Interim Project report on Credit Score Prediction

Submitted By
Online June '21 'B' Batch – Group 1

Group Members

1. Albert Antony
2. Akash Saxena
3. Arjoo Singh
4. Sumanth Inuganti
5. Impu D M
6. B Meghana

Research Supervisor

Mr. Animesh Tiwari

Great Lakes Institute of Management

Table of Contents

1.INTRODUCTION

In today's world, taking loans from financial institutions has become a very common phenomenon. Everyday a large number of people apply for loans, for a variety of purposes. But all these applicants are not reliable and everyone cannot be approved. Every year, we read about a number of cases where people do not repay bulk of the loan amount to the banks due to which they suffer huge losses. The risk associated with making a decision on loan approval is immense.

Credit Risk assessment is a crucial issue faced by Banks nowadays which helps them to evaluate if a loan applicant can be a defaulter at a later stage so that they can go ahead and grant the loan or not. This helps the banks to minimize the possible losses and can increase the volume of credits. Nowadays there are many risks related to bank loans, especially for the banks so as to reduce their capital loss. The analysis of risks and assessment of default becomes crucial thereafter. Banks hold huge volumes of customer behavior related data from which they are unable to arrive at a judgement if an applicant can be defaulter or not.

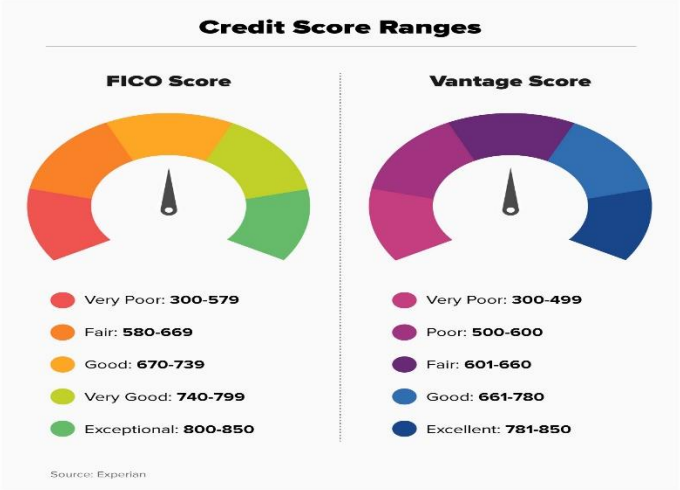
So the idea of this project is to gather loan data from multiple data sources and use various machine learning algorithms on this data to extract important information. This model can be used by the organizations in making the right decision to approve or reject the loan request of the customers. In this paper, we examine a real bank credit data and conduct several machine learning algorithms on the data for that determine credit worthiness of customers to formulate bank risk automated system.

2.LITERATURE SUMMARY

2.1 History and Applications of Credit Score

Credit Scores were invented in 1950's. In 1956, engineer Bill Fa teamed up with mathematician Earl Isaac to create Fair, Isaac and Company, with the goal of creating a standardized, impartial credit scoring system.[1] Today, the application of credit scoring has evolved from traditional decision making of accepting or rejecting an application for credit to inclusion of other facets of the credit process such as the pricing of financial services to reflect the risk profile of the consumer or business and the setting of credit limits.

Both FICO Score and Vantage Score are designed to measure how financially stable the person is and how likely to be able to pay back debt. [1]



2.2 Analysis of Customer Segmentation of Bank using Data Mining Technique

Competitiveness in the banking industry has made the task of raising loanable funds more difficult as customer needs has become more complex. Objectives of this research includes identification of customer characteristics based on their job profile, identification of customer behavior (current balance in savings account for each segment) drilled down by customer profile (age, tier, gender, balance and education background). Based on customer needs identification, management analyses customer profiles by using data mining to define the market.[4]

Author: Arta Moro Sundjaja | Binus University

2.3 Handle Imbalance Classification Problem in Machine Learning

While performing the machine learning or any operation on the data that is imbalanced the model will be inaccurate and biased. This problem is predominant in scenarios where anomaly detection is crucial for data. Standard classifier algorithms like Decision Tree and Logistic Regression have a bias towards classes which have number of instances. This happens because Machine Learning Algorithms. [6]

