An Analysis of Borrowers and Lending Activities

I. Introduction

As institutionalized loans become harder to avail of following the 2008 Subprime mortgage crisis, certain individuals remain deprived of the opportunity to take a personal loan despite being amply able to repay it, simply because they do not fulfill the criteria set by the banking institutions for personal loans. This has given birth to the concept of a lending club, a peer-to-peer online marketplace that allows borrowers to apply online for personal loans of up to $40,000. Lending Club allows investors to invest their capital on such applicants for a high return on investment. In this project, by analyzing its loan datasets, we can provide important indicators to investors regarding Lending Club loan applications currently under their considerations.

II. Project Objective

The purpose of this project is to help investors make wise decisions on their investments. Given the non-institutional nature of these loans, there is a risk of loss of investment involved and it would not be a wise decision for small and medium investors to invest without conducting an in-depth analysis of this investment opportunity.

LendingClub is a peer-to-peer lending site for individuals to apply for a loan that may be funded by any potential investors, including single investors or groups of founders. In order to keep track of the applicant's credit records, LendingClub needs to collect related financial information such as age, employment status, income, and borrowing purpose. These sources help

LendingClub to assess the risk and estimate if the applicant could repay the loan with a specific amount and corresponding percentage rate. Our project uses these collected sources to build predictive models and understand the significant variables that lead to default.

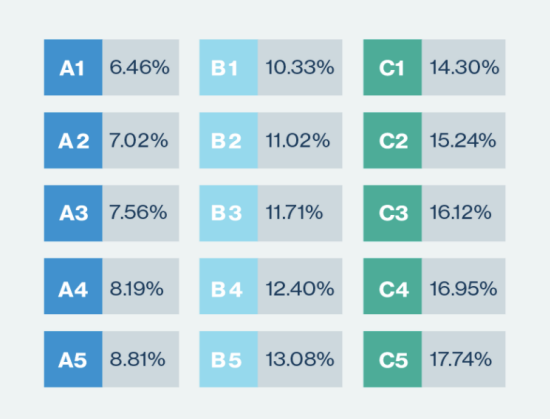
EDA is the first step to be done since it reveals relationships within data and therefore gives out indication about what predictions we can make with the data. We then would like to use two machine learning methods to predict loan default rates and make a comparison between the results and give out the optimal model. With the results of our project, investors can get instruction on the direction of a lending portfolio on site.

III. Data Source and Data Exploration

The data was collected from the official website of Lending Club. We choose the data from the second quarter of 2020 which contains 13,303 individuals with their complete loan data. The data contains 150 features in total. Here are some of the important features: id, loan amount, funded amount, interest rate, term, grade, subgrade, employment title, homeownership, employment length, annual income, purpose, loan status, zip code, FICO score, etc.

Although we have as many as 150 columns in the dataset, a large number of them are not usable because of data format (like zip code with hidden numbers) and missing values. To make our analysis more interpretable, we drop those columns with 80% or more missing values which reduce our data dimension from 150 to 97. To further understand our data, we select several columns of information that we are interested in and apply EDA to each of them in order to tell any specific patterns on loan/borrowers distribution.

Based on the credit portfolio of each applicant, LendingClub will distribute each corresponding loan a grade, from A1 to C5 with a corresponding interest rate. In our data, the "int_rate" variable stands for **Interest Rate** on the Loan. After summarizing the data, we find the range of interest rate is from 6.46% to 30.99%, with the mean interest rate equals to 12.76% and the median interest rate equals to 10.81%. LendingClub will pay Notes investors principal and interest payment equal to pro-rata portion of each borrower's payment related to the corresponding loan. Each loan's subgrade and interest rate are displayed by the reference table [Table 3.1].

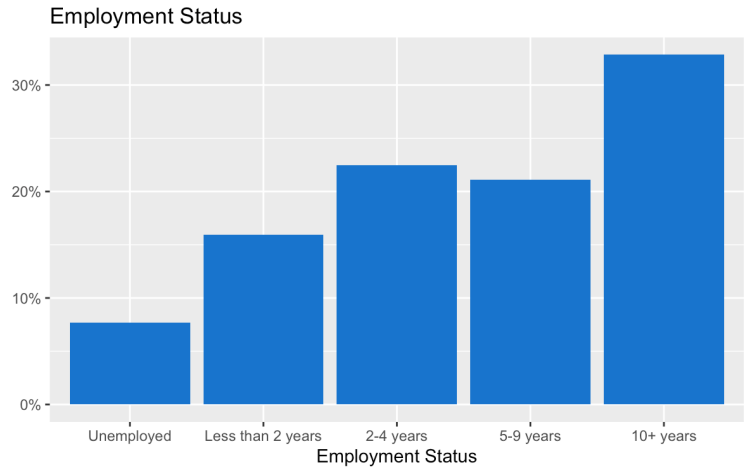


[Table 3.1]

