Payoff_Takehome_Assessment

Vaibhav Walvekar

January 10, 2017

Dataset details: The lending club dataset is a collection of installment loan records, including credit grid data (e.g. FICO, revolving balance, etc.) and loan performance (e.g. loan status).

The data is stored in a postgres database on AWS. Please use the below information to connect to the database with your tool of choice to access the data (R, Python, SQL, etc.)

There are 4 tables for you to use: . lending_club_2007_2011 . lending_club_2012_2013 . lending_club_2014 . lending_club_2015

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## filter(): dplyr, stats
## lag():    dplyr, stats

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'MASS'

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

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
## Loaded glmnet 2.0-5
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
## Loading required package: RPostgreSQL
## Loading required package: DBI
```

```
dim(lending_club_consolidated)
summary(lending_club_consolidated)
```

Based on the summary, cleaning consolidated lending dataset

```
#Deleting row containing Memberid as NA (all columns are NA for this observation)
lending_club_consolidated <- lending_club_consolidated[!is.na(
  lending_club_consolidated$member_id),]

#Converting id to numeric datatype
lending_club_consolidated$id = as.numeric(lending_club_consolidated$id)

#Creating a new column from issue_d as a date datatype
lending_club_consolidated$issue_date<-as.Date(as.yearmon(lending_club_consolidated$issue_d,
                                                         format = "%b-%Y"))

#Creating a new column from earliest_cr_line as a date datatype
lending_club_consolidated$earliest_cr_line_date<-
  as.Date(as.yearmon(lending_club_consolidated$earliest_cr_line, format = "%b-%Y"))

#Converting interest rate to numeric datatype
lending_club_consolidated$int_rate = as.numeric(gsub("\\%", "",
                                                     lending_club_consolidated$int_rate))

#Converting revol_util to numeric datatype
lending_club_consolidated$revol_util = as.numeric(gsub("\\%", "",
                                                       lending_club_consolidated$revol_util))

#Converting term to numeric datatype
lending_club_consolidated$term <- as.numeric(substr
                                             (lending_club_consolidated$term,0,3))
```

```r
#Renaming columns to indicate correct units
lending_club_consolidated <- dplyr::rename(lending_club_consolidated,
                                           int_rate_percent=int_rate, revol_util_percent =
                                             revol_util, term_in_months = term)

attach(lending_club_consolidated)
```

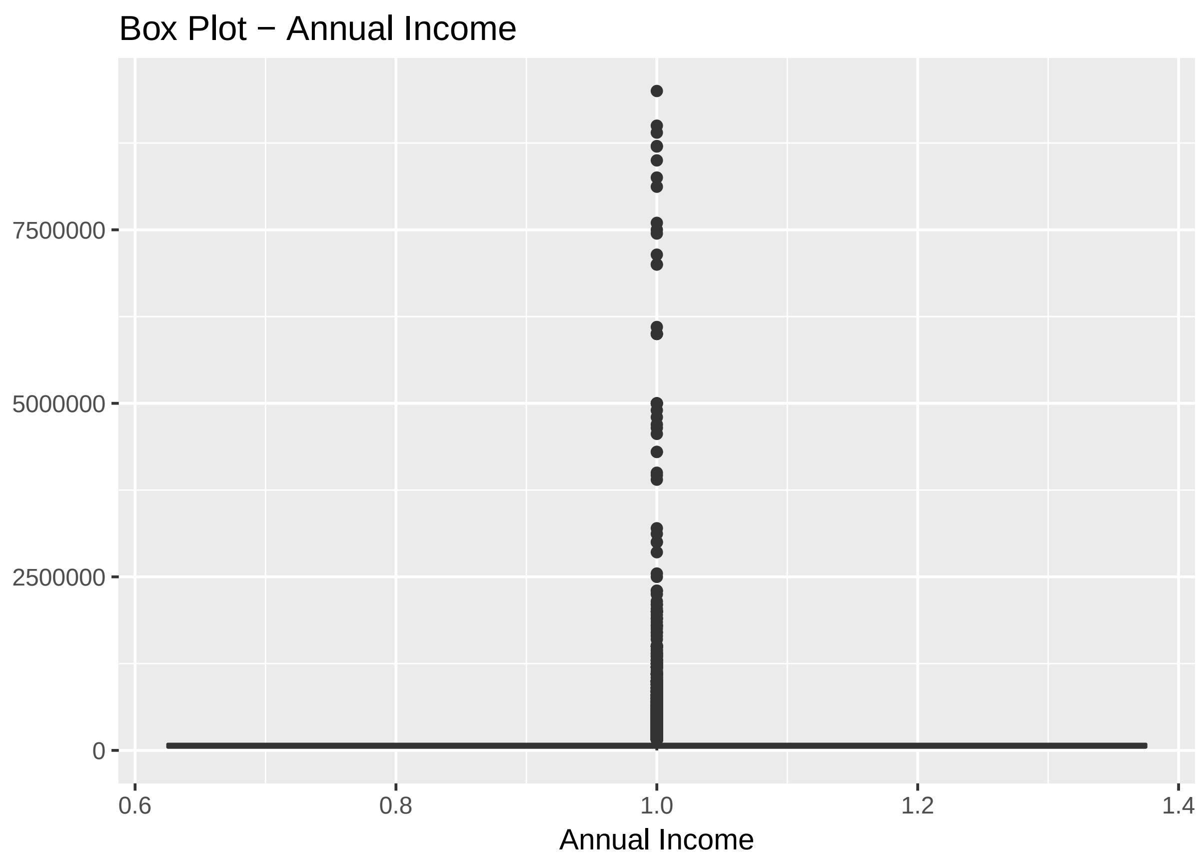
Below are two sections of data questions relating to the Lending Club dataset and another question that is not related to it. Please try and answer all the questions. Quality is much more important than quantity.

Going through the dataset we can understand that Lending club dataset contains information about loan given out to people who have number of different purposes. The information has been captured from 2007 to 2015. There are 111 different columns. Some of the key columns with regards to below analysis are loan_amnt, grade, term, issue_d, loan_status, etc.

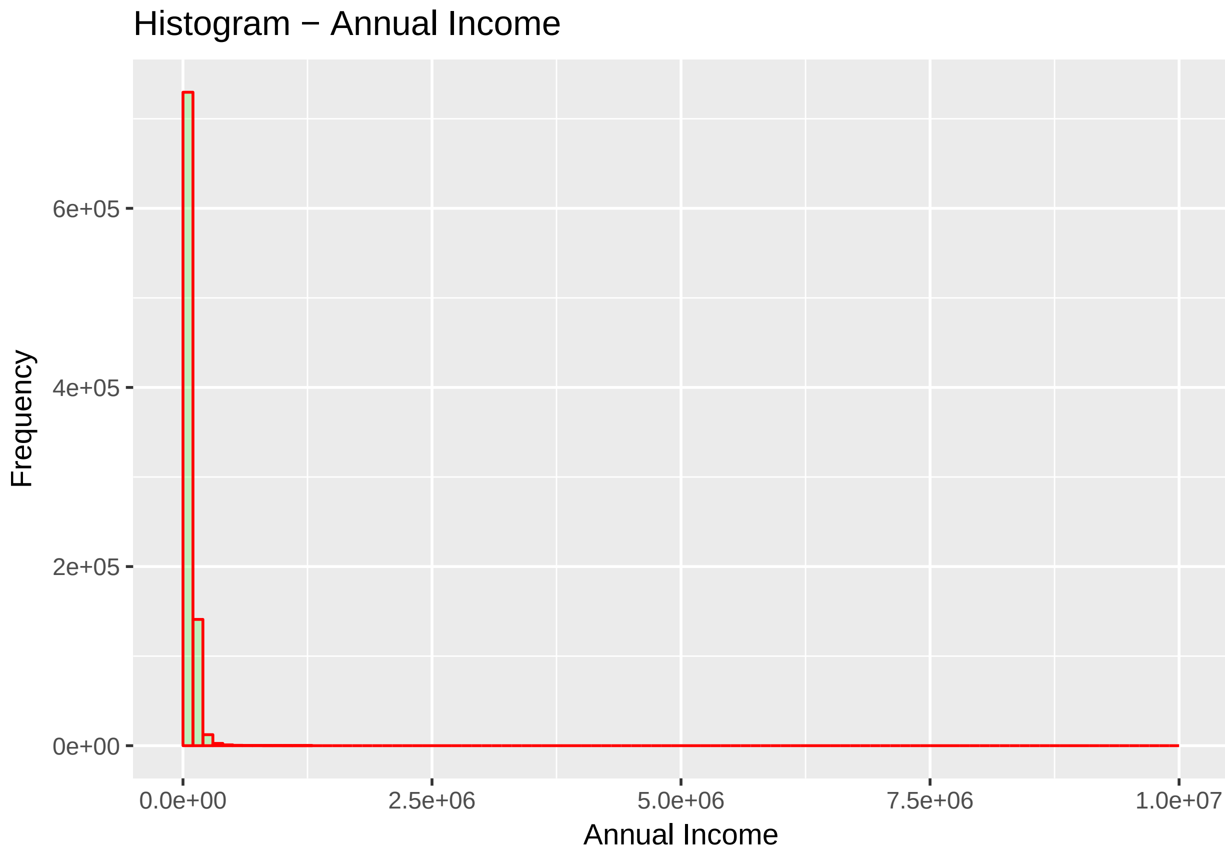
1. Does the data have any outliers?

Outliers are observations that are abnormally out of range of the other values for a random sample from the population. To find out outliers I looked at the summary of the consolidated lending dataset. This helped understand that mostly none of the features had such abnormal observations, except for a couple of important ones like annual_inc and tot_hi_cred_lim.

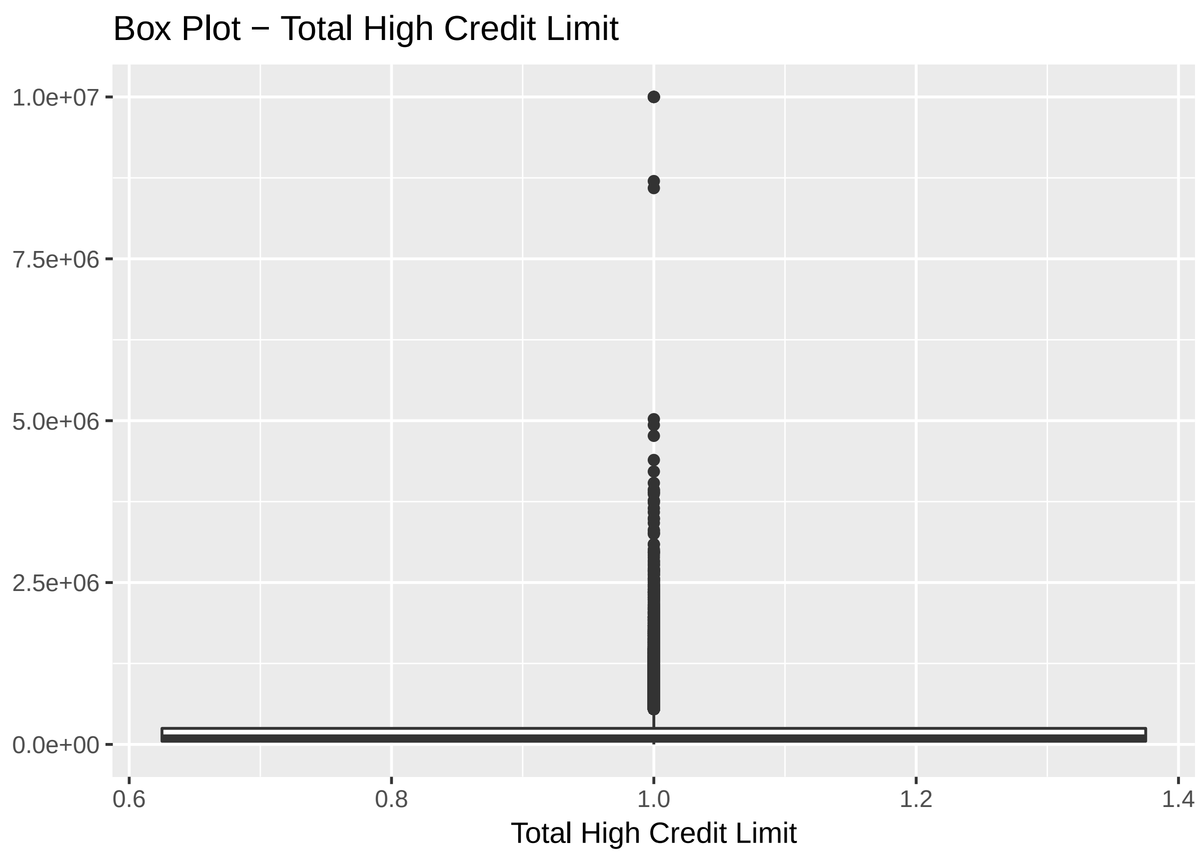
```r
#Removing na from annual_inc column
lending_club_consolidated_filtered_annual_inc <- lending_club_consolidated[!is.na(annual_inc),]

