4/12/2020

# LENDING CLUB LOAN DATA PREDICTION AND ANALYSIS

DATA SCIENCE ENGINEERING METHODS AND TOOLS

Aditya Salunkhe
Sahil Bagwan
Shruti Kasar
Sumedh Telang

## Contents

## Topic:

- Loan Data Prediction and Analysis

## Professor:

- Dr. Liu Handan

## Group Members:

- Aditya Salunkhe (001348277)
- Sahil Bagwan (001055189)
- Shruti Kasar (001306991)
- Sumedh Telang (001059338

## Introduction:

Lending Club is an American based peer-to-peer lending company which has helped over 2.5+ million customer in a span of over 10 years (2005 -2015) through personal, auto refinancing, business and medical financing loans. Investors lend money directly to borrowers via the online platform. Lending Club offers loans up to $40,000 across 2 loan terms (36/60 months). Loans are issued according to various grades, corresponding to different interest rates.

In this project we attempt to use our analysis find various factors that influence the loan market and to answer the following questions:

- Identify if the Lending club loan will be charged off
- Analyze factors that affect the interest rate, credit grade and predict the interest rate
- Should I invest with Lending Club?

The loan industry of the United States has gone through several distinct phases of evolution that have allowed it to reach its current status as the largest and most complex market all over the world. Loan analysis is an evaluation method that determines if loans are made on feasible terms and if potential borrowers can and are willing to pay back the loan. It checks the eligibility of the potential borrower against the criteria set forth for lending. Loan analysis helps in assessing the skills and financial knowledge of the borrower to determine the level of risk involved. Hence this project will help us understand the important factors in this everlasting loan scene.

## How it works:

- Customers interested in a loan complete a simple application at LendingClub.com
- LC leverage online data and technology to quickly assess risk, determine a credit rating and assign appropriate interest rates.
- Qualified applicants receive offers in just minutes and can evaluate loan options with no impact to their credit score
- Investors ranging from individuals to institutions select loans in which to invest and can earn monthly returns
- The entire process is online, using technology to lower the cost of credit and pass the savings back in the form of lower rates for borrowers and solid returns for investors.

## Objectives:

- Identify if the Lending club loan will be charged off.
- Analyze factors that affect the interest rate, credit grade and predict the interest rate over a period of years.
- Should I invest with Lending Club?

## Methodology:

Using concepts of Machine Learning, an efficient model can be implemented that would enable the company to identify if LC loan will be charged off as well as identify potential loan defaulters. We also intent to use the data to analyze the factors that influence the interest rate and the credit grade and predict the interest rate over a period of years.

We intend to use to the following supervised ML algorithms to train the model for predictive analysis.

- Linear Regression
- Logistic Regression
- Naïve Bayes Classifier
- Random Forest
- Gradient Boosting

## Description & Specifications of the Dataset:

The dataset contains the Analyzing Lending Club's complete loan data for all loans issued through the 2007-2015, including the current loan status (Current, Late, Fully Paid, etc.) and latest payment information. The file containing loan data through the "present" contains complete loan data for all loans issued through the previous completed calendar quarter. Additional features include credit scores, number of finance inquiries, address including zip codes, and state, and collections among others. The file is a matrix of about 890 thousand observations and 75 variables. A data dictionary is provided in a separate file.

In this project the features such as credit score, amount, finances, payment quarters, financial inquires etc. will be considered for analysis which will gives a better diving into understanding eligibility criterion, as well as various grades corresponding to the various interest rates.

Using concepts of Machine Learning, an efficient model can be implemented that would enable the company to identify if LC loan will be charged off as well as identify potential loan defaulters.

## Dataset Key Specifications:

Files Included:

