Loan Payment Prediction

Don Marco Loleng (loleng2), Devyn Theis (dtheis2), Anuj Patel (anpatl10)

12/15/2018

Introduction and Literature Review:

In today's society the idea of paying back a loan can be daunting, especially for borrowers. Lenders provide the loans to borrowers in exchange for a promise of repayment; lenders make money when a loan is fully paid back and lose money when not. Our project hopes to make it easier on them by predicting whether or not a customer will default or pay back a loan. The dataset used for our model contains roughly 900 thousand observations and 75 variables, from this Kaggle source. The data comes from Lending Club they provide loans in a peer to peer interaction. The scientific goal is to predict whether or not a customer will default or pay back a loan and the statistical learning method we will use to accomplish this is logistic regression. In the dataset our outcome variable will be "Loan Status" since it is a categorical variable that we can turn to 1 and 0 with some data cleaning. Below are several of what we think are important parameters for our problem (we have also attached a data dictionary for reference):

- Loan_amnt - amount of money requested by borrower
- Funded_amnt - total amount committed to a loan
- Funded_amnt_inv - total amount committed by investors for a business loan
- Int_rate - interest rate on the loan
- Grade - metric used to rate loan based on borrowers credit history. Scale of A to F
- Term - length of loan
- Installment - fixed amount for each payment
- Home_ownership - whether or not borrower rents or owns housing
- Annual_inc - annual income of borrower
- Issue_d - when the loan was issued
- Loan_status - whether or not loan was paid back in full
- Pymnt_plan - whether or not borrower was on a payment plan
- Purpose - what the loan was being used for(car, credit card, small_business, etc.)
- Title - item bought with loan (bike, computer, personal)
- Dti - debt to income ratio of borrower
- Inq_last_6_mnths - number of inquiries last 6 months
- Mnths_since_last_delinq - months since last delinquency
- Open_acc - number of open accounts under borrower name
- Revol_bal - amount per revolving balance
- Total_pymnt - total payment amount
- Last_pymntd - date of last payment
- Last_pymnt_amnt - last payment amount
- Application_type - type of application (individual etc.)

To summarize, the amount of money borrowed can impact how long/if you can pay back the amount in time. Interest rate impacts borrowers in a sense that good credit history applicants have lower interest rates thus being more responsible for their loan. Having a lower loan grade can dramatically impact a borrower's ability to pay back loan on time due to more constraints.

In all, we think these variables along with several more will help in accurately predicting whether or not a borrower can pay back their loan on time.

Literature Review:

The scholarly article we reviewed was about Credit Risk Analysis using Machine and Deep learning models. Since the dataset was imbalanced (classification categories are not equally represented) the method of SMOTE algorithm was used to create synthetic samples rather than over-sampling with replacement. There were 7 models used for comparison: logistic regression with regularization, random forest, gradient boost, neural network approach with four different complexities. To rank the models, based on companies' credit worthiness, ROC curve, AUC, and RMSE were used as selection criteria. After the data cleaning, there were 181 variables and the first 10 were selected by each model. The results were interesting in that the model with highest AUC was the random forest model on both the testing and training set. Even more interesting the AUC values for all the models were at least above .84. From this we can infer what our models will potentially look like as we are also using ROC curve and RMSE as selection criteria. The only difference being we aren't using more advanced methods like random forest or neural networks. Some of the variables used in the authors model were equity, cash-flows, proft, and liabilities. In summary, the class tree based algorithms outperforms the rest. After reading this article we have a very good idea on where to start with our dataset most importantly dealing with an imbalanced dataset with a lot of NA values. This was a very interesting article and we have a more clear roadmap on how to predict our desired outcome.

Link to Report

Loading the Data

```
loan = read.csv("./loan.csv")
```

```
levels(loan$loan_status)
```

```
##  [1] "Charged Off"
##  [2] "Current"
##  [3] "Default"
##  [4] "Does not meet the credit policy. Status:Charged Off"
##  [5] "Does not meet the credit policy. Status:Fully Paid"
##  [6] "Fully Paid"
##  [7] "In Grace Period"
##  [8] "Issued"
##  [9] "Late (16-30 days)"
## [10] "Late (31-120 days)"
```

Since we want to predict "Default" or "Charged Off" vs "Fully Paid", we can ignore the others for the time being.

```
# Drop observations that are not relavant to our analysis.
new_data = loan[loan$loan_status %in% c("Default", "Charged Off", "Fully Paid"),]

# Clean column to reflect the remaining levels.
new_data$loan_status = droplevels(new_data$loan_status, c("Current", "Does not meet the credit policy. S
                                                          "Does not meet the credit policy. Status:Fully
                                                          "Issued", "Late (16-30 days)", "Late (31-120 d
dim(new_data)
```

```
## [1] 254190     74
```

```
#Create a numeric binary response for loan status (0 for "Fully Paid" loans, 1 for "Default" or "Charge
new_data$loan_status_numeric = ifelse(new_data$loan_status == "Fully Paid", 0, 1)
```

```
## [1] "Default"    "Fully Paid"
```

Summary Statistics and Data Visualization:

We found that there were many NA values in multiple columns:

```
# Show which columns are above 81% NA
NAs <- loan[,colSums(is.na(loan)) >  nrow(loan)*.81]
colnames(NAs)
```

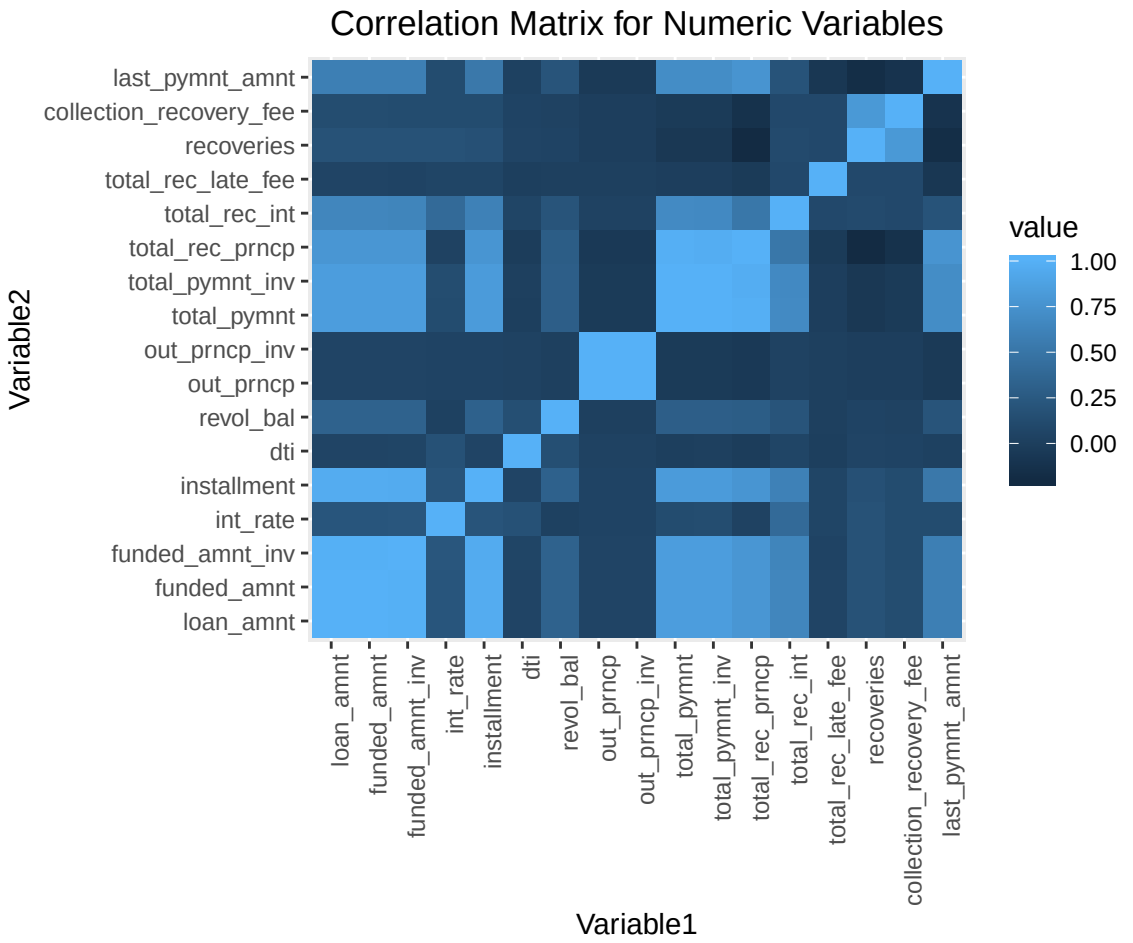
```
##  [1] "mths_since_last_record" "annual_inc_joint"
##  [3] "dti_joint"              "open_acc_6m"
##  [5] "open_il_6m"             "open_il_12m"
##  [7] "open_il_24m"            "mths_since_rcnt_il"
##  [9] "total_bal_il"           "il_util"
## [11] "open_rv_12m"            "open_rv_24m"
## [13] "max_bal_bc"             "all_util"
## [15] "inq_fi"                 "total_cu_tl"
## [17] "inq_last_12m"
```

```
##                                mean    median     variance
## loan_amnt                1.357059e+04 12000.000 6.615198e+07
## funded_amnt              1.353021e+04 12000.000 6.578866e+07
## funded_amnt_inv          1.343761e+04 11975.000 6.608839e+07
## int_rate                 1.377601e+01    13.530 1.939599e+01
## installment              4.182732e+02   365.230 5.998219e+04
## dti                      1.656053e+01    16.220 6.075252e+01
## revol_bal                1.517027e+04 10938.000 3.514404e+08
## out_prncp                5.560524e+01     0.000 9.142142e+05
## out_prncp_inv            5.558916e+01     0.000 9.137081e+05
## total_pymnt              1.357286e+04 11428.462 8.993796e+07
## total_pymnt_inv          1.347570e+04 11328.530 8.993917e+07
## total_rec_prncp          1.146496e+04 10000.000 6.956833e+07
## total_rec_int            1.948614e+03  1311.635 4.273406e+06
## total_rec_late_fee       7.202963e-01     0.000 2.963157e+01
## recoveries               1.585710e+02     0.000 5.570828e+05
## collection_recovery_fee  1.667295e+01     0.000 1.294963e+04
## last_pymnt_amnt          6.432985e+03  3897.990 5.413799e+07
```