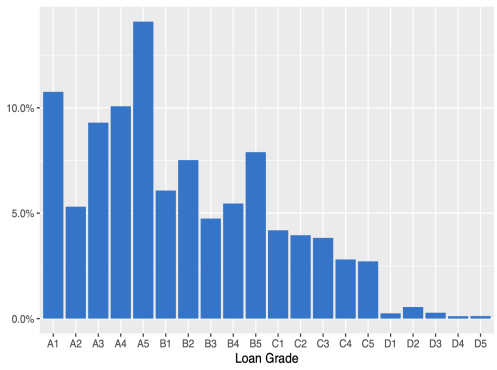
FICO is a credit scoring model using credit information such as reports from your lenders, banks, and other financial institutions to the main three credit bureaus, which are Experian, Equifax, and Transunion. FICO score is a credit score typically distributed from 300 to 850. Some important factors will affect the credit score such as past payment background check, age, credit rate, percentage of credit limit used, current balances of debts, and recent credit performance and dynamics. From the FICO system, the median FICO credit score range is from 670 to 739, which means the scores from 740 to 799 indicate good credit, while anything below 670 is considered as subprime. In our dataset, we find the range of FICO score is from 664 to

749, with mean equals to 723 and the median equals to 719. After analyzing the data, we find the proportion of the “Subprime” loan is less than 10%.

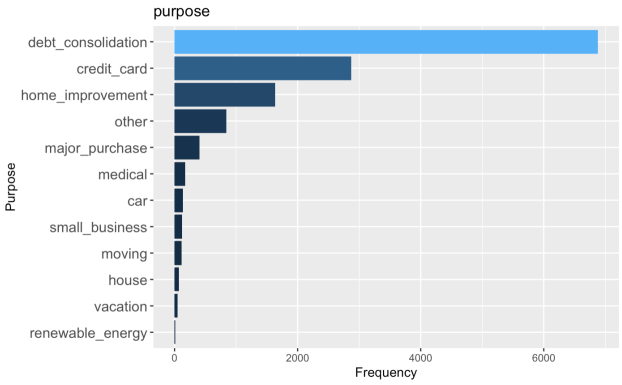
The **Employment Status** displays Employment length in years. In our dataset, we assume “n/a” columns as unemployed and we calculate proportions across different employment statuses. We conclude that more than 50% of loanees have been employed for more than 5 years. Also according to figure1 in the appendix, the manager accounts for the highest proportion of employment titles. Director is the second large population of the loanee.



In Lending Club, each loan is assigned a **grade** with a corresponding interest rate based on applicants’ background, including credit history. From the histogram [Table 3.3], we can see that most loans are within-grade A.

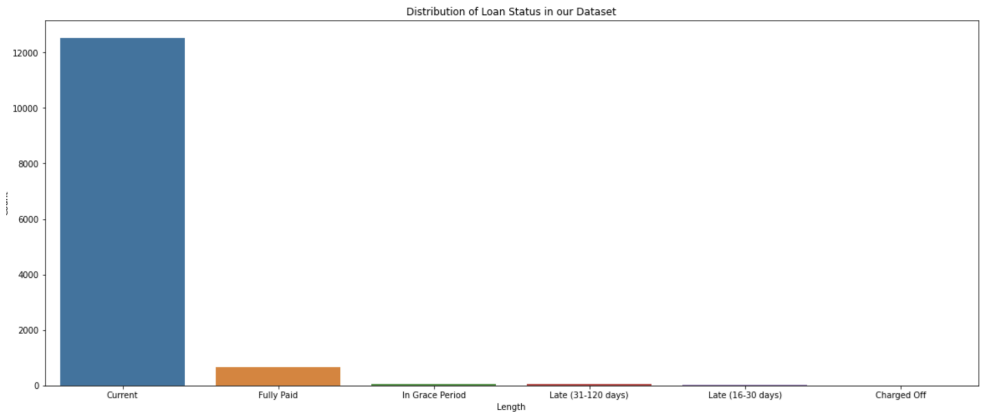


[Table 3.3]



[Table 3.4]

The histogram [Table 3.4] shows that debt consolidation is the most frequently used reason for borrows. Credit card payment and home improvement are the other two important reasons.



We plotted a histogram of counts of different loan status in the dataset. As the figure shows, we have most of the loans in "Current" status which is not settled. With this highly imbalanced proportion, we need additional processing in later prediction analysis.

Results of other EDAs

Funded Amount: ranged from $1,000 to $40,000 with the median of $15,000. (Table 1 in appendix)

The term length of Loans: either 36 months or 60 months, about 73% are 36 months. (Figure 2 in appendix)