- Loan.csv
- LCDataDictionary.xlsx

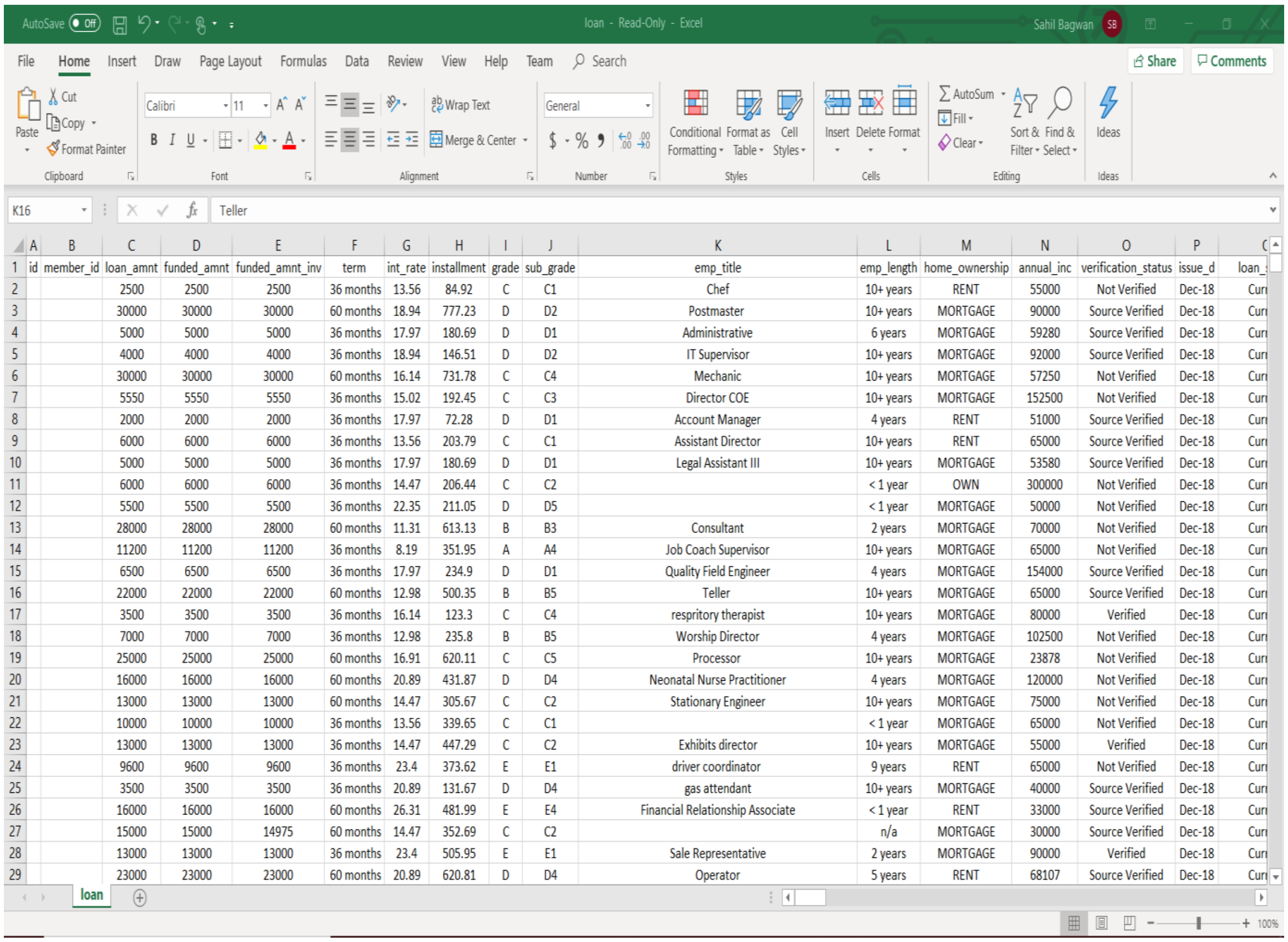
Number of Columns: 145

Number of Rows: 421,097

Dataset Size – 1.11 GB

Type of Data – Text

Dataset Link - https://www.kaggle.com/wendykan/lending-club-loan-data



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | installment | grade | sub_grade | emp_title | emp_length | home_ownership | annual_inc | verification_status | issue_d | loan_ |
| 2 | | | 2500 | 2500 | 2500 | 36 months | 13.56 | 84.92 | C | C1 | Chef | 10+ years | RENT | 55000 | Not Verified | Dec-18 | Cur |
| 3 | | | 30000 | 30000 | 30000 | 60 months | 18.94 | 777.23 | D | D2 | Postmaster | 10+ years | MORTGAGE | 90000 | Source Verified | Dec-18 | Cur |
| 4 | | | 5000 | 5000 | 5000 | 36 months | 17.97 | 180.69 | D | D1 | Administrative | 6 years | MORTGAGE | 59280 | Source Verified | Dec-18 | Cur |
| 5 | | | 4000 | 4000 | 4000 | 36 months | 18.94 | 146.51 | D | D2 | IT Supervisor | 10+ years | MORTGAGE | 92000 | Source Verified | Dec-18 | Cur |
| 6 | | | 30000 | 30000 | 30000 | 60 months | 16.14 | 731.78 | C | C4 | Mechanic | 10+ years | MORTGAGE | 57250 | Not Verified | Dec-18 | Cur |
| 7 | | | 5550 | 5550 | 5550 | 36 months | 15.02 | 192.45 | C | C3 | Director COE | 10+ years | MORTGAGE | 152500 | Not Verified | Dec-18 | Cur |
| 8 | | | 2000 | 2000 | 2000 | 36 months | 17.97 | 72.28 | D | D1 | Account Manager | 4 years | RENT | 51000 | Source Verified | Dec-18 | Cur |
| 9 | | | 6000 | 6000 | 6000 | 36 months | 13.56 | 203.79 | C | C1 | Assistant Director | 10+ years | RENT | 65000 | Source Verified | Dec-18 | Cur |
| 10 | | | 5000 | 5000 | 5000 | 36 months | 17.97 | 180.69 | D | D1 | Legal Assistant III | 10+ years | MORTGAGE | 53580 | Source Verified | Dec-18 | Cur |
| 11 | | | 6000 | 6000 | 6000 | 36 months | 14.47 | 206.44 | C | C2 | | < 1 year | OWN | 300000 | Not Verified | Dec-18 | Cur |
| 12 | | | 5500 | 5500 | 5500 | 36 months | 22.35 | 211.05 | D | D5 | | < 1 year | MORTGAGE | 50000 | Not Verified | Dec-18 | Cur |
| 13 | | | 28000 | 28000 | 28000 | 60 months | 11.31 | 613.13 | B | B3 | Consultant | 2 years | MORTGAGE | 70000 | Not Verified | Dec-18 | Cur |
| 14 | | | 11200 | 11200 | 11200 | 36 months | 8.19 | 351.95 | A | A4 | Job Coach Supervisor | 10+ years | MORTGAGE | 65000 | Not Verified | Dec-18 | Cur |
| 15 | | | 6500 | 6500 | 6500 | 36 months | 17.97 | 234.9 | D | D1 | Quality Field Engineer | 4 years | MORTGAGE | 154000 | Source Verified | Dec-18 | Cur |
| 16 | | | 22000 | 22000 | 22000 | 60 months | 12.98 | 500.35 | B | B5 | Teller | 10+ years | MORTGAGE | 65000 | Source Verified | Dec-18 | Cur |
| 17 | | | 3500 | 3500 | 3500 | 36 months | 16.14 | 123.3 | C | C4 | respritory therapist | 10+ years | MORTGAGE | 80000 | Verified | Dec-18 | Cur |
| 18 | | | 7000 | 7000 | 7000 | 36 months | 12.98 | 235.8 | B | B5 | Worship Director | 4 years | MORTGAGE | 102500 | Not Verified | Dec-18 | Cur |
| 19 | | | 25000 | 25000 | 25000 | 60 months | 16.91 | 620.11 | C | C5 | Processor | 10+ years | MORTGAGE | 23878 | Not Verified | Dec-18 | Cur |
| 20 | | | 16000 | 16000 | 16000 | 60 months | 20.89 | 431.87 | D | D4 | Neonatal Nurse Practitioner | 4 years | MORTGAGE | 120000 | Not Verified | Dec-18 | Cur |
| 21 | | | 13000 | 13000 | 13000 | 60 months | 14.47 | 305.67 | C | C2 | Stationary Engineer | 10+ years | MORTGAGE | 75000 | Not Verified | Dec-18 | Cur |
| 22 | | | 10000 | 10000 | 10000 | 36 months | 13.56 | 339.65 | C | C1 | | < 1 year | MORTGAGE | 65000 | Not Verified | Dec-18 | Cur |
| 23 | | | 13000 | 13000 | 13000 | 36 months | 14.47 | 447.29 | C | C2 | Exhibits director | 10+ years | MORTGAGE | 55000 | Verified | Dec-18 | Cur |
| 24 | | | 9600 | 9600 | 9600 | 36 months | 23.4 | 373.62 | E | E1 | driver coordinator | 9 years | RENT | 65000 | Not Verified | Dec-18 | Cur |
| 25 | | | 3500 | 3500 | 3500 | 36 months | 20.89 | 131.67 | D | D4 | gas attendant | 10+ years | MORTGAGE | 40000 | Source Verified | Dec-18 | Cur |
| 26 | | | 16000 | 16000 | 16000 | 60 months | 26.31 | 481.99 | E | E4 | Financial Relationship Associate | < 1 year | RENT | 33000 | Source Verified | Dec-18 | Cur |
| 27 | | | 15000 | 15000 | 14975 | 60 months | 14.47 | 352.69 | C | C2 | | n/a | MORTGAGE | 30000 | Source Verified | Dec-18 | Cur |
| 28 | | | 13000 | 13000 | 13000 | 36 months | 23.4 | 505.95 | E | E1 | Sale Representative | 2 years | MORTGAGE | 90000 | Verified | Dec-18 | Cur |
| 29 | | | 23000 | 23000 | 23000 | 60 months | 20.89 | 620.81 | D | D4 | Operator | 5 years | RENT | 68107 | Source Verified | Dec-18 | Cur |

## Data Sources:

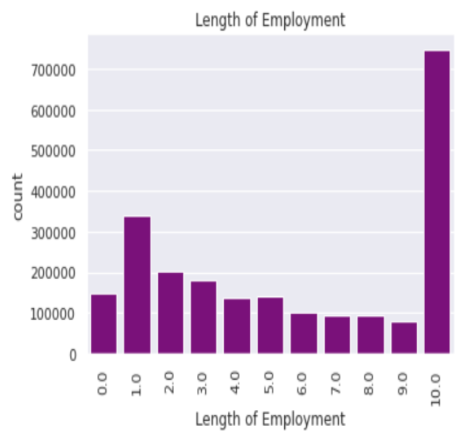
1. Lending Loan Dataset – Kaggle
2. Loan.csv
3. Browse Notes
4. Loan Stats

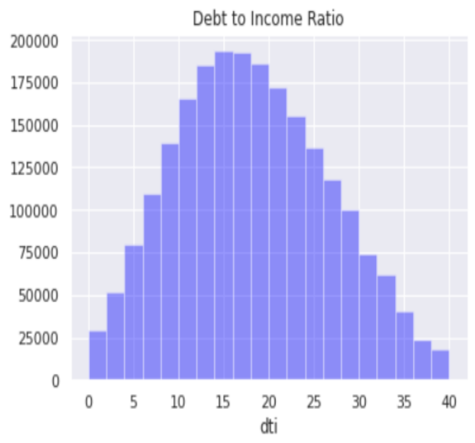
## Working:

1 – Data Cleaning, Analysis & Statistics

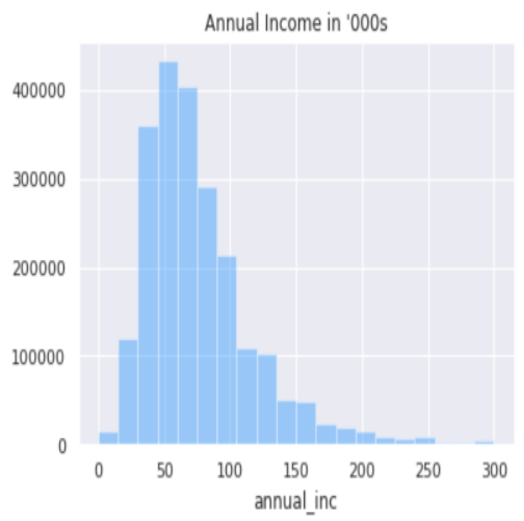
- Removing columns containing more than 40% of NULL values
- Cleaning data not containing uniform values
- Elimination of current loan status as it doesn't help in predictions
- Removing NaN values from the dataset
- Converting data frame of emp_length into integer format by removing inconsistencies
- Removing last 2 rows with no data
- Finding correlation for the entire dataset and elimination of features with same meaning to avoid duplicity