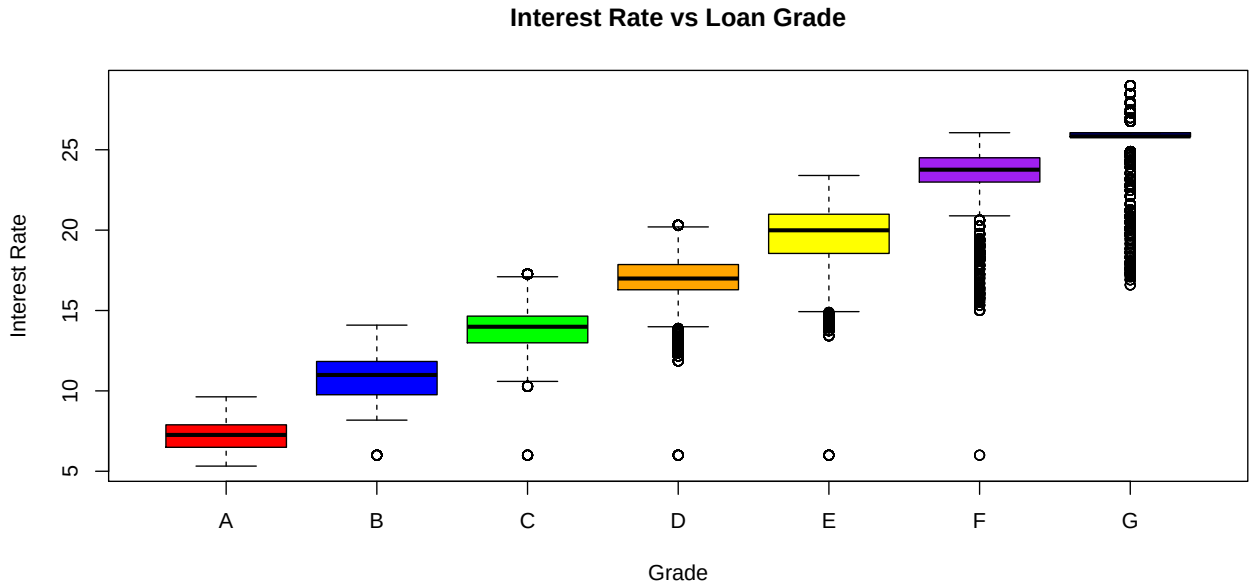
Above you can see some summary statistics for the numerical variables in our data. It shows the mean, median, and variance of each numerical variable.

```
##         Var1   Freq
## 1    Default  46467
## 2 Fully Paid 207723
```

The above table shows the distribution of our response variable which is Loan Status. It has two factors of "Default" and "Fully Paid". For our purposes, we will be converting this to numeric values, where "Default" will be 1 and "Fully Paid" will be 0.

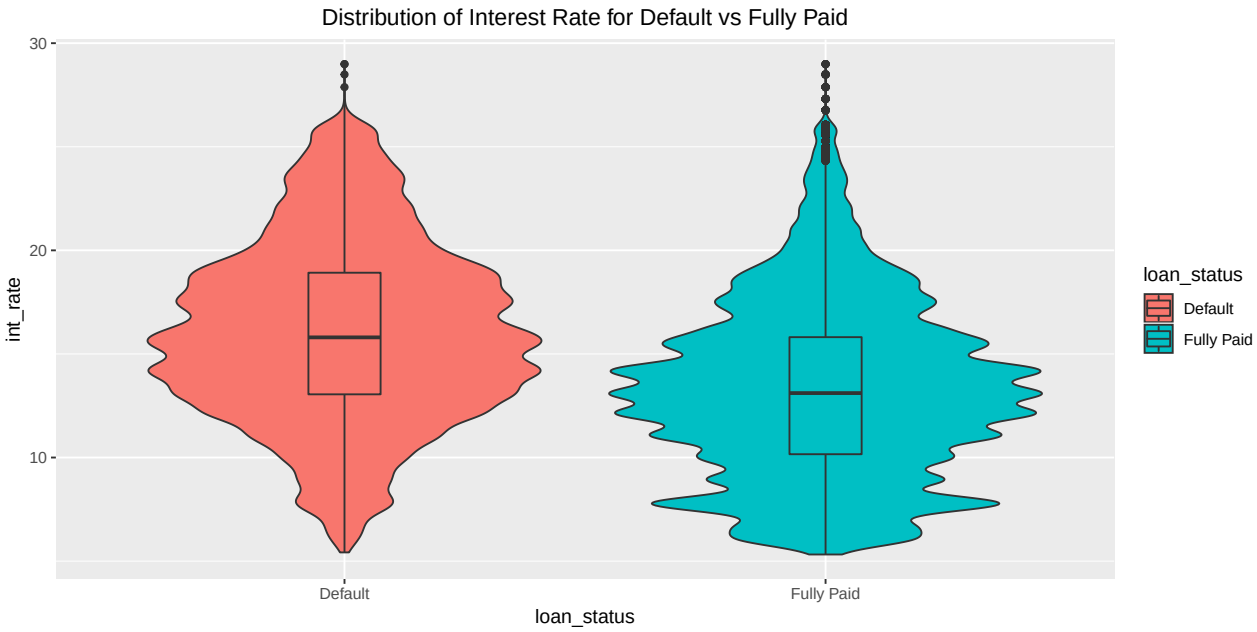


From the heat map we can see the correlation between each variable to deduce any possibilities of multi-collinearity. We can also analyze which variables are not correlated with one another to start our exploratory data analysis.

