MIS431 - Final Project

[redacted]

1. Introduction

In this report, we are performing an analysis for LendingClub as to which factors they should choose when approving a three-year loan to their customers. We will be analyzing LendingClubs various variables in order to guide them in the right direction using Exploratory Analysis and Machine Learning models. The issue we are attempting to figure out for LendingClub is what exactly might be causing borrowers to default on their loans; And how we can help Lending Club pinpoint the best explanitory variables to make their decision on approving a loan. The importance of this report will be shown in this report, for which our analysis can save LendingClub a lot of money from preventing in giving out risky loans.

Before we dive into the report, these are the questions we have asked ourselves and are going to be answering in the Exploratory Data Analysis section.

1.) What is the distribution of the Debt-to-Income ratio when organized by NotFullyPaid and CreditPolicy?

2.) What are the basic metrics of the Debt to Income ratio based on CreditPolicy and NotFullyPaid?

3.) What is the relationship between ones Fico Credit Score and the Interest Rate that they are charged?

4.) What sort of insights can you get with summary statistics on grouped CreditPolicy and NotFullyPaid variables?

5.) What is the distribution of the level of delinquency on payment?

6.) Which category of loan purposes have the highest customer average grouped by NotFullyPaid?

These questions are an important part of our analysis and we will prove why they matter in order to make a better decision in the loan approval process.

2. Exploratory Data Analysis

2.1) Effect of meeting requirements to Debt/Income ratio

What is the distribution of the Debt-to-Income ratio when organized by NotFullyPaid and CreditPolicy?

Data

In this section, based on Figure 1, we are using Debt to Income ratios grouped by loans that are not fully paid and meeting of credit policy. The reason why we chose these variables is because we would like to see the distribution of the Debt-Income ratios across the two categories of NotFullyPaid and MeetsRequirements.

Method

In order to extract full insights from this data, we convert the binary variables into text variables. From there, this allows simplicity in building the chart as it is less confusing as seeing 0's and 1's. Additionally one can interpret the graph more clearly as shown in Figure 1.

Analysis

From Figure 1, we can clearly see the distribution of the Debt/Income ratios across the chosen categories. The distribution in Meets Requirements and Fully Paid loans shows the largest distribution for payback values. The data is not quite normally distributed however, as the left half of it looks like a cosine wave. It does have somewhat of a right skewedness to it though. Additionally, the other categories do not have nearly as high of a distribution.

Results

It comes as no surprise that the data in Figure 1 only shows large distribution values for those who pay off their loans and meet credit requirements. These customers are the ones who are likely to get approved for their next loan since they have a good payoff history. For those in the other three categories shown in Figure 1, they prove that they are not responsible in paying off their debt and should not be approved for another loan as the low distribution numbers show. The primary importance of this chart for LendingClub is that they can see patterns in which customers will have a greater chance of paying off their loans.

2.2) Metric Ratings of Credit Policy and Payback

What are the basic metrics of the Debt to Income ratio based on CreditPolicy and NotFullyPaid?

Data

We now look at Table 1 in order to get insights as to what the Debt to Income ratio metrics will be. The reasoning behind this table is to showcase if there is a possible confidence interval of each category which may help explain NotFullyPaid customers.

Method

In order to create this table, we group the data by credit policy and not fully paid. We then calculate the summary statistics giving us the average debt to income ratio and the standard deviation of the debt to income ratio based on the two grouped categories. We convert the two categories dummy variables to clear text labels which allows us to view the table efficiently.

Analysis

Table 1 shows us the general idea as to how current decisions are made in regards to allowing approval for a customers loan based on the Debt to Income ratio. The average DTI ratio for those who meet requirements and fully pay off their loan is 12.23. 6696 customers from the database fall in this category. For those who meet requirements but do not pay off their loan have a slightly higher DTI ratio of 12.73. For the categories Doesn't Meet Requirements, the ones that fully pay their loans have an average DTI of 13.79. Those who do not fully pay off their loan have an average DTI of 14.10.

Results

Table 1 shows us that those who do not meet requirements tend to have a higher Debt to Income ratio. Additionally, they have a higher standard deviation as well. It makes sense that the ones who do not meet CreditPolicy requirements have higher DTI ratios because they most likely borrow money to payoff money or something of that sort. This dataset however shows us that the lowest average DTI ratio in the table to be the ones who fully pay off their loans and meet credit requirements. This is important to LendingClub because they can see which borrowers with a particular Debt/Income ratio will have a greater chance of paying their loans off.

2.3) Relationship between Fico Score and Interest Rates

What is the relationship between ones Fico Credit Score and the Interest Rate that they are charged?

Data

Figure 2 shows us the relationship between Fico Credit Score and Interest Rate. The reasoning behind this is to study the relationship between the two categroies and see if their relationship is similar or different. Additionally we would like to showcase how high the Interest Rate goes based on ones Fico Credit Score.

Method

In Figure 2, we set the X-axis to be equal to the Fico Credit Score and the Y-axis equal to the charged Interest Rate. Figure 2 is then split amongst the categories of loans not fully paid off, with "No" being fully paid and "Yes" being unpaid.

Analysis

We can observe that those who have not fully paid off their loans, and those that have show similar relationships. The key difference is that there are many more points in the plot regarding those that have paid off their loans. The data in those who have fully paid off their loans are skewed more towards the left, including some outliers who get approved for very low interests even though they have a credit score under 700. For those who do not pay off their loans fully, their data is skewed more towards the right where there are not as many of those who get lower interest rates.

Results

Both graphs in Figure 2 have a negative linear relationship as expected. The higher the credit score, the lower the interest rate regardless of being able to pay back the debt. This may be something to look at in your approval process as there should be a general higher interest rate for those who do not fully pay off their previous loans. This chart is important to LendingClub because it will show them a conformation that Fico Scores and Interest Rates are negatively correlated.

2.4) Credit Policy Metrics

What sort of insights can you get with summary statistics on grouped CreditPolicy and NotFullyPaid variables?

Data

We showcase Table 2 in order to see the average fico score and the average interest rate grouped together and test there to be any possible relationships between the two. Additionally we are looking to extract any insights as to which criteria is the best to follow.

Method

After the data is grouped for the two categories, we calculate the average Fico score and the average interest rate charged. We group them in Table 2 to show a clear relationship between the variables, along with naming our dummy variables into categorical variables.

Analysis

From Table 2, we can see that the average Fico credit score for individuals who meet credit policy criteria to be 719. Additionally the table shows us that this category of customers have the lowest average interest rate with a value of 11.73%, which is the lowest on the chart. Those who meet criteria but do not fully pay off thier loan have a lower average credit score and a higher average interest rate at 12.8%. The customers who do not meet the credit policy criteria have even lower credit scores less than 690 and higher average interest rates.

Results

In Table 2, we learn that those who pay off their loans on time will on average have a lower interest rate and a higher credit score. Those who do not pay off thier loans will be given a higher average interest rate and a lower credit score. The best insights from this model is to follow the credit policy category in deciding on whether to approve or not approve a loan. This table is important to LendingClub because it will give them a good guide as to what level of Fico credit score they should be willing to accept in order to approve the customer.

2.5) Analysis of Delinquency on Payment

What is the distribution of the level of delinquency on payment?

Data

In Figure 3, we showcase data on the delinquency on payment and those who pay off their loans vs. those who do not. The goal of Figure 3 is to provide insights as to which purpose of taking on the debt is the best category for approval.