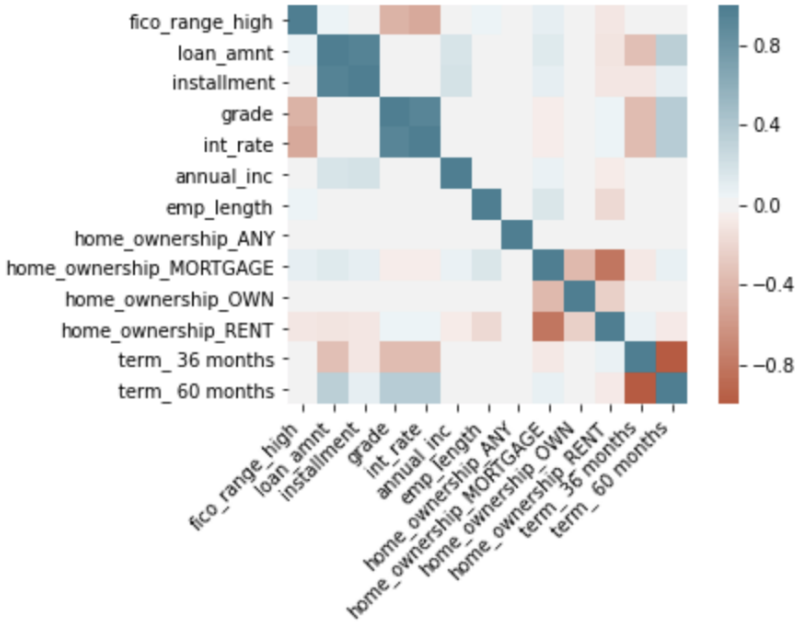
Home Ownership: most borrowers have mortgages and less of them own their house or apartment. (Figure 3 in appendix)

Loan Status: it indicates where your loan is in the process. More than 90% of Lending club's loans in the data are in the process of being paid back. Default and late ones are the least. Also, more than 98% of matured loans are fully paid. (Figure 4 & 5 in appendix)

IV. Materials and Methods

There are several types of features, including categorical, discrete, continuous, and binary. To handle categorical variables, we separate them into ordinal and nominal variables. For ordinal ones like loan grade (A, B, C, D), we use mapping to assign values. For nominal ones like home-ownership, we use a one-hot encoding.

After preliminary data processing, let's see the correlation structure between the variables that we are interested in and selected to be included in the next modeling step. It shows us any dependencies between different variables and helps us reduce the dimensionality a little bit more.



From the above correlation matrix, we can see that loan amount and installment are positively correlated with each other. Intuitively, this is because larger amounts of loans generally need more time to repay. Also, grades and interest rate have a very high correlation as well which is explained and analyzed in our EDA part that interest rate is decided by grades once the grades are assigned. Terms are inversely related because borrowers could only choose one of the two terms of loans. To avoid collinearity, we exclude one of the correlated variables.

With an overview of the dataset and how loan applicants are distributed according to different attributes, we are interested in another important and practical question: Are borrowers trustworthy? To answer this question, we implemented machine learning models to predict the loan status based on the information provided. Therefore, we as investors could identify loans/borrowers that may not be safe enough to invest. For prediction, we used two different decision tree models: the Rpart method and the Random Forest Classification method. Before running each model, we splitted our data into a training-test set (80%-20%) to see how our models perform.

Rpart method builds regression models on a general two-stage procedure, and the resulting models can be represented by binary trees. In our case, since our goal was to build a regression model on loan status, we divided loan_status into 2 groups: “current” and “fully paid” are represented by “1” and “charged off”, “in grace period”, “late” are represented by “0”. Then we separated our data into a training set and a testing set. Initially, we found that our data was extremely unbalanced. In the training set, the ratio between group “1” to group “0” was 10499:86. We tried the “undersampling and oversampling” method, which is included in the package “ROSE”. The undersampling and oversampling are techniques used to adjust the class distribution of the dataset. The ratio changed to 4953:5074. Then we tried “systemic balancing”

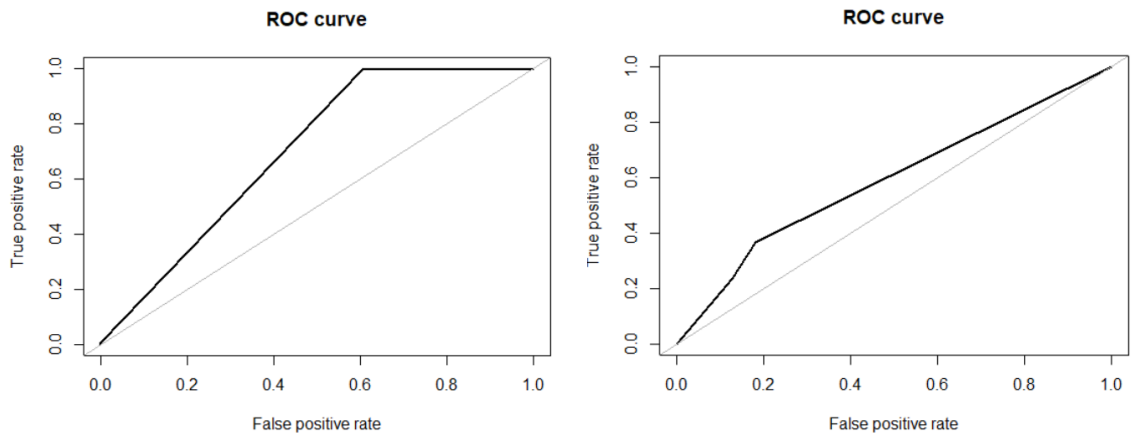
methods to rebalance our training set. This method is also a type of oversampling, which creates artificial data points to balance the data classes. The ratio changed to 5361:5264. We used training sets generated from both methods to build models and the one with higher accuracy would be chosen.