2.4 Methodology to predict Loan defaulters in Indian Banking System

The main objective of this research is to propose a methodology for predicting the credit score which can therebybe used to identify the defaulter's and Non defaulters and gauge the crisis which banking systems are facing.
There are many forecasting methods which can be used to predict the variable of interest. They are broadly divided into two categories: 1) Regression models: Simple Linear Regression (SLR), Multiple Linear Regression, 2) Time series-based methods: Moving Average, Simple Exponential Smoothing, Holt,and Holt-winter method.[7]

3. SCOPE AND OBJECTIVES

3.1 Problem Statement

In the above context we have seen that all Public Sector and Commercial Banks have been inculcating huge loss for unable to identify loan defaulters. So, in our project we will be estimating risk simulation involved in identifying the loyal customer based on their credit score and other features in giving out loans by the banks.

3.2 Aim of the Project

The loan is one of the most important products of the banking. All the banks are trying to figure out effective business strategies to persuade customers to apply for their loans. However, there are some customers who behave negatively after their application are approved. To prevent this situation, banks have to find some methods to predict customers' behaviors. We need to identify defaulters of the bank. And also build a recommendation engine that will help the financial institutions to identify the loan defaulters before approving the loan applications. Machine learning algorithms have a pretty good performance on this purpose, which are widely-used by the banking. Here, we will work on loan behaviors prediction using machine learning models.

We aim to build a model which can give a good accuracy to predict the good credit which help financial institutions in preventing the huge losses.

3.3 Objectives

The objective of this project is to build a model for the lenders to help them understand who are eligible to avail loans and how much credit can be lent to that customer. This way a bank can significantly reduce its credit lending risk.

Hence in this project we present a data mining framework for credit score prediction from a given set of data using the data mining techniques and machine learning. The core objectives of this project are illustrated below:

- Exploring the dataset and analyze the inconsistencies like missing values, etc.
- Data pre-processing such as Outlier Detection, Outlier Removal, Imputations, Splitting Training & Test Datasets, Balancing Training Dataset
- Features Selection such as Correlation Analysis of Features, Ranking Features

- Building Classification Model
  - Predicting Class Labels of Test Dataset
  - Evaluating Predictions.

4.Data Sources and Description

4.1 Data Set

Variables	Data Type	Description
Loan ID	Unique identifier (Object)	Loan ID specifies the ID of particular Loan availed by the corresponding Customer.
Customer ID	Unique identifier (Object)	Loan ID specifies the ID of particular Loan availed by the corresponding Customer.
Loan Status	Object	Loan Status provides information about the status of the Loan availed by the Customer.
Current Loan Amount	Float	Current Loan Amount is any loan that is fully paid till date according to the institution.
Term	Object	Short term or Long term, term chosen by the customer to repay it
Credit Score	Float	Indicator of a person's creditworthiness, or their ability to repay debt
Annual Income	Float	The total amount of money a customer makes each year before deductions are taken out of the pay
Years in current Job	Object	Displays in Duration of current job of the customers
Home Ownership	Object	Type of Home ownership of the customer
Purpose	Object	Purpose of the Loan being availed
Monthly debt	Float	Monthly debts are recurring monthly payments, such as credit card payments, loan payments
Years of Credit History	Float	Number of years in which a person pays back loans, credits, etc over time
Months since last delinquent	Float	Number of months since a borrower was late or overdue on a payment, such as income taxes, a mortgage, an automobile loan, or a credit card account
Number of Open Accounts	Float	Open account is deferred payment basis. Number of Open Accounts of a Customer
Number of Credit Problems	Float	Number of Credit Problems of a Customer
Current Credit Balance	Float	The current amount of all charges and payments made to the customer's account up to that day
Maximum Open Credit	Float	Maximum pre-approved Amount by the bank to the customer
Bankruptcies	Float	Number of times customers have gone Bankrupt
Tax Liens	Float	Number of time a customer is exempted from tax

Below is the detailed description of each of the variables.