Method

The data shown in Figure 3 is plotted based on the different purposes of the debt. The data is organized by its X-axis being the level of delinquency (the number of times the borrower has been 30+ days past due on a payment for the past two years) and the Y-Axis is the number of total delinquent customers in each category. The data is plotted into box plots in order to gain insights of the distribution.

Analysis

All_Other and Debt_Consolidation and Credit_Card have the highest count for delinquencies being that they have paid off their loans. Additionally, these two variables have the highest numbers of fail to payoff in a 30 day period past due. The rest of the categories have lower amounts of unpaid off loans for 30 days, with a lower count of delenquencies. The educational category has the lowest amount of the number of times that they have been 30+ days past due.

Results

Those with higher levels of being past due for 30+ days have the highest count of delinquencies. This is a good reasoning because it makes sense that those who do not pay off their loans are unlikely to be less delinquent. Categories such as educational, home_improvent, major_purchase, and small_business are all of the categories who do not have a high level of being past due. It is explainable that these categories have a lower level of delinquency. This chart serves as an important indicator to LendingClub on what loans to approve to a customer based on their missed payoff history.

2.6) Summary of each credit purpose

Which category of loan purposes have the highest customer average grouped by NotFullyPaid?

Data

In Table 3, we showcase our NotFullyPaid loans along with the purpose of taking out a loan. This table is designed to show any trends between the categories presented in order to extract an insightful relationship and to figure out which category is the highest.

Method

Table 3 was built to showcase how many customers are approved for a loan based on their NotFullyPaid and Credit Purpose categories. The columns are grouped by NotFullyPaid and Purpose. Additonally, the average number of customers is calculated using the mean function for each group presented.

Analysis

In the category No for Not Fully Paid, we see that the highest approved customers are on average 335 for debt consolidation and all other variables. The lowest approved are for educational, home_improvement, major_purchase, and small_business. In the category with Yes for Not Fully Paid we see that debt consolidation holds the crown for the highest average customers approved. The rest of the variables however do not have a high approval average.

Results

Using Table 3, our question for this section can be answered as debt consolidation being the highest average approval for those categories. Debt consolidation lump sums your credit into a single payment, which shows the true meaning of this data. It makes sense for people who want to erase their debt in one fail swoop. Debt consolidation additionally has the highest average customers approved for those who have not fully paid off their loans. This table is valuable to LendingClub because it will show them which categories they have approved based on debt purpose, which they can then tune to their future customers.

3. ML Model Building Exercise

ML Metrics Table

```
data.frame(ModelUsed=c("LogReg","QDA","KNN"), F1Measure=c(0.7451,0.6681,0.7856),AUC_ROC=c(0.6452,0.6355
```

```
##   ModelUsed F1Measure AUC_ROC Precision Sensitivity Specificity
## 1    LogReg    0.7451  0.6452    0.8857      0.6431      0.6431
## 2       QDA    0.6681  0.6355    0.8899      0.5348      0.6536
## 3       KNN    0.7856  0.5744    0.8576      0.7247      0.3698
```

Methods Explained

The Logistic Regression is an algorithm which splits the data into two categories. The Quadratic Discriminant Analysis (QDA) groups the data in order to find the optimal clusters to effectively classify if one is able to be approved for a loan. The KNN is a model which gathers values that are near a calculated point and groups them into clusters using the distance formula in order to classify the loan customer.

F1-Measure Score

Based on our F1 measure, the perfect model to classify those who pay off their loans and those that do not is the KNN algorithm. The KNN algorithm has the highest F1-Measure, which is a statistic that tells how much the overall performance is; The F1-Measure is between 0 and 1. The Logistic Regression model comes in second place and the QDA model comes in third place. So far the KNN has the best performance based on the F1 score.

Area under ROC Curve / ROC Curve

In terms of the grade calculated via the ROC curve area, the Logistic Regression model has the highest grade while the KNN model has the lowest grade. What this means is that in this section, the Logistic Regression model has the highest performance in terms of classifying loans that are not fully paid. In terms of the grade for all three models, LogReg and QDA's grades are under 0.70 so they both recieve a "D" grade rating while the KNN recieves an F rating since its grade is lower than 0.6.

When taking a look at the plots of the ROC Curve, the QDA model has the best ROC curve as its curve has the highest slope and highest specificity, meaning that it will have the highest area and highest grade of a B. Second comes the Logistic Regression model which also earns a B, and third comes in the KNN model which also earns a B.

Confusion Matrix

The sensitivity metric tells us how good a test is at detecting positives. The sensitivity score for the KNN model has the highest sensitivity out of the three models. What this means is that the KNN model is highly accurate at detecting positives, which in this case are those who have not paid off their loans. The Logistic Regression model comes in second and the KNN model comes in third.

The specificity test is a test which is good at avoiding false alarms. The KNN model does not perform as well in the specificity category. The QDA model takes the crown for having the highest specificity rate. The Logistic Regression model comes in second and the KNN model comes in third.

The precision test is a test which shows how many positively classified were relevant. The table shows us that all of the models have a similar precision level give or take a few decimal places. All of the models work correctly in making sure that everything in the calculations is relevant. In terms of ranking them, QDA comes first, Logistic Regression comes in second, and the KNN comes in third.

VIP Interpretations

As shown in the Logistic Regression VIP Graph, the most important explanitory variables we have generated are LogAnnualIncome and Purpose_credit_card. These two variables take up most of the data required to make an accurate prediction. However, we did not choose these as we feel that the LogAnnualIncome serves as a good predictor because the logarithm scale throws our data off. The Purpose variable however is shown in Figure 3 as an accurate measure of classifying approved loans and non-approved loans.

Model Recommendations

Based on our results shown with our three models, the best model to use for deciding on whether to approve or disapprove a customers loan application is the KNN model. The reason why is due to the fact that the area under the ROC curve is invalid as all three models scored a D or F in them. The KNN has the highest F1 score (think of R^2) and the highest Sensitivity score. However, since the dataset is larger, it is optimal to use the Logistic Regression model instead.

4. Recommendations

In response to LendingClub's first question, based on the data examined in this report, we would highly recommend that you take a look at the variables Debt/Income ratio, Credit Policy, and Delenquency. These three variables would serve well in your consideration as to approve or disapprove their loan. Additionally, you should take a look at Log Annual Income and Purpose_credit_card since they are highly ranked in our VIP model. These variables are shown to be the most significant variables in our KNN Model as effective methods to use for classification of the loan non-paid and paid customers. Unfortunately we do not have a method to view the best variables in the QDA and KNN models. This is a good recommendation because you now have an insight as to which variables are most effective. This section will benefit LendingClub in a way that their business can be impacted knowing the correct method to classify a loan.

In response to LendingClub's second question, the best model use to classify NotFullyPaid effectively, we recommend using the KNN. It has the greatest F1 score. Since the Logistic Regression model does not have the highest sensitivity, but has a specificity and precision, you may want to take a look at the Logistic Regression model depending on your requirements. This is a good recommendation to you because you can now maybe even add additional categories you wish to test and you will get the most accurate result from the Logistic Regression model since you have a large dataset. This will also benefit the business because it will allow them to access machine learning while their competitors may not be so technical.

5. Conclusion

In conclusion, we hope you will take the listed variables into consideration when faced with approving or disapproving a customers loan application. The data your team provided has had much insights inside it. Some of the variables may not be as best at using them in a different scenario, but our overall report overcame those challenges. Now we will dive into a summary of both our explanatory section and our machine learning section.