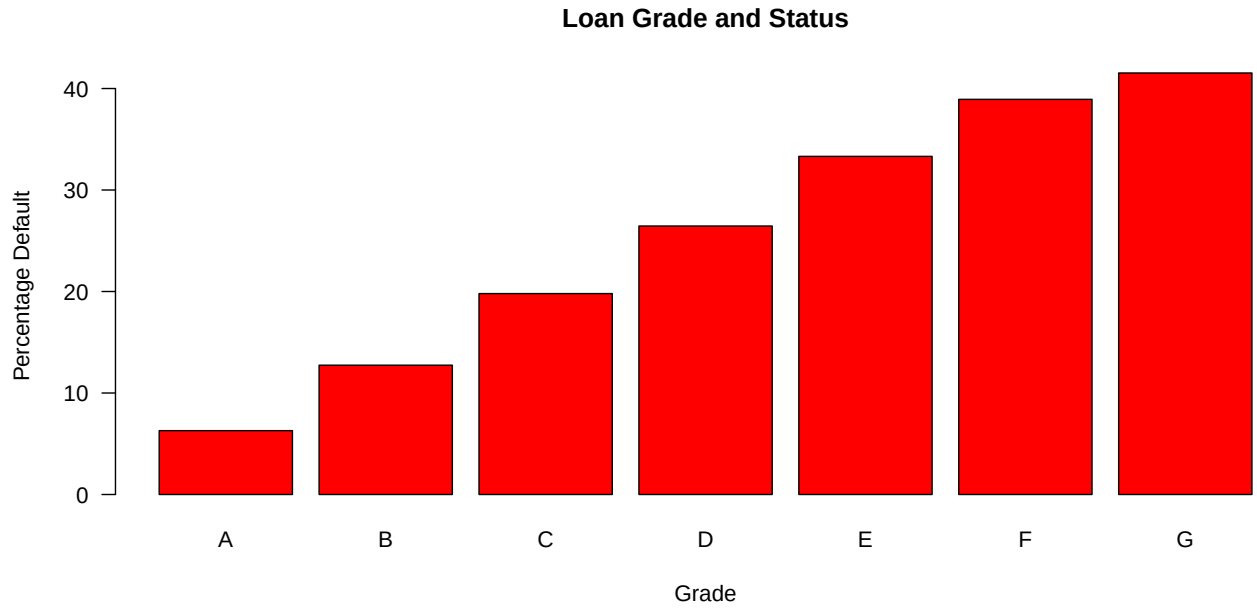


Looking at the relationship between interest rate vs loan grade we can summarize that the higher grade your

loan has the higher the interest rate. This will be helpful when reducing our dataset to a more manageable matrix as these two variables have a clear linear trend. This is also consistent with our intuition that borrowers with higher loan grade are deemed more risky by the creditor thus having a higher interest rate.

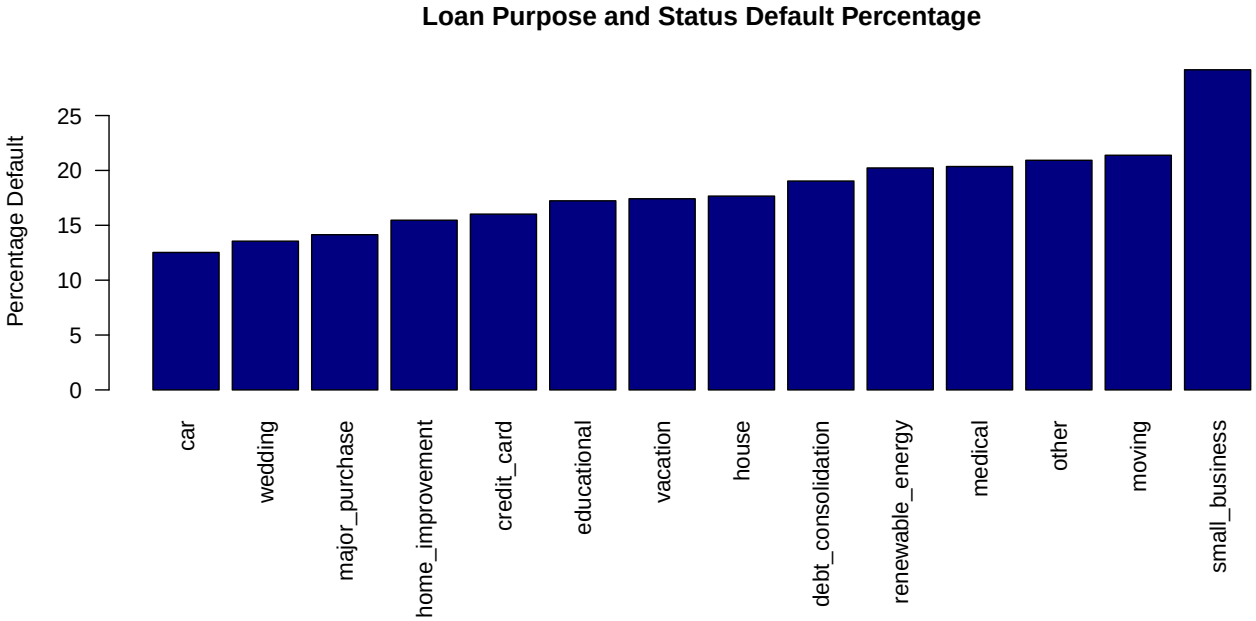


The "violin" plot shows the distribution of our response levels with interest rate. As you can see there is a clear trend between level and interest with the median being lower in Fully Paid than Default. Because we have seen lots of trends with interest rate and our response we can deem it very important in our model for prediction of binary loan payoff.

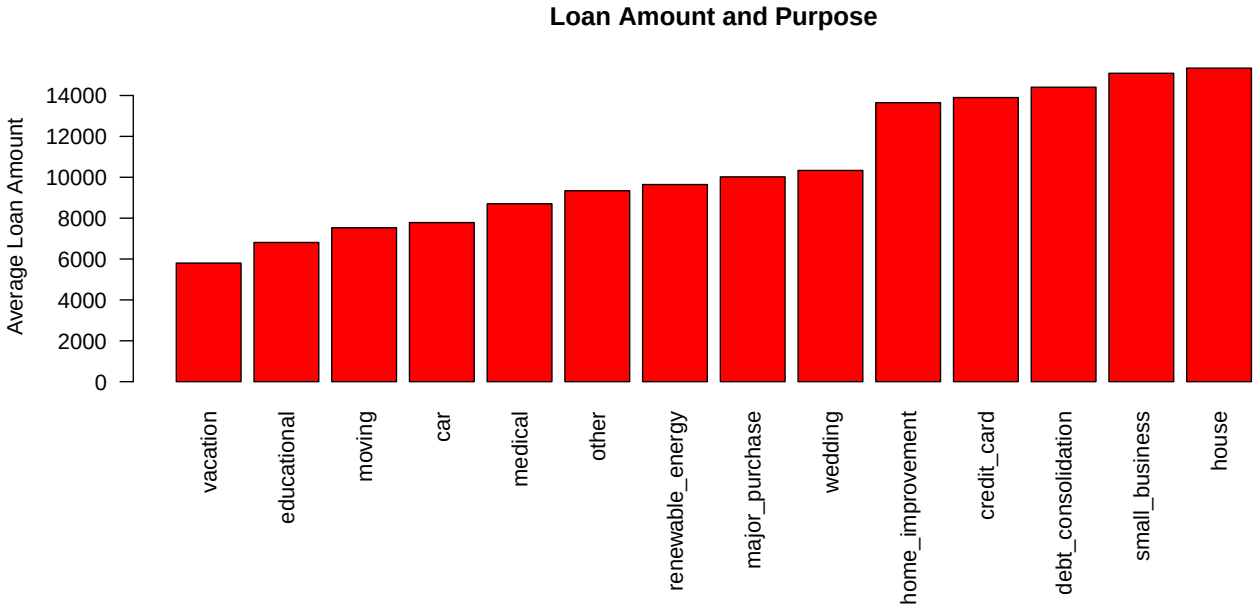


This plot shows the relationship between Loan Grade and the percentage of Defaults for each Loan Grade category. We can see here that the percentage of Defaults continuously increases for each Grade group as the grade goes from A the best, to G the worst. This can be explained by looking at the relationship between Interest Rate and Loan Grade. As Loan Grade gets worse, the Interest Rate also increases. Because of this, as loan grade goes to G, it results in higher Default percentage. This makes sense because with our previous