Random Forest model is an ensemble of decision trees which uses bootstrap to achieve parallel growing of trees (bagging). 'Random' refers to random sampling of data entries (with replacement) and random sampling of features. Compared to gradient boost models, random forest aims to reduce variance and also no need of data standardization.

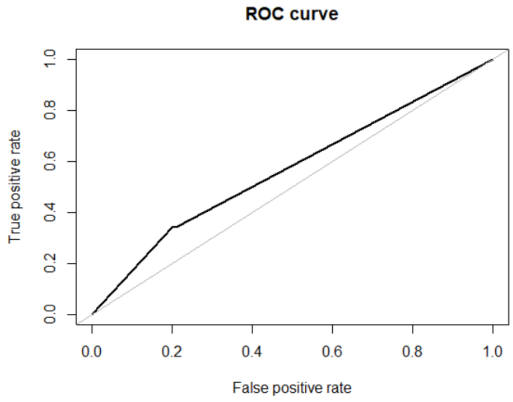
V. Results

Rpart model

We first built a model based on original data: the ROC curve shows below. The AUC is 0.697. Since an unbalanced dataset will bias the prediction model towards the more common class, which is "current" and "fully paid" in our case. Therefore, this is not the best model, even though the AUC might be the highest compared to the AUC of the other two models.



Then we built our second model based on our first balanced training set using "undersampling and oversampling" method: the ROC curve shows as above. The AUC is 0.592. We built our third model using our second training set by using "systematic method": the ROC curve shows below. The AUC is 0.568.



In comparison, we used the "undersampling and oversampling" method to balance our data. There are the following two reasons that might cause the general decline of AUC: it might be caused by the feature selection process and it could also be caused by the fact that we didn't balance our test data. In order to ensure the accuracy and authenticity of our model, we only balanced our training set, which might trigger the difference in classification results.

Random Forest Classification model

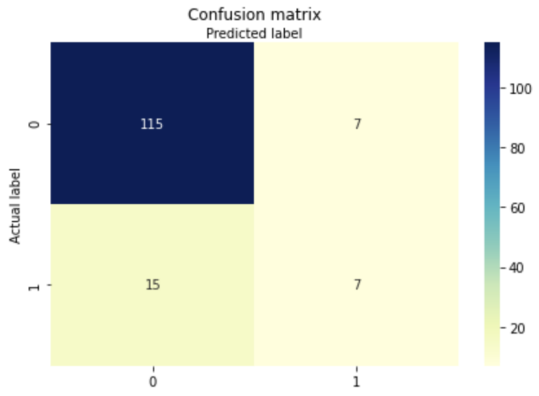
Since we define our target as binary, as we are interested in whether loans are recommended to the investors on the Lending Club. In this case, random forest classification is suitable for the prediction of the loan default rate. More specifically, we set loan status for fully paid as 0 while another status such as late, default, charged off as 1 to help quantify default rate/probability in the prediction. A small difference between the previous the rpart model and the random forest model we use is that this time, we only consider mature loans (with a status

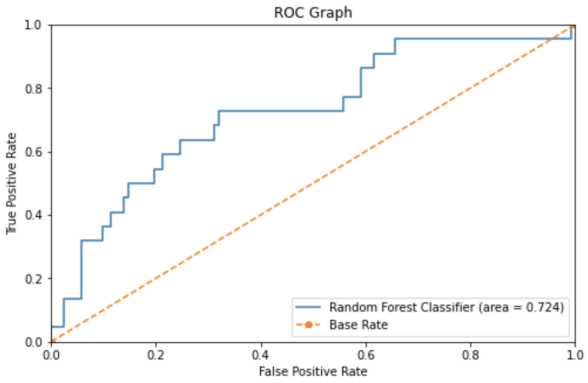
other than current) to see whether there will be a difference in the results of prediction.

Among all the matured loans, 85% are fully paid and 15% are default/charged off. Similar to the previous rpart model, we handled imbalance data by oversampling before training our model. If we use imbalance data, we will still get relatively high accuracy, but merely because of the model putting everything into the majority (0). To improve our model performance, we use hyperparameter tuning to find the optimal model with parameters that give the best score of the metric. Here we use *Recall* as we would like to identify as many potential default loans as possible among truly bad loans and maximize the true positive rate of default. Scikit-Learn library in Python provides a GridSearch cross-validation function to help to find the best models and parameters.

	Accuracy	ROC AUC	Recall
Benchmark	82.64%	0.703	0.182
Tuned Model	84.72%	0.724	0.318