When we picked our variables to generate the tables and charts located in Appendix/Appendices, it required great intuition as to select which variables made most sense in gaining insights from the patterns in it. The selection process took great analytical thinking and common sense. We did not know at first if it would have generated us those beautiful charts, that was the selection process. We believe that out of Figure 1, 2, & 3, Figure 3 showcases the best metrics in the boxplot because it takes into account the type of loan (i.e. CreditCard, Debt Consolidation, and Home Improvement).

In terms of gathering the right steps to execute our Machine Learning model, we were able to use all of the variables in the dataset. The most valuable variables we found were LogAnnualIncome, InquiriesLast6Months, Debt/Income ratio, CreditPolicy, and Purpose. These several variables will give you a great insight as to what criteria you wish to set for approving or disapproving a loan.

Overall the initial analysis held very well in our report. The questions we asked proved to be important ones which the Figures and Tables in the appendix section greatly emphasized. The tests we ran on our classification models also proved to point LendingClub in the correct direction. We hope that you use the recommended variables and models highlighted in our report in order to have the best approval criteria for the NotFullyPaid section of the dataset. We thank you for reading our report and hope that it will prevent you from giving a delinquent debt non-paying client a loan.

6. Appendix/Appendices

```r
## Add R libraries here
library(tidyverse)
library(tidymodels)
library(ggplot2)
library(dplyr)
library(discrim)
library(skimr)
library(klaR)
library(kknn)
library(vip)
library(themis)


# Load data
loans_df <- read_csv("/cloud/project/Datasets/Loans.csv") %>%
  mutate(NotFullyPaid = as.factor(if_else(NotFullyPaid == 1,"Yes","No")))
```

Visuals

```r
# Creating a table named payback based on CreditPolicy, NotFullyPaid, and Debt to Income Ratio
payback <- loans_df %>%
  mutate(CreditPolicy=factor(CreditPolicy, levels=c(1, 0), labels=c("Meets Requirements","Doesn't Meet 
  group_by(CreditPolicy, NotFullyPaid, Dti) %>%
  dplyr::select(CreditPolicy, NotFullyPaid, Dti)
```

```r
# Creating a grid histogram plot relating to CreditPolicy, NotFullyPaid, and Debt to Income Ratio
payback %>%
ggplot(mapping=aes(x=Dti)) +
  geom_histogram(bins=150) +
  facet_grid(NotFullyPaid ~ CreditPolicy) +
  labs(title="Figure 1", x="Debt To Income Ratio",y="Count") +
  theme(plot.title = element_text(hjust = 0.5))
```

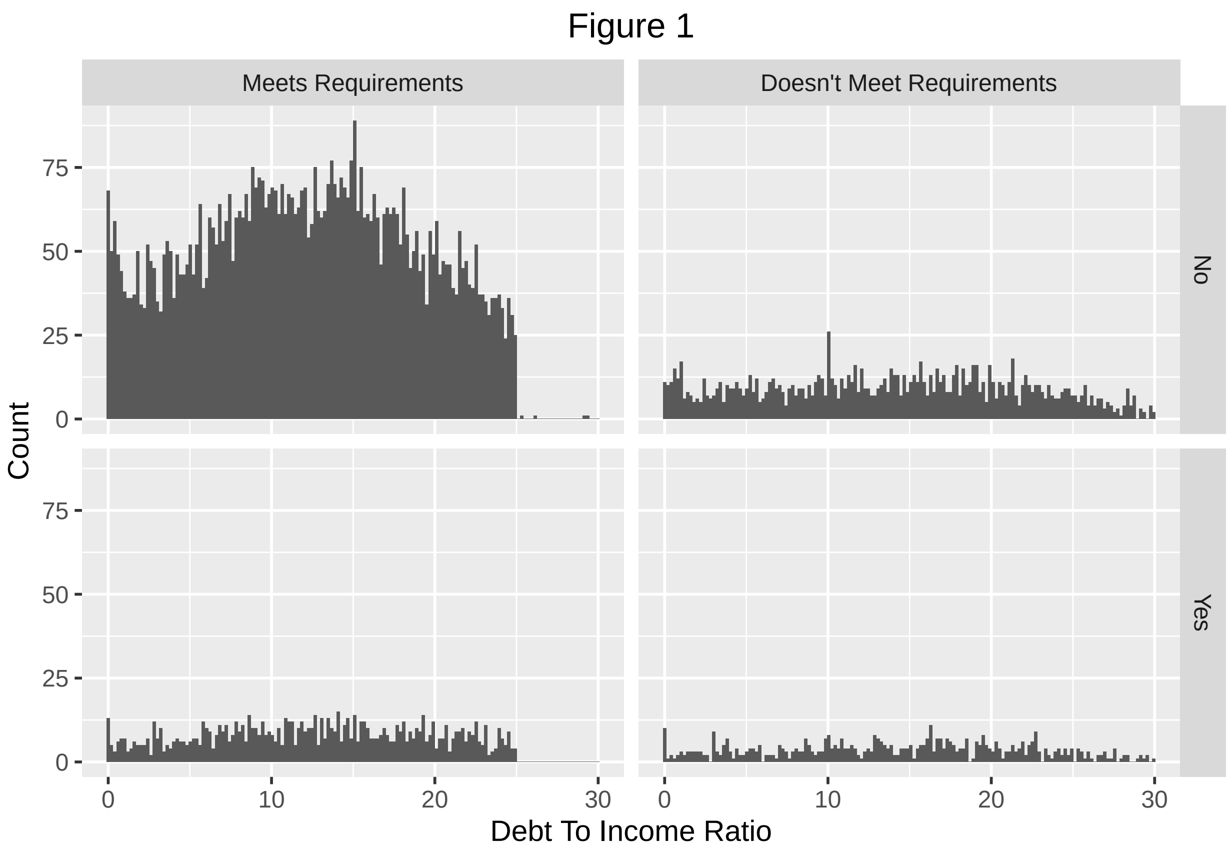


Table 1

```r
# Creating a table of the summarized results of Figure 1
pb <- payback %>%
  group_by(CreditPolicy, NotFullyPaid) %>%
  summarise(mean_dti=mean(Dti), sd_dti=sd(Dti), total=n()) %>%
  dplyr::select(CreditPolicy, NotFullyPaid, mean_dti, sd_dti,total)
```

```
## `summarise()` has grouped output by 'CreditPolicy'. You can override using the
## `.groups` argument.
```

```r
pb
```

```
## # A tibble: 4 x 5
## # Groups:   CreditPolicy [2]
##   CreditPolicy             NotFullyPaid mean_dti sd_dti total
##   <fct>                    <fct>           <dbl>  <dbl> <int>
## 1 Meets Requirements       No               12.2   6.62  6696
## 2 Meets Requirements        Yes              12.7    6.68  1014
## 3 Doesn't Meet Requirements No               13.8    7.80  1349
## 4 Doesn't Meet Requirements Yes              14.1    7.53   519
```

```
# Creating a data table out of CreditPolicy, NotFullyPaid, Credit Score, and Interest Rate
score <- loans_df %>%
         dplyr::select(CreditPolicy, NotFullyPaid, Fico, IntRate)
```

```
# Creating a scatter plot of the Credit Score against the Interest Rate
score %>% ggplot(mapping=aes(x=Fico, y=IntRate)) +
  geom_point(color='blue') +
  facet_wrap(~ NotFullyPaid) +
  labs(title="Figure 2", x="FICO Credit Score", y="Interest Rate") +
  theme(plot.title = element_text(hjust = 0.5))
```