*Perform Exploratory Analysis to get relevant data.*

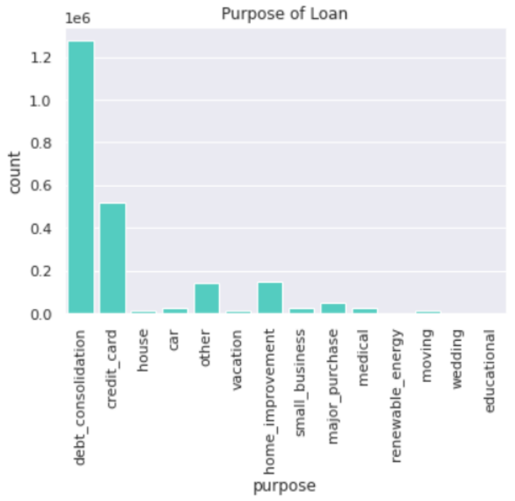




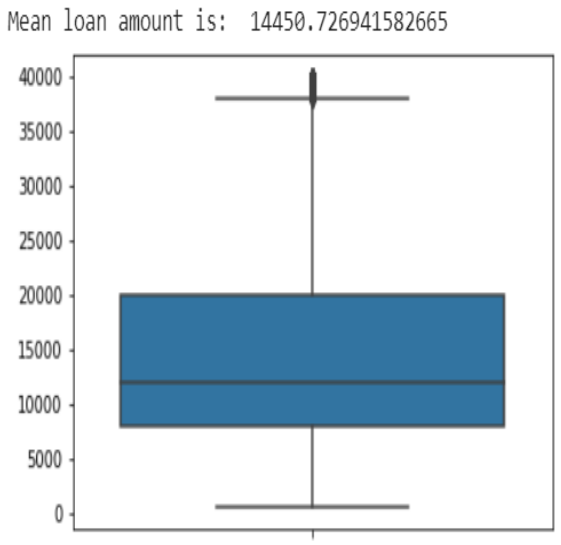
More loans were disbursed to individuals who have worked 10 or more years



Annual Income of people in the 2000’s

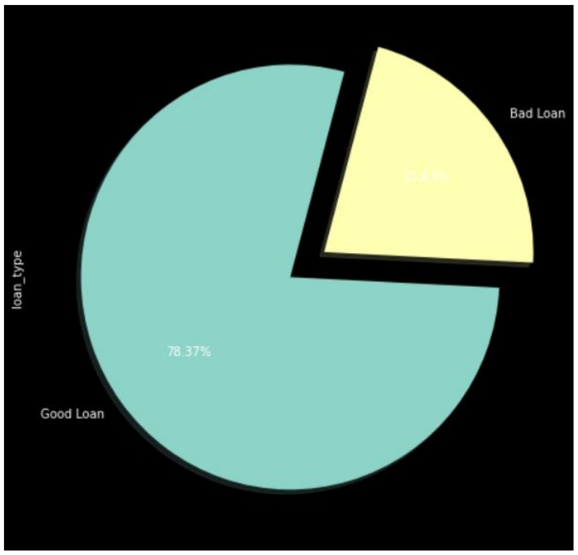


Purpose of the loan taken

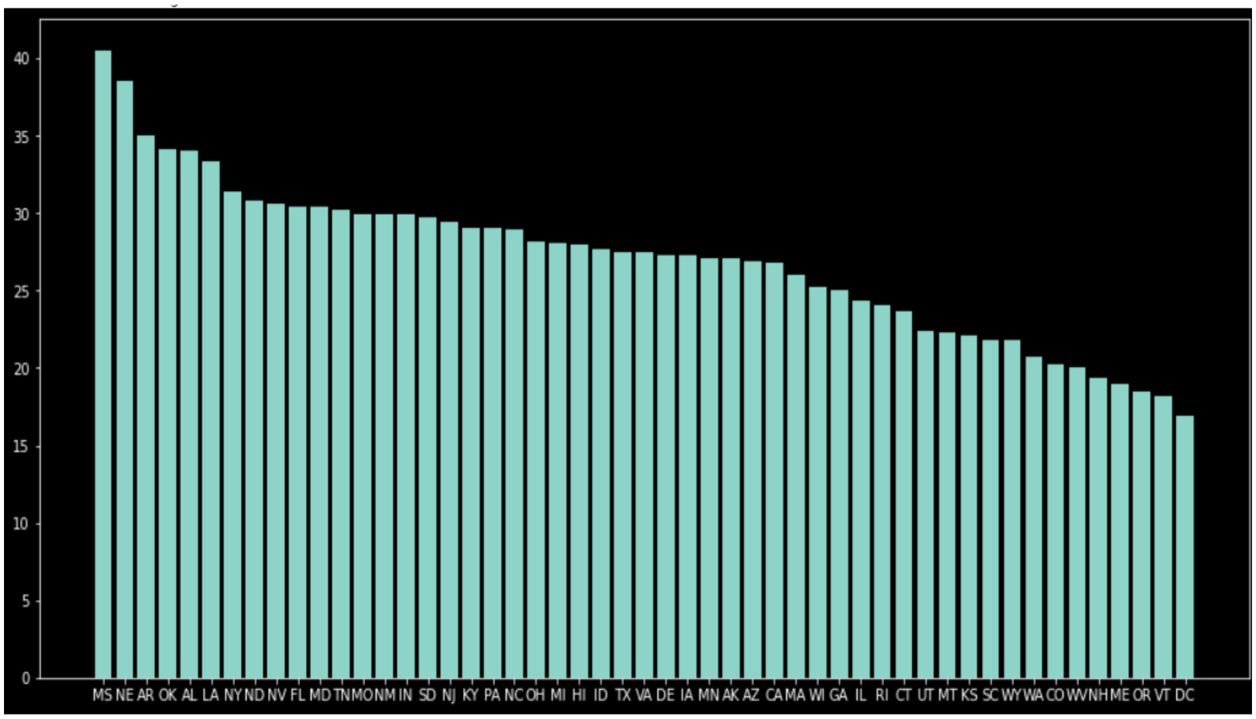


```
Mean loan amount is:  14450.726941582665
```

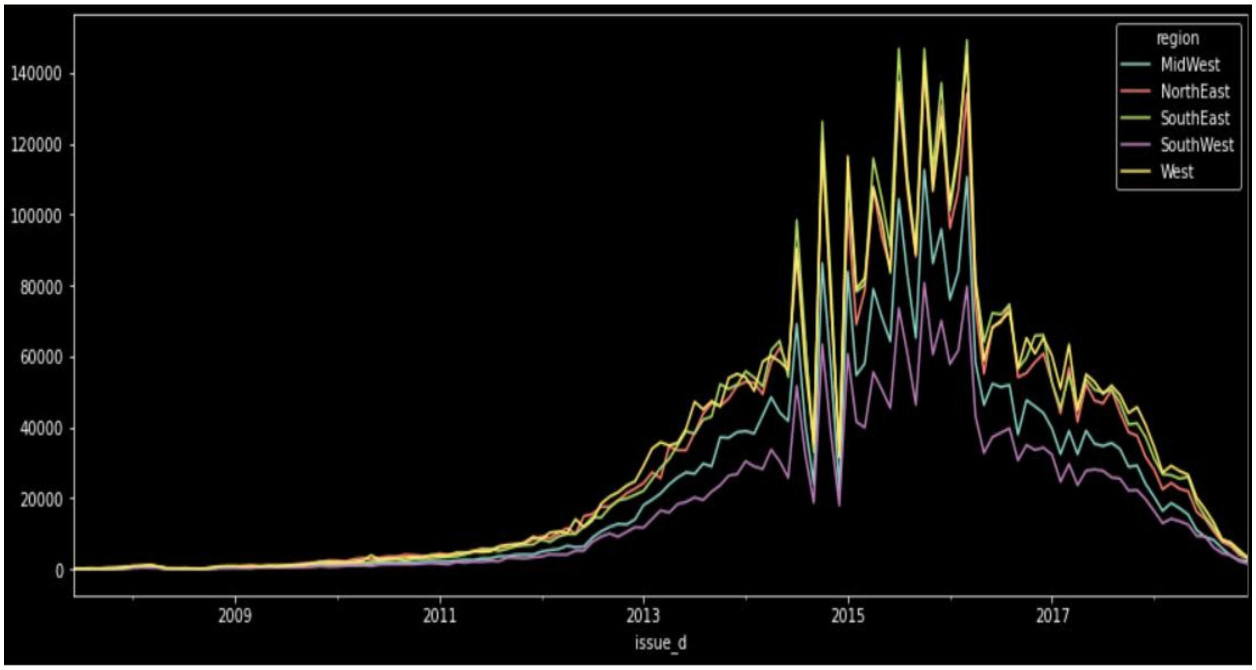
Average loan amount



Good Loans vs Bad Loans Comparison



State wise bad loan distribution with Mississippi, Nebraska and Arkansas being the least safe



