933399     0.81481481
## 19     0.90    0.7993132 0.2054381   0.0000000000     0.88552189
## 20     0.95    0.7825258 0.0776699   0.0000000000     0.95959596
## 21     1.00    0.7733689 0.0000000   0.0000000000     1.00000000
```

The random forest model, which was fit on the training data, will now be used on the test data.

```
#The test results table is made.
loan_rf_test_results <- data.frame(loan_test,
                                   rf_pred_0.5 = predict(loan_rf_training,
                                                         newdata = loan_test,
                                                         type = "response"),
                                   predict(loan_rf_training,
                                           newdata = loan_test,
                                           type = "prob"))
loan_rf_test_results %>%  dplyr::select(loan_default, rf_pred_0.5, Yes, No) %>% slice(1:10)
```

```
##    loan_default rf_pred_0.5   Yes    No
## 1            No          No 0.000 1.000
## 2            No          No 0.044 0.956
## 3            No          No 0.000 1.000
## 4            No          No 0.000 1.000
## 5            No          No 0.034 0.966
## 6           Yes         Yes 0.928 0.072
## 7            No          No 0.000 1.000
## 8           Yes         Yes 0.728 0.272
## 9            No          No 0.066 0.934
## 10           No          No 0.002 0.998
```

The random forest test results will now be passed through the confusion matrix to determine the F1 score and False Negative observations. The optimal cut-off from the training data set will be used. These numbers will be compared with the next 2 models.

```r
cf_matrix(actual_vec = loan_rf_test_results$loan_default,
          pred_prob_vec = loan_rf_test_results$Yes,
          positive_val = "Yes", cut_prob = .1)
```

```
## $confusion_matrix
##           metric observations      rate pct_total_obs
## 1        Correct          892 0.7935943    0.79359431
## 2  Misclassified          232 0.2064057    0.20640569
## 3  True Positive          152 0.6007905    0.13523132
## 4  True Negative          740 0.8495982    0.65836299
## 5 False Negative          101 0.3992095    0.08985765
## 6 False Positive          131 0.1504018    0.11654804
##
## $F1_summary
##      metric     value
## 1 Precision 0.5371025
## 2    Recall 0.6007905
## 3  F1 Score 0.5671642
```

Results:

Training - F1 = 0.738 False Negative = 0.152

Test - F1 = 0.567 False Negative = 0.399

Decision Tree Classification: Predicting loan_default

```r
#First, the Decision Tree model will be fit on the training data.
set.seed(314)

loan_tree_training <- rpart(loan_default ~ fico_score + highest_ed_level + us_region_residence,
                            data = loan_training,
                            method = "class",
                            control = rpart.control(cp = 0, minbucket = 4))
```

The results table is created and the optimal cp is found. The optimal cp in the training model is 0.0084.

```r
cp_results <- loan_tree_training$cptable %>% data.frame()
round(cp_results, 5)
```

```
##         CP nsplit rel.error  xerror    xstd
## 1  0.24747      0   1.00000 1.00000 0.03608
## 2  0.03451      1   0.75253 0.77778 0.03284
## 3  0.03030      3   0.68350 0.72391 0.03192
## 4  0.01178      5   0.62290 0.67340 0.03099
## 5  0.00842      6   0.61111 0.64310 0.03041
## 6  0.00758      8   0.59428 0.64815 0.03051
## 7  0.00673     10   0.57912 0.64310 0.03041
## 8  0.00505     11   0.57239 0.63636 0.03028
## 9  0.00449     16   0.54545 0.64141 0.03038
## 10 0.00337     19   0.53199 0.64310 0.03041
## 11 0.00224     25   0.51178 0.62795 0.03011
## 12 0.00168     28   0.50505 0.62626 0.03008
## 13 0.00084     42   0.48148 0.64983 0.03054
## 14 0.00056     46   0.47811 0.67003 0.03093
## 15 0.00000     57   0.47138 0.68519 0.03122
```