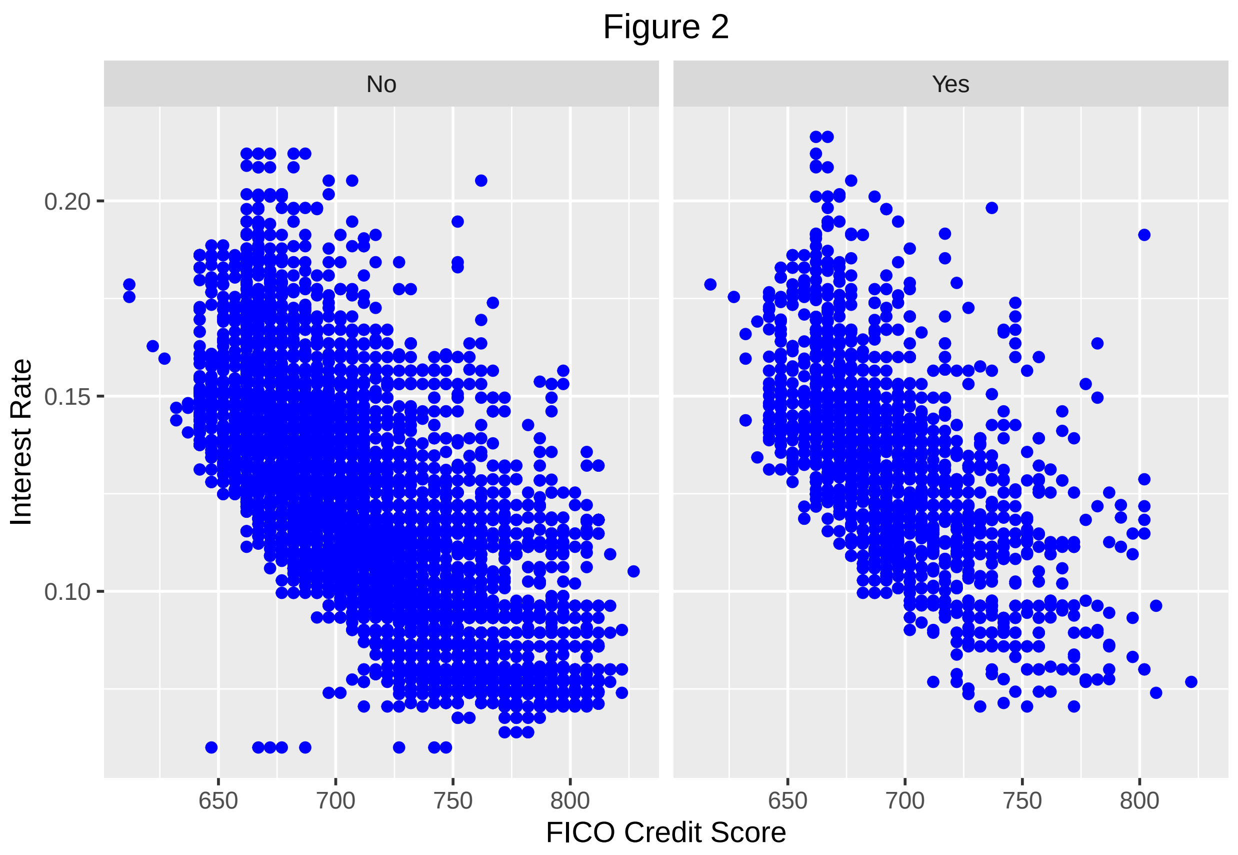


Table 2

```
# Creating a table summarizing the average Credit Score and the average Interest Rate
score %>%
  mutate(CreditPolicy=factor(CreditPolicy, levels=c(1, 0), labels=c("Meets Criteria","Does not meet Cri
  group_by(CreditPolicy, NotFullyPaid) %>%
  summarise(avg_fico_score = mean(Fico), avg_interest_rate=mean(IntRate))
```

```
## `summarise()` has grouped output by 'CreditPolicy'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 4 x 4
## # Groups:   CreditPolicy [2]
##   CreditPolicy           NotFullyPaid avg_fico_score avg_interest_rate
##   <fct>                  <fct>                 <dbl>             <dbl>
## 1 Meets Criteria         No                     719.             0.117
## 2 Meets Criteria         Yes                    707.             0.128
## 3 Does not meet Criteria No                     685.             0.138
## 4 Does not meet Criteria Yes                    680.             0.141
```

```
# Creating a sub table based on delinquency and credit purpose
deal <- loans_df  %>%
        dplyr::select(NotFullyPaid, Delinq2yrs, Purpose)
```

```
# Creating a summary table off of delinquency and credit purpose
delinq <- deal %>%
  group_by(NotFullyPaid, Delinq2yrs, Purpose) %>%
  summarise(total=n())
```

```
## `summarise()` has grouped output by 'NotFullyPaid', 'Delinq2yrs'. You can
## override using the `.groups` argument.
```

```
# Creating a grid boxplot chart based on credit purpose, delinquency, and NotFullyPaid
delinq %>%
  ggplot(mapping=aes(x=Delinq2yrs, y=total, fill=NotFullyPaid)) +
  geom_boxplot() +
  scale_x_log10() +
  scale_y_log10() +
  facet_wrap(~ Purpose) +
  labs(title="Figure 3", x="Delinquency on Payment", y="Total Count of Delinquency") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 3

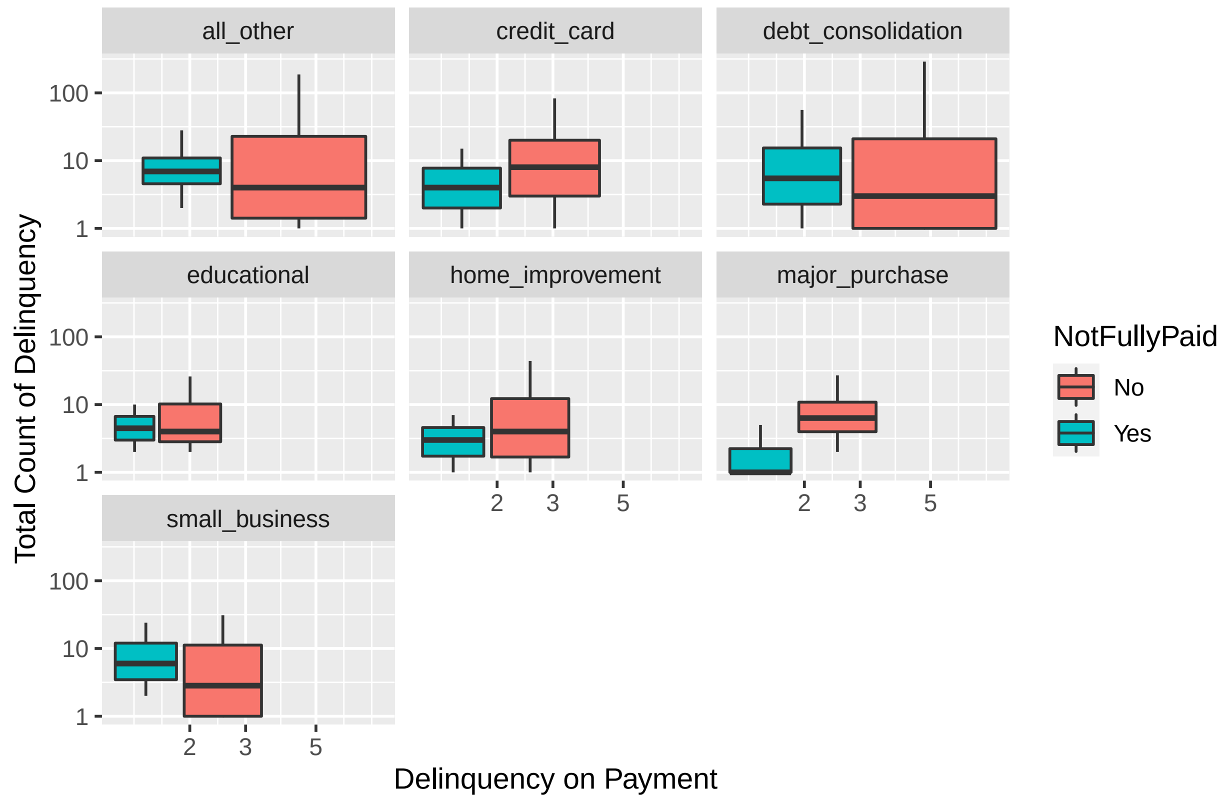


Table 3