#Box Plot to find outliers - Annual Income
ggplot(data=lending_club_consolidated_filtered_annual_inc,
       aes(y=lending_club_consolidated_filtered_annual_inc$annual_inc, x=1)) + geom_boxplot() +
  labs(title = "Box Plot - Annual Income", x = "Annual Income",  y = "")
```



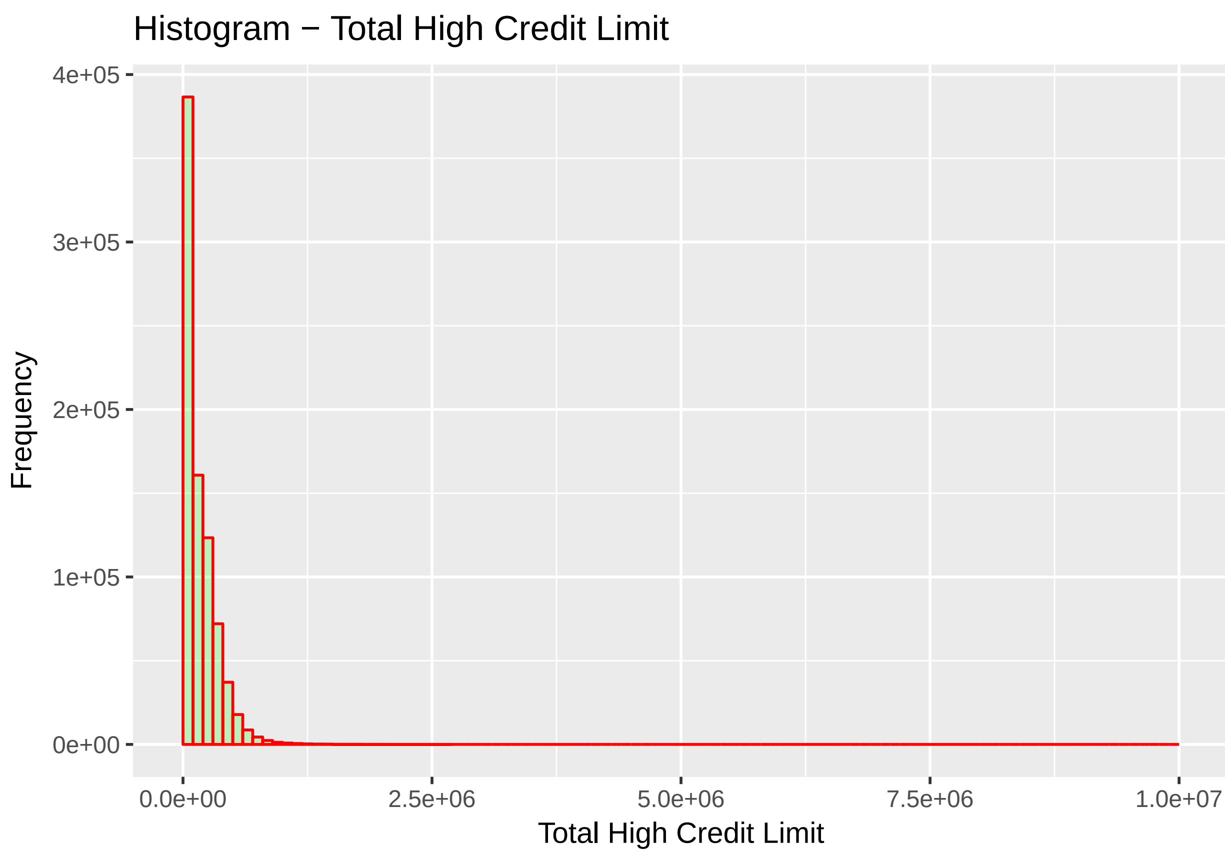

```
#Histogram to check for outliers - Annual Income
ggplot(data=lending_club_consolidated_filtered_annual_inc,
       aes(lending_club_consolidated_filtered_annual_inc$annual_inc)) +
  geom_histogram(breaks=seq(0, 10000000, by = 100000),
                 col="red",
                 fill="green",
                 alpha = .2) +
  labs(title = "Histogram - Annual Income", x = "Annual Income",  y = "Frequency")
```




```
#Removing na from tot_hi_cred_lim column
lending_club_consolidated_filtered_tot_cred <- lending_club_consolidated[!is.na(tot_hi_cred_lim),]

#Box Plot to find outliers - Total High Credit Limit
ggplot(data=lending_club_consolidated_filtered_tot_cred,
       aes(y=lending_club_consolidated_filtered_tot_cred$tot_hi_cred_lim, x=1)) + geom_boxplot() +
  labs(title = "Box Plot - Total High Credit Limit", x = "Total High Credit Limit",  y = "")
```




```
#Histogram to check for outliers - Total High Credit Limit
ggplot(data=lending_club_consolidated_filtered_tot_cred,
       aes(lending_club_consolidated_filtered_tot_cred$tot_hi_cred_lim)) +
  geom_histogram(breaks=seq(0, 10000000, by = 100000),
                 col="red",
                 fill="green",
                 alpha = .2) +
  labs(title = "Histogram - Total High Credit Limit", x = "Total High Credit Limit",
       y = "Frequency")
```

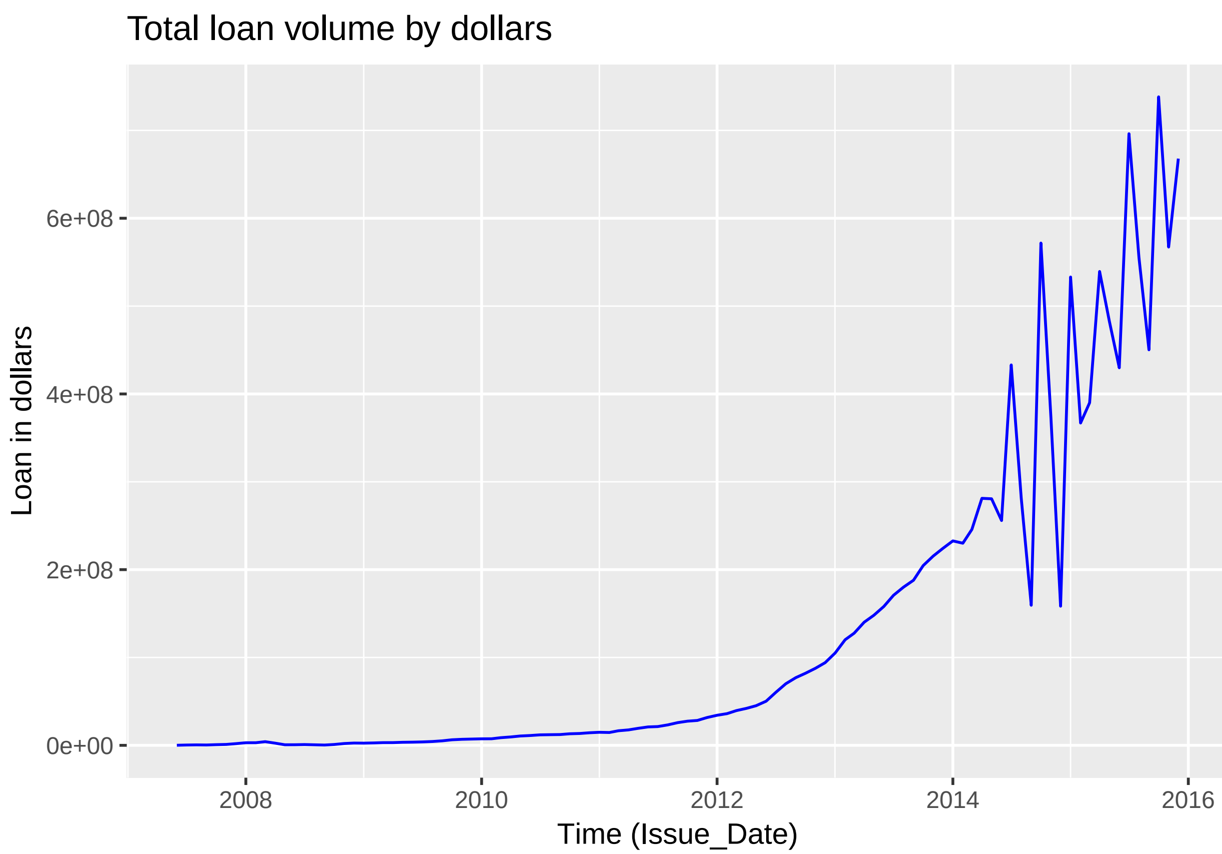


From the above graphics we can see that some of the observations for annual_inc and tot_hi_cred_lim are outliers. This becomes very evident from the box plot where the 1st and 3rd quartiles are very near to the baseline and other values are abnormally higher. Logically, an annual income of $9500000 is abnormally high for a person applying for a loan. It also is clear from 3rd quartile value being almost 100 times lesser. Similarly for a credit limit of $9999999, is abnormally high when compared to 3rd quartile values is around $250000. Thus there are some outliers in the dataset which may have been captured due to wrong entry by the loan applicant.

2. What is the monthly total loan volume by dollars and by average loan size?

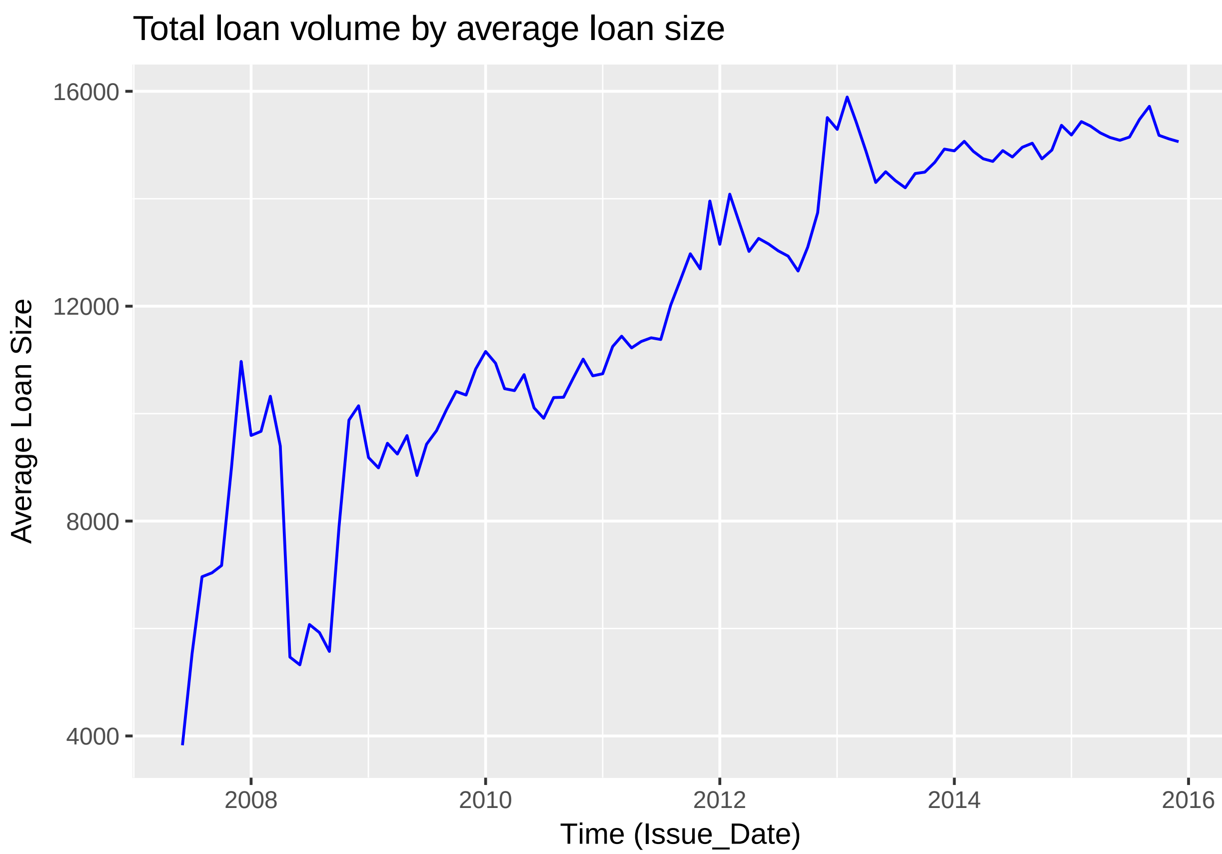
For us to look at the monthly trend of loan volume, we need group together loans issued in individual months. Following on that we can calculate monthly total loan volume by dollars and monthly total loan volume by average loan size.