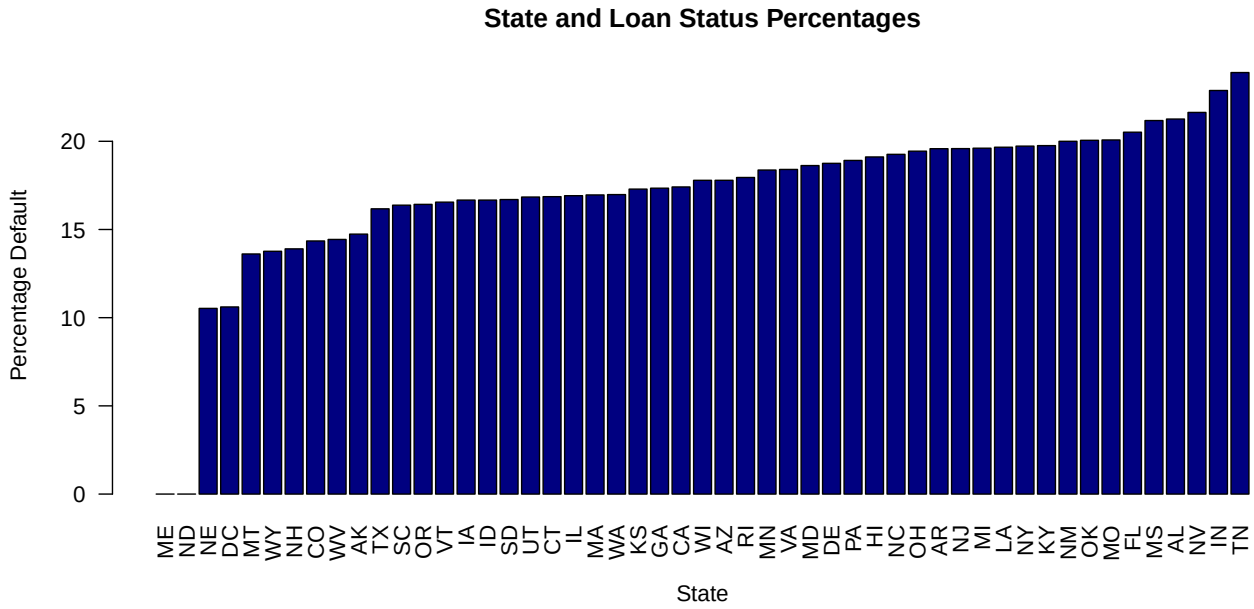
interest rate box plots, it showed that Default percentage increases as the Interest Rate of a loan increases.



The above plot shows the relationship between Loan Purpose and the percentage of Defaults for each loan purpose category. This is interesting to look at because it shows that small business loans are the most likely loans to be Default and car loans are the least likely. This could be because small business loans tend to be larger amounts and are the most risky because they could go bankrupt and never pay off the loan. Car loans are less risky because the loan amount is likely smaller and more easier to pay off in time.



The above plot shows the relationship between Loan Purpose and the Average Loan Amount for each loan purpose category. This interesting to look at because it shows that house loans are typically the largest loans, and small business loans are close behind. Also, vacation loans and educational loans are typically the smallest loans.



This is an interesting plot as it describes the relationship between each state and the percentage of default borrowers. The plot is sorted in increasing order of percentage default with TN having the highest amount. Our intuition wants us to believe that Nevada would have the highest percentage because of activities such as gambling, but our plot shows that it is third highest right behind Indiana.

Proposed Analysis:

An important factor in our analysis is interpretability of the factors that contribute to the risk of loan default/charge off. Thus, one of the main goals of our analysis was to identify a well-performing model with interpretable model inputs, as well as a managable amount of predictors to reduce model complexity. In addition, increasing model complexity of our linear model would increase variance, thus making our model very dependent on the training dataset. A large factor in our cleaning process was to further decrease the amount of less significant predictors to allow our model to train within our memory limit (memory errors produced with large amounts of predictors), in addition to the reasons mentioned above.

Data Cleaning:

The first step in any prediction outcome is to first analyze the data you are given. Since the data source was from Kaggle we ran some preliminary analysis to determine if any cleaning had to be done. After data exploratoration and consultation of the data dictionary, we identified multiple attributes to remove prior to analysis:

```
cols_to_drop = c("id", "member_id", "url", "emp_title", "desc", "title", "mths_since_last_delinq",
                 "mths_since_last_record", "next_pymnt_d", "verification_status_joint",
                 "policy_code", "annual_inc_joint", "dti_joint")

new_data = new_data[, -which(colnames(new_data) %in% cols_to_drop)]
dim(new_data)
```

```
## [1] 254190     62
```

- *id*: This is a unique loan id local to the website, this has no impact on the loan itself.
- *member_id*: ID number of the user
- *url*: URL of the loan

- *emp_title*: We believe the variation of employers for each individual will be hard to gain any important information from. It is a categorical variable with tens of thousands of levels.
- *desc*: This is a user provided description of a loan, again, without some advanced analysis applied to the text, there is very little information we can gain from this.
- *title*: This is a user provided title of a loan, again, without some advanced analysis applied to the text, there is very little information we can gain from this.
- *mths_since_last_delinq*: Many of these values are empty, and we believe that other columns encode information about delinquencies without this level of granularity.
- *next_pymnt_d*: The next payment date is almost completely empty because we are considering completed loans, which have either been charged off/defaulted or paid in full.
- *verification_status_joint*: There is only one observation within our completed loans that indicates joint status, so we removed the columns pertaining to joint-specific accounts.
- *annual_inc_joint*: There is only one joint loan in the completed loans.
- *dti_joint*: There is only one joint loan in the completed loans.
- *policy_code*: All of the policy codes in our subset were the same.

Next, we identified some factor variables Bin date variables by year, convert to numeric: * "last_credit_pull_d" * "earliest_cr_line" * "last_pymnt_d"

```
# Convert date variables into numeric variables reflecting year.
new_data$last_credit_pull_yr = as.numeric(substr(new_data$last_credit_pull_d, 5,8))
new_data$earliest_cr_line_yr = as.numeric(substr(new_data$earliest_cr_line, 5,8))
new_data$last_pymnt_yr = as.numeric(substr(new_data$last_pymnt_d, 5,8))
new_data$issue_yr = as.numeric(substr(new_data$issue_d, 5,8))
```

```
# Remove old date variables
new_data$last_credit_pull_d = NULL
new_data$earliest_cr_line = NULL
new_data$last_pymnt_d = NULL
new_data$issue_d = NULL
```

Feature Selection

```
# Numeric columns only
numerics = new_data[,sapply(new_data, is.numeric)]
dim(numerics)
```

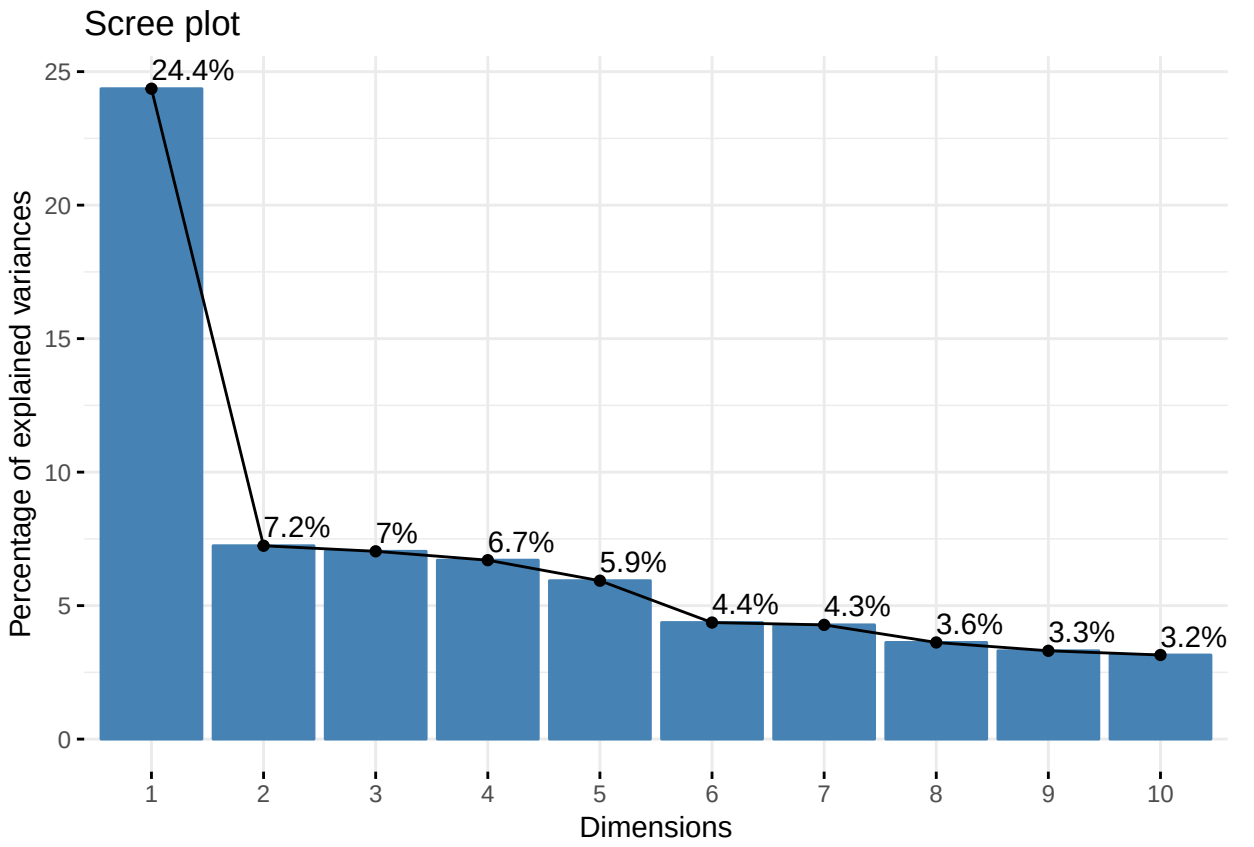
```
## [1] 254190     49
```

```
na_colnames = colnames(new_data[,which(colSums(is.na(new_data)) >  nrow(new_data)*0.81)])
na_colnames
```