Region wise loan distribution

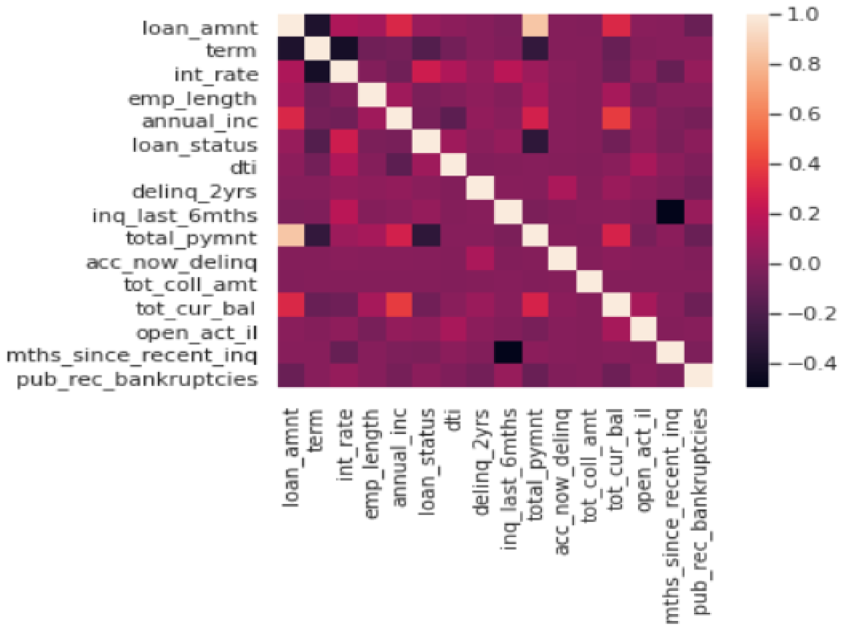
2 – Train and Test the clean Dataset

## *Concepts Used:*

**Confusion Matrix:** A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

| | *Class 1 Predicted* | *Class 2 Predicted* |
|---|---|---|
| **Class 1 Actual** | **TP** | **FN** |
| **Class 2 Actual** | **FP** | **TN** |

**Correlation Matrix:** A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (Xi) in the table is correlated with each of the other values in the table (Xj). This allows you to see which pairs have the highest correlation.



**Bayes' Theorem:** It is a way of finding a probability when we know certain other probabilities.

The formula is:

$P(A|B) = P(A)\ P(B|A)\ P(B)$

Which tells us: how often A happens given that B happens, written P(A|B),

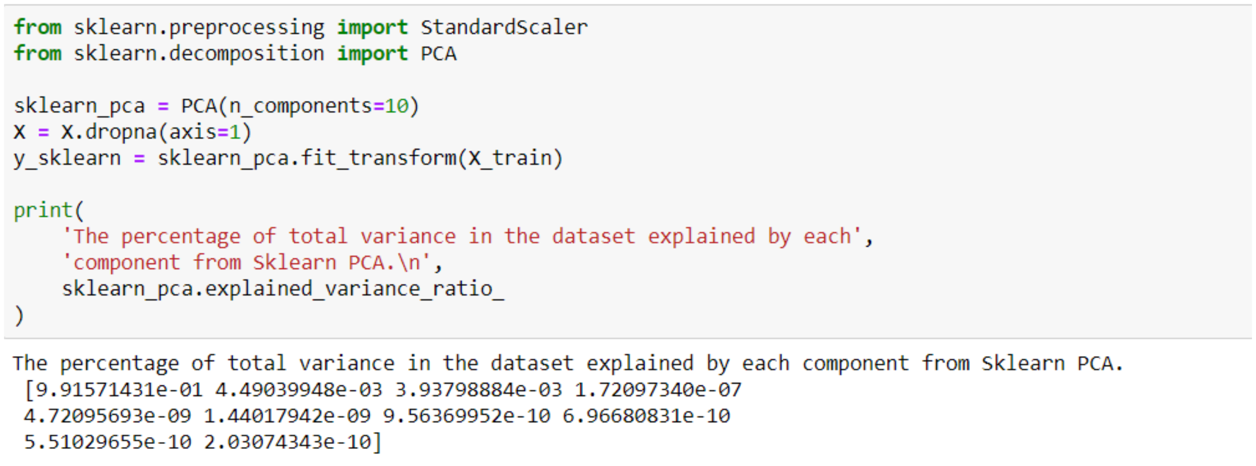
When we know: how often B happens given that A happens, written P(B|A)

and how likely A is on its own, written P(A)

and how likely B is on its own, written P(B)

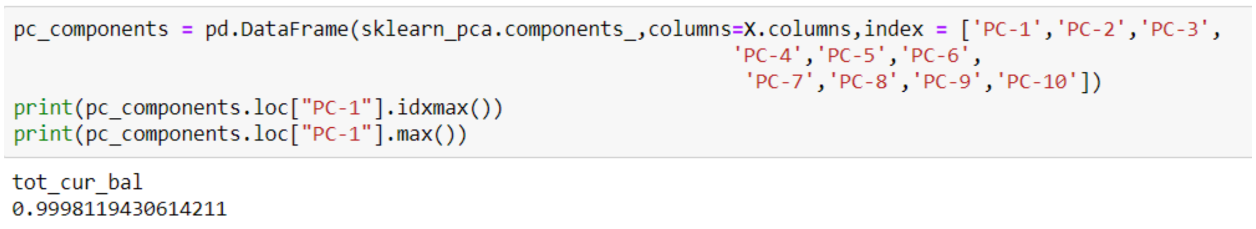
**Principle Component Analysis:**

PCA was performed to identify top 10 components explaining the variance in the data.
99% of the variance can be explained by only one component that is tot_cur_bal.



```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

sklearn_pca = PCA(n_components=10)
X = X.dropna(axis=1)
y_sklearn = sklearn_pca.fit_transform(X_train)

print(
    'The percentage of total variance in the dataset explained by each',
    'component from Sklearn PCA.\n',
    sklearn_pca.explained_variance_ratio_
)
```

```
The percentage of total variance in the dataset explained by each component from Sklearn PCA.
 [9.91571431e-01 4.49039948e-03 3.93798884e-03 1.72097340e-07
 4.72095693e-09 1.44017942e-09 9.56369952e-10 6.96680831e-10
 5.51029655e-10 2.03074343e-10]
```



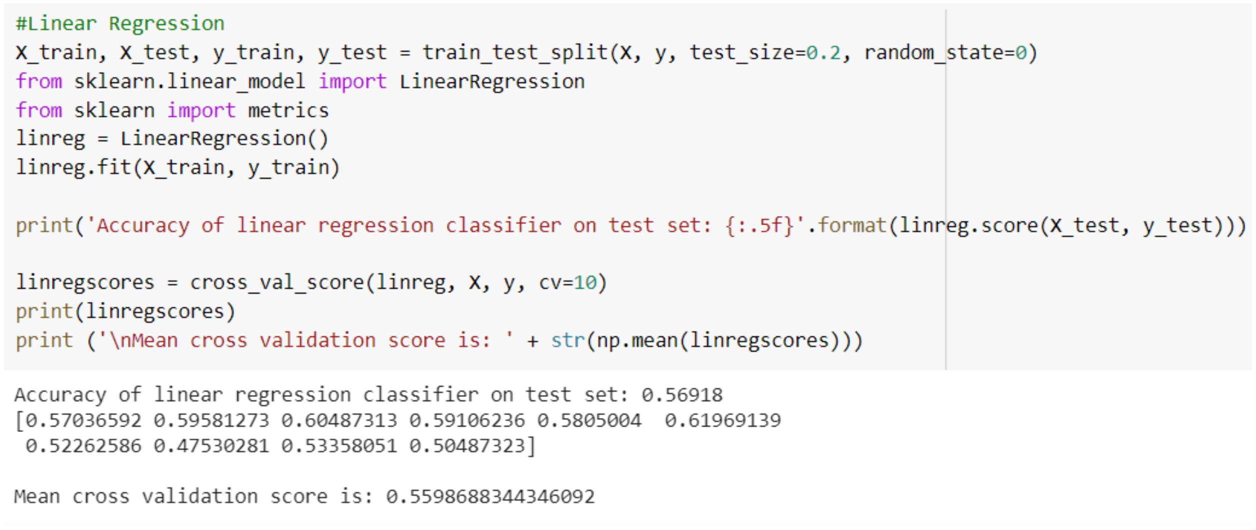
```
pc_components = pd.DataFrame(sklearn_pca.components_,columns=X.columns,index = ['PC-1','PC-2','PC-3',
                                                              'PC-4','PC-5','PC-6',
                                                               'PC-7','PC-8','PC-9','PC-10'])
print(pc_components.loc["PC-1"].idxmax())
print(pc_components.loc["PC-1"].max())
```

```
tot_cur_bal
0.9998119430614211
```