```r
# Creating a summary table for the average total customers based on delinquency
delinq %>%
  group_by(NotFullyPaid, Purpose) %>%
  dplyr::select(NotFullyPaid, Purpose, total) %>%
  summarise(avg_customers=mean(total))
```

```
## `summarise()` has grouped output by 'NotFullyPaid'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 14 x 3
## # Groups:   NotFullyPaid [2]
##    NotFullyPaid Purpose            avg_customers
##    <fct>        <chr>                      <dbl>
##  1 No           all_other                  243
##  2 No           credit_card                186
##  3 No           debt_consolidation         335.
##  4 No           educational                 68.5
##  5 No           home_improvement           104.
##  6 No           major_purchase              77.6
##  7 No           small_business              89.4
##  8 Yes          all_other                   77.4
##  9 Yes          credit_card                 36.5
## 10 Yes          debt_consolidation         121.
## 11 Yes          educational                 23
## 12 Yes          home_improvement            26.8
## 13 Yes          major_purchase              12.2
## 14 Yes          small_business              43
```

Machine Learning Models

Train/Test Split

```
# Set seed to G number
set.seed(00800533)

# Split the data into training and testing sets
loans_split <- initial_split(loans_df, prop=0.75, strata=NotFullyPaid)

# Data splitting into training and testing data frames
loans_training <- loans_split %>% training()
loans_testing <- loans_split %>% testing()
```

Cross-Validation

```
# Set seed to G number
set.seed(00800533)

# Create a cross validation set
loan_folds <- vfold_cv(loans_df, v=5)
loan_folds
```

```
## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits              id
##   <list>              <chr>
## 1 <split [7662/1916]> Fold1
## 2 <split [7662/1916]> Fold2
## 3 <split [7662/1916]> Fold3
## 4 <split [7663/1915]> Fold4
## 5 <split [7663/1915]> Fold5
```

Feature Engineering

```
# Creating a recipe for our feature engineering section, this will be inputted into all of the classifi
loans_recipe <- recipe(NotFullyPaid ~ ., data=loans_training) %>%
  step_YeoJohnson(all_numeric(), -all_outcomes()) %>%
  step_normalize(all_numeric(), -all_outcomes()) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_smote(NotFullyPaid,over_ratio = 1)
loans_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         13
##
## Operations:
##
## Yeo-Johnson transformation on all_numeric(), -all_outcomes()
## Centering and scaling for all_numeric(), -all_outcomes()
## Dummy variables from all_nominal(), -all_outcomes()
## SMOTE based on NotFullyPaid
```

Check Transformations

```r
# Prep and bake training dataframe
loans_recipe %>%
  prep() %>%
  bake(new_data=loans_training)
```

```
## # A tibble: 7,182 x 19
##    CreditPolicy  IntRate Insta~1 LogAnn~2    Dti    Fico DaysW~3 Revol~4 Revol~5
##           <dbl>    <dbl>   <dbl>    <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1        0.496 -0.114     1.88   0.690    0.987  0.769   0.594    0.965   0.278
##  2        0.496 -0.811    -0.689  0.690   -0.591  0.148  -0.686    1.11    0.898
##  3        0.496  0.752    -1.21   0.606    0.393 -1.23   -0.0176  -0.344  -0.130
##  4        0.496 -1.71     -0.997  1.60     0.662  0.531   0.759    1.51    0.244
##  5        0.496 -0.322    -1.39   0.784    0.698 -0.729  -0.0511   1.85    0.247
##  6        0.496  0.00625  -1.42  -1.19    -0.306  0.0124 -0.670   -0.242  -0.732
##  7        0.496  0.469     0.417 -0.811    1.32  -0.892   0.951    0.364   0.836
##  8        0.496  0.386    -0.102  1.49    -0.430 -1.41    0.0810  -0.299  -0.929
##  9        0.496 -1.41      0.217  0.00536  0.464  1.41    0.890   -0.197  -0.994
## 10        0.496 -2.02     -1.32   0.957   -0.841  0.993   0.117   -0.592  -1.60
## # ... with 7,172 more rows, 10 more variables: InqLast6mths <dbl>,
## #   Delinq2yrs <dbl>, PubRec <dbl>, NotFullyPaid <fct>,
## #   Purpose_credit_card <dbl>, Purpose_debt_consolidation <dbl>,
## #   Purpose_educational <dbl>, Purpose_home_improvement <dbl>,
## #   Purpose_major_purchase <dbl>, Purpose_small_business <dbl>, and abbreviated
## #   variable names 1: Installment, 2: LogAnnualInc, 3: DaysWithCrLine,
## #   4: RevolBal, 5: RevolUtil
```

Confusion Matrix Analytic Functions

```r
# Calculates the specificity from the confusion matrix
SPECIFICITY <- function(x) {
  H <- x$table
  TP <- H[1]
  FN <- H[2]
  FP <- H[3]
  TN <- H[4]
  return(TN/(TN + FP))
}

# Calculates the sensitivity from the confusion matrix
SENSITIVITY <- function(x) {
  H <- x$table
  TP <- H[1]
  FN <- H[2]
  FP <- H[3]
  TN <- H[4]
  return(TP/(TP+FN))
}

# Calculates the precision from the confusion matrix
PRECISION <- function(x) {
  H <- x$table
  TP <- H[1]
  FN <- H[2]
  FP <- H[3]
  TN <- H[4]
  return(TP/(TP+FP))
}
```

Logistic Regression Model

Specify Logistic Regression Model

```r
logistic_model <- logistic_reg() %>%
                  set_engine('glm') %>%
                  set_mode('classification')

logistic_model
```

```
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

Create a Logistic Regression Workflow

```r
logistic_wf <- workflow() %>%
               add_model(logistic_model) %>%
               add_recipe(loans_recipe)

logistic_wf
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_YeoJohnson()
## * step_normalize()
## * step_dummy()
## * step_smote()
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

Fit Logistic Regression Model