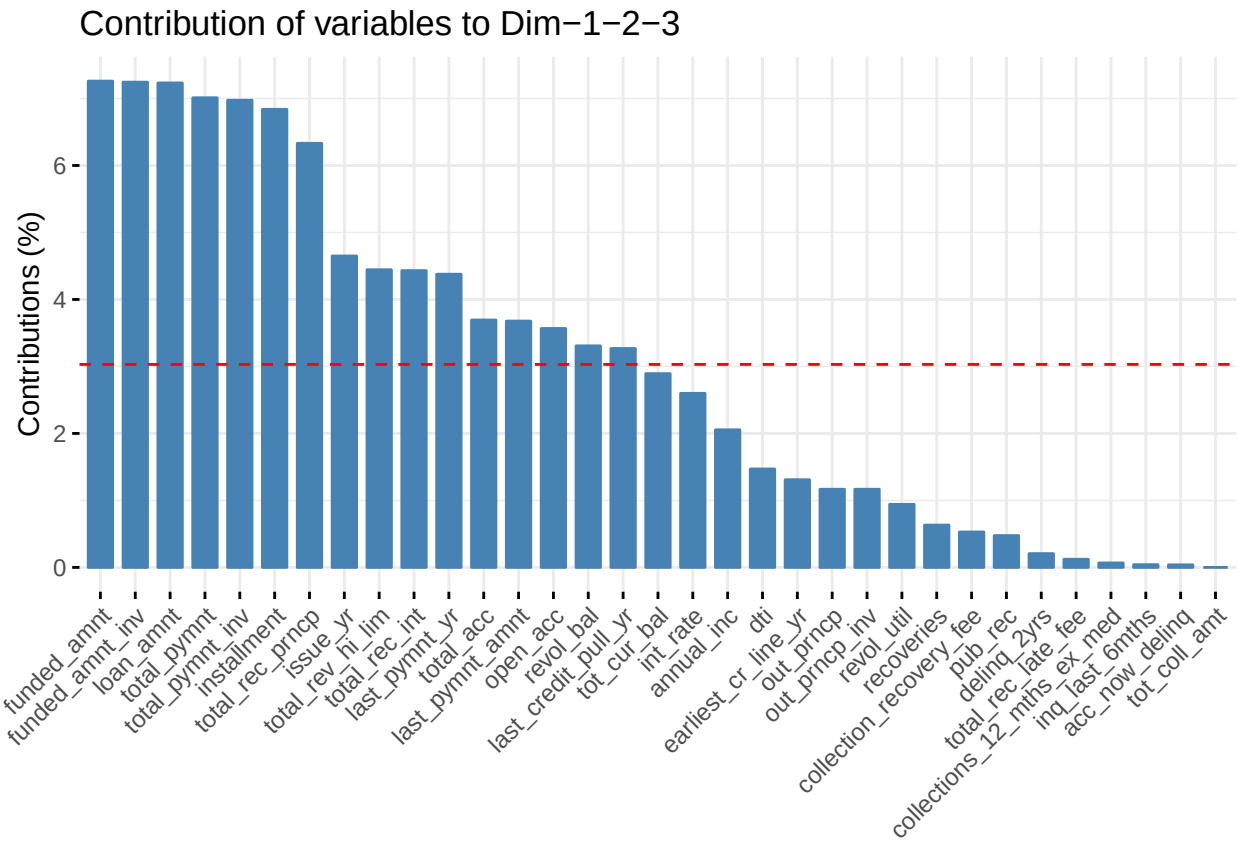
```
##  [1] "mths_since_last_major_derog" "open_acc_6m"
##  [3] "open_il_6m"                  "open_il_12m"
##  [5] "open_il_24m"                 "mths_since_rcnt_il"
##  [7] "total_bal_il"                "il_util"
##  [9] "open_rv_12m"                 "open_rv_24m"
## [11] "max_bal_bc"                  "all_util"
## [13] "inq_fi"                      "total_cu_tl"
## [15] "inq_last_12m"
```

```
# Remove response variable
numerics$loan_status_numeric = NULL
# Remove any columns with NA values greater than 81%
numerics_no_na = numerics[,-which(colnames(numerics) %in% na_colnames)]
```

```
pca = PCA(numerics_no_na, scale.unit = TRUE, graph = FALSE)
fviz_eig(pca, addlabels = TRUE)
```




```
fviz_contrib(pca, choice = "var", axes = 1:3)
```




```
# Extract top 15 contributors to the first three PCs
contrib_colnames = fviz_contrib(pca, choice = "var", axes = 1:3)$data$name
contribs = fviz_contrib(pca, choice = "var", axes = 1:3)$data$contrib
contrib_colnames = contrib_colnames[order(contribs, decreasing = TRUE)]
top_15_colnames = contrib_colnames[1:15]
top_15 = numerics[,which(colnames(numerics) %in% top_15_colnames)]
top_15_colnames
```

```
##  [1] funded_amnt      funded_amnt_inv  loan_amnt        total_pymnt
##  [5] total_pymnt_inv  installment      total_rec_prncp  issue_yr
##  [9] total_rev_hi_lim total_rec_int    last_pymnt_yr    total_acc
## [13] last_pymnt_amnt  open_acc         revol_bal
## 33 Levels: loan_amnt funded_amnt funded_amnt_inv int_rate ... issue_yr
```

```
# Extract categorical variables
categoricals = new_data[,sapply(new_data, is.factor)]
```

```
# Remove additional columns
categoricals$zip_code = NULL
categoricals$application_type = NULL
categoricals$loan_status = NULL
categoricals$sub_grade = NULL
categoricals$pymnt_plan = NULL
```

- *zip_code*: Zip code data is too granular for our analysis, to capture locational information we can use state.
- *application_type*: all but one observation in our completed loans are "INDIVIDUAL", the single observation being "JOINT". This will not add any significant information with one observation.
- *sub_grade*: Sub grade data is too granular for our analysis, to capture loan quality information we can

use “grade”.