## *Algorithms Used:*

**Linear Regression:**

Linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

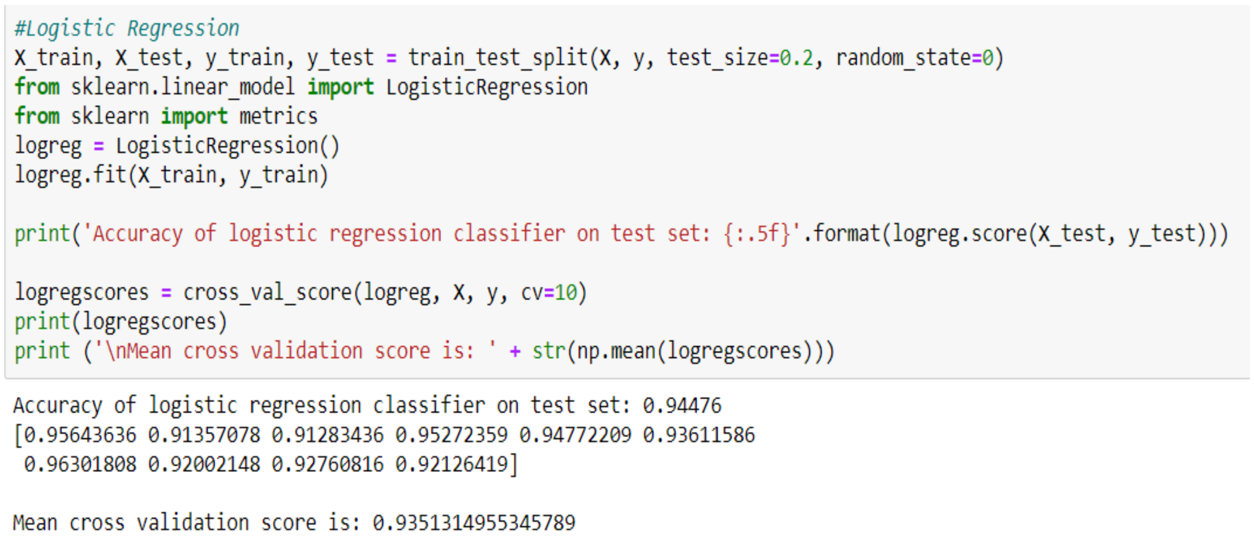


```
#Linear Regression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
from sklearn.linear_model import LinearRegression
from sklearn import metrics
linreg = LinearRegression()
linreg.fit(X_train, y_train)

print('Accuracy of linear regression classifier on test set: {:.5f}'.format(linreg.score(X_test, y_test)))

linregscores = cross_val_score(linreg, X, y, cv=10)
print(linregscores)
print ('\nMean cross validation score is: ' + str(np.mean(linregscores)))
```

```
Accuracy of linear regression classifier on test set: 0.56918
[0.57036592 0.59581273 0.60487313 0.59106236 0.5805004  0.61969139
 0.52262586 0.47530281 0.53358051 0.50487323]

Mean cross validation score is: 0.5598688344346092
```

**Logistic Regression:**

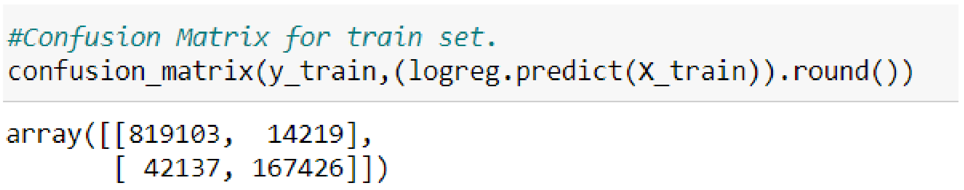
Logistic regression is the appropriate regression analysis to conduct when the dependent variable is binary. Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.



```
#Logistic Regression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

print('Accuracy of logistic regression classifier on test set: {:.5f}'.format(logreg.score(X_test, y_test)))

logregscores = cross_val_score(logreg, X, y, cv=10)
print(logregscores)
print ('\nMean cross validation score is: ' + str(np.mean(logregscores)))
```

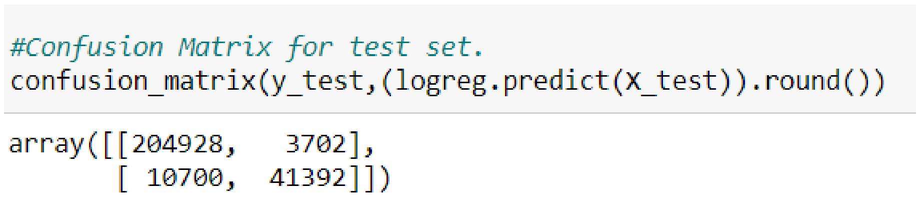
```
Accuracy of logistic regression classifier on test set: 0.94476
[0.95643636 0.91357078 0.91283436 0.95272359 0.94772209 0.93611586
 0.96301808 0.92002148 0.92760816 0.92126419]

Mean cross validation score is: 0.9351314955345789
```

The accuracy for the dataset using Logistic regression for which the loan will be waived off is 0.94476.



```
#Confusion Matrix for train set.
confusion_matrix(y_train,(logreg.predict(X_train)).round())
```

```
array([[819103,  14219],
       [ 42137, 167426]])
```

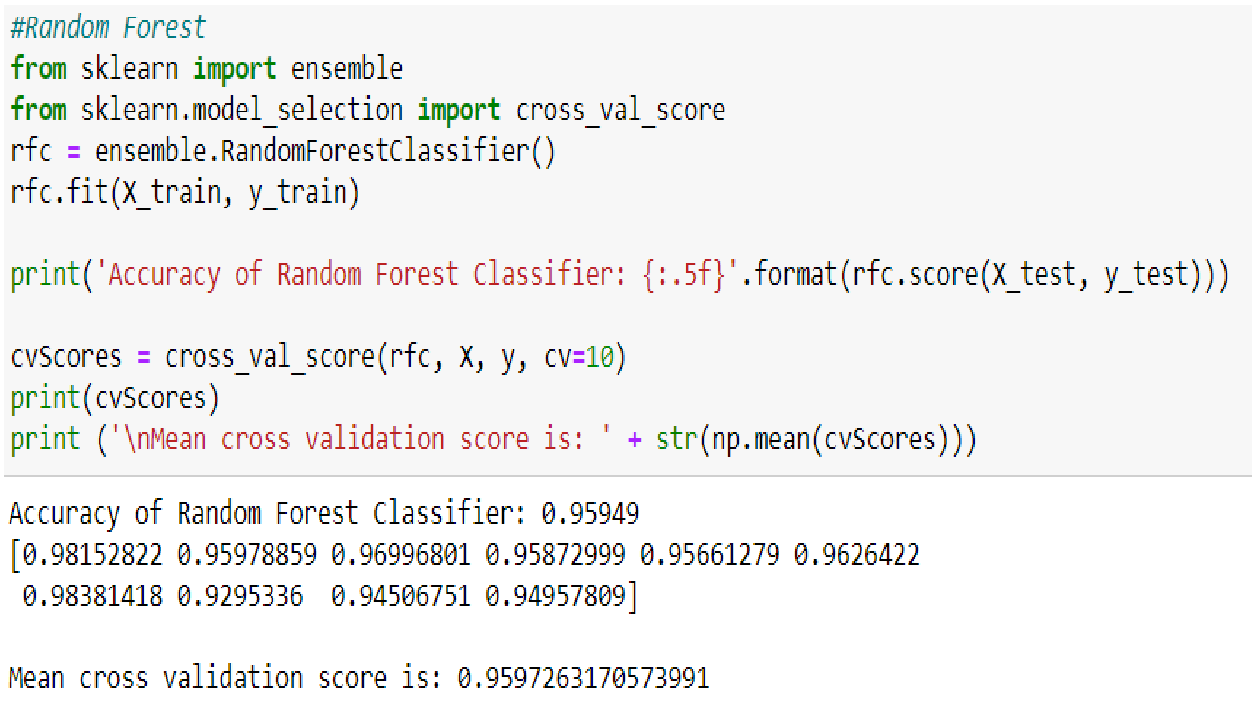


```
#Confusion Matrix for test set.
confusion_matrix(y_test,(logreg.predict(X_test)).round())
```

```
array([[204928,   3702],
       [ 10700,  41392]])
```

**Random Forest Algorithm:**

Random forest is a supervised learning algorithm which is used for both classification as well as regression. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.



```
#Random Forest
from sklearn import ensemble
from sklearn.model_selection import cross_val_score
rfc = ensemble.RandomForestClassifier()
rfc.fit(X_train, y_train)

print('Accuracy of Random Forest Classifier: {:.5f}'.format(rfc.score(X_test, y_test)))

cvScores = cross_val_score(rfc, X, y, cv=10)
print(cvScores)
print ('\nMean cross validation score is: ' + str(np.mean(cvScores)))
```

```
Accuracy of Random Forest Classifier: 0.95949
[0.98152822 0.95978859 0.96996801 0.95872999 0.95661279 0.9626422
 0.98381418 0.9295336  0.94506751 0.94957809]

Mean cross validation score is: 0.9597263170573991
```

The accuracy for the dataset using Random Forest algorithm for which the loan will be waived off is 0.95949.