```
logistic_fit <- logistic_wf %>%
                fit(data=loans_training)
logistic_fit
```

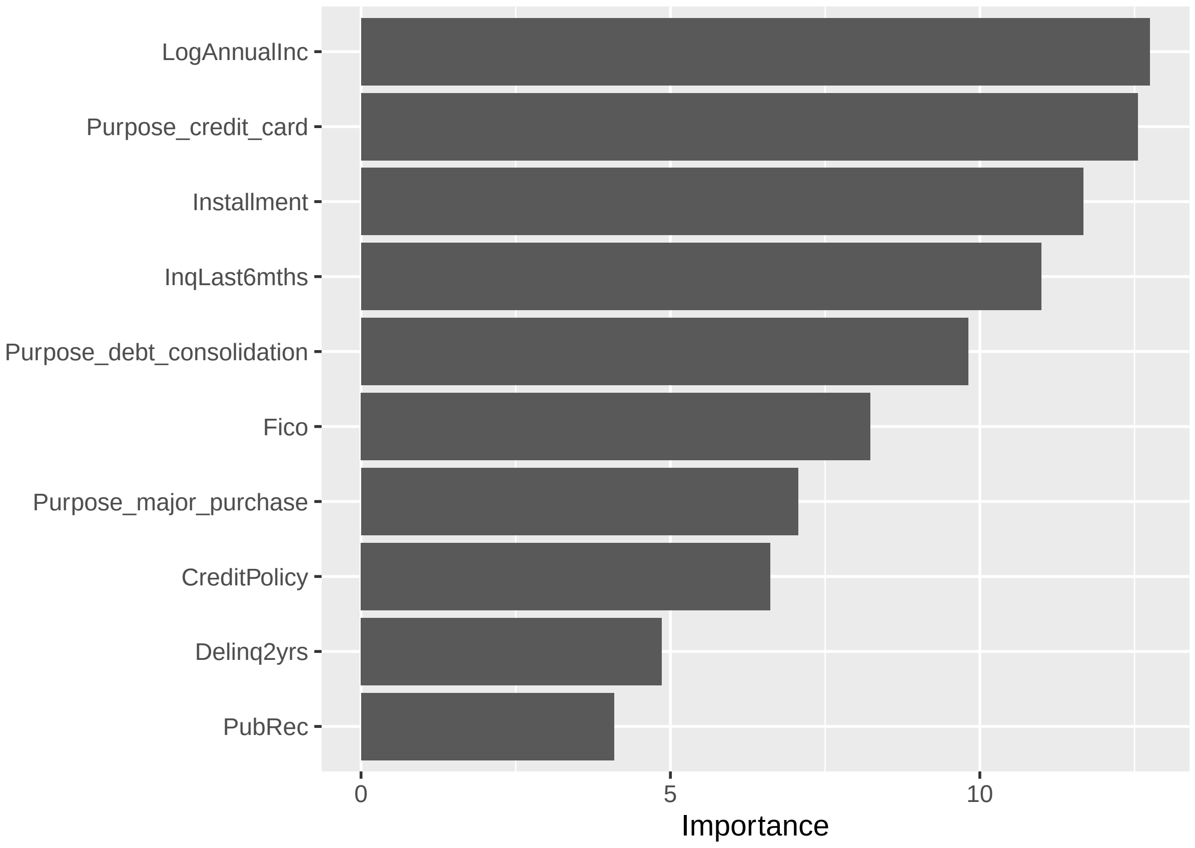
```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_YeoJohnson()
## * step_normalize()
## * step_dummy()
## * step_smote()
##
## -- Model -----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##                (Intercept)                CreditPolicy
##                    0.20265                    -0.14753
##                    IntRate                 Installment
##                    0.07713                     0.31846
##               LogAnnualInc                         Dti
##                   -0.34551                     0.01100
##                       Fico               DaysWithCrLine
##                   -0.31544                     0.04643
##                   RevolBal                   RevolUtil
##                    0.11665                     0.09087
##               InqLast6mths                 Delinq2yrs
##                    0.24615                    -0.11122
##                     PubRec          Purpose_credit_card
##                    0.08097                    -0.95969
## Purpose_debt_consolidation         Purpose_educational
##                   -0.53725                    -0.26579
##   Purpose_home_improvement      Purpose_major_purchase
##                   -0.19716                    -0.86924
##     Purpose_small_business
##                    0.30316
##
## Degrees of Freedom: 12065 Total (i.e. Null);  12047 Residual
## Null Deviance:       16730
## Residual Deviance: 15200     AIC: 15240
```

Logistic Regression VIP Graph

```
lg_fit <- logistic_fit %>% pull_workflow_fit()
```

```
## Warning: `pull_workflow_fit()` was deprecated in workflows 0.2.3.
## Warning: Please use `extract_fit_parsnip()` instead.
```

```
vip(lg_fit)
```



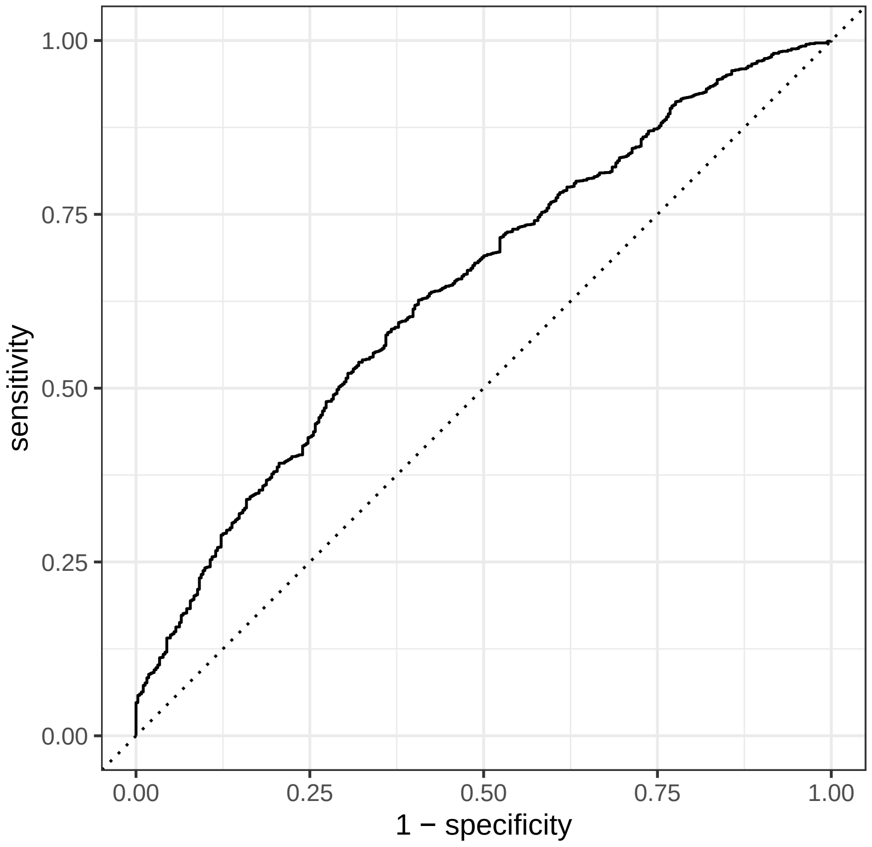
Logistic Regression Predictions

```
pred_categories <- predict(logistic_fit, new_data=loans_testing)
pred_probabilities <- predict(logistic_fit, new_data=loans_testing, type='prob')
logistic_results <- loans_testing %>%
                    dplyr::select(NotFullyPaid) %>%
                    bind_cols(pred_categories) %>%
                    bind_cols(pred_probabilities)
logistic_results
```