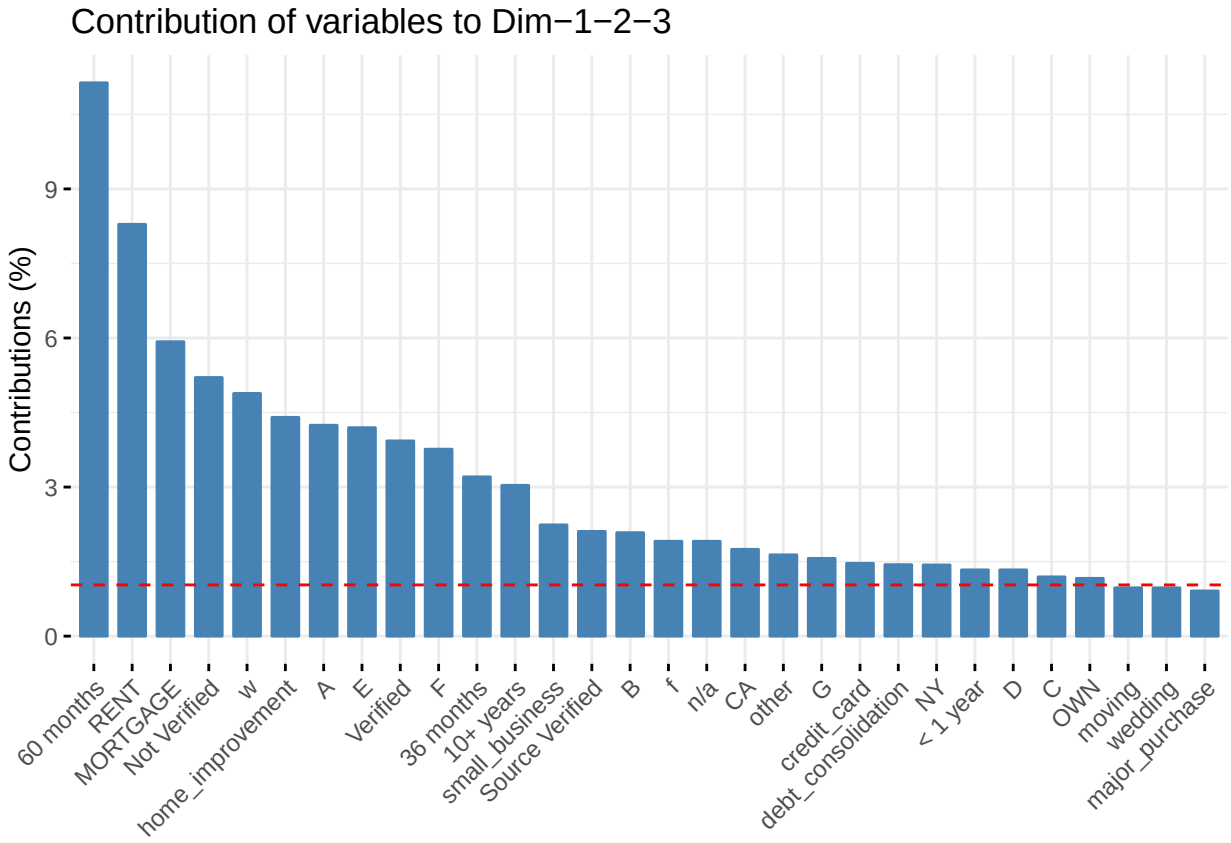
- *pymnt_plan*: Only two observations with value “y”, the rest are “n”

```
colnames(categoricals)
```

```
## [1] "term"                "grade"                "emp_length"
## [4] "home_ownership"      "verification_status" "purpose"
## [7] "addr_state"          "initial_list_status"
```

```
mca = MCA(categoricals, graph = FALSE)
```

```
fviz_contrib(mca, choice = "var", axes = 1:3, top = 30)
```




```
##  [1] "loan_amnt"           "funded_amnt"         "funded_amnt_inv"
##  [4] "term"                "installment"         "grade"
##  [7] "emp_length"          "home_ownership"      "verification_status"
## [10] "purpose"             "addr_state"          "open_acc"
## [13] "revol_bal"           "total_acc"           "initial_list_status"
## [16] "total_pymnt"         "total_pymnt_inv"     "total_rec_prncp"
## [19] "total_rec_int"       "last_pymnt_amnt"     "last_pymnt_yr"
## [22] "issue_yr"
```

As seen, the “last_pymnt_yr” variable has some remaining NA values:

```
colSums(is.na(model_1_data))
```

```
##          loan_amnt        funded_amnt    funded_amnt_inv
##                  0                  0                  0
##               term        installment              grade
##                  0                  0                  0
```

```
##          emp_length      home_ownership verification_status
##                   0                   0                   0
##             purpose          addr_state            open_acc
##                   0                   0                   0
##           revol_bal           total_acc initial_list_status
##                   0                   0                   0
##         total_pymnt     total_pymnt_inv     total_rec_prncp
##                   0                   0                   0
##       total_rec_int     last_pymnt_amnt       last_pymnt_yr
##                   0                   0                 536
##            issue_yr
##                   0
```

Since the number of occurences is low relative to the size of our dataset, we impute the value based on the column mean:

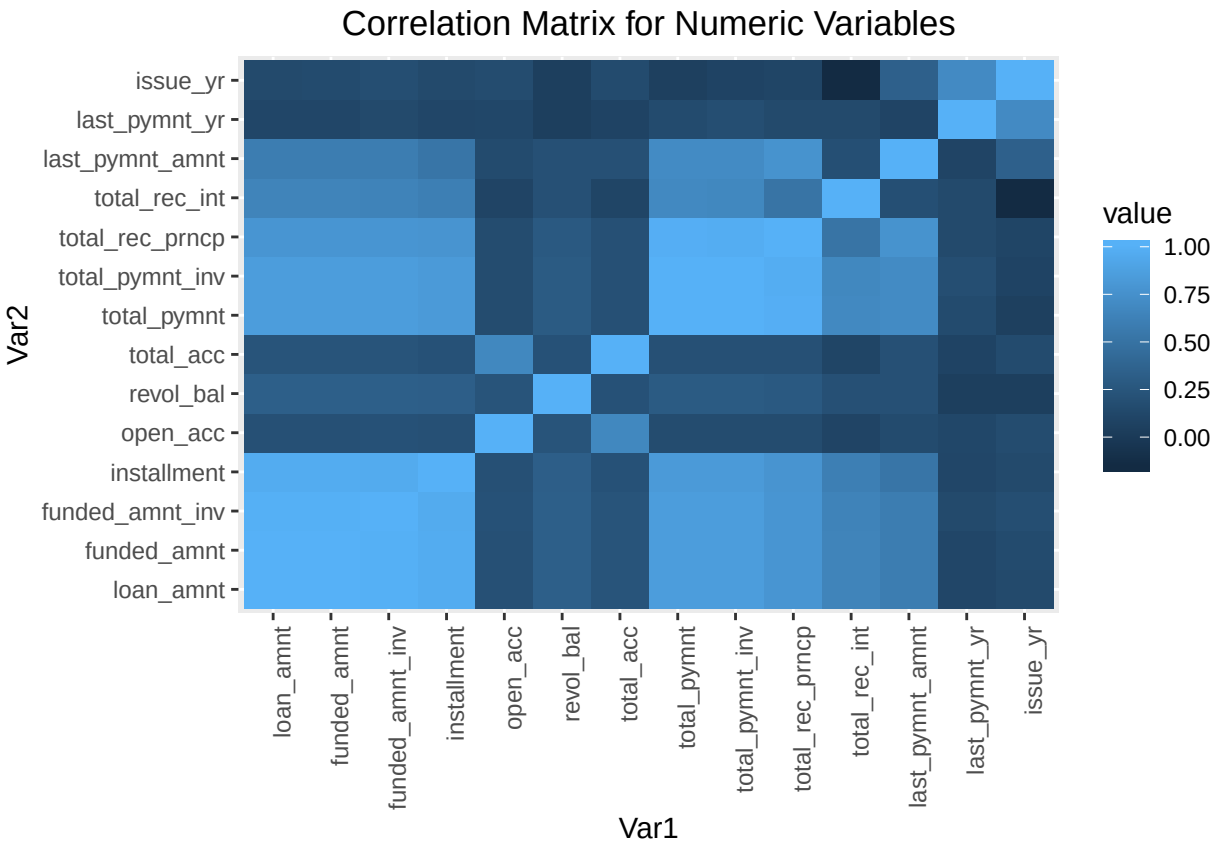
```
model_1_data$last_pymnt_yr[is.na(model_1_data$last_pymnt_yr)] = mean(model_1_data$last_pymnt_yr, na.rm='
```

Model Proposals:

Here we display the correlation heatmap which will influence our model choice:

```
# Create new correlation heatmap
mod_1_numerics = model_1_data[,sapply(model_1_data, is.numeric)]
cormat <- round(cor(mod_1_numerics),2)
melted_cormat <- melt(cormat)

# plot
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) + ggtitle("Correlation Matrix for Numeric
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  theme(plot.title = element_text(hjust = 0.5)) + geom_tile()
```



Due to the appearance of highly correlated predictors, we attempt to handle those using Ridge Regression, and other penalized models.

```
# Convert response to binary factor for caret classification
model_1_data$loan_status_numeric = ifelse(new_data$loan_status == "Fully Paid", 0, 1)
caret_model_data = model_1_data
caret_model_data$loan_status_numeric = as.factor(model_1_data$loan_status_numeric)
levels(caret_model_data$loan_status_numeric) = c("good", "bad")

# Split train and test data
c_test_idx = sample(1:nrow(caret_model_data),size = .2 * nrow(caret_model_data))
c_test = caret_model_data[c_test_idx,]
c_train = caret_model_data[-c_test_idx,]
```

Here we do a 3-fold cross validated ridge regression model, with lambda values 1 through 5. Due to very high computational cost and runtime, we reduce the number of folds and increase the incrementing of the tuning parameter lambda. In the original model, we trained a larger interval of lambda values at smaller increments, but here we reduce the scope of our interval but include the lambda from the original model in the interval:

```
# 3-fold CV to decrease knit time
# note that this was trained with a larger range of lambdas,
# but it was reduced due to selection of small lambda
control = trainControl(method = "cv", number = 3, verboseIter = TRUE, classProbs = TRUE)
grid = expand.grid(seq(0,5,by=1)) # Using ridge reg
colnames(grid) = c("lambda")
ridge_train = c_train
ridge_train$loan_status_numeric = as.numeric(ifelse(ridge_train$loan_status_numeric == "bad", 1, 0))
```

```
ridgeFit <- train(loan_status_numeric ~ ., data = ridge_train, method = 'ridge',
                  family = "binomial",
                  preProc = c("center", "scale"), metric = "kappa",
                  trControl = control,
                  tuneGrid = grid)
```

We attempted to use Cohen's Kappa metric to account for class probabilities, however the model did not allow for this, and instead used accuracy.