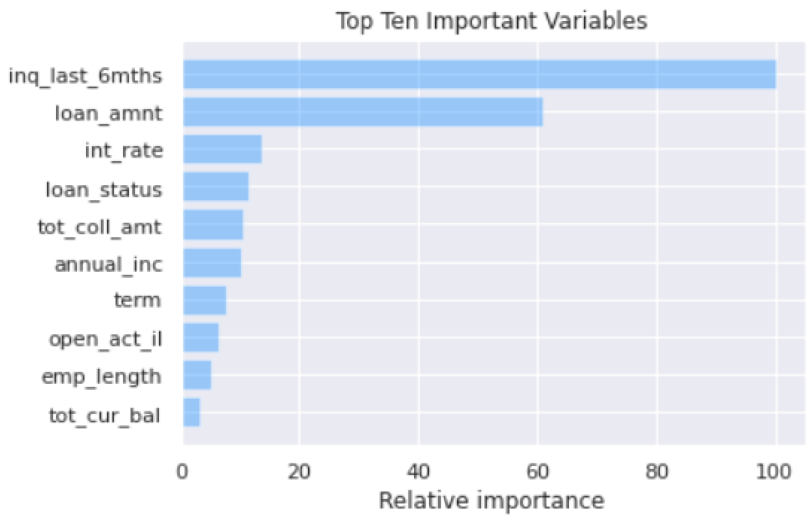
```
#Confusion Matrix for train set.
confusion_matrix(y_train,(rfc.predict(X_train)).round())
```

```
array([[833322,      0],
       [    12, 209551]])
```

```
#Confusion Matrix for test set.
confusion_matrix(y_test,(rfc.predict(X_test)).round())
```

```
array([[208540,     90],
       [ 10472,  41620]])
```

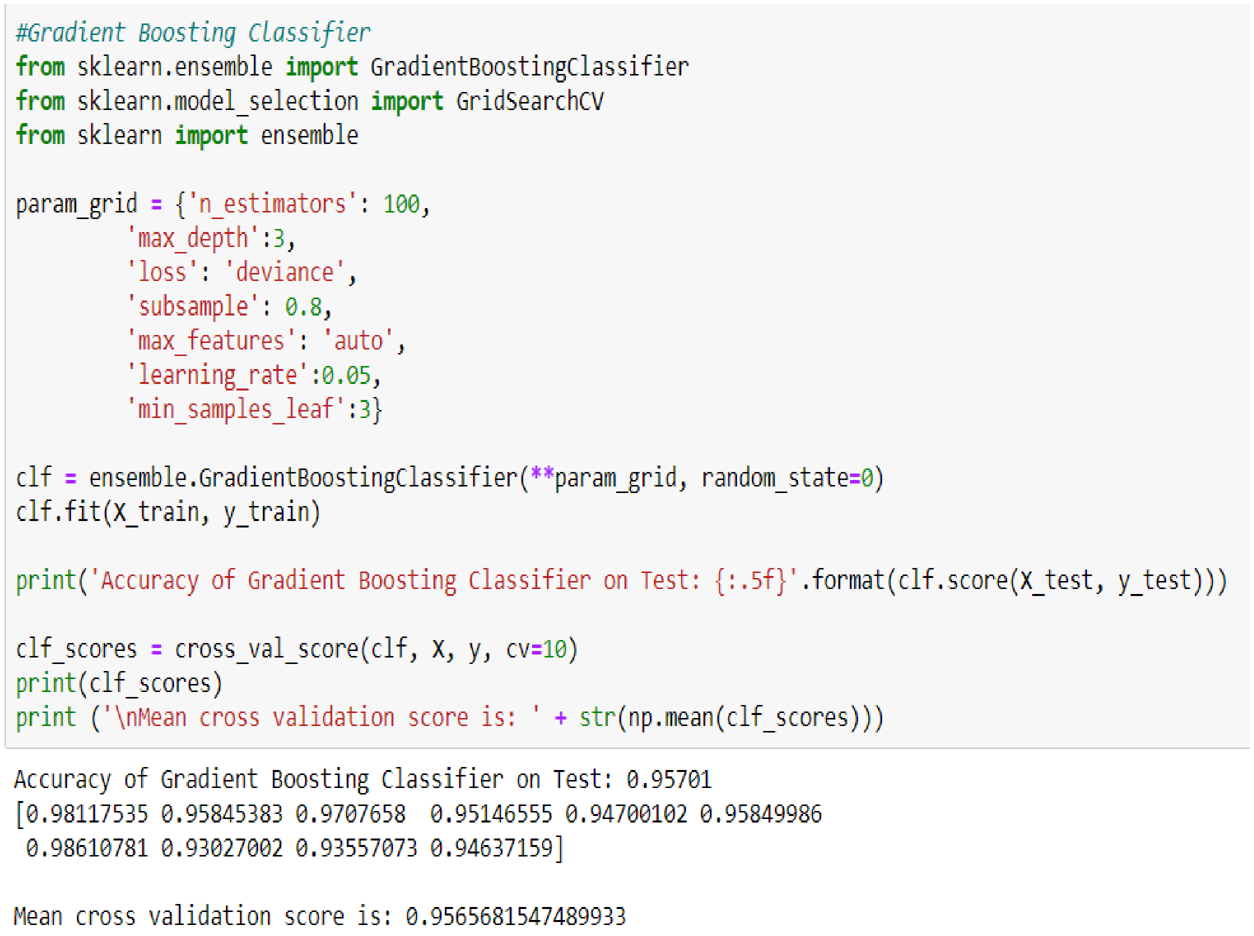
```
feature_importance = rfc.feature_importances_

#Make importances relative to max importance.
feature_importance = 100 * (feature_importance / feature_importance.max())
indices = np.argsort(feature_importance)[-10:]
plt.barh(np.arange(10), feature_importance[indices],color='dodgerblue',alpha=.4)
plt.yticks(np.arange(10), np.array(df.columns)[indices])
_ = plt.xlabel('Relative importance'), plt.title('Top Ten Important Variables')
```



Top 10 important variables based on random forest classifier

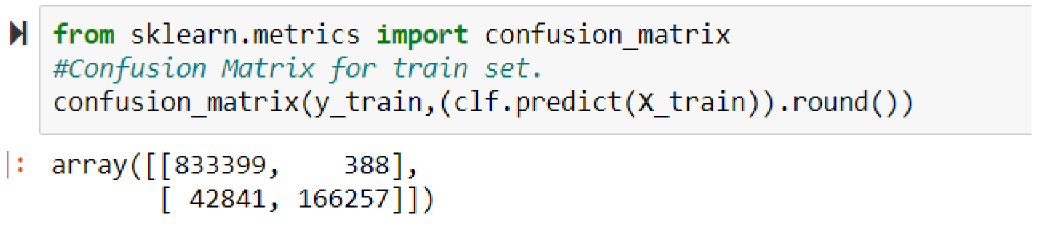
**Gradient Boosting Classifier:**

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set.



```
#Gradient Boosting Classifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import ensemble

param_grid = {'n_estimators': 100,
          'max_depth':3,
          'loss': 'deviance',
          'subsample': 0.8,
          'max_features': 'auto',
          'learning_rate':0.05,
          'min_samples_leaf':3}

clf = ensemble.GradientBoostingClassifier(**param_grid, random_state=0)
clf.fit(X_train, y_train)

print('Accuracy of Gradient Boosting Classifier on Test: {:.5f}'.format(clf.score(X_test, y_test)))

clf_scores = cross_val_score(clf, X, y, cv=10)
print(clf_scores)
print ('\nMean cross validation score is: ' + str(np.mean(clf_scores)))
```

```
Accuracy of Gradient Boosting Classifier on Test: 0.95701
[0.98117535 0.95845383 0.9707658  0.95146555 0.94700102 0.95849986
 0.98610781 0.93027002 0.93557073 0.94637159]

Mean cross validation score is: 0.9565681547489933
```



```
from sklearn.metrics import confusion_matrix
#Confusion Matrix for train set.
confusion_matrix(y_train,(clf.predict(X_train)).round())
```

```
array([[833399,    388],
       [ 42841, 166257]])
```

The accuracy for the dataset using Gradient Boosting Classifier for which the loan will be waived off is 0.95701.