```r
loan_tree_training$cptable
```

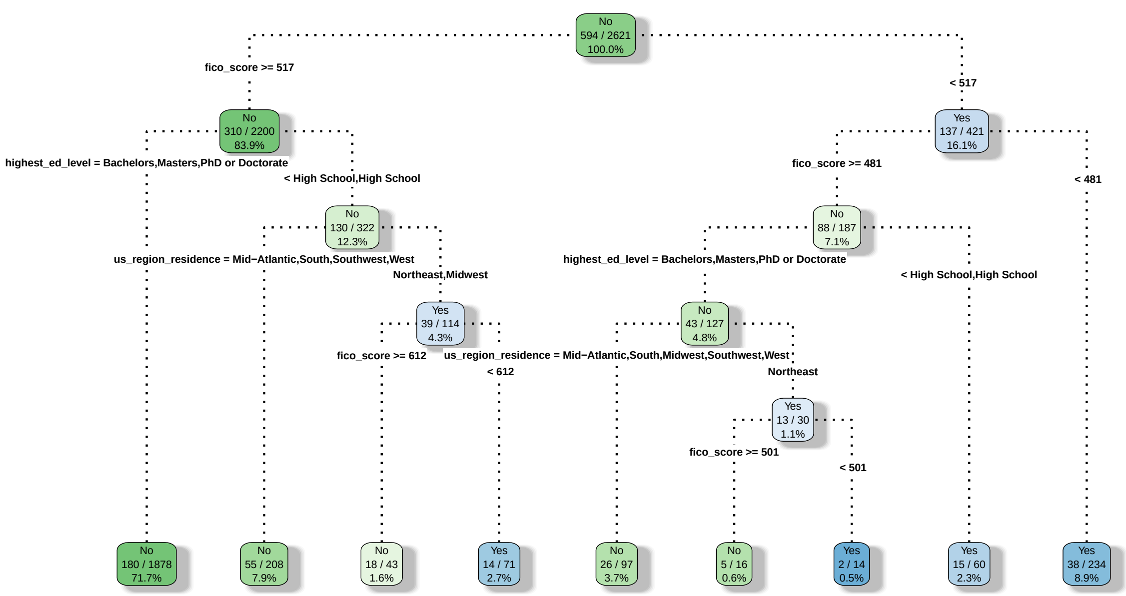
```
##              CP nsplit rel error    xerror       xstd
## 1  0.2474747475      0 1.0000000 1.0000000 0.03608279
## 2  0.0345117845      1 0.7525253 0.7777778 0.03284183
## 3  0.0303030303      3 0.6835017 0.7239057 0.03191798
## 4  0.0117845118      5 0.6228956 0.6734007 0.03099448
## 5  0.0084175084      6 0.6111111 0.6430976 0.03041157
## 6  0.0075757576      8 0.5942761 0.6481481 0.03051029
## 7  0.0067340067     10 0.5791246 0.6430976 0.03041157
## 8  0.0050505051     11 0.5723906 0.6363636 0.03027893
## 9  0.0044893378     16 0.5454545 0.6414141 0.03037852
## 10 0.0033670034     19 0.5319865 0.6430976 0.03041157
## 11 0.0022446689     25 0.5117845 0.6279461 0.03011152
## 12 0.0016835017     28 0.5050505 0.6262626 0.03007781
## 13 0.0008417508     42 0.4814815 0.6498316 0.03054305
## 14 0.0005611672     46 0.4781145 0.6700337 0.03093081
## 15 0.0000000000     57 0.4713805 0.6851852 0.03121520
```

```r
cp_results %>% filter(xerror == min(xerror)) %>% mutate(lower_value = xerror - xstd,
                                                         upper_value = xerror + xstd)
```

```
##            CP nsplit rel.error    xerror       xstd lower_value
## 1 0.001683502     28 0.5050505 0.6262626 0.03007781   0.5961848
##   upper_value
## 1   0.6563404
```

```r
#The Decision Tree is pruned using the new cp.
loan_pruned <- prune(loan_tree_training, cp = 0.0084)

rpart.plot(loan_pruned, type = 4, extra = 103, digits = -3,
           box.palette = "GnBu",
           branch.lty = 3, branch.lwd = 3,
           shadow.col = "gray", gap = 0, tweak = 1.0)
```



```r
#The Decision Tree results table is made for the training data.
loan_tree_results <- data.frame(loan_training,
                                predict(loan_pruned,
                                        newdata = loan_training,
                                        type = "prob"))

loan_tree_results %>%  dplyr::select(loan_default, Yes, No) %>% slice(1:10)
```

```
##    loan_default        Yes        No
## 1            No 0.09584665 0.9041534
## 2            No 0.09584665 0.9041534
## 3            No 0.09584665 0.9041534
## 4            No 0.09584665 0.9041534
## 5            No 0.09584665 0.9041534
## 6           Yes 0.85714286 0.1428571
## 7           Yes 0.75000000 0.2500000
## 8            No 0.09584665 0.9041534
## 9            No 0.09584665 0.9041534
## 10           No 0.09584665 0.9041534
```

The confusion matrix is used on the results table to determine the F1 score and optimal cut-off. In this case, the F1 score is 0.6457 and the cut-off is 0.375.