```
## # A tibble: 2,396 x 4
##    NotFullyPaid .pred_class .pred_No .pred_Yes
##    <fct>        <fct>          <dbl>     <dbl>
##  1 No           No             0.754    0.246
##  2 No           Yes            0.437    0.563
##  3 No           No             0.585    0.415
##  4 No           No             0.686    0.314
##  5 No           No             0.828    0.172
##  6 No           No             0.541    0.459
##  7 No           No             0.870    0.130
##  8 No           No             0.863    0.137
##  9 No           No             0.852    0.148
## 10 No           No             0.936    0.0644
## # ... with 2,386 more rows
```

Logistic Regression ROC Curve Plot

```
roc_logreg <- roc_curve(logistic_results, truth=NotFullyPaid, estimate=.pred_No)
roc_logreg %>% autoplot()
```



Logistic Regression ROC AUC

```
roc_auc(logistic_results, truth=NotFullyPaid, .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.644
```

Logistic Regression Confusion Matrix

```
logreg_cf <- conf_mat(logistic_results, truth=NotFullyPaid, estimate=.pred_class)
print(logreg_cf)
```

```
##             Truth
## Prediction   No  Yes
##        No  1295  169
##        Yes  717  215
```

Logistic Regression Confusion Matrix Metrics

```
cat("Precision: ", PRECISION(logreg_cf), "\n")
```

```
## Precision:  0.8845628
```

```
cat("Sensitivity: ", SENSITIVITY(logreg_cf), "\n")
```

```
## Sensitivity:  0.6436382
```

```
cat("Specificity: ", SPECIFICITY(logreg_cf), "\n")
```

```
## Specificity:  0.5598958
```

Logistic Regression F1-Score

```
f_meas(logistic_results, truth=NotFullyPaid, estimate=.pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 f_meas  binary         0.745
```

QDA Model

Specify QDA Model

```
qda_model <- discrim_regularized(frac_common_cov = 0) %>%
             set_engine('klaR') %>%
             set_mode('classification')

qda_model
```

```
## Regularized Discriminant Model Specification (classification)
##
## Main Arguments:
##   frac_common_cov = 0
##
## Computational engine: klaR
```

Create QDA Workflow

```
qda_wf <- workflow() %>%
          add_model(qda_model) %>%
          add_recipe(loans_recipe)
qda_wf
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: discrim_regularized()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_YeoJohnson()
## * step_normalize()
## * step_dummy()
## * step_smote()
##
## -- Model ----------------------------------------------------------------------
## Regularized Discriminant Model Specification (classification)
##
## Main Arguments:
##   frac_common_cov = 0
##
## Computational engine: klaR
```

Fit QDA Model

```
qda_fit <- qda_wf %>%
           last_fit(split=loans_split)
qda_fit
```

```
## # Resampling results
## # Manual resampling
## # A tibble: 1 x 6
##   splits              id               .metrics .notes   .predictions .workflow
##   <list>              <chr>            <list>   <list>   <list>       <list>
## 1 <split [7182/2396]> train/test split <tibble> <tibble> <tibble>     <workflow>
```

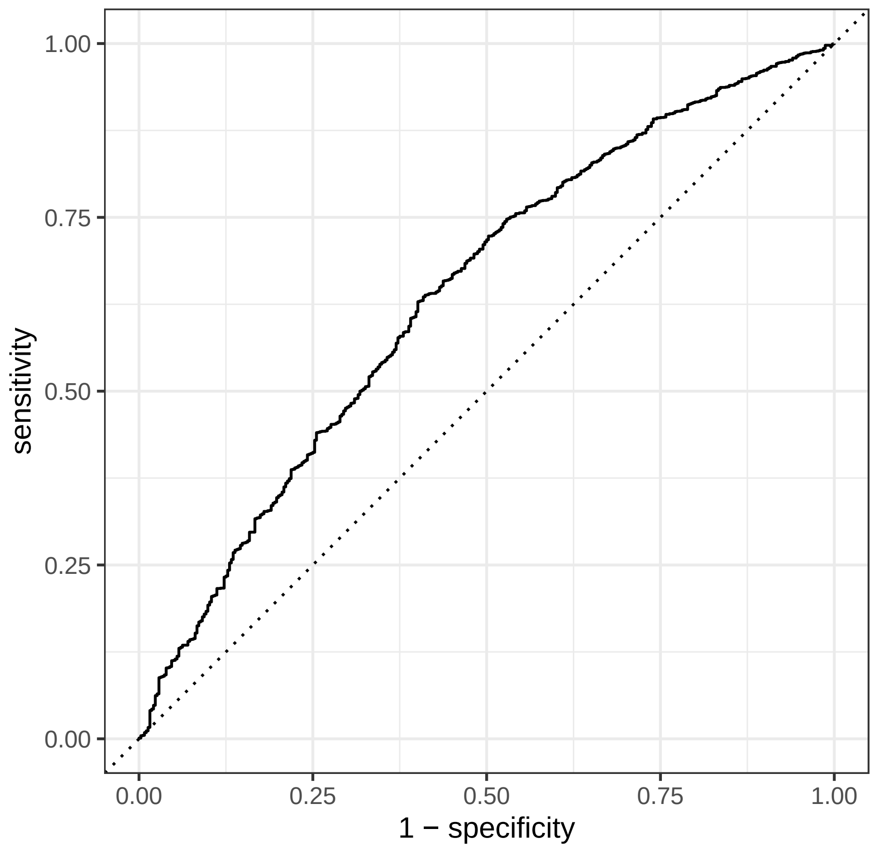
Collect QDA Predictions

```
qda_results <- qda_fit %>% collect_predictions()
qda_results
```

```
## # A tibble: 2,396 x 7
##    id               .pred_No .pred_Yes  .row .pred_class NotFullyPaid .config
##    <chr>               <dbl>     <dbl> <int> <fct>       <fct>        <chr>
##  1 train/test split    0.821   0.179       2 No          No           Preproces~
##  2 train/test split    0.415   0.585       3 Yes         No           Preproces~
##  3 train/test split    0.501   0.499      15 No          No           Preproces~
##  4 train/test split    0.720   0.280      23 No          No           Preproces~
##  5 train/test split    0.904   0.0956     24 No          No           Preproces~
##  6 train/test split    0.184   0.816      25 Yes         No           Preproces~
##  7 train/test split    0.989   0.0110     29 No          No           Preproces~
##  8 train/test split    0.980   0.0204     30 No          No           Preproces~
##  9 train/test split    0.928   0.0723     31 No          No           Preproces~
## 10 train/test split    0.993   0.00729    37 No          No           Preproces~
## # ... with 2,386 more rows
```

QDA ROC Curve

```
qda_results %>%
  roc_curve(truth=NotFullyPaid, estimate=.pred_No) %>%
  autoplot()
```



QDA ROC-AUC Area Underneath Curve

```
roc_auc(qda_results, truth=NotFullyPaid, .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.639
```

QDA Confusion Matrix

```
qda_cf_matrix <- conf_mat(qda_results, truth=NotFullyPaid, .pred_class)
qda_cf_matrix
```

```
##             Truth
## Prediction   No  Yes
##        No  1097  137
##        Yes  915  247
```

QDA Confusion Matrix Metrics