```
#Grouping data by new created variable issue_date
by_issue_date = group_by(lending_club_consolidated,issue_date)

#Calculating total loan sum for each month
total_loan_by_dollars <- summarize(by_issue_date,totalsum = sum(loan_amnt,na.rm = TRUE))

#Plotting the trend of total loan volume by dollars
ggplot(total_loan_by_dollars, aes(x = issue_date, y = totalsum)) +
  geom_line( colour="blue") + labs(title = "Total loan volume by dollars",
                                   x = "Time (Issue_Date)",  y = "Loan in dollars")
```



From the above graphic we can see that the total loan issued per month was almost constant in the period from 2007-12, but after that there is a steep rise until mid of 2014 after which it has been quite fluctuating.

```
#Calculating mean loan amount for each month
total_loan_by_avgsize <- summarize(by_issue_date,mean = mean(loan_amnt,na.rm = TRUE))

#Plotting the trend of total loan volume by average loan size
ggplot(total_loan_by_avgsize, aes(x = issue_date, y = mean)) + geom_line(colour="blue") +
  labs(title = "Total loan volume by average loan size", x = "Time (Issue_Date)",
       y = "Average Loan Size")
```



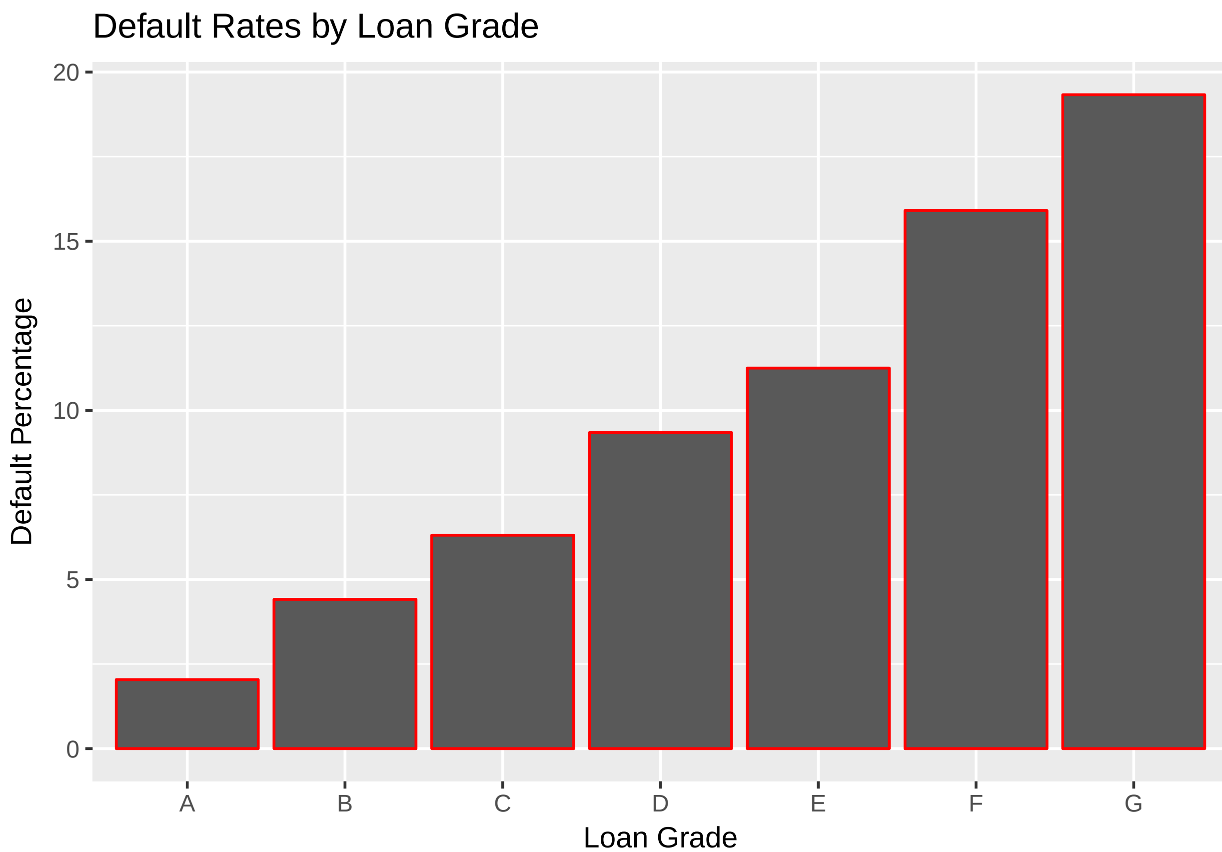
From above graphic we can see that total loan volume by average loan size has been steadily increaseing over the years although a dip is seen in the period between 2008-09. This dip may be on account of the 2008 financial crisis where the average loan issued took a hit.

3. What are the default rates by Loan Grade?

To calculate the default rates, we use the loan_status column which identifies the current status of the loan. As per my knowledge, the status of the loan changes from current to late to default to charged off, if the loan is not payed before due date. Thus in order to calculate the percentage of default in each grade, I have also considered loans which were charged off. I am considering charged of loans because at some stage these loans were in default stage and due to no payment from the loan applicant the status have been moved to charged off.

```
#Calculating proportion of loans in "Default", "Charged off" or
#"Does not meet the credit policy. Status:Charged Off" category per loan grade
prop_by_grade_filtered <- lending_club_consolidated %>%
          group_by(grade,loan_status) %>%
          summarise (n = n()) %>%
          mutate(proportion = n *100/ sum(n)) %>%
          filter(loan_status == "Default" | loan_status== "Charged Off" |
            loan_status == "Does not meet the credit policy. Status:Charged Off")

#Grouping by grade from above output to calculate sum of percentage of
#all three loan status as considered above
by_grade_aggregated = group_by(prop_by_grade_filtered,grade)
default_prop_by_grade <- summarize(by_grade_aggregated,DefaultPercentage =
                                     sum(proportion,na.rm = TRUE))
```