**Naïve Bayes Classifier:**

A naive Bayes classifier is an algorithm that uses Bayes' theorem to classify objects.

```
#Naive Bayes Classifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import BernoulliNB
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#Testing Naive Bayes Classifier
BNB = BernoulliNB()
BNB.fit(X_train, y_train)

y_pred = BNB.predict(X_test)

print('Accuracy of Bernoulli classifier: {:.5f}'.format(BNB.score(X_test, y_test)))

BNB_scores = cross_val_score(BNB, X, y, cv=10)
print(BNB_scores)
print ('\nMean cross validation score is: ' + str(np.mean(BNB_scores)))
```

```
Accuracy of Bernoulli classifier: 0.80001
[0.77593759 0.78327874 0.7910418  0.79806077 0.79382637 0.76480696
 0.74682612 0.78873121 0.7837757  0.784305  ]

Mean cross validation score is: 0.7810590259448037
```

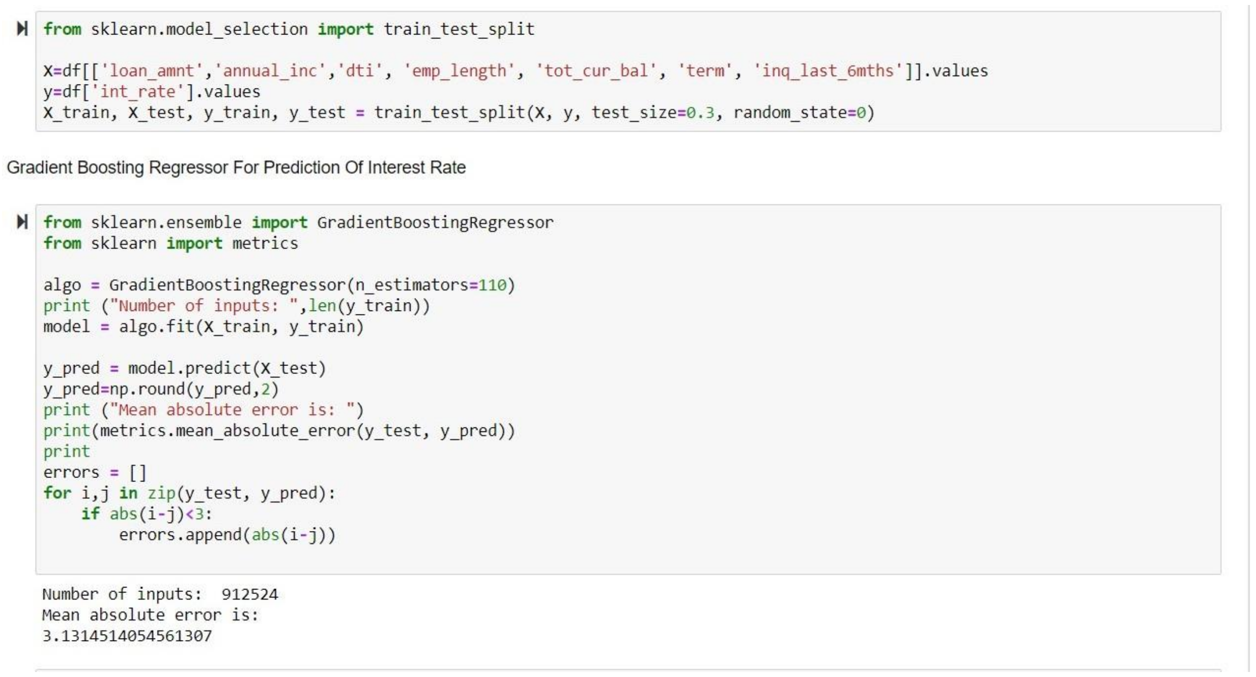
```
from sklearn.metrics import confusion_matrix
#Confusion Matrix for train set.
confusion_matrix(y_train,(BNB.predict(X_train)).round())
```

```
array([[743366,  90374],
       [117848,  91297]])
```

The accuracy for the dataset using Naïve Bayes algorithm for which the loan will be waived off is 0.80001.

**Interest Rate Prediction:**

Using Gradient Boosting Classifier, we predict the interest rate for a loan based on the given factors like loan amount, annual income, debt to income ratio, employment length, total current balance, term etc. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set.

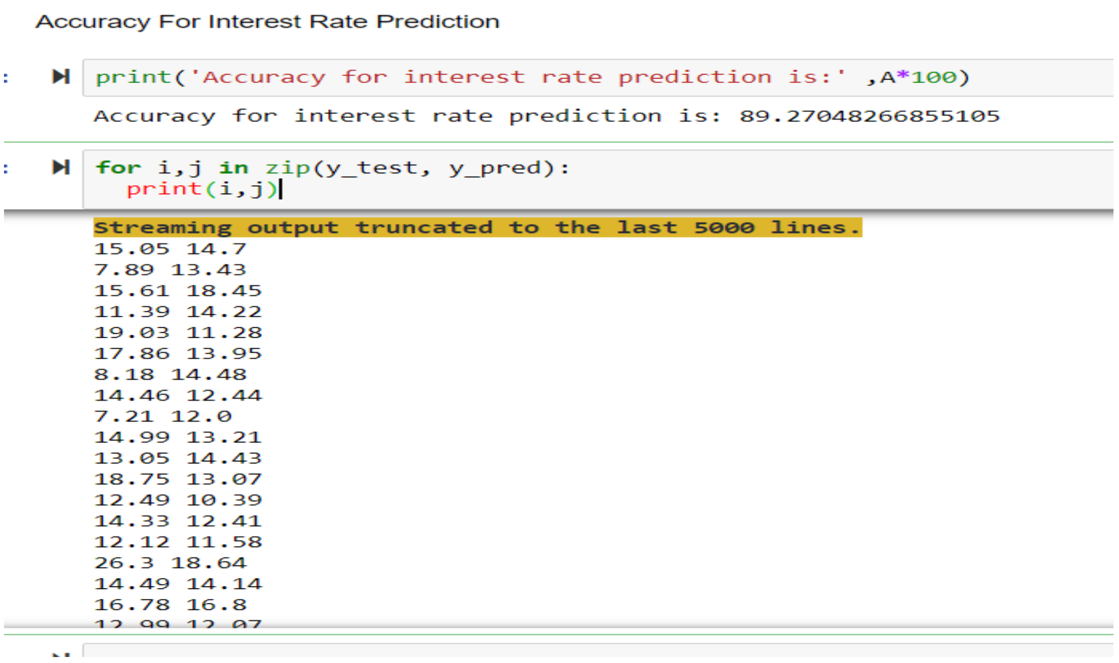


```
from sklearn.model_selection import train_test_split

X=df[['loan_amnt','annual_inc','dti', 'emp_length', 'tot_cur_bal', 'term', 'inq_last_6mths']].values
y=df['int_rate'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Gradient Boosting Regressor For Prediction Of Interest Rate

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn import metrics

algo = GradientBoostingRegressor(n_estimators=110)
print ("Number of inputs: ",len(y_train))
model = algo.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_pred=np.round(y_pred,2)
print ("Mean absolute error is: ")
print(metrics.mean_absolute_error(y_test, y_pred))
print
errors = []
for i,j in zip(y_test, y_pred):
    if abs(i-j)<3:
        errors.append(abs(i-j))
```

```
Number of inputs:  912524
Mean absolute error is:
3.1314514054561307
```



Accuracy For Interest Rate Prediction

```
print('Accuracy for interest rate prediction is:' ,A*100)
```