As seen, the final lambda selected was 0, thus we are not shrinking any of the model inputs. Though we thought Ridge would assist in ridding highly correlated variables, the benefits are not seen in our model since the "best shrinkage is"best" value of lambda was selected as 0.

Below we show performance metrics based on a binary classification with a decision threshold equal to the proportion of "bad" loans in our dataset (prior probability):

```
# Prior class (default loan) probability
# prb_default = sum(as.numeric(ifelse(c_train$loan_status_numeric == "good",0,1))) / nrow(c_train)
prb_default = 0.1824472
```

```
# Adjust decision boundary based on prior probability
confusionMatrix(as.factor(ifelse(predict(ridgeFit, newdata = c_test) > prb_default, "bad", "good")),
                reference = c_test$loan_status_numeric)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  good   bad
##       good 40095   164
##       bad   1377  9202
##
##                Accuracy : 0.9697
##                  95% CI : (0.9682, 0.9712)
##     No Information Rate : 0.8158
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.904
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9668
##             Specificity : 0.9825
##          Pos Pred Value : 0.9959
##          Neg Pred Value : 0.8698
##              Prevalence : 0.8158
##          Detection Rate : 0.7887
##    Detection Prevalence : 0.7919
##       Balanced Accuracy : 0.9746
##
##        'Positive' Class : good
##
```

Above we show a combination of metrics that we can use to calculate model performance. As seen, the accuracy metric is very high, but that is expected with this dataset, due to the imbalanced response between "Fully Paid" and "Default/Charged Off" loans. We read into the Cohen's Kappa metric, as it takes into account some imbalance in the dataset, however we could not tune the model with the above train function based on this metric. However, it seems reasonably high for this model. It is interesting to note that the

model has a higher "false-positive" error rate, in that it falsely identifies loans as "Default/Charged Off", more often than falsely identifying a loan as "Fully Paid", which is an opposing trend to the imbalance of the data. Note that the Sensitivity and Specifity metrics use "good" loans ("Fully Paid") as the positive response, as opposed to previous models where the binary value was 1 for "bad" loans.

Similarly we use a demonstrational range of lambda values for Lasso, as the orginal model took far too long to train. The folds are also reduced to 3. We attempted Lasso Regression to penalize predictors, as Ridge did not:

```
control = trainControl(method = "cv", number = 3, verboseIter = TRUE, classProbs = TRUE)
grid = expand.grid(1, seq(0,10,by=2)) # Using lasso reg
colnames(grid) = c("alpha", "lambda")

lassoFit <- train(loan_status_numeric ~ ., data = c_train, method = 'glmnet',
                  family = "binomial",
                  preProc = c("center", "scale"), metric = "kappa",
                  trControl = control,
                  tuneGrid = grid)
```

Again, attempted to use Cohen's Kappa metric to account for class probabilities, however the model did not allow for this, and instead used accuracy.

Again, penalization did not change our model since the lambda parameter was tuned to 0 shrinkage. Due to these to failed attempts at reducing model complexity and handling correlated predictors, we looked outside of penalized regression for ways to reduce complexity.

Again we show performance metrics based on a binary classification with a decision threshold equal to the proportion of "bad" loans in our dataset (prior probability):

```
# Adjust decision boundary based on prior probability
confusionMatrix(as.factor(ifelse(predict(lassoFit, newdata = c_test, type = "prob")$bad > prb_default,
                reference = c_test$loan_status_numeric)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  good   bad
##       good 41425    87
##       bad     47  9279
##
##                Accuracy : 0.9974
##                  95% CI : (0.9969, 0.9978)
##     No Information Rate : 0.8158
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##  Mcnemar's Test P-Value : 0.0007542
##
##             Sensitivity : 0.9989
##             Specificity : 0.9907
##          Pos Pred Value : 0.9979
##          Neg Pred Value : 0.9950
##              Prevalence : 0.8158
##          Detection Rate : 0.8148
##    Detection Prevalence : 0.8166
##       Balanced Accuracy : 0.9948
##
```

```
##        'Positive' Class : good
##
```

This model appears to perform better on almost all metrics, most notably with a Kappa value increase of approximately 0.09. However, such high accuracy raises some concerns and suspicions, especially with the mentioned imbalance in our dataset. However, the Specificty (true negative rate) is increased from the previous model, which is an important metric, and indicates that the model is not "blindly" assigning loan status to "good" due to the imbalance.

We attempt to reconcile the suspicious model performance by fitting a different model using weighted logistic regression to handle the imbalance (roughly 4 to 1):

```
wts = ifelse(c_train$loan_status_numeric == "bad", 4, 1)
weightedFit1 = glm(loan_status_numeric~., family = "binomial", data = c_train, weights = wts)
```

Because we encoded the imbalance to our model, we can now return our decision boundary to a default 50% probability:

```
confusionMatrix(as.factor(ifelse(predict(weightedFit1, newdata = c_test, type = "response") > .5, "bad"
                reference = c_test$loan_status_numeric)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  good   bad
##       good 41396    79
##       bad     76  9287
##
##                Accuracy : 0.997
##                  95% CI : (0.9964, 0.9974)
##     No Information Rate : 0.8158
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9899
##  Mcnemar's Test P-Value : 0.8724
##
##             Sensitivity : 0.9982
##             Specificity : 0.9916
##          Pos Pred Value : 0.9981
##          Neg Pred Value : 0.9919
##              Prevalence : 0.8158
##          Detection Rate : 0.8143
##    Detection Prevalence : 0.8158
##       Balanced Accuracy : 0.9949
##
##        'Positive' Class : good
##
```

This model performs very similar to the attempt at Lasso regression, however we chose to continue with this model in favor of the either two because the penalized models did not actually implement any real penalization. The 'glm' function with weighted logistic regression returns a much more manageable model that we can modify more easily than the returned models of the 'caret' package. With such similar performance metrics, including Cohen's Kappa and specificity, which we expressed interest in above, we progress with the simpler model.

The next step to further simplify our model was to apply a backward step search using AIC as a metric (we do not run the code in this report but instead gather the final formula that results from this function call):

```
# This code is commented out, as it took upwards of 1.5 hours to complete

# aic_back_model = step(weightedFit1, direction = "backward", trace = FALSE)
```

Here we train the result of the AIC backwards search:

```
aic_back_model = glm(formula = loan_status_numeric ~ funded_amnt + funded_amnt_inv +
    installment + grade + emp_length + home_ownership + verification_status +
    purpose + addr_state + open_acc + initial_list_status + total_pymnt +
    total_pymnt_inv + total_rec_prncp + total_rec_int + last_pymnt_amnt +
    last_pymnt_yr + issue_yr, family = "binomial", data = c_train,
    weights = wts)
```

The resulting model has four less predictors:

```
length(coef(aic_back_model))
```

```
## [1] 100
```

```
length(coef(weightedFit1))
```

```
## [1] 104
```

```
coef(weightedFit1)[!names(coef(weightedFit1)) %in% names(coef(aic_back_model))]
```

```
##      loan_amnt term 60 months      revol_bal      total_acc
##   2.985819e-05   5.825959e-01  -7.504631e-08   2.022393e-03
```

“loan _amnt“,”term 60 months“,”revol_bal“, and”total_acc" were removed.

```
confusionMatrix(as.factor(ifelse(predict(aic_back_model, newdata = c_test, type = "response") > .5, "ba
                reference = c_test$loan_status_numeric)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  good   bad
##       good 41398    80
##       bad     74  9286
##
##                Accuracy : 0.997
##                  95% CI : (0.9965, 0.9974)
##     No Information Rate : 0.8158
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9899
##  Mcnemar's Test P-Value : 0.687
##
##             Sensitivity : 0.9982
##             Specificity : 0.9915
##          Pos Pred Value : 0.9981
##          Neg Pred Value : 0.9921
##              Prevalence : 0.8158
##          Detection Rate : 0.8143
##    Detection Prevalence : 0.8159
##       Balanced Accuracy : 0.9948
##
##        'Positive' Class : good
```

##

The performance metrics remain high, with slightly less model complexity, thus we select this as our final logistic regression model.