```r
cf_matrix(actual_vec = loan_tree_results$loan_default,
          pred_prob_vec = loan_tree_results$Yes,
          positive_val = "Yes", search_cut = TRUE)
```

```
##    cut_prob correct_rate   F1_score false_pos_rate false_neg_rate
## 1      0.00    0.2266311 0.36951788   1.0000000000      0.0000000
## 2      0.05    0.2266311 0.36951788   1.0000000000      0.0000000
## 3      0.10    0.8057993 0.61929693   0.1623088308      0.3030303
## 4      0.15    0.8057993 0.61929693   0.1623088308      0.3030303
## 5      0.20    0.8057993 0.61929693   0.1623088308      0.3030303
## 6      0.25    0.8057993 0.61929693   0.1623088308      0.3030303
## 7      0.30    0.8603586 0.64534884   0.0518006907      0.4393939
## 8      0.35    0.8626478 0.64566929   0.0463739517      0.4478114
## 9      0.40    0.8626478 0.64566929   0.0463739517      0.4478114
## 10     0.45    0.8653186 0.63720452   0.0340404539      0.4781145
## 11     0.50    0.8653186 0.63720452   0.0340404539      0.4781145
## 12     0.55    0.8653186 0.63720452   0.0340404539      0.4781145
## 13     0.60    0.8653186 0.63720452   0.0340404539      0.4781145
## 14     0.65    0.8653186 0.63720452   0.0340404539      0.4781145
## 15     0.70    0.8653186 0.63720452   0.0340404539      0.4781145
## 16     0.75    0.8653186 0.63720452   0.0340404539      0.4781145
## 17     0.80    0.8538726 0.58050383   0.0266403552      0.5538721
## 18     0.85    0.7771843 0.03947368   0.0009866798      0.9797980
## 19     0.90    0.7733689 0.00000000   0.0000000000      1.0000000
## 20     0.95    0.7733689 0.00000000   0.0000000000      1.0000000
## 21     1.00    0.7733689 0.00000000   0.0000000000      1.0000000
```

Now, the Decision Tree model will be fit on the test data.

```r
#The results table is created.
loan_tree_test <- data.frame(loan_test,
                             predict(loan_pruned,
                                     newdata = loan_test,
```

```r
                                        type = "prob"))
loan_tree_test %>% dplyr::select(loan_default, Yes, No) %>% slice(1:10)
```

```
##    loan_default        Yes        No
## 1            No 0.09584665 0.9041534
## 2            No 0.09584665 0.9041534
## 3            No 0.09584665 0.9041534
## 4            No 0.09584665 0.9041534
## 5            No 0.09584665 0.9041534
## 6           Yes 0.83760684 0.1623932
## 7            No 0.09584665 0.9041534
## 8           Yes 0.85714286 0.1428571
## 9            No 0.26442308 0.7355769
## 10           No 0.09584665 0.9041534
```

```r
loan_tree_test <- loan_tree_test %>% mutate(tree_pred_0.3 = ifelse(Yes >= 0.375, "Yes", "No"))
table(loan_tree_test$loan_default, loan_tree_test$tree_pred_0.3)
```

```
##
##         No Yes
##   No   825  46
##   Yes 149 104
```

The confusion matrix will show the F1 score and the count of False Negative observations.