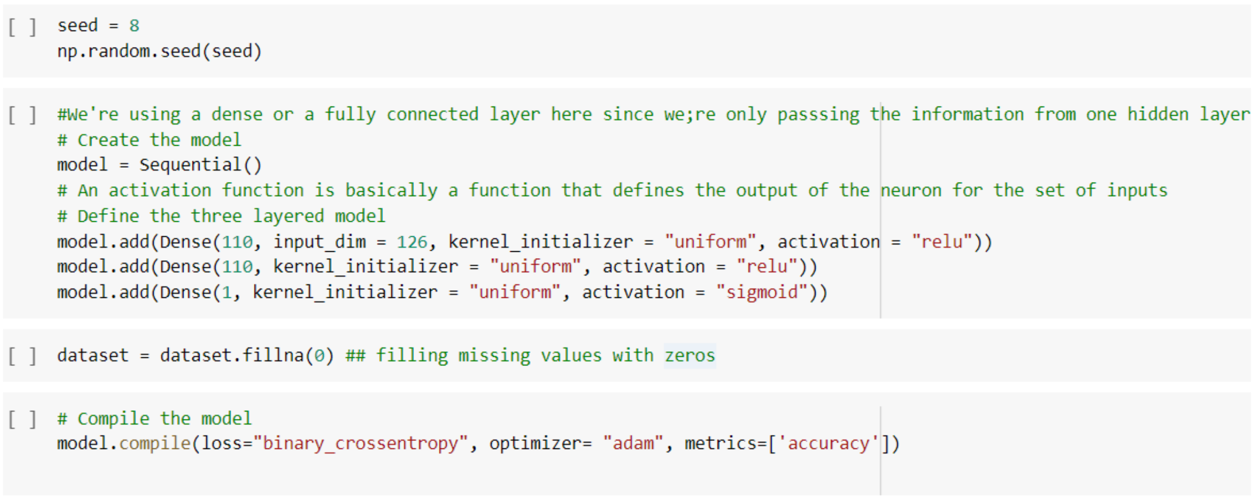
```
Accuracy for interest rate prediction is: 89.27048266855105
```

```
for i,j in zip(y_test, y_pred):
  print(i,j)
```

```
Streaming output truncated to the last 5000 lines.
15.05 14.7
7.89 13.43
15.61 18.45
11.39 14.22
19.03 11.28
17.86 13.95
8.18 14.48
14.46 12.44
7.21 12.0
14.99 13.21
13.05 14.43
18.75 13.07
12.49 10.39
14.33 12.41
12.12 11.58
26.3 18.64
14.49 14.14
16.78 16.8
12.99 12.07
```

**Deep Learning:**

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. We're using a 3-layered dense neural network for classification of the potential loan defaulters by training the model for 50 epochs.

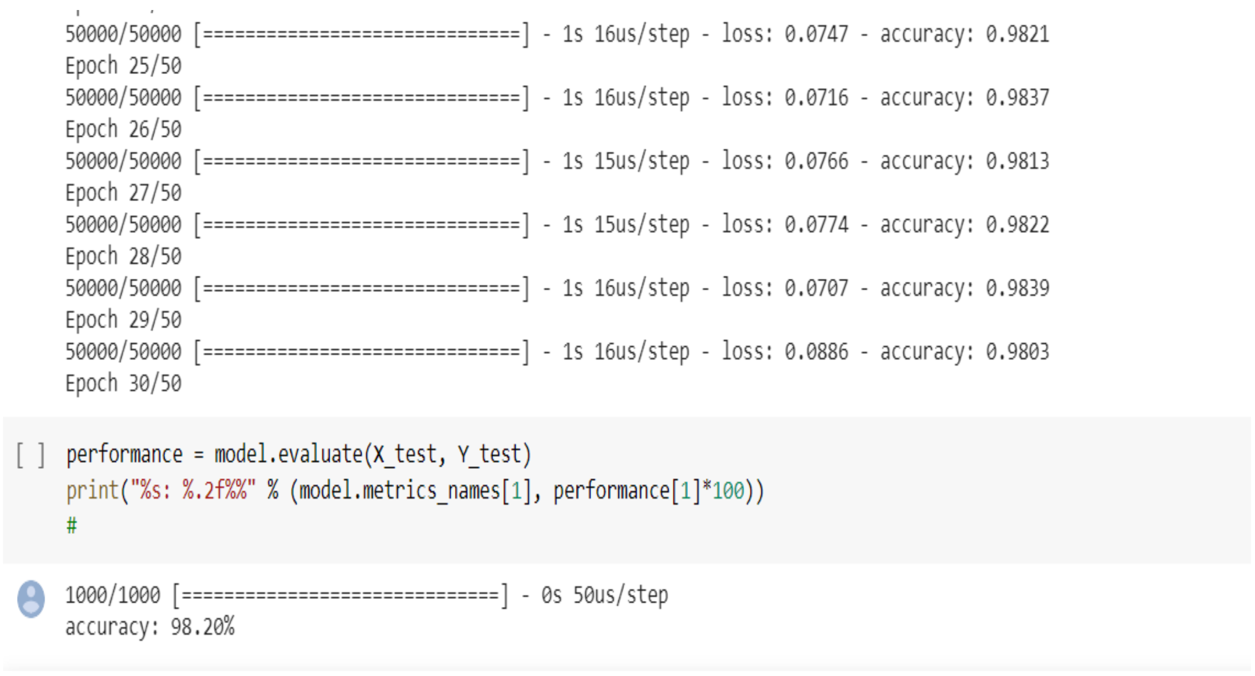


```
[ ] seed = 8
    np.random.seed(seed)
```

```
[ ] #We're using a dense or a fully connected layer here since we;re only passsing the information from one hidden layer
    # Create the model
    model = Sequential()
    # An activation function is basically a function that defines the output of the neuron for the set of inputs
    # Define the three layered model
    model.add(Dense(110, input_dim = 126, kernel_initializer = "uniform", activation = "relu"))
    model.add(Dense(110, kernel_initializer = "uniform", activation = "relu"))
    model.add(Dense(1, kernel_initializer = "uniform", activation = "sigmoid"))
```

```
[ ] dataset = dataset.fillna(0) ## filling missing values with zeros
```

```
[ ] # Compile the model
    model.compile(loss="binary_crossentropy", optimizer= "adam", metrics=['accuracy'])
```



```
50000/50000 [==============================] - 1s 16us/step - loss: 0.0747 - accuracy: 0.9821
Epoch 25/50
50000/50000 [==============================] - 1s 16us/step - loss: 0.0716 - accuracy: 0.9837
Epoch 26/50
50000/50000 [==============================] - 1s 15us/step - loss: 0.0766 - accuracy: 0.9813
Epoch 27/50
50000/50000 [==============================] - 1s 15us/step - loss: 0.0774 - accuracy: 0.9822
Epoch 28/50
50000/50000 [==============================] - 1s 16us/step - loss: 0.0707 - accuracy: 0.9839
Epoch 29/50
50000/50000 [==============================] - 1s 16us/step - loss: 0.0886 - accuracy: 0.9803
Epoch 30/50
```

```
[ ] performance = model.evaluate(X_test, Y_test)
    print("%s: %.2f%%" % (model.metrics_names[1], performance[1]*100))
    #
```

```
1000/1000 [==============================] - 0s 50us/step
accuracy: 98.20%
```

## Result and Analysis:

**Loan Waive off Accuracy:**

Logistic regression - 0.94476
Gradient Boosting Classifier - 0.95701 (The best of all)
Random Forest algorithm - 0.95949
Naïve Bayes algorithm - 0.80001
Linear Regression – 0.56918

**Interest Rate Prediction:**

Accuracy for interest rate prediction is 89.27 % using Gradient Boost Classifier.

**Potential Loan Defaulter Prediction:**

Accuracy for the forecasting of the potential loan defaulters based on their current loan status using neural networks is 98.2%

## Conclusion:

Lending Club's data is a great source of information on personal credit. Additionally, this data set is bound to grow exponentially over the next years. In this project we have tried to predict the loan waive off accuracy by using different algorithms and have got the result to a 95% accuracy. With the interest rate prediction, we can easily predict what the interest rate would be for an individual loan. Also, examining the current status of an individual's loan repayment we are forecasting whether will this individual be a loan defaulter soon. Analyzing the loan data, Lending Club had 78% of the loans that were in good standing with respect to the loan paid, installments and waive off accuracy. Hence one can say that it is worth investing with Lending Club.

## Limitations:

- Data only from 2005-2018
- Data has not been tested during periods of financial crisis, as interest rates and default rates would likely increase during those periods.
- Credit scores were not given for 2018's data. This would likely be another feature that could affect the model.

## References:

https://www.kaggle.com/wendykan/lending-club-loan-data#loan.csv

https://www.edureka.co/blog/logistic-regression-in-python/

https://geekflare.com/choosing-ml-algorithms/