Conclusions:

Here we print the coefficients sorted by value, without the "state" predictors:

```
coef_df = as.data.frame(aic_back_model$coefficients)
state_row_idx = 42:91
coef_no_state = aic_back_model$coefficients[-state_row_idx]
sort(coef_no_state)
```

```
##                 home_ownershipNONE                       last_pymnt_yr
##                        -2.046104606                        -0.527362768
##                  purposeeducational              purposerenewable_energy
##                        -0.382719749                        -0.317358975
##               purposemajor_purchase                 initial_list_statusw
##                        -0.285538110                        -0.201838370
##                       purposemoving verification_statusSource Verified
##                        -0.183097986                        -0.169160492
##             purposehome_improvement                      total_rec_prncp
##                        -0.127709732                        -0.097396683
##                       total_rec_int         verification_statusVerified
##                        -0.094357513                        -0.036085812
##                  purposecredit_card                    emp_length5 years
##                        -0.035799893                        -0.011365252
##                   emp_length3 years                      funded_amnt_inv
##                        -0.004850863                        -0.004379442
##                     last_pymnt_amnt                      total_pymnt_inv
##                        -0.002028673                         0.003922376
##                         installment                         funded_amnt
##                         0.005898922                         0.006406107
##                            open_acc                      purposevacation
##                         0.022647654                         0.027960670
##                   emp_length1 year                    emp_length6 years
##                         0.031333735                         0.043799677
##                   emp_length4 years                       purposeother
##                         0.055089384                         0.063764009
##                              gradeE                              gradeF
##                         0.076568953                         0.079505245
##           purposedebt_consolidation                         total_pymnt
##                         0.082952230                         0.091000526
##                        purposehouse                      purposemedical
##                         0.097229331                         0.106578528
##                   emp_length9 years                    emp_length2 years
##                         0.147265013                         0.185685862
##                              gradeD                              gradeB
##                         0.201983887                         0.232188825
##                   emp_length8 years                              gradeC
##                         0.238086951                         0.250909765
##                  emp_length10+ years                    emp_length7 years
##                         0.260548404                         0.305229153
```

```
##              purposesmall_business                          issue_yr
##                        0.409911270                       0.488949069
##                      purposewedding                     emp_lengthn/a
##                        0.534770878                       0.797623121
##                             gradeG                home_ownershipOTHER
##                        1.600155604                       3.545121844
##                   home_ownershipRENT             home_ownershipMORTGAGE
##                        4.214176165                       4.253559865
##                   home_ownershipOWN                       (Intercept)
##                        4.271184739                      71.417760706
```

Some important inputs to our model:

- *home_ownership*: Any sort of home ownership (other, mortgage, own, rent) contribute the most to the probability of a "bad" loan, lack of home ownership decreases probability of a "bad" loan.
- *grade*: Decrease in loan grades corresponds to higher probability of a "bad" loan. This makes sense as Lending Club uses risk metrics to calculate loan grade.
- *purpose*: Wedding and small business loans contribute to higher probability of "bad loan", and educational loans, loans for major purchases, and renewable energy tend to get paid in full.
- *issue_yr*: Though this is a high contributor to loan default probability, we cannot confidently apply this metric to current loans, as the lifetime of the loan is yet to be decided.
- *last_pymnt_yr*: More recent payments are indicative to less likelihood of defaults.

The state predictors reflect the information shown on the percentage default plot by state in the prior sectio:

```
state_row_idx = 42:91
coef_state = aic_back_model$coefficients[state_row_idx]
sort(coef_state)
```

```
##  addr_stateND  addr_stateME  addr_stateID  addr_stateMS  addr_stateWY
## -13.382004572 -10.790442573 -10.564985290  -1.417245897  -0.951177052
##  addr_stateIA  addr_stateIN  addr_stateDC  addr_stateSC  addr_stateDE
##  -0.639112488  -0.125671439  -0.020199146  -0.007991824   0.178397586
##  addr_stateSD  addr_stateNE  addr_stateRI  addr_stateNM  addr_stateMT
##   0.196563631   0.224973782   0.229315803   0.342827979   0.408251056
##  addr_stateKY  addr_stateMA  addr_stateNJ  addr_stateGA  addr_stateWI
##   0.456263678   0.462344696   0.469745905   0.485945138   0.524435308
##  addr_stateTX  addr_stateAZ  addr_stateLA  addr_stateWV  addr_stateMD
##   0.562833578   0.563577366   0.574710810   0.585594605   0.587968316
##  addr_stateMI  addr_stateOH  addr_stateNY  addr_statePA  addr_stateIL
##   0.613974424   0.615232207   0.618564565   0.637374786   0.645132448
##  addr_stateCT  addr_stateNC  addr_stateVT  addr_stateWA  addr_stateOK
##   0.676104190   0.679401969   0.681381292   0.705970132   0.728489888
##  addr_stateCA  addr_stateFL  addr_stateCO  addr_stateKS  addr_stateVA
##   0.738660229   0.770460354   0.837429465   0.874936977   0.884544203
##  addr_stateOR  addr_stateAL  addr_stateMO  addr_stateUT  addr_stateAR
##   0.900298602   0.915422428   0.952607708   0.966772364   1.117521553
##  addr_stateTN  addr_stateNV  addr_stateNH  addr_stateMN  addr_stateHI
##   1.121236703   1.138813005   1.188875332   1.201509576   1.265822487
```

Some points of interest are the very strong negative correlations of ND, ME, and ID to the likelihood of default. However, this due to a lack of observations from those states. There are less than 100 loans from the three states combined, so we cannot assume the behavior of those loans reflects the greater population.

- The final model contains 100 coefficients and an accuracy rate of .997 which is abnormally high. In

addition, when we sorted the state coefficients we found they exactly matched our visualization which tells us our model coefficients reflect the behavior of that portion of our dataset. One oddity was with our factor variable grade, the order of increasing coefficient size should be A,B,C,D,E,F,G which is what our visualization of grade vs percentage default explains. In our model, a grade of C had greater contribution to the likelihood of a default than E or F.

- As you can see from our final model prediction results, our final model mispredicted that 79 "Fully Paid" loans would be "Default" and mispredicted that 72 "Default" Loans would be "Fully Paid". Although it seems like this is not a lot proportionally, this could be a large factor, especially since Lending Club draws directly from individual investors who may suffer from this risk (though we assume Lending Club will absorb most of the damage). We can also use these results to confirm to banks that the methods used to determine loan payment are very successful and that better management of tracking borrowers' spending habits will decrease the amount of people who default on their loan.

Discussion

- One of the largest pitfalls of our analysis was the effect of our time constraints, especially with training on such a large dataset. We ran backward selection on the best model we found in order to further remove variables, but this took upwards 1.5 hours to run. Because of this, we found it difficult to attempt any more complex (i.e. non-linear) or varying models, as it simply would take too much time/resources. Another pitfall in our analysis came from our data set being imbalanced. For our response variable, Loan Status, we had about 4 times more "Fully Paid" loans compared to "Default" loans and this may have made our analysis less accurate, however we attempted to deal with this by adjusting the decison boundaries and also attempting weighted regression. We could have done more research to better deal with this issue, but again were constrained by time. Dealing with NA values was an additional pitfall because we had to spend a lot of time on deciding on how to deal with the columns that had them, and at times it was difficult to properly determine the significance of these NA values. Our analysis could have been improved if we had found a better way to deal with the NA values in our data, as they were very prevalent.
- With the modeling prcoess in particular we performed the analysis under the assumption of underlying linear relations to our predictors, but as mentioned prior, more complex or a variation of models could not be exhaustively assessed based on resource/time constraints. If we were concerned primarily with model accuracy or with the granularity of exaclty how an input affected loan status, we should have attempted to identify the distributions of the covariates and adjust our model selection accordingly. However, since our goal was to get a more general sense of contributions to lona defaults, our linear model was acceptable.
- In reference to our predictions, another pitfall of this model is that it is trained solely based off of completed loans (needed for supervised learning), however the behavior of current loans is likely much different. Some of the influential predictors referenced information that could only be viewed holistically across the lifetime of the loan, and would be much harder to analyze during the lifetime of the loan.