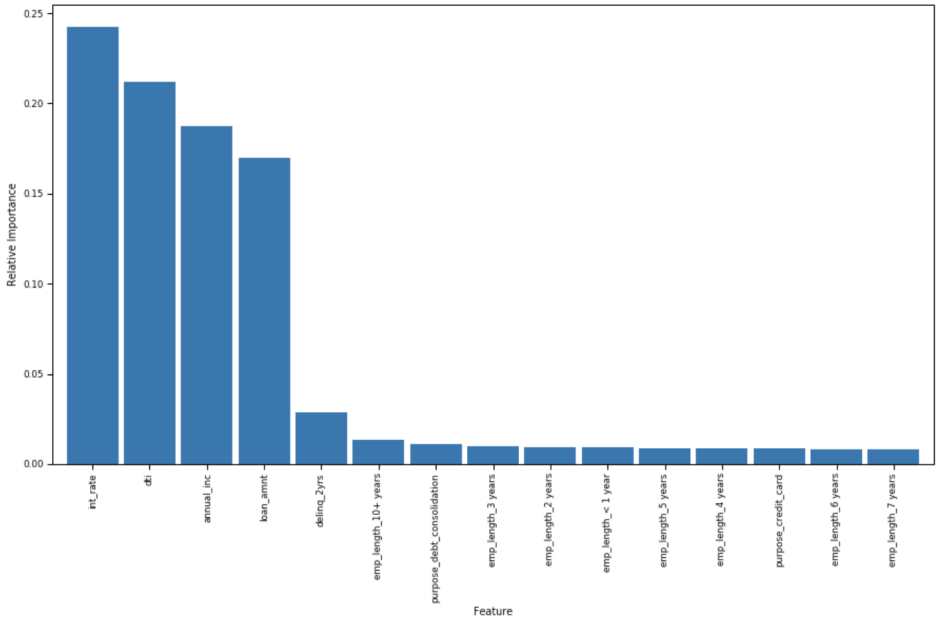
```
Accuracy of Random Forest Classifier on test set: 84.72%
```





Now we have our optimal model and its prediction, which feature has the greatest impact on the outcomes. We can select the top 5 of them by their relative importance. For instance,

interest rate, debt to income ratio, amount of loan, annual income and 2 years delinquency are the most important factors in predicting loan defaulters.



To figure out the relationship between important factors and our target outcome, we used average values for three different levels of default probabilities for each factor. More specifically, the lower the interest rate (higher the loan grade) and debt to income ratio, the lower the default probability. The higher the borrower's annual income, the less likely the borrower is going to default the loan.

VI. Discussion and Conclusion

In conclusion, apart from statistical data exploration and bivariate analysis, We have successfully built machine learning models to predict the default probability for applicants on their loans which can be further used by LendingClub for their analysis. Also, we might want to research other techniques to improve the model performance and prediction power of the

algorithm. One limitation of our analysis is the limited number of our data, where only a few applicants defaulted on their loan in the Second Quarter of 2020. We can improve our analysis by using an extended dataset that consists of more historical records in the past five years to see if there will be any change in our prediction of the current loans to be paid off or defaulted or even charged off. Then these new data points can be used for predicting them or even used to train the model again to improve its accuracy.

Another limitation is that we only use information (columns) selected from EDA which may result in selection bias. To make our model more robust, it would be better to include all the feasible features and use some techniques to reduce the dimensionality. However, since we had various types of data type including categorical, ordinal and numeric, we cannot apply PCA directly for dimensionality reduction but we can try some different types of methods like 't-SNE' which has better performance dealing with high dimensional data to reduce the dimensionality.

With potential inference on features, we need to conduct further research because correlation cannot tell causation. Economic factors and borrower's assets record could change over time. Also, with a potential investment made to borrowers, we may check on a regular basis to adjust the model and therefore complete a development cycle. To further improve our analysis and dig deeper, we could implement the multi classification method to predict detailed loan status. Our analysis could be used by the LendingClub to design a scoring system to facilitate the selection of the most credible individuals for investment strategy.

VII. Reference

1) LendingClub Data Collection and Statistics. https://www.lendingclub.com/statistics/additional-statistics?

2) Note Interest Rates and Fees. (n.d.). Retrieved December 03, 2020, from https://www.lendingclub.com/investing/investor-education/interest-rates-and-fees

3) How to Understand Your Credit Score [+Common Questions]. (n.d.). Retrieved December 03, 2020, from https://www.lendingclub.com/loans/resource-center/understanding-credit-scores

4) *Prediction of LendingClub loan defaulters*. Kaggle. Retrieved December 30, 2018, from https://www.kaggle.com/deepanshu08/prediction-of-lendingclub-loan-defaulters

VIII. Appendix

Figures

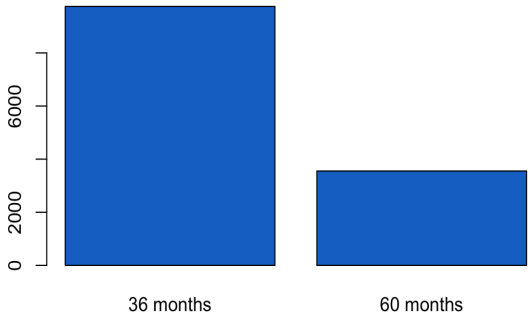
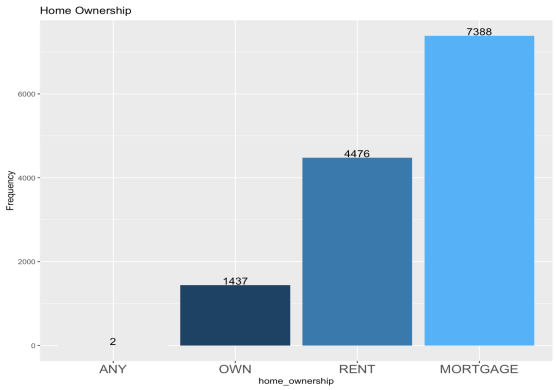