Table 1

The dataset extracted from the GitHub. The dataset contains 19 features and 100514 records different customers. It contains information about the loan status, loan term, current loan amount, and Annual income credit score of the various customers. Here, the main purpose is to predict credit score which determined the credit worthiness of the customer to repay or default the loan it helps the banks to identify the loyal customers

```
Number of Rows: 100514
Number of Columns: 19
```

4.1 Variables considered for Analysis

The dataset contains 19 features and 100514 records different customers. We need to predict target variable i.e. credit score from the information that we have. There many missing values, variable description and improper data types. Data cleansing has been performed on the given dataset. There are few variables which has no description and meaning in data, and we have removed them since it has null values. The dataset has multiple features which needs to be analyzed and the feature has to be minimized. Factor analysis and VIF technique has been used to identify the best features that are required to get the best prediction model.

4.2 Target Variables:

Our target variable is Credit Score which we need to identify the customer will be defaulters or non-Defaulter which is the credit risk for the bank if the customer is Non Defaulter.

4.3 Missing Data:

Our data has many missing values and data discrepancies. We performed imputation to fix the missingdata by using median, mean and mode depending on the type of variables. Total missing data in our dataset is:

5. Data Pre-processing

The dataset contains 18 variables. We will be working with the 17 variables to predict credit score and build our models.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100514 entries, 0 to 100513
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   Loan ID                       100000 non-null  object
 1   Customer ID                   100000 non-null  object
 2   Loan Status                   100000 non-null  object
 3   Current Loan Amount           100000 non-null  float64
 4   Term                          100000 non-null  object
 5   Credit Score                  80846 non-null   float64
 6   Annual Income                 80846 non-null   float64
 7   Years in current job          95778 non-null   object
 8   Home Ownership                100000 non-null  object
 9   Purpose                       100000 non-null  object
 10  Monthly Debt                  100000 non-null  float64
 11  Years of Credit History       100000 non-null  float64
 12  Months since last delinquent  46859 non-null   float64
 13  Number of Open Accounts       100000 non-null  float64
 14  Number of Credit Problems     100000 non-null  float64
 15  Current Credit Balance        100000 non-null  float64
 16  Maximum Open Credit           99998 non-null   float64
 17  Bankruptcies                  99796 non-null   float64
 18  Tax Liens                     99990 non-null   float64
dtypes: float64(12), object(7)
memory usage: 14.6+ MB
```

Some of the variables are not proper and had various duplicates, when we verify there were around 18001 duplicates in ‘Loan ID’ and after dropping the duplicates only 81999 data were left. Since 'Loan ID' is used only for identification and not needed in modeling. We have drop 'Loan ID' column. Hence 17 variables are left in our data.

An observation which differs from overall pattern on a sample dataset is called outlier. The outliers may suggest experimental errors, variability in a measurement, or an anomaly.

5.1 Finding null values and Missing Values in the target feature.

By using isnull() function we can find out the missing values that are present in our dataset and below figure shows those missing values for each attribute.

	Count	Percentage
Loan Status	0	0.000000
Current Loan Amount	0	0.000000
Term	0	0.000000
Credit Score	14842	23.906706
Annual Income	14842	23.906706
Years in current job	2862	4.609958
Home Ownership	0	0.000000
Purpose	0	0.000000
Monthly Debt	0	0.000000
Years of Credit History	0	0.000000
Months since last delinquent	33043	53.223910
Number of Open Accounts	0	0.000000
Number of Credit Problems	0	0.000000
Current Credit Balance	0	0.000000
Maximum Open Credit	2	0.003221
Bankruptcies	124	0.199733
Tax Liens	4	0.006443

Count and percentage of missing values

Figure 3

'Credit Score' is taken as target feature. As 23.9% missing value is present in target feature, we will drop the null values instead of imputing it to avoid biased modelling. From above we can see that column 'Months since last delinquent' is having high percentage of missing values i.e.; 53.22%. There are missing values in Credit Score, Annual Income, Years in current job, Months since last delinquent, Maximum Open Credit, Bankruptcies and Tax Liens.
By dropping the null values of 'Credit Score', the null values of 'Annual Income' were also removed.