```
#Plotting Default rates by Loan Grade
ggplot(default_prop_by_grade, aes(x = grade, y= DefaultPercentage)) +
  geom_bar(stat = "identity", colour="red") +
  labs(title = "Default Rates by Loan Grade", x = "Loan Grade",  y = "Default Percentage")
```



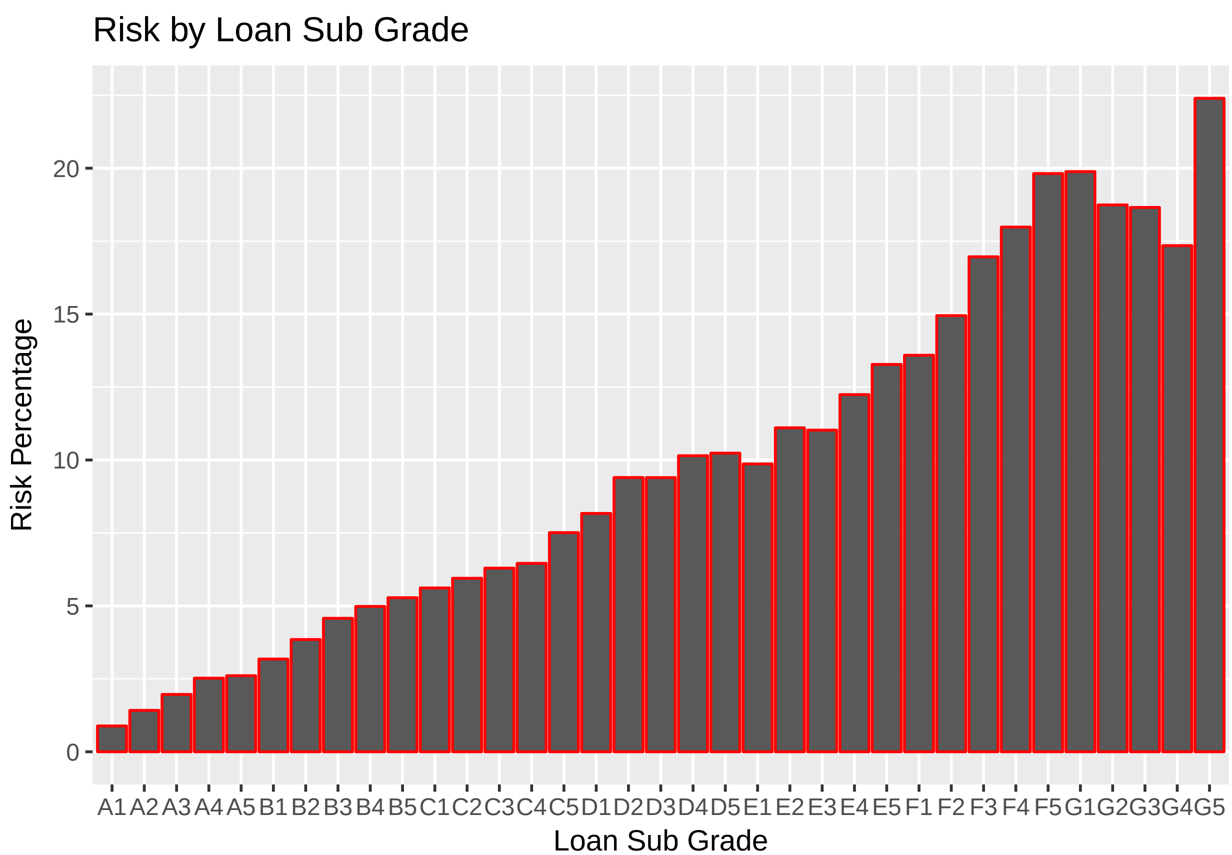
The bar plot shows the sum of default percentages per grade. We can see that the default percentages increase from grade A through G. This is expected as per lending club website because grade A is more risk free than grades through G.

4. Are we charging an appropriate rate for risk?

To answer this question we need to have a measure for risk. Assuming that we consider the likelihood of a loan getting "Charged off", "Default" or "Late" as the risk, we can calculate percentage of loans having such status per subgrade. Using this risk measure we can make plot of subgrade vs. risk. We can also find the correlation between risk and mean interest rate per subgrade, to answer our quastion more appropriately

```
#Calculating proportion of loans in "Late", Default", "Charged off" or
#"Does not meet the credit policy. Status:Charged Off" category per loan sub grade
prop_by_subgrade_filtered <- lending_club_consolidated %>%
          group_by(sub_grade,loan_status) %>%
          summarise (n = n()) %>%
          mutate(proportion = n *100/ sum(n)) %>%
          filter(loan_status == "Default" | loan_status== "Charged Off" |
            loan_status == "Does not meet the credit policy. Status:Charged Off"
                                                           | loan_status == "Late")

#Grouping by sub grade from above output to calculate sum of percentage
#of all four loan status considered as risk
```

```r
by_sub_grade_aggregated = group_by(prop_by_subgrade_filtered,sub_grade)
risk_prop_by_sub_grade <- summarize(by_sub_grade_aggregated,Risk =
                                      sum(proportion,na.rm = TRUE))

#Grouping by subgrade using the original df to calculate accurate mean interest rate per sub grade
by_sub_grade = group_by(lending_club_consolidated,sub_grade)
int_rate_by_sub_grade <- summarize(by_sub_grade,Mean_Interest_Rate =
                                     mean(int_rate_percent,na.rm = TRUE))

#Combining risk and interest rate dfs
risk_int_rate_df <- merge(risk_prop_by_sub_grade, int_rate_by_sub_grade, by = c("sub_grade"))

#Plotting Risk Percentage vs. Sub Grade
ggplot(risk_int_rate_df, aes(x = sub_grade, y= Risk)) + geom_bar(stat = "identity",
                                                                 colour="red") +
  labs(title = "Risk by Loan Sub Grade", x = "Loan Sub Grade",  y = "Risk Percentage")
```




```r
#Finding correlation between Risk and Mean Interest rate per sub grade
cor(risk_int_rate_df$Risk, risk_int_rate_df$Mean_Interest_Rate)
```

```
## [1] 0.976253
```

The correlation between Risk and Mean Interest rate is as high as 0.976. Thus we can say that we are actually charging appropriate rate for the risk. With increase in risk there is an increase in interest rate charged to the customers. From the graphic of Risk by Loan Sub Grade, it is expected that risk percentage should increase as we move from sub grades A1-A5 through G1-G5. This is very much the case, except for ris being less for G2, G3 and G4 than G1 and F5. Thus it could be case of miss categorizing customers to wrong sub grades. But overall, I would say we are charging an appropriate rate for risk.

5. What are the top 5 predictors of default rate by order of importance? Explain the model that you used and discuss how you validated it.

As assumed in the above questions, default on loan is followed by charged off, thus considering charged off status as also default, I am creating a new variable on the dataset which is a categorical variable indicating 1 for default and 0 for not default.

```
default_category = rep (0,nrow(lending_club_consolidated ))

default_category [lending_club_consolidated$loan_status == "Default" |
                    loan_status== "Charged Off" | loan_status ==
                    "Does not meet the credit policy. Status:Charged Off"]=1
default_category<- as.factor(default_category)
lending_club_consolidated = data.frame(lending_club_consolidated ,default_category)

#Removing object for memory management
rm(default_category)
```

As we discovered there are some outliers in our data, I would like to assume that these have been introduced due to human error and thus can removed from the dataset. For annual income, as the 3rd quartile is $90000, I would like to ignore values beyond $1,50,000, this is keeping in mind any person having an annual salary greater than $1,50,000 is less likely to apply for loan of 500 to 30000 dollars. Another outlier found was in tot_hi_cred_lim. The 3rd quartile value is $247777, thus a value of $500000 sounds reasonable for an upper limit.

```
lending_club_consolidated_no_outliers <- filter(lending_club_consolidated,
                                                annual_inc<150000,
                                                tot_hi_cred_lim<500000)
```

Now to create a model to predict default rate, I plan to logically cut down the features to 20-25 as many of the features in the dataset arent very useful for prediction. The required subset of features according to my understanding are captured into the new dataframe as below:

```
#Required columns for model generation
mycols <- c("loan_amnt", "int_rate_percent", "grade", "sub_grade", "annual_inc",
            "verification_status", "term_in_months", "dti", "earliest_cr_line_date",
            "inq_last_6mths", "open_acc", "pub_rec", "revol_bal","revol_util_percent",
            "total_acc", "initial_list_status", "application_type", "acc_now_delinq",
            "delinq_2yrs","installment", "addr_state","issue_date",
            "default_category")

#Creating new dataframe
lending_club_model_df <- lending_club_consolidated_no_outliers[mycols]

#Removing a dataframe for memory management
rm(lending_club_consolidated_no_outliers)

#Ommiting all NA values from the dataset
lending_club_model_df <- na.omit(lending_club_model_df)
```

NA values hinder in building an efficient model and since there are only small portion rows containing NA's, I am ignoring them.

As we are trying to predict a qualitative variable, if there is a default or not on a loan, I plan to use Logistic regression for building the model. Firsty, setting the seed as to achieve same result on each run and avoid different random sampling on each iteration. Secondly, segregating dataset into train and test. Thirdly, running glm function for logistic regression.

Note : Due to computational restrictions, I am reducing the size of dataset to 10% of the actual lending_club_model_df.

```
set.seed(1)
#Reducing size of the dataset because of computational restrictions
reduced_population_size <- sample(nrow(lending_club_model_df),
                                  nrow(lending_club_model_df)*0.1)
reduced_lending_club_model_df <- lending_club_model_df[reduced_population_size, ]

#Segregating training and test data
train <- sample(nrow(reduced_lending_club_model_df),
                nrow(reduced_lending_club_model_df)*0.7)
lending_club_model_df.train <- reduced_lending_club_model_df[train, ]
lending_club_model_df.test <- reduced_lending_club_model_df[-train, ]

#Fitting logistic regression model
logit.fit <- glm ( default_category ~ .,
                  data = lending_club_model_df.train , family = "binomial" )

summary(logit.fit)
```

```
##
## Call:
## glm(formula = default_category ~ ., family = "binomial", data = lending_club_model_df.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5154  -0.3529  -0.2275  -0.1472   3.3735
##
## Coefficients: (6 not defined because of singularities)
##                                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        4.140e+01  1.901e+00  21.781  < 2e-16
## loan_amnt                         -9.204e-06  1.917e-05  -0.480 0.631098
## int_rate_percent                  -1.511e-02  3.473e-02  -0.435 0.663594
## gradeB                             1.443e+00  4.180e-01   3.453 0.000554
## gradeC                             1.838e+00  4.874e-01   3.771 0.000163
## gradeD                             2.444e+00  5.714e-01   4.278 1.88e-05
## gradeE                             2.915e+00  6.713e-01   4.342 1.41e-05
## gradeF                             3.133e+00  7.768e-01   4.033 5.50e-05
## gradeG                             3.338e+00  9.204e-01   3.627 0.000286
## sub_gradeA2                        3.481e-01  4.206e-01   0.828 0.407864
## sub_gradeA3                        4.681e-01  4.038e-01   1.159 0.246334
## sub_gradeA4                        2.758e-01  3.827e-01   0.721 0.471124
## sub_gradeA5                        7.174e-01  3.680e-01   1.950 0.051230
## sub_gradeB1                       -2.940e-01  1.967e-01  -1.495 0.134978
## sub_gradeB2                       -4.479e-01  1.718e-01  -2.606 0.009151
## sub_gradeB3                       -1.441e-01  1.451e-01  -0.993 0.320612
## sub_gradeB4                       -1.266e-01  1.342e-01  -0.943 0.345428
## sub_gradeB5                               NA         NA      NA       NA
## sub_gradeC1                       -3.319e-01  1.533e-01  -2.165 0.030410
## sub_gradeC2                       -1.421e-01  1.377e-01  -1.032 0.302072
## sub_gradeC3                        4.102e-02  1.249e-01   0.328 0.742601
## sub_gradeC4                        1.574e-02  1.204e-01   0.131 0.896039
```