Figure 1: Employment Title

Figure 2: Term Length



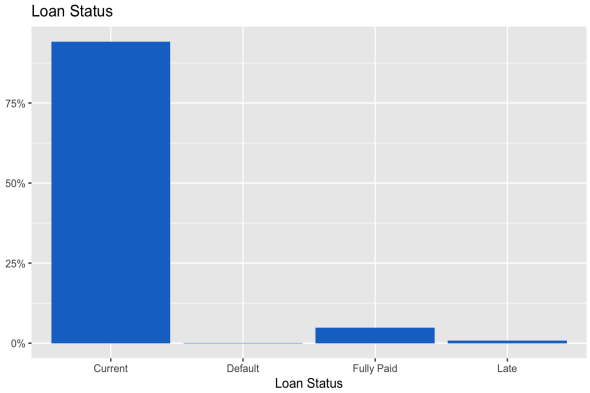


Figure 3: Home Ownership

Figure 4: Loan Status

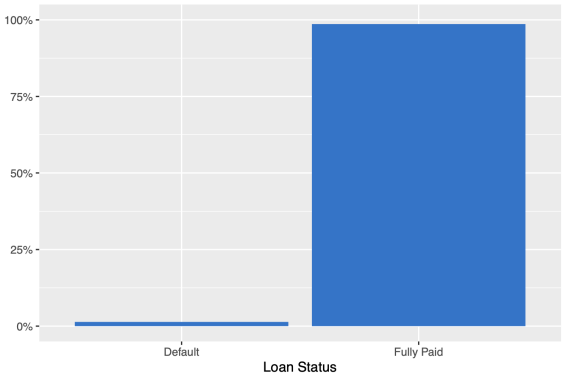


Figure 5: Loan Status of Matured Loans Only

Tables

Table 1: Founded Amount

min	1st quantile	median	mean	3rd quantile	max
1000	8000	15000	16518	24000	40000

Codes (in R & Python)

R code:

```
library(readr)
library(dplyr)
library(tm)
library(wordcloud)
library(RColorBrewer)
library(zoo)
library(ggplot2)
library(scales)
library(extrafont)
```

```
X2020Q2 <- read_csv("Desktop/5291/2020Q2.csv")
#View(X2020Q2)

new <- X2020Q2[!is.na(X2020Q2$funded_amnt),] # removes missing lines
#View(new)
```

Rpart Model

```
#install.packages("tidyverse")
library(tidyverse)
#install.packages("plotly")
library(plotly)
library(zoo)
#install.packages("caret")
library(caret)
library(dplyr)
library(rpart)
install.packages('pROC')
library(pROC)
#delete columns with large number of NA and zeros
setwd("C:/Users/Cynth/Desktop")
loan<-read.csv("loan2020.csv")
summary(loan)
#transfer int_rate to numeric
loan$int_rate <-  as.character(loan$int_rate)
loan$int_rate <-  as.numeric(substr(loan$int_rate, 1, nchar(loan$int_rate)-1))
##eliminate NA
sum(is.na(loan))
loan_1 = loan
for(i in 1:ncol(loan_1)){
  loan_1[is.na(loan_1[,i]), i] <- mean(loan_1[,i], na.rm = TRUE)
}
```

```r
str(loan_1)
loan_demand = loan_1
table(loan_1$grade)
loan_demand$grade <- as.character(loan_demand$grade)
loan_demand$st <- loan_demand$grade
#loan_demand$st <- ifelse(test = loan_demand$grade == ("A"|"B"), yes = 1, no = 0)
table(loan_demand$loan_status)
loan_demand[loan_demand$loan_status == 'Charged Off',]$st <- "0"
loan_demand[loan_demand$loan_status == 'Current',]$st <- "1"
loan_demand[loan_demand$loan_status == 'Fully Paid',]$st <- "1"
loan_demand[loan_demand$loan_status == 'In Grace Period',]$st <- "0"
loan_demand[loan_demand$loan_status == 'Late (16-30 days)',]$st <- "0"
loan_demand[loan_demand$loan_status == 'Late (31-120 days)',]$st <- "0"
loan_demand$st<-as.factor(loan_demand$st)
sum(is.na(loan_demand))
table(loan_demand$st)
```

```r
#data partition
set.seed(123)
#install.packages("caTools")
library(caTools)
train<-sample.split(loan_demand,SplitRatio=0.8,group=NULL)
trainset<-subset(loan_demand,train==T)
testset<-loan_demand[!train,]
table(trainset$st)
#install.packages("ROSE")
library(ROSE)
#data rebalancing using over sampling
data_2 <- ovun.sample(st ~ ., data = trainset, method = "both",p=0.5,N=10000,seed=1)$data
table(data_2$st)

trainset_2 = subset(trainset, select = -c(grade))
data_3 <- ROSE(st ~ ., data = trainset_2,seed=1)$data
table(data_3$st)
## With original data
tree<-rpart(st~last_pymnt_d+last_pymnt_amnt+total_rec_prncp+fico_range_high+fico_range_low
            +il_util+total_pymnt+total_pymnt_inv+total_rec_int+out_prncp+emp_title+earliest_cr_line+dti+out_prncp_inv+mo_sin_old_rev_tl_op,data=trainset)
summary(tree)
pred.tree.both <- predict(tree, newdata = testset)
roc.curve(testset$st, pred.tree.both[,2])#ROC Curve and value
```

```
## using both undersampling and oversampling
tree1<-rpart(st~last_pymnt_d+last_pymnt_amnt+total_rec_prncp+fico_range_high+fico_range_low
              +il_util+total_pymnt+total_pymnt_inv+total_rec_int+out_prncp+emp_title+earliest_cr_line+dti+out_prncp_inv+mo_sin_old_rev_tl_op,data=data_2)
summary(tree1)
pred.tree.both1 <- predict(tree1, newdata = testset)
roc.curve(testset$st, pred.tree.both1[,2])#ROC Curve and value


## using some automatedly created data
tree2<-rpart(st~last_pymnt_d+last_pymnt_amnt+total_rec_prncp+fico_range_high+fico_range_low
             +il_util+total_pymnt+total_pymnt_inv+total_rec_int+out_prncp+emp_title+earliest_cr_line+dti+out_prncp_inv+mo_sin_old_rev_tl_op,data=data_3)
summary(tree2)
pred.tree.both2 <- predict(tree2, newdata = testset)
roc.curve(testset$st, pred.tree.both2[,2])#ROC Curve and value
```