```
data['Credit Score'].describe()
```

```
count    47241.000000
mean      1258.477234
std       1779.949375
min        585.000000
25%        709.000000
50%        729.000000
75%        742.000000
max       7510.000000
Name: Credit Score, dtype: float64
```

```
data['Credit Score'].describe()
```

```
count    47241.000000
mean       717.946847
std         28.073787
min        585.000000
25%        705.000000
50%        725.000000
75%        739.000000
max        751.000000
Name: Credit Score, dtype: float64
```

Treatment of 'Credit Score' Variable

Figure 4

A credit score in India ranges from 300-850 and one should always take measure to bring their credit score closer to 850, but in our data the average Credit Score was not within the range of 300 to 850. Around 3990 rows had an error in their data and these rows were corrected by dividing them with 10.

In a similar way, we have treated the missing values by replacing them with 0,1 and mean as well. After these 47241 rows were left.

```
# Checking for null values.
data.isnull().sum()
```

```
Loan Status                     0
Current Loan Amount             0
Term                            0
Credit Score                    0
Annual Income                   0
Years in current job            0
Home Ownership                  0
Purpose                         0
Monthly Debt                    0
Years of Credit History         0
Months since last delinquent    0
Number of Open Accounts         0
Number of Credit Problems       0
Current Credit Balance          0
Maximum Open Credit             0
Bankruptcies                    0
Tax Liens                       0
dtype: int64
```

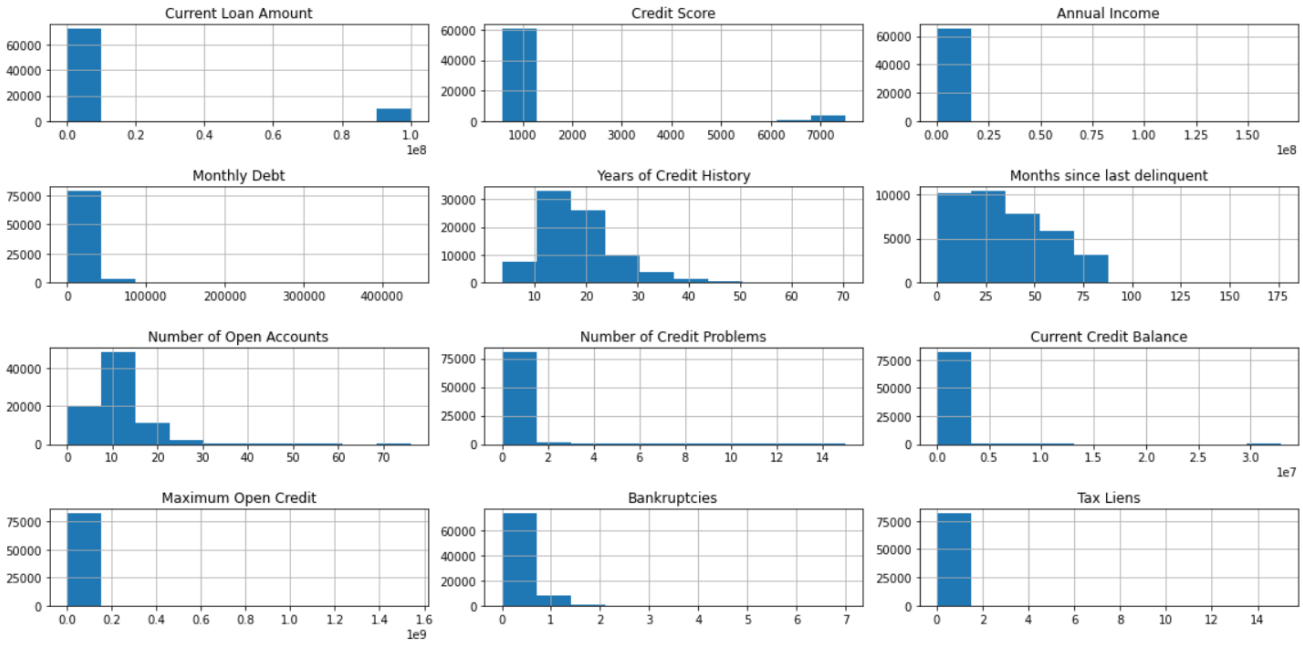
Count after treatment of missing values

Figure 5

6 Exploratory Data Analysis

6.1 Univariate Data Analysis

- Data distribution of numerical variables using histogram has beendemonstrated below:



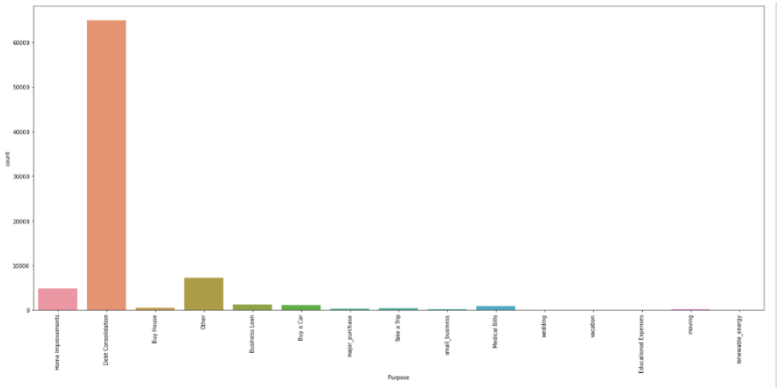