```
## sub_gradeC5                               NA         NA      NA       NA
## sub_gradeD1                       -3.283e-01  1.542e-01  -2.129 0.033282
## sub_gradeD2                       -1.833e-01  1.465e-01  -1.251 0.210997
## sub_gradeD3                       -1.869e-01  1.426e-01  -1.311 0.189934
## sub_gradeD4                       -6.079e-02  1.374e-01  -0.442 0.658177
## sub_gradeD5                               NA         NA      NA       NA
## sub_gradeE1                       -4.798e-01  1.935e-01  -2.480 0.013140
## sub_gradeE2                       -2.246e-01  1.781e-01  -1.261 0.207344
## sub_gradeE3                       -2.887e-01  1.794e-01  -1.609 0.107655
## sub_gradeE4                       -3.942e-01  1.865e-01  -2.114 0.034517
## sub_gradeE5                               NA         NA      NA       NA
## sub_gradeF1                       -4.763e-01  2.873e-01  -1.658 0.097363
## sub_gradeF2                       -3.955e-01  2.956e-01  -1.338 0.180948
## sub_gradeF3                        1.584e-02  2.867e-01   0.055 0.955950
## sub_gradeF4                       -8.655e-02  3.043e-01  -0.284 0.776085
## sub_gradeF5                               NA         NA      NA       NA
## sub_gradeG1                       -2.394e-01  5.604e-01  -0.427 0.669293
## sub_gradeG2                        3.722e-01  5.491e-01   0.678 0.497953
## sub_gradeG3                       -8.440e-02  6.103e-01  -0.138 0.890015
## sub_gradeG4                       -7.772e-01  7.285e-01  -1.067 0.286083
## sub_gradeG5                               NA         NA      NA       NA
## annual_inc                        -8.036e-06  1.054e-06  -7.626 2.43e-14
## verification_statusSource Verified  1.045e-01  5.683e-02   1.839 0.065850
## verification_statusVerified        4.891e-02  5.513e-02   0.887 0.375006
## term_in_months                     1.125e-03  4.911e-03   0.229 0.818793
## dti                                1.073e-02  2.913e-03   3.683 0.000231
## earliest_cr_line_date              3.301e-05  8.745e-06   3.775 0.000160
## inq_last_6mths                     9.576e-02  1.965e-02   4.874 1.09e-06
## open_acc                           7.575e-04  5.692e-03   0.133 0.894118
## pub_rec                           -6.944e-02  4.483e-02  -1.549 0.121417
## revol_bal                          4.648e-07  1.984e-06   0.234 0.814744
## revol_util_percent                -1.114e-03  1.048e-03  -1.063 0.287604
## total_acc                          4.259e-03  2.550e-03   1.670 0.094867
## initial_list_statusw              -7.343e-03  4.352e-02  -0.169 0.865993
## application_typeJOINT             -1.075e+01  1.883e+02  -0.057 0.954488
## acc_now_delinq                    -1.448e-01  3.051e-01  -0.474 0.635215
## delinq_2yrs                       -2.148e-03  2.439e-02  -0.088 0.929835
## installment                        7.007e-04  5.908e-04   1.186 0.235592
## addr_stateAL                      -8.113e-03  4.679e-01  -0.017 0.986165
## addr_stateAR                      -1.436e-01  5.031e-01  -0.285 0.775295
## addr_stateAZ                       2.822e-01  4.511e-01   0.626 0.531585
## addr_stateCA                       2.761e-01  4.357e-01   0.634 0.526350
## addr_stateCO                       1.655e-02  4.578e-01   0.036 0.971172
## addr_stateCT                      -7.450e-02  4.699e-01  -0.159 0.874034
## addr_stateDC                       1.469e-01  6.470e-01   0.227 0.820333
## addr_stateDE                       6.808e-01  5.421e-01   1.256 0.209171
## addr_stateFL                       3.918e-01  4.383e-01   0.894 0.371390
## addr_stateGA                       5.231e-02  4.490e-01   0.116 0.907260
## addr_stateHI                       6.533e-01  4.896e-01   1.335 0.182034
## addr_stateID                      -1.075e+01  6.211e+02  -0.017 0.986192
## addr_stateIL                       5.052e-02  4.460e-01   0.113 0.909798
## addr_stateIN                       2.648e-01  4.585e-01   0.577 0.563614
## addr_stateKS                       1.823e-01  4.801e-01   0.380 0.704193
## addr_stateKY                       1.245e-01  4.805e-01   0.259 0.795596
```

```
## addr_stateLA                          4.774e-01  4.634e-01   1.030 0.302938
## addr_stateMA                          3.092e-01  4.515e-01   0.685 0.493517
## addr_stateMD                          3.319e-01  4.505e-01   0.737 0.461273
## addr_stateME                         -1.026e+01  1.246e+02  -0.082 0.934380
## addr_stateMI                          6.456e-02  4.512e-01   0.143 0.886216
## addr_stateMN                          3.753e-02  4.595e-01   0.082 0.934915
## addr_stateMO                          2.615e-01  4.566e-01   0.573 0.566785
## addr_stateMS                         -2.159e-01  5.851e-01  -0.369 0.712112
## addr_stateMT                         -5.518e-01  6.723e-01  -0.821 0.411853
## addr_stateNC                          9.923e-02  4.492e-01   0.221 0.825165
## addr_stateND                         -1.010e+01  1.762e+02  -0.057 0.954272
## addr_stateNE                         -1.877e-01  1.099e+00  -0.171 0.864432
## addr_stateNH                         -4.426e-01  5.845e-01  -0.757 0.448886
## addr_stateNJ                          1.905e-01  4.463e-01   0.427 0.669426
## addr_stateNM                          5.232e-01  4.940e-01   1.059 0.289506
## addr_stateNV                          4.331e-01  4.554e-01   0.951 0.341599
## addr_stateNY                          3.448e-01  4.378e-01   0.788 0.430967
## addr_stateOH                          9.028e-02  4.463e-01   0.202 0.839674
## addr_stateOK                          3.787e-01  4.731e-01   0.801 0.423387
## addr_stateOR                          2.459e-01  4.644e-01   0.530 0.596410
## addr_statePA                          2.799e-01  4.445e-01   0.630 0.528908
## addr_stateRI                          2.402e-01  5.079e-01   0.473 0.636255
## addr_stateSC                         -4.538e-01  4.917e-01  -0.923 0.356021
## addr_stateSD                         -1.143e-01  6.384e-01  -0.179 0.857957
## addr_stateTN                          4.183e-01  4.556e-01   0.918 0.358533
## addr_stateTX                          9.984e-02  4.386e-01   0.228 0.819928
## addr_stateUT                          2.351e-01  4.876e-01   0.482 0.629654
## addr_stateVA                          3.085e-01  4.471e-01   0.690 0.490162
## addr_stateVT                          4.688e-01  5.766e-01   0.813 0.416157
## addr_stateWA                          1.086e-01  4.541e-01   0.239 0.811037
## addr_stateWI                         -1.088e-01  4.718e-01  -0.231 0.817694
## addr_stateWV                         -1.829e-01  5.362e-01  -0.341 0.733073
## addr_stateWY                         -1.532e-01  6.103e-01  -0.251 0.801787
## issue_date                           -2.862e-03  1.033e-04 -27.706  < 2e-16
##
## (Intercept)                          ***
## loan_amnt
## int_rate_percent
## gradeB                               ***
## gradeC                               ***
## gradeD                               ***
## gradeE                               ***
## gradeF                               ***
## gradeG                               ***
## sub_gradeA2
## sub_gradeA3
## sub_gradeA4
## sub_gradeA5                          .
## sub_gradeB1
## sub_gradeB2                          **
## sub_gradeB3
## sub_gradeB4
## sub_gradeB5
## sub_gradeC1                          *
```

```
## sub_gradeC2
## sub_gradeC3
## sub_gradeC4
## sub_gradeC5
## sub_gradeD1                        *
## sub_gradeD2
## sub_gradeD3
## sub_gradeD4
## sub_gradeD5
## sub_gradeE1                        *
## sub_gradeE2
## sub_gradeE3
## sub_gradeE4                        *
## sub_gradeE5
## sub_gradeF1                        .
## sub_gradeF2
## sub_gradeF3
## sub_gradeF4
## sub_gradeF5
## sub_gradeG1
## sub_gradeG2
## sub_gradeG3
## sub_gradeG4
## sub_gradeG5
## annual_inc                         ***
## verification_statusSource Verified .
## verification_statusVerified
## term_in_months
## dti                                ***
## earliest_cr_line_date              ***
## inq_last_6mths                     ***
## open_acc
## pub_rec
## revol_bal
## revol_util_percent
## total_acc                          .
## initial_list_statusw
## application_typeJOINT
## acc_now_delinq
## delinq_2yrs
## installment
## addr_stateAL
## addr_stateAR
## addr_stateAZ
## addr_stateCA
## addr_stateCO
## addr_stateCT
## addr_stateDC
## addr_stateDE
## addr_stateFL
## addr_stateGA
## addr_stateHI
## addr_stateID
## addr_stateIL
```

```
## addr_stateIN
## addr_stateKS
## addr_stateKY
## addr_stateLA
## addr_stateMA
## addr_stateMD
## addr_stateME
## addr_stateMI
## addr_stateMN
## addr_stateMO
## addr_stateMS
## addr_stateMT
## addr_stateNC
## addr_stateND
## addr_stateNE
## addr_stateNH
## addr_stateNJ
## addr_stateNM
## addr_stateNV
## addr_stateNY
## addr_stateOH
## addr_stateOK
## addr_stateOR
## addr_statePA
## addr_stateRI
## addr_stateSC
## addr_stateSD
## addr_stateTN
## addr_stateTX
## addr_stateUT
## addr_stateVA
## addr_stateVT
## addr_stateWA
## addr_stateWI
## addr_stateWV
## addr_stateWY
## issue_date                        ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22649  on 52396  degrees of freedom
## Residual deviance: 19118  on 52293  degrees of freedom
## AIC: 19326
##
## Number of Fisher Scoring iterations: 13
```

```
#Predicting on test data
logit.probs <- predict(logit.fit, newdata = lending_club_model_df.test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

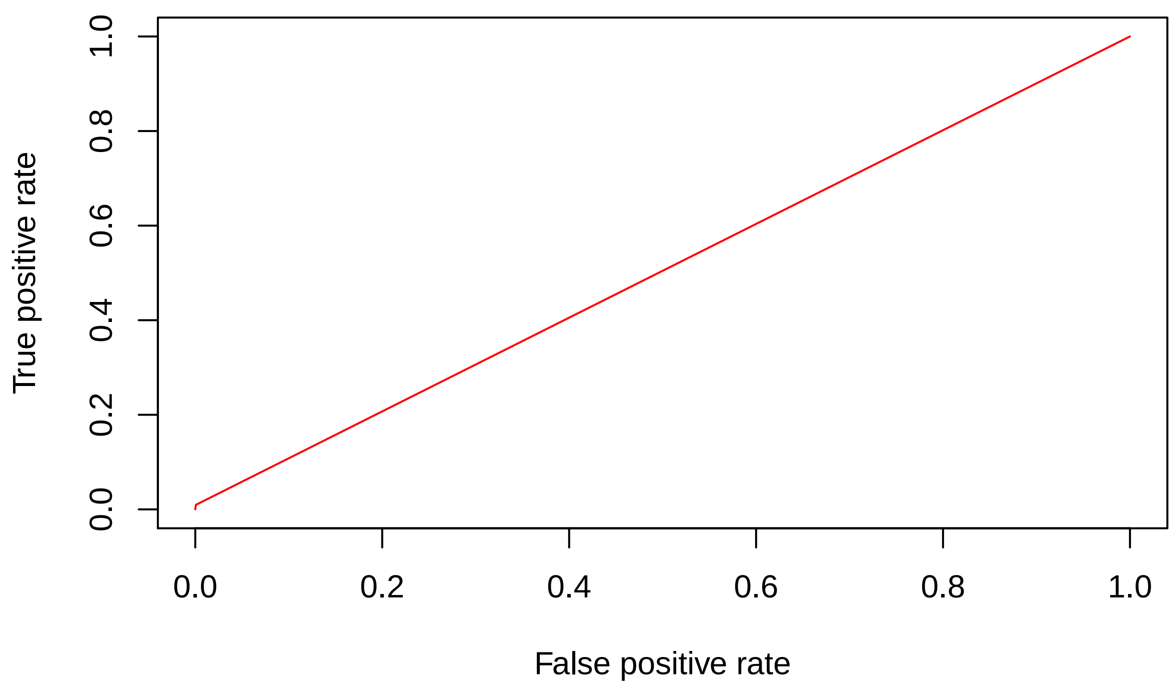
```r
logit.probs <- ifelse(logit.probs > 0.5, 1, 0)

#Confusion Matrix
confmatrix_default_category<- table(lending_club_model_df.test$default_category,
                                    logit.probs)
confmatrix_default_category
```

```
##    logit.probs
##         0     1
##   0 21186    14
##   1  1245    12
```

```r
#Accuracy of the model
sum(diag(confmatrix_default_category))/sum(confmatrix_default_category)
```

```
## [1] 0.9439373
```

```r
#Checking performance of the model by plotting ROC curve
pr <- prediction(logit.probs, lending_club_model_df.test$default_category)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf, col=rainbow(5))
```




```r
#Area under the curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.5044431
```