interest rate

```{r}
summary(as.numeric(gsub("%","",new$int_rate)))
```

FICO score

```{r}
summary(new$fico_range_high)
```

```{r}
new$subprime <- ifelse(new$fico_range_high < 670, "Yes", "No")
nrow(new$subprime == "Yes")/nrow(new)
ggplot(new, aes(x = factor(subprime))) +
  geom_bar(aes(y = (..count..)/sum(..count..)), fill = "dodgerblue3") +
  scale_y_continuous(labels = percent) +
  labs(x = "Subprime", y = "") +
  ggtitle("Subprime")
```

employment title

```
loantitleCorpus <- Corpus(VectorSource(new$emp_title))
loantitleCorpus <- tm_map(loantitleCorpus, content_transformer(tolower))
loantitleCorpus <- tm_map(loantitleCorpus, removePunctuation)
wordcloud(loantitleCorpus, max.words = 100, random.order = FALSE,rot.per=0,
          colors=brewer.pal(8, "Dark2"))
```

Loan grade

```
ggplot(new, aes(x = factor(sub_grade)))+
  geom_bar(aes(y = (..count..)/sum(..count..)), fill = "dodgerblue3")+
  scale_y_continuous(labels = percent) +
  labs(x = "Loan Grade", y = "")+
  ggtitle("Loan Grade")
```

Borrower's purpose

```
new %>%
  group_by(purpose) %>%
  summarize(freq = n()) %>%
  top_n(50) %>%
  ggplot(aes(reorder(purpose, freq), y = freq, fill = freq)) +
  geom_bar(stat = "identity", position = "dodge") +
  xlab("Purpose") +
  ylab("Frequency") +
  coord_flip() +
  theme(legend.position ='none', axis.text.y = element_text(size = 12)) +
  ggtitle("purpose")


```

Funded Amount

```
summary(new$funded_amnt)
```

Term length of Loans

```
table(new$term)
```

Home Ownership

```
new %>%
  group_by(home_ownership) %>%
  summarize(freq = n()) %>%
  ggplot(aes(reorder(home_ownership, freq), y = freq, fill = freq)) +
  geom_bar(stat = "identity", position = "dodge") +
  xlab("home_ownership") +
  ylab("Frequency") +
  #theme_fivethirtyeight() +
  theme(legend.position ='none', axis.text.x = element_text(size = 15)) +
  geom_text(aes(label = freq), vjust = -0.1, size = 4.5) +
  #scale_fill_gradientn(name = '',colours = rev(brewer.pal(10,'Spectral'))) +
  ggtitle("Home Ownership")
```

Loan Status

```
table(new$loan_status)
new$status <- gsub("Does not meet the credit policy. Status","",new$loan_status)
new$status[new$status == "Charged off"] <- "Default"
new$status[new$status == "In Grace Period"] <- "Late"
new$status[new$status == "Late(16-30 days)"] <- "Late"
new$status[new$status == "Late(31-120 days)"] <- "Late"
ggplot(new, aes(x = factor(status)))+
  geom_bar(aes(y = (..count..)/sum(..count..)),fill = "dodgerblue3")+
  scale_y_continuous(labels = percent)+
  labs(x = "Loan Status", y = "")+
  ggtitle("Loan Status")
```

Matured Loan Status

```
ggplot(aes(x = factor(status)), data = subset(new, status!="Current"&status!="Late"))+
  geom_bar(aes(y = (..count..)/sum(..count..)), fill = "dodgerblue3")+
  scale_y_continuous(limits = c(0,1), labels = percent)+
  labs(x = "Loan Status", y = "")+
  ggtitle("Loan Status (Only Matured Loans)")
```