```
cat("Precision: ", PRECISION(qda_cf_matrix), "\n")
```

```
## Precision:  0.8889789
```

```
cat("Sensitivity: ", SENSITIVITY(qda_cf_matrix), "\n")
```

```
## Sensitivity:  0.5452286
```

```
cat("Specificity: ", SPECIFICITY(qda_cf_matrix), "\n")
```

```
## Specificity:  0.6432292
```

QDA F1-Score

```
f_meas(qda_results, truth=NotFullyPaid, estimate=.pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 f_meas  binary         0.676
```

KNN Model

Specify KNN Model

```
knn_model <- nearest_neighbor(neighbors = tune()) %>%
             set_engine('kknn') %>%
             set_mode('classification')
knn_model
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = tune()
##
## Computational engine: kknn
```

Create KNN Workflow

```
knn_wf <- workflow() %>%
          add_model(knn_model) %>%
          add_recipe(loans_recipe)
knn_wf
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_YeoJohnson()
## * step_normalize()
## * step_dummy()
## * step_smote()
##
## -- Model -----------------------------------------------------------------------
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = tune()
```

```
##
## Computational engine: kknn
```

Tune KNN Hyperparameters

```
gridTune <- c(10,15,25,45,60)

set.seed(00800533)

knn_tuning <- knn_wf %>%
              tune_grid(resamples=loan_folds, grid=gridTune)

knn_tuning
```

```
## # Tuning results
## # 5-fold cross-validation
## # A tibble: 5 x 4
##   splits              id    .metrics          .notes
##   <list>              <chr> <list>            <list>
## 1 <split [7662/1916]> Fold1 <tibble [20 x 5]> <tibble [0 x 1]>
## 2 <split [7662/1916]> Fold2 <tibble [20 x 5]> <tibble [0 x 1]>
## 3 <split [7662/1916]> Fold3 <tibble [20 x 5]> <tibble [0 x 1]>
## 4 <split [7663/1915]> Fold4 <tibble [20 x 5]> <tibble [0 x 1]>
## 5 <split [7663/1915]> Fold5 <tibble [20 x 5]> <tibble [0 x 1]>
```

Select Best KNN Model

```
best_k <- knn_tuning %>%
          select_best(metric='roc_auc')

best_k
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1        14 Preprocessor1_Model10
```

Finalize KNN Workflow

```
final_knn_wf <- knn_wf %>%
                finalize_workflow(best_k)

final_knn_wf
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_YeoJohnson()
## * step_normalize()
## * step_dummy()
## * step_smote()
##
## -- Model -----------------------------------------------------------------------
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 14
##
## Computational engine: kknn
```

Fit KNN Model

```
knn_fit <- final_knn_wf %>%
           last_fit(split=loans_split)

knn_fit
```

```
## # Resampling results
## # Manual resampling
## # A tibble: 1 x 6
##   splits              id               .metrics .notes   .predictions .workflow
##   <list>              <chr>            <list>   <list>   <list>       <list>
## 1 <split [7182/2396]> train/test split <tibble> <tibble> <tibble>     <workflow>
```

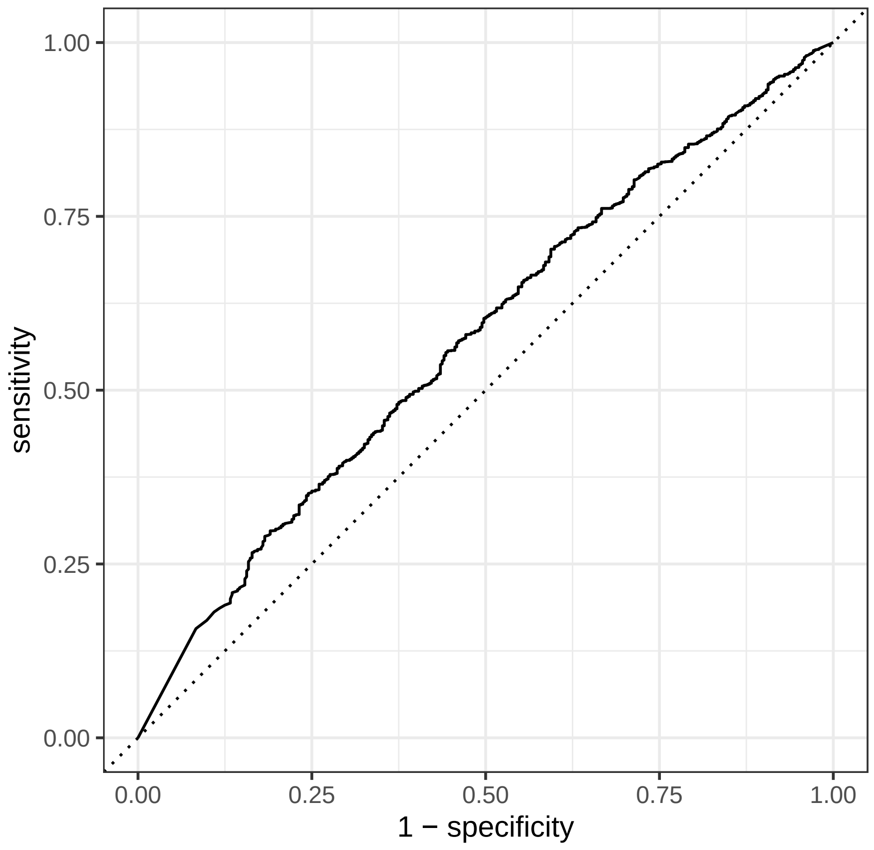
Collect KNN Predictions

```
knn_results <- knn_fit %>% collect_predictions()
knn_results
```

```
## # A tibble: 2,396 x 7
##    id               .pred_No .pred_Yes  .row .pred_class NotFullyPaid .config
##    <chr>               <dbl>     <dbl> <int> <fct>       <fct>        <chr>
##  1 train/test split    0.814    0.186      2 No          No           Preproces~
##  2 train/test split    0.482    0.518      3 Yes         No           Preproces~
##  3 train/test split    0.514    0.486     15 No          No           Preproces~
##  4 train/test split    1        0         23 No          No           Preproces~
##  5 train/test split    1        0         24 No          No           Preproces~
##  6 train/test split    0.961    0.0391    25 No          No           Preproces~
##  7 train/test split    1        0         29 No          No           Preproces~
##  8 train/test split    0.929    0.0705    30 No          No           Preproces~
##  9 train/test split    1        0         31 No          No           Preproces~
## 10 train/test split    1        0         37 No          No           Preproces~
## # ... with 2,386 more rows
```

KNN ROC Curve

```
knn_results %>%
  roc_curve(truth=NotFullyPaid, estimate=.pred_No) %>% autoplot()
```



KNN ROC AUC Area Under Curve

```
roc_auc(knn_results, truth=NotFullyPaid, .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.574
```

KNN Confusion Matrix

```
knn_cf_mat <- conf_mat(knn_results, truth=NotFullyPaid, estimate=.pred_class)
knn_cf_mat
```

```
##           Truth
## Prediction   No  Yes
##        No  1457  240
##        Yes  555  144
```

KNN Confusion Matrix Metrics

```
cat("Precision: ", PRECISION(knn_cf_mat), "\n")
```

```
## Precision:  0.858574
```

```
cat("Sensitivity: ", SENSITIVITY(knn_cf_mat), "\n")
```

```
## Sensitivity:  0.7241551
```

```
cat("Specificity: ", SPECIFICITY(knn_cf_mat), "\n")
```

```
## Specificity:  0.375
```

KNN F1-Score

```
f_meas(knn_results, truth=NotFullyPaid, estimate=.pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 f_meas  binary         0.786
```

— End of the Project —