From the model generated, there are some variables which have little statistical significance as the pvalue is greater than 0.05. Thus these can be ignored while building the final model. Now, to understand the accuracy and performance of the model, we can look at the confusion matrix. It has a very good accuracy level though there are 1287 falsepositive and 3 Falsenegatives which are misclassified. To understand the performance of the model, we look at the ROC curve. As the AUC is very small, the performance of the model is not great. High performing models have ROC curve touching the top left orner and covering more area. Thus addition of deletion of features are required for the model. We can also look at AIC which is a measure of goodness of fit and can be used for model selection.

Now to reduce the number of features and select the best set of features, we can choose between backward subset selection method or lasso regression. In backward stepwise selection, a model with all features is considered initially and then based on performance of the model one or more features are removed and the process is continued untill we get the best mode.

In case of Lasso, the coefficients of the features which are not as significant are reduced to zero.

```
#Backward
step(glm ( default_category ~ .,
                  data = lending_club_model_df.train , family = "binomial" ),
     direction = "backward")
```

```
## Start:  AIC=19326.03
## default_category ~ loan_amnt + int_rate_percent + grade + sub_grade +
##     annual_inc + verification_status + term_in_months + dti +
##     earliest_cr_line_date + inq_last_6mths + open_acc + pub_rec +
##     revol_bal + revol_util_percent + total_acc + initial_list_status +
##     application_type + acc_now_delinq + delinq_2yrs + installment +
##     addr_state + issue_date
##
##
## Step:  AIC=19326.03
## default_category ~ loan_amnt + int_rate_percent + sub_grade +
##     annual_inc + verification_status + term_in_months + dti +
##     earliest_cr_line_date + inq_last_6mths + open_acc + pub_rec +
##     revol_bal + revol_util_percent + total_acc + initial_list_status +
##     application_type + acc_now_delinq + delinq_2yrs + installment +
##     addr_state + issue_date
##
##                         Df Deviance   AIC
## - addr_state            49    19194 19304
## - sub_grade             34    19176 19316
## - delinq_2yrs            1    19118 19324
## - open_acc               1    19118 19324
## - initial_list_status    1    19118 19324
## - term_in_months         1    19118 19324
## - revol_bal              1    19118 19324
## - int_rate_percent       1    19118 19324
## - loan_amnt              1    19118 19324
## - acc_now_delinq         1    19118 19324
## - application_type       1    19119 19325
## - revol_util_percent     1    19119 19325
## - installment            1    19119 19325
## - verification_status    2    19122 19326
## <none>                        19118 19326
## - pub_rec                1    19121 19327
## - total_acc              1    19121 19327
```

```
## - dti                    1    19132 19338
## - earliest_cr_line_date  1    19133 19339
## - inq_last_6mths         1    19141 19347
## - annual_inc             1    19179 19385
## - issue_date             1    19824 20030
##
## Step:  AIC=19303.69
## default_category ~ loan_amnt + int_rate_percent + sub_grade +
##     annual_inc + verification_status + term_in_months + dti +
##     earliest_cr_line_date + inq_last_6mths + open_acc + pub_rec +
##     revol_bal + revol_util_percent + total_acc + initial_list_status +
##     application_type + acc_now_delinq + delinq_2yrs + installment +
##     issue_date
##
##                         Df Deviance   AIC
## - sub_grade             34    19253 19295
## - delinq_2yrs            1    19194 19302
## - initial_list_status    1    19194 19302
## - term_in_months         1    19194 19302
## - open_acc               1    19194 19302
## - revol_bal              1    19194 19302
## - int_rate_percent       1    19194 19302
## - acc_now_delinq         1    19194 19302
## - loan_amnt              1    19194 19302
## - application_type       1    19195 19303
## - revol_util_percent     1    19195 19303
## - verification_status    2    19197 19303
## - installment            1    19195 19303
## - total_acc              1    19196 19304
## <none>                        19194 19304
## - pub_rec                1    19197 19305
## - dti                    1    19206 19314
## - earliest_cr_line_date  1    19210 19318
## - inq_last_6mths         1    19215 19323
## - annual_inc             1    19253 19361
## - issue_date             1    19912 20020
##
## Step:  AIC=19295.39
## default_category ~ loan_amnt + int_rate_percent + annual_inc +
##     verification_status + term_in_months + dti + earliest_cr_line_date +
##     inq_last_6mths + open_acc + pub_rec + revol_bal + revol_util_percent +
##     total_acc + initial_list_status + application_type + acc_now_delinq +
##     delinq_2yrs + installment + issue_date
##
##                         Df Deviance   AIC
## - loan_amnt              1    19253 19293
## - delinq_2yrs            1    19253 19293
## - term_in_months         1    19253 19293
## - initial_list_status    1    19253 19293
## - revol_bal              1    19253 19293
## - open_acc               1    19254 19294
## - acc_now_delinq         1    19254 19294
## - revol_util_percent     1    19254 19294
## - installment            1    19254 19294
```

```
## - application_type       1    19254 19294
## - total_acc              1    19255 19295
## <none>                        19253 19295
## - verification_status    2    19258 19296
## - pub_rec                1    19256 19296
## - dti                    1    19268 19308
## - earliest_cr_line_date  1    19271 19311
## - inq_last_6mths         1    19277 19317
## - annual_inc             1    19317 19357
## - int_rate_percent       1    19605 19645
## - issue_date             1    20762 20802
##
## Step:  AIC=19293.39
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     term_in_months + dti + earliest_cr_line_date + inq_last_6mths +
##     open_acc + pub_rec + revol_bal + revol_util_percent + total_acc +
##     initial_list_status + application_type + acc_now_delinq +
##     delinq_2yrs + installment + issue_date
##
##                         Df Deviance   AIC
## - delinq_2yrs            1    19253 19291
## - initial_list_status    1    19253 19291
## - revol_bal              1    19253 19291
## - term_in_months         1    19254 19292
## - open_acc               1    19254 19292
## - acc_now_delinq         1    19254 19292
## - revol_util_percent     1    19254 19292
## - application_type       1    19254 19292
## - total_acc              1    19255 19293
## <none>                        19253 19293
## - verification_status    2    19258 19294
## - pub_rec                1    19256 19294
## - installment            1    19267 19305
## - dti                    1    19268 19306
## - earliest_cr_line_date  1    19271 19309
## - inq_last_6mths         1    19277 19315
## - annual_inc             1    19318 19356
## - int_rate_percent       1    19759 19797
## - issue_date             1    20762 20800
##
## Step:  AIC=19291.41
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     term_in_months + dti + earliest_cr_line_date + inq_last_6mths +
##     open_acc + pub_rec + revol_bal + revol_util_percent + total_acc +
##     initial_list_status + application_type + acc_now_delinq +
##     installment + issue_date
##
##                         Df Deviance   AIC
## - initial_list_status    1    19253 19289
## - revol_bal              1    19253 19289
## - open_acc               1    19254 19290
## - term_in_months         1    19254 19290
## - acc_now_delinq         1    19254 19290
## - revol_util_percent     1    19254 19290
```

```
## - application_type       1    19254 19290
## - total_acc              1    19255 19291
## <none>                        19253 19291
## - verification_status    2    19258 19292
## - pub_rec                1    19256 19292
## - installment            1    19268 19304
## - dti                    1    19268 19304
## - earliest_cr_line_date  1    19271 19307
## - inq_last_6mths         1    19277 19313
## - annual_inc             1    19318 19354
## - int_rate_percent       1    19763 19799
## - issue_date             1    20767 20803
##
## Step:  AIC=19289.43
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     term_in_months + dti + earliest_cr_line_date + inq_last_6mths +
##     open_acc + pub_rec + revol_bal + revol_util_percent + total_acc +
##     application_type + acc_now_delinq + installment + issue_date
##
##                         Df Deviance   AIC
## - revol_bal              1    19254 19288
## - open_acc               1    19254 19288
## - term_in_months         1    19254 19288
## - acc_now_delinq         1    19254 19288
## - revol_util_percent     1    19254 19288
## - application_type       1    19254 19288
## - total_acc              1    19255 19289
## <none>                        19253 19289
## - verification_status    2    19258 19290
## - pub_rec                1    19256 19290
## - installment            1    19268 19302
## - dti                    1    19268 19302
## - earliest_cr_line_date  1    19271 19305
## - inq_last_6mths         1    19277 19311
## - annual_inc             1    19318 19352
## - int_rate_percent       1    19767 19801
## - issue_date             1    20885 20919
##
## Step:  AIC=19287.46
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     term_in_months + dti + earliest_cr_line_date + inq_last_6mths +
##     open_acc + pub_rec + revol_util_percent + total_acc + application_type +
##     acc_now_delinq + installment + issue_date
##
##                         Df Deviance   AIC
## - term_in_months         1    19254 19286
## - open_acc               1    19254 19286
## - acc_now_delinq         1    19254 19286
## - revol_util_percent     1    19254 19286
## - application_type       1    19254 19286
## - total_acc              1    19255 19287
## <none>                        19254 19288
## - verification_status    2    19258 19288
## - pub_rec                1    19256 19288
```

```
## - installment           1    19268 19300
## - dti                   1    19269 19301
## - earliest_cr_line_date  1    19271 19303
## - inq_last_6mths        1    19277 19309
## - annual_inc            1    19320 19352
## - int_rate_percent      1    19779 19811
## - issue_date            1    20886 20918
##
## Step:  AIC=19285.59
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     dti + earliest_cr_line_date + inq_last_6mths + open_acc +
##     pub_rec + revol_util_percent + total_acc + application_type +
##     acc_now_delinq + installment + issue_date
##
##                         Df Deviance   AIC
## - open_acc               1    19254 19284
## - acc_now_delinq         1    19254 19284
## - revol_util_percent     1    19254 19284
## - application_type       1    19255 19285
## - total_acc              1    19255 19285
## - verification_status    2    19258 19286
## <none>                        19254 19286
## - pub_rec                1    19256 19286
## - installment            1    19269 19299
## - dti                    1    19269 19299
## - earliest_cr_line_date  1    19271 19301
## - inq_last_6mths         1    19279 19309
## - annual_inc             1    19323 19353
## - int_rate_percent       1    19908 19938
## - issue_date             1    20919 20949
##
## Step:  AIC=19283.76
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     dti + earliest_cr_line_date + inq_last_6mths + pub_rec +