```r
cf_matrix(actual_vec = loan_tree_test$loan_default,
          pred_prob_vec = loan_tree_test$Yes,
          positive_val = "Yes", cut_prob = 0.375)
```

```
## $confusion_matrix
##           metric observations       rate pct_total_obs
## 1        Correct          929 0.82651246    0.82651246
## 2  Misclassified          195 0.17348754    0.17348754
## 3  True Positive          104 0.41106719    0.09252669
## 4  True Negative          825 0.94718714    0.73398577
## 5 False Negative          149 0.58893281    0.13256228
## 6 False Positive           46 0.05281286    0.04092527
##
## $F1_summary
##      metric     value
## 1 Precision 0.6933333
## 2    Recall 0.4110672
## 3  F1 Score 0.5161290
```

Results:

Training - F1 = 0.646 False Negative = 0.448

Test - F1 = 0.516 False Negative = 0.588

KNN Classification: Predicting loan_default

```r
#First, the optimal k is found. In this case, it is 28.
set.seed(314)

train.kknn(loan_default ~ fico_score + highest_ed_level + us_region_residence,
          data = loan_training,
          kmax = 40)
```

```
##
## Call:
## train.kknn(formula = loan_default ~ fico_score + highest_ed_level +     us_region_residence, data = l
##
## Type of response variable: nominal
## Minimal misclassification: 0.1320107
## Best kernel: optimal
## Best k: 28
```

```
#Next, a model is fit on the training data using the optimal k. A results table is also created.
set.seed(314)

loan_knn_training <- kknn(loan_default ~ fico_score + highest_ed_level + us_region_residence,
                          train = loan_training,
                          test = loan_training,
                          k = 28,
                          distance = 2)
loan_knn_training_results <- data.frame(loan_training,
                                        knn_pred_0.5 = loan_knn_training$fitted.values,
                                        loan_knn_training$prob)

loan_knn_training_results %>% dplyr::select(loan_default, knn_pred_0.5, Yes, No) %>% slice(1:10)
```

```
##    loan_default knn_pred_0.5        Yes         No
## 1            No           No 0.00000000 1.00000000
## 2            No           No 0.00000000 1.00000000
## 3            No           No 0.00000000 1.00000000
## 4            No           No 0.09869163 0.90130837
## 5            No           No 0.06698121 0.93301879
## 6           Yes          Yes 0.84861473 0.15138527
## 7           Yes          Yes 0.94202921 0.05797079
## 8            No           No 0.11111075 0.88888925
## 9            No           No 0.14406312 0.85593688
## 10           No           No 0.01211420 0.98788580
```

A confusion matrix is constructed to find the F1 score and optimal cut-off for the KNN model on the training data set. In this case, the F1 score is 0.714 and the optimal cut-off is 0.4.

```
cf_matrix(actual_vec = loan_knn_training_results$loan_default,
          pred_prob_vec = loan_knn_training_results$Yes,
          positive_val = "Yes", search_cut = TRUE)
```

```
##    cut_prob correct_rate   F1_score false_pos_rate false_neg_rate
## 1      0.00    0.2266311 0.36951788    1.000000000     0.00000000
## 2      0.05    0.5757345 0.51652174    0.548593981     0.00000000
## 3      0.10    0.7069821 0.59706191    0.366551554     0.04208754
## 4      0.15    0.7729874 0.64688427    0.269363592     0.08249158
## 5      0.20    0.8119039 0.68131868    0.210162802     0.11279461
## 6      0.25    0.8450973 0.70622287    0.148001973     0.17845118
## 7      0.30    0.8576879 0.70972763    0.115934879     0.23232323
## 8      0.35    0.8698970 0.71223629    0.083374445     0.28956229
## 9      0.40    0.8771461 0.71352313    0.063640849     0.32491582
## 10     0.45    0.8790538 0.70290534    0.048347311     0.36868687
## 11     0.50    0.8748569 0.67779961    0.038973853     0.41919192
## 12     0.55    0.8760015 0.67005076    0.030093735     0.44444444
## 13     0.60    0.8718047 0.64102564    0.020720276     0.49494949
```

```
## 14      0.65     0.8645555 0.59977452     0.013320178      0.55218855
## 15      0.70     0.8527280 0.54047619     0.009373458      0.61784512
## 16      0.75     0.8405189 0.47355164     0.005920079      0.68350168
## 17      0.80     0.8302175 0.41370224     0.003946719      0.73569024
## 18      0.85     0.8149561 0.31786217     0.001973360      0.80976431
## 19      0.90     0.7977871 0.19452888     0.000000000      0.89225589
## 20      0.95     0.7840519 0.09003215     0.000000000      0.95286195
## 21      1.00     0.7756581 0.02000000     0.000000000      0.98989899
```

Now, the KNN model is fit on the test data set using the optimal k and cut-off from the training set.