Python code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
loans = pd.read_csv('../../Downloads/LoanStats_securev1_2020Q2.csv',skiprows=1)
```

```
# Transform ordinal variable
mapping_dict = {
    "emp_length": {
        "10+ years": 10,
        "9 years": 9,
        "8 years": 8,
        "7 years": 7,
        "6 years": 6,
        "5 years": 5,
        "4 years": 4,
        "3 years": 3,
        "2 years": 2,
        "1 year": 1,
        "< 1 year": 0
    },
    "grade":{
        "A": 1,
        "B": 2,
        "C": 3,
        "D": 4
    }
}

df = df.replace(mapping_dict)
df[['emp_length','grade']].head()
```

```
# One-hot Encoding for Nonordinal variable
n_columns = ["home_ownership", "term"]
dummy_df = pd.get_dummies(df[n_columns])
df = pd.concat([df, dummy_df], axis=1)
```

```
df = df.drop(n_columns, axis=1)
```

```
df['int_rate'] = df['int_rate'].str.rstrip('%').astype('float')
```

```
df.dropna(inplace=True)
df.shape
```

```
(12284, 12)
```

```
df = df[df.loan_status!='Current']
```

```
def coding(col, codeDict):

    colCoded = pd.Series(col, copy=True)
    for key, value in codeDict.items():
        colCoded.replace(key, value, inplace=True)

    return colCoded

df["loan_status"] = coding(df["loan_status"],
                           {'Fully Paid':0,'In Grace Period':1,'Late (31-120 days)':1,
                            'Late (16-30 days)':1,'Charged Off':1})
df['loan_status'].value_counts()
```

```
corr = df.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
# Feature Scaling
# from sklearn.preprocessing import MinMaxScaler
# scaler = MinMaxScaler()

# X = scaler.fit_transform(X)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)
```

```
from imblearn.over_sampling import SMOTE
# handle imbalanced data
sm = SMOTE(random_state=42)    # over-sampling
X_train, y_train = sm.fit_sample(X_train, y_train)
```

```
# Common sklearn Model Helpers
from sklearn import feature_selection
from sklearn import model_selection
from sklearn import metrics
# sklearn modules for ML model selection
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score


rf = RandomForestClassifier()
rf = rf.fit(X_train,y_train)
```

```
cnf_matrix = metrics.confusion_matrix(y_test, rf.predict(X_test))
class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu",fmt="d")
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label');
print('Accuracy of Random Forest Classifier on test set: {:.2f}%'.format(rf.score(X_test, y_test)*100))
```

```
from sklearn.metrics import confusion_matrix, classification_report, precision_recall_curve
from sklearn.metrics import auc, roc_auc_score, roc_curve, recall_score, log_loss
from sklearn.metrics import f1_score, accuracy_score, roc_auc_score, make_scorer
from sklearn.metrics import average_precision_score

rf.fit(X_train, y_train) # fit optimised model to the training data
probs = rf.predict_proba(X_test) # predict probabilities
probs = probs[:, 1]
rrf_roc_auc = roc_auc_score(y_test, probs) # calculate AUC score using test dataset
rf_recall = recall_score(y_test, rf.predict(X_test))
rf_f1 = f1_score(y_test, rf.predict(X_test))
print('AUC score: %.3f' % rf_roc_auc)
print('Recall Score: %.3f' % rf_recall)
print('F1 Score: %.3f' % rf_f1)
print(classification_report(y_test, rf.predict(X_test)))
```

```
importances = rf.feature_importances_
indices = np.argsort(importances)[::-1][:5] # Sort feature importances in descending order and choose top 5
names = [df_col[i] for i in indices] # Rearrange feature names so they match the sorted feature importances
plt.figure(figsize=(15, 7))
plt.title("Feature Importance")
plt.bar(range(5), importances[indices])
plt.xticks(range(5), names, rotation=45)
plt.show()
```

```
# Create ROC Graph
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, rf.predict_proba(X_test)[:,1])
plt.figure(figsize=(8, 5))

# Plot Logistic Regression ROC
plt.plot(fpr, tpr, label='Random Forest Classifier (area = %0.3f)' % rf_roc_auc)
# Plot Base Rate ROC
plt.plot([0,1], [0,1],'p--',label='Base Rate')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Graph')
plt.legend(loc="lower right")
plt.show()
```

```
from sklearn.model_selection import GridSearchCV

rf_classifier = RandomForestClassifier()
param_grid = {'n_estimators': [50, 100, 150],
              'min_samples_split':[2,4,6],
              'max_depth': [5, 10, 15]}

grid_obj = GridSearchCV(rf_classifier,
                        param_grid=param_grid,
                        scoring='recall',
                        cv=5)

grid_fit = grid_obj.fit(X_train, y_train)
rf = grid_fit.best_estimator_

print('='*20)
print("best params: " + str(grid_obj.best_estimator_))
print("best params: " + str(grid_obj.best_params_))
print('best score:', grid_obj.best_score_)
print('='*20)
```

```
====================
best params: RandomForestClassifier(max_depth=10, n_estimators=150)
best params: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 150}
best score: 0.8742268041237112
====================
```