##     revol_util_percent + total_acc + application_type + acc_now_delinq +
##     installment + issue_date
##
##                         Df Deviance   AIC
## - acc_now_delinq         1    19254 19282
## - revol_util_percent     1    19254 19282
## - application_type       1    19255 19283
## - verification_status    2    19258 19284
## <none>                        19254 19284
## - pub_rec                1    19256 19284
## - total_acc              1    19257 19285
## - installment            1    19269 19297
## - dti                    1    19270 19298
## - earliest_cr_line_date  1    19272 19300
## - inq_last_6mths         1    19279 19307
## - annual_inc             1    19323 19351
## - int_rate_percent       1    19910 19938
## - issue_date             1    20919 20947
##
## Step:  AIC=19281.99
```

```
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     dti + earliest_cr_line_date + inq_last_6mths + pub_rec +
##     revol_util_percent + total_acc + application_type + installment +
##     issue_date
##
##                         Df Deviance   AIC
## - revol_util_percent     1    19255 19281
## - application_type       1    19255 19281
## - verification_status    2    19258 19282
## <none>                        19254 19282
## - pub_rec                1    19256 19282
## - total_acc              1    19257 19283
## - installment            1    19269 19295
## - dti                    1    19270 19296
## - earliest_cr_line_date  1    19272 19298
## - inq_last_6mths         1    19279 19305
## - annual_inc             1    19323 19349
## - int_rate_percent       1    19911 19937
## - issue_date             1    20920 20946
##
## Step:  AIC=19280.46
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     dti + earliest_cr_line_date + inq_last_6mths + pub_rec +
##     total_acc + application_type + installment + issue_date
##
##                         Df Deviance   AIC
## - application_type       1    19255 19279
## - verification_status    2    19258 19280
## <none>                        19255 19281
## - pub_rec                1    19257 19281
## - total_acc              1    19258 19282
## - installment            1    19269 19293
## - dti                    1    19271 19295
## - earliest_cr_line_date  1    19273 19297
## - inq_last_6mths         1    19281 19305
## - annual_inc             1    19327 19351
## - int_rate_percent       1    19949 19973
## - issue_date             1    20942 20966
##
## Step:  AIC=19279.43
## default_category ~ int_rate_percent + annual_inc + verification_status +
##     dti + earliest_cr_line_date + inq_last_6mths + pub_rec +
##     total_acc + installment + issue_date
##
##                         Df Deviance   AIC
## - verification_status    2    19259 19279
## <none>                        19255 19279
## - pub_rec                1    19258 19280
## - total_acc              1    19259 19281
## - installment            1    19270 19292
## - dti                    1    19271 19293
## - earliest_cr_line_date  1    19274 19296
## - inq_last_6mths         1    19282 19304
## - annual_inc             1    19328 19350
```

```
## - int_rate_percent       1    19950 19972
## - issue_date             1    20945 20967
##
## Step:  AIC=19279.4
## default_category ~ int_rate_percent + annual_inc + dti + earliest_cr_line_date +
##     inq_last_6mths + pub_rec + total_acc + installment + issue_date
##
##                         Df Deviance   AIC
## - pub_rec                1    19261 19279
## <none>                        19259 19279
## - total_acc              1    19263 19281
## - dti                    1    19276 19294
## - installment            1    19277 19295
## - earliest_cr_line_date  1    19278 19296
## - inq_last_6mths         1    19286 19304
## - annual_inc             1    19331 19349
## - int_rate_percent       1    19988 20006
## - issue_date             1    20987 21005
##
## Step:  AIC=19279.36
## default_category ~ int_rate_percent + annual_inc + dti + earliest_cr_line_date +
##     inq_last_6mths + total_acc + installment + issue_date
##
##                         Df Deviance   AIC
## <none>                        19261 19279
## - total_acc              1    19265 19281
## - dti                    1    19279 19295
## - installment            1    19280 19296
## - earliest_cr_line_date  1    19281 19297
## - inq_last_6mths         1    19288 19304
## - annual_inc             1    19332 19348
## - int_rate_percent       1    19988 20004
## - issue_date             1    21037 21053
```

```
##
## Call:  glm(formula = default_category ~ int_rate_percent + annual_inc +
##     dti + earliest_cr_line_date + inq_last_6mths + total_acc +
##     installment + issue_date, family = "binomial", data = lending_club_model_df.train)
##
## Coefficients:
##           (Intercept)       int_rate_percent             annual_inc
##             3.597e+01              1.285e-01             -8.181e-06
##                   dti  earliest_cr_line_date         inq_last_6mths
##             1.138e-02              3.725e-05              9.822e-02
##             total_acc            installment             issue_date
##             3.740e-03              4.346e-04             -2.537e-03
##
## Degrees of Freedom: 52396 Total (i.e. Null);  52388 Residual
## Null Deviance:       22650
## Residual Deviance: 19260     AIC: 19280
```

```r
#Creating matrix for Lasso
X <- model.matrix(default_category ~., data = lending_club_model_df.train)[,-1]

lending_club_model_df.train$default_category =
```

```r
  as.numeric(lending_club_model_df.train$default_category)
#Applying logistic regression using glmnet, which gives same result as glm
#when used with alpha = 1
fit <- glmnet(X, lending_club_model_df.train$default_category, alpha = 1,family="binomial")

#Cross validating to find best lambda which will reduce insignificant coefficients to zero
cv.out <- cv.glmnet(X, lending_club_model_df.train$default_category, alpha = 1)
bestlambda <- cv.out$lambda.min
bestlambda
```

```
## [1] 0.0008946991
```

```r
#Using best lambda and fitting logistic to find optimum fit model
fit_best <- glmnet(X, lending_club_model_df.train$default_category, lambda = bestlambda)
coef(fit_best)
```

```
## 110 x 1 sparse Matrix of class "dgCMatrix"
##                                              s0
## (Intercept)                        3.199098e+00
## loan_amnt                          .
## int_rate_percent                   5.657190e-03
## gradeB                            -5.037203e-03
## gradeC                            -4.396063e-03
## gradeD                             .
## gradeE                             .
## gradeF                             3.614005e-03
## gradeG                             1.679783e-02
## sub_gradeA2                        1.667884e-03
## sub_gradeA3                        .
## sub_gradeA4                       -2.337108e-03
## sub_gradeA5                        .
## sub_gradeB1                        4.055671e-03
## sub_gradeB2                       -1.527932e-03
## sub_gradeB3                        .
## sub_gradeB4                       -1.788586e-03
## sub_gradeB5                       -2.580882e-04
## sub_gradeC1                       -1.921257e-03
## sub_gradeC2                        .
## sub_gradeC3                        .
## sub_gradeC4                        .
## sub_gradeC5                       -4.452849e-03
## sub_gradeD1                        .
## sub_gradeD2                        .
## sub_gradeD3                        .
## sub_gradeD4                        2.093182e-03
## sub_gradeD5                        .
## sub_gradeE1                        .
## sub_gradeE2                        1.247702e-02
## sub_gradeE3                        .
## sub_gradeE4                        .
## sub_gradeE5                        2.032347e-02
## sub_gradeF1                        .
## sub_gradeF2                        .
## sub_gradeF3                        4.084290e-02
## sub_gradeF4                        1.848990e-02
```

```
## sub_gradeF5                          2.519920e-02
## sub_gradeG1                          .
## sub_gradeG2                          6.313056e-02
## sub_gradeG3                          .
## sub_gradeG4                         -1.707109e-02
## sub_gradeG5                          .
## annual_inc                          -2.347851e-07
## verification_statusSource Verified  .
## verification_statusVerified          1.385440e-03
## term_in_months                       .
## dti                                  4.672844e-04
## earliest_cr_line_date                1.109377e-06
## inq_last_6mths                       5.735032e-03
## open_acc                             3.457467e-05
## pub_rec                             -2.408653e-03
## revol_bal                            .
## revol_util_percent                   .
## total_acc                            .
## initial_list_statusw                 .
## application_typeJOINT                .
## acc_now_delinq                       .
## delinq_2yrs                          .
## installment                          1.003812e-05
## addr_stateAL                        -2.145068e-03
## addr_stateAR                        -5.331795e-03
## addr_stateAZ                         9.153944e-04
## addr_stateCA                         2.269143e-03
## addr_stateCO                        -2.363550e-03
## addr_stateCT                        -4.851382e-03
## addr_stateDC                         .
## addr_stateDE                         1.163719e-02
## addr_stateFL                         8.728791e-03
## addr_stateGA                        -2.954301e-03
## addr_stateHI                         1.664913e-02
## addr_stateID                         .
## addr_stateIL                        -1.854261e-03
## addr_stateIN                         .
## addr_stateKS                         .
## addr_stateKY                         .
## addr_stateLA                         8.664641e-03
## addr_stateMA                         8.527192e-04
## addr_stateMD                         2.106628e-03
## addr_stateME                         .
## addr_stateMI                        -2.008754e-04
## addr_stateMN                        -1.223701e-03
## addr_stateMO                         .
## addr_stateMS                        -1.154355e-03
## addr_stateMT                        -1.244104e-02
## addr_stateNC                         .
## addr_stateND                         .
## addr_stateNE                         .
## addr_stateNH                        -8.906811e-03
## addr_stateNJ                         .
## addr_stateNM                         4.019218e-03
```

```
## addr_stateNV                          5.898851e-03
## addr_stateNY                          4.510388e-03
## addr_stateOH                         -3.990656e-04
## addr_stateOK                          2.927416e-03
## addr_stateOR                         .
## addr_statePA                         .
## addr_stateRI                         .
## addr_stateSC                         -1.592442e-02
## addr_stateSD                         .
## addr_stateTN                          3.199969e-03
## addr_stateTX                         -6.876407e-04
## addr_stateUT                         .
## addr_stateVA                          1.801897e-03
## addr_stateVT                         .
## addr_stateWA                         .
## addr_stateWI                         -4.536945e-03
## addr_stateWV                         -6.368572e-03
## addr_stateWY                         -7.415899e-04
## issue_date                           -1.364127e-04
```

Though the model coefficients vary between backward stepwise selection model and lasso model, we are able to find the best features required for predicting default category of the loan. Depending the coefficeints magnitude though we can gauge the importance of the predictors, but it wont be completely correct.