```r
set.seed(314)

knn_loan_test <- kknn(loan_default ~ fico_score + highest_ed_level + us_region_residence,
                      train = loan_training,
                      test = loan_test,
                      k = 28, distance = 2)
```

```r
#The results table is made.
knn_results_test <- data.frame(loan_test,
                               knn_pred_0.5 = knn_loan_test$fitted.values,
                               knn_loan_test$prob)

knn_results_test <- knn_results_test %>% mutate(knn_pred_0.4 = ifelse(Yes >= 0.4, "Yes", "No"))

knn_results_test %>% dplyr::select(loan_default, knn_pred_0.4, Yes, No) %>% slice(1:10)
```

```
##    loan_default knn_pred_0.4         Yes         No
## 1            No           No 0.094580808 0.90541919
## 2            No           No 0.249267718 0.75073228
## 3            No           No 0.000000000 1.00000000
## 4            No           No 0.009844126 0.99015587
## 5            No           No 0.100234086 0.89976591
## 6           Yes          Yes 0.917649374 0.08235063
## 7            No           No 0.000000000 1.00000000
## 8           Yes          Yes 0.846158723 0.15384128
## 9            No           No 0.207715990 0.79228401
## 10           No           No 0.014418174 0.98558183
```

A confusion matrix is made to determine the F1 score and the number of False Negative observations.

```r
cf_matrix(actual_vec = knn_results_test$loan_default,
          pred_prob_vec = knn_results_test$Yes,
          positive_val = "Yes",
          cut_prob = 0.4)
```

```
## $confusion_matrix
##           metric observations       rate pct_total_obs
## 1        Correct          931 0.82829181    0.82829181
## 2  Misclassified          193 0.17170819    0.17170819
## 3  True Positive          126 0.49802372    0.11209964
## 4  True Negative          805 0.92422503    0.71619217
## 5 False Negative          127 0.50197628    0.11298932
## 6 False Positive           66 0.07577497    0.05871886
##
## $F1_summary
##      metric     value
```

```
## 1 Precision 0.6562500
## 2    Recall 0.4980237
## 3  F1 Score 0.5662921
```

Results:

Training - F1 = 0.714 False Negative = 0.325

Test - F1 = 0.566 False Negative = 0.50

Summary of Findings and Recommendations

Training F1 Score False Negative Rate Random Forest 0.738 0.152 Decision Tree 0.646 0.448 KNN 0.714 0.325

Test F1 Score False Negative Rate False Negative Percent of Observations Random Forest 0.567 0.399 9% Decision Tree 0.516 0.588 13% KNN 0.566 0.500 11%

The Exploratory Data Analysis was successful in demonstrating what interactions existed in the data set. Findings from the EDA were then supported by the Gini index. Although the EDA showed that variables like adjusted_annual_inc and age_category have a relationship with loan_default, the model appeared to make accurate decisions without these variables. By only using fico_score, highest_ed_level, and us_region_residence, the model remained simple and relatively accurate.

The classification model that provided the best results was the Random Forest.

```r
cf_matrix(actual_vec = loan_rf_test_results$loan_default,
          pred_prob_vec = loan_rf_test_results$Yes,
          positive_val = "Yes", cut_prob = .1)
```

```
## $confusion_matrix
##           metric observations      rate pct_total_obs
## 1        Correct          892 0.7935943    0.79359431
## 2  Misclassified          232 0.2064057    0.20640569
## 3  True Positive          152 0.6007905    0.13523132
## 4  True Negative          740 0.8495982    0.65836299
## 5 False Negative          101 0.3992095    0.08985765
## 6 False Positive          131 0.1504018    0.11654804
##
## $F1_summary
##      metric     value
## 1 Precision 0.5371025
## 2    Recall 0.6007905
## 3  F1 Score 0.5671642
```

The Random Forest model has the highest F1 score and the lowest False Negative Rate. As a bank, the situation that would result in the highest risk is predicting that a customer will not default on their loan; however, the customer actually defaults on their loan. This situation can be observed in the model as the False Negative. The Random Forest model has the lowest False Negative Rate. Out of the total observations, the Random Forest model predicted 9% False Negatives. 79% of the observations were classified correctly.

The results of the Random Forest model are a good start for the bank. In order to improve predictions, it would be best if the bank could provide even more data. Perhaps there are also unknown variables that are important in predicting if a customer will default. In the meantime, the bank now has a fairly accurate way to predict default rates.