As we havent found a sufficiently satisfactory model, I would like to fit random forest to predcit default category.

```
lending_club_model_df.train$grade <- as.factor(lending_club_model_df.train$grade)
lending_club_model_df.test$grade <- as.factor(lending_club_model_df.test$grade)
lending_club_model_df.train$default_category =
  as.factor(lending_club_model_df.train$default_category)

#Fitting Random Forest
rf.fit <- randomForest(default_category ~ loan_amnt +
                         int_rate_percent+grade+annual_inc+term_in_months+dti+
                         inq_last_6mths+revol_util_percent+total_acc+issue_date+
                     installment+earliest_cr_line_date,
                       data = lending_club_model_df.train)

#Predicting using random forest model
rf.probs <- predict(rf.fit, newdata = lending_club_model_df.test)

#Calculating Accuracy using confusion matrix
confmatrix_rf_new<-table(rf.probs,lending_club_model_df.test$default_category)
confmatrix_rf_new
```

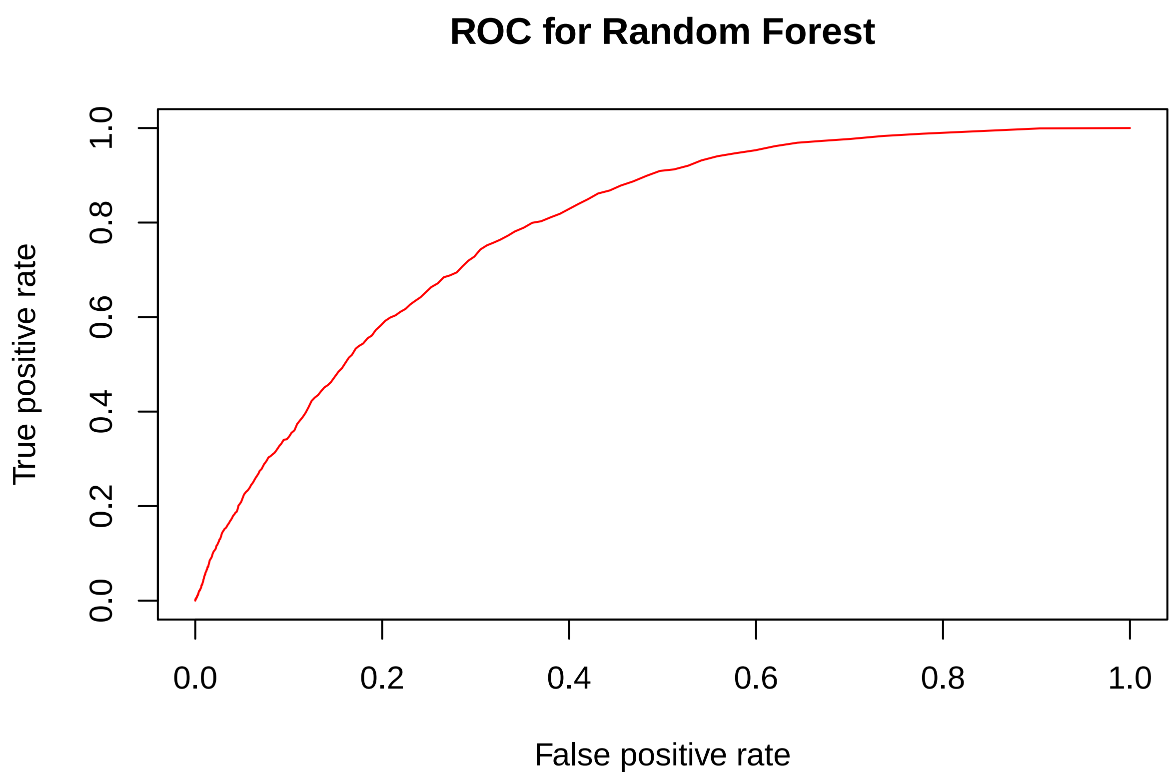
```
##
## rf.probs     0     1
##        1 21192  1253
##        2     8     4
```

```
sum(diag(confmatrix_rf_new))/sum(confmatrix_rf_new)
```

```
## [1] 0.9438482
```

```
#Plotting performance of model using ROC curve
probRF <- predict(rf.fit, newdata = lending_club_model_df.test, type='prob')
```

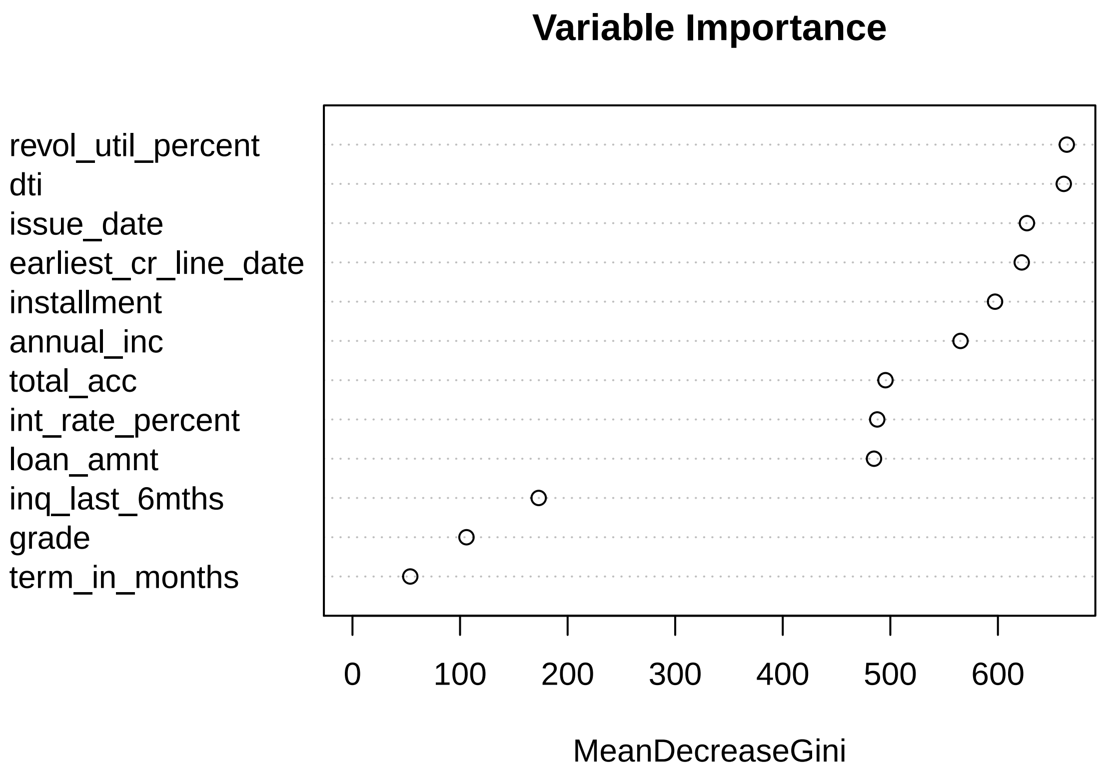
```
predRF <- prediction(probRF[,2],lending_club_model_df.test$default_category)
perfRF <- performance(predRF, measure = "tpr", x.measure = "fpr")
plot(perfRF, col=rainbow(5),main = "ROC for Random Forest" )
```




```
#Finding importance of variables in the model
importance(rf.fit)
```

```
##                       MeanDecreaseGini
## loan_amnt                    484.79778
## int_rate_percent             487.79333
## grade                        105.98319
## annual_inc                   565.24821
## term_in_months                53.71374
## dti                          661.23778
## inq_last_6mths               173.10591
## revol_util_percent           664.02851
## total_acc                    495.46609
## issue_date                   626.97326
## installment                  597.32102
## earliest_cr_line_date        622.16026
```

```
#Plotting importance of variables in the model
varImpPlot(rf.fit,main = "Variable Importance")
```



From the model generated by fitting random forest, the accuracy of the predictions is quite comparable to the Logistic model but performance of this model is far better. This can be seen in the graphic as the ROC curve covers larger area. Moving on to the importance of the predictors, the top five predictors according to variable importance plot based on the random forest model are dti, revol_util_percent, earliest_cr_line_date, issue_date and installment.

6. Select one of the below topics and concisely explain it to:

I would like to explain Logistic Regression.

a. someone with significant mathematical experience

The outcomes of many of the experiments/research are qualitative or categorical and they can be predicted or categorized into classes using methods like Logistic Regression. Thus logistic regression is used to predict a variable which has discrete values and is not continuous. Logistic regression approach calculates the probability of each of the categories of the response variable. This probability is then used to categorize the response variable Y. The function used to predict qualitative variables has to have outputs between 0 and 1. Thus we use logistic function,

$$p(X) = e^{\beta 0+\beta 1*X}/1 + e^{\beta 0+\beta 1*X}$$

$\beta 0$ and $\beta 1$ are the unknown coefficients. To evaluate these coefficients we can estimate based on available training data using methods like Maximum likelihood. The idea behind finding estimates is that we find estimates for $\beta 0$ and $\beta 1$ such that the predicted probability $\hat{p(x_i)}$ is as close indicative of the class that the response belongs to. For example, if we consider a scenario where we are predicting a default on a loan payment as in the above examples. The estimates calculated for $\beta 0$ and $\beta 1$ once put in above equation should give a response

$$p(X)$$

closer to 1 for defaultors and close to 0 for individuals who did not default. The maximum likelihood function used to evaluate $\beta 0$ and $\beta 1$ is as below:

$$l(\beta 0, \beta 1) = \pi_{i,j=1} p(x_i) * \pi_{i`,j`=0} p(1 - x_{i`})$$

The $\beta 0$ and $\beta 1$ are calculated by maximizing the above function. Once the estimates are caluclated, we an use them to classify new test observations by calculating the p(X).

Logistic function will always produce an S shaped curve which would swiftly move from one category represented by 0 to another category represented by 1 for bimodal categorical variables. Some manipulation of logistic function leads to below formula:

$$p(X)/1 - p(X) = e^{\beta 0 + \beta 1 * X}$$

The equation on the left hand side is called the odds and they can range from 0 to infinity. From the above example odd close to 0 indicate lwould mean ow probability of default and high probability of default for odds nearing infinity. Another important concept to know about logistic regression is that log-odds are linear in X. Log-odds is also known as logit.

$$log(p(X)/1 - p(X)) = \beta 0 + \beta 1 * X$$

Thus interpreting the above result we can say that a unit change in X causes the log odds to change by $\beta 1$. Thus in conclusion we can say that there is no linear relationship between p(X) and X. An increase or decrease in X will cause p(X) to increase or decrease depending on the sign of $\beta 1$.

b. someone with little mathematical experience.

As one starts with a research project, there are number of instances when the response variable is qualitative or categorical. The prediction of qualitative response variable involves segreagating responses into different classes and this is achieved by many different methods of which one is logistic regression. For the basis of classification, logistic regression predicts the probability of each of the categories of a qualitative variable. A simple example of classification which can be solved using logistic regression is of classifying if the email recieved by a person is a spam or not. To classify this email, we use the data from previous emails as training observations. The data that can be useful could like the subject line, specific words in email, domain of email sender, etc.Using this data a model is developed which has an output between 0 and 1. Lets assume according to our model we considered 0 as no spam and 1 as spam. Using any new email as a test observation, if the model outputs a value greater than 0.5, we can classify the email as spam or else not a spam. Logistic regression can be applied to classify response variables where there are more than two classes, though in industry some of the other methods like discriminant analysis are preferred.

c. Topics: Logistic Regression, Ridge vs Lasso Regression, Principal Component Analysis, Factor Analysis, K-means Clustering, Support Vector Machines, Markov Process, Hidden Markov Model, Decision trees, Random forest or the curse of dimensionality.