Univariate analysis of numerical variables

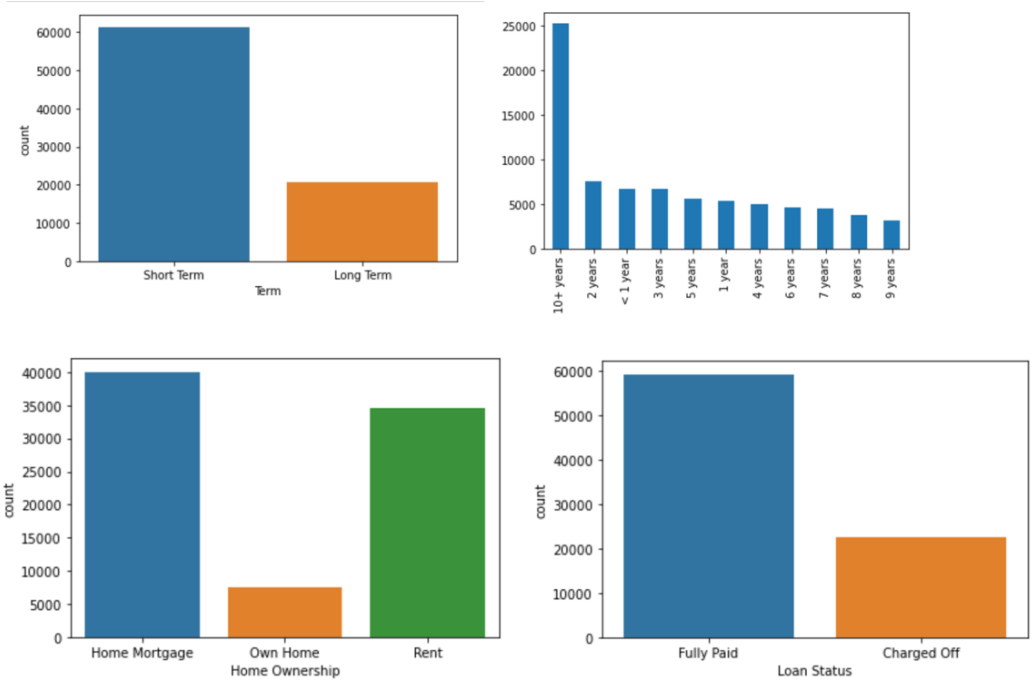
Figure 6

Here we have seen the distribution of data of numeric variables. The distribution is asymmetric i.e left or right skewed.

- Data distribution of Categorical variables has been mentioned below:

Figure 7



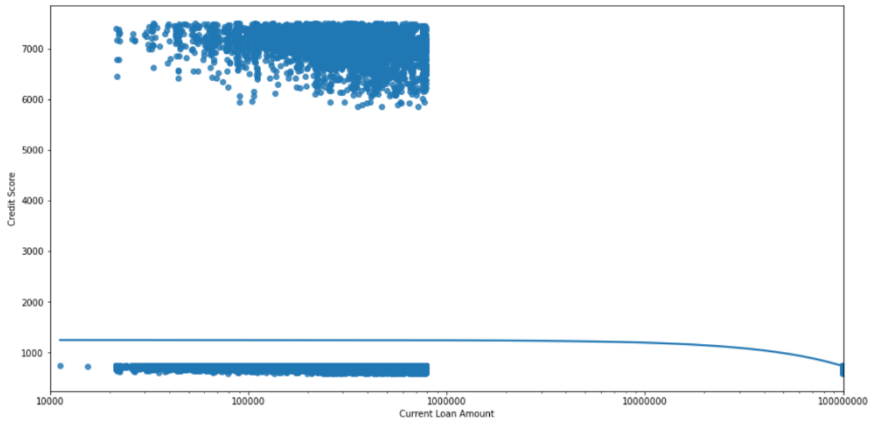


Univariate analysis of categorical variables

Figure 8

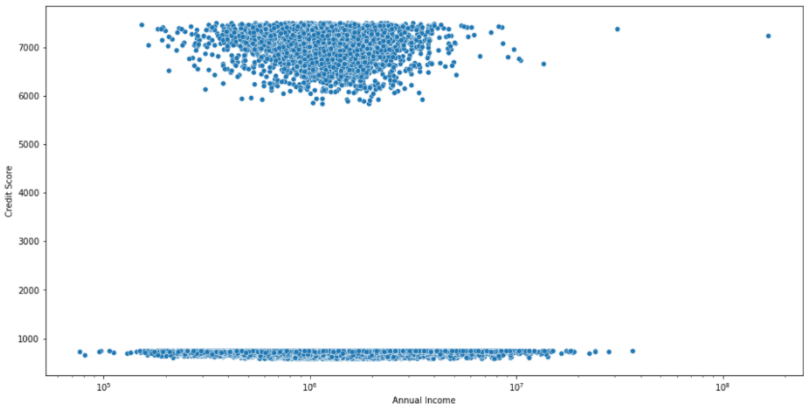
- By figure 7 its clear that purpose of loan for Debt Consolidation is huge
- Most of the loans are taken for short Terms
- 10 + years count is more compared to others.
- Around 50 percent of the customers have a home mortgage and 40 percent customers have Rented houses.
- Most of the loans have been paid fully paid. Around 27 percent loans have been charged off which accounts for the bad debts.

6.2 Bivariate Analysis

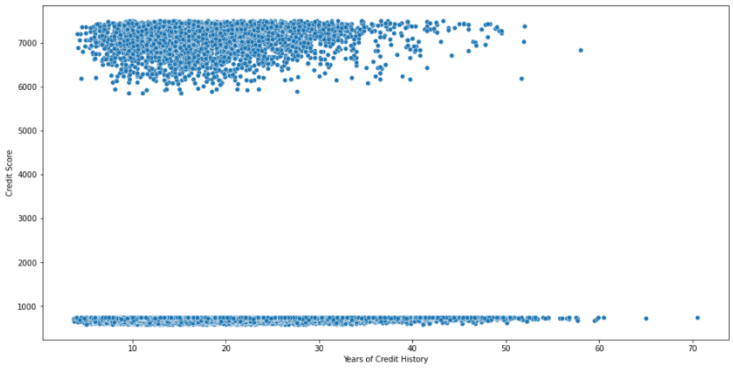


Credit Score Vs Current loan amount

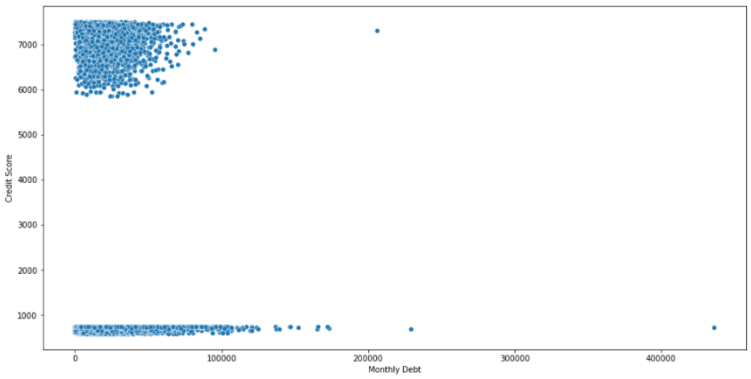
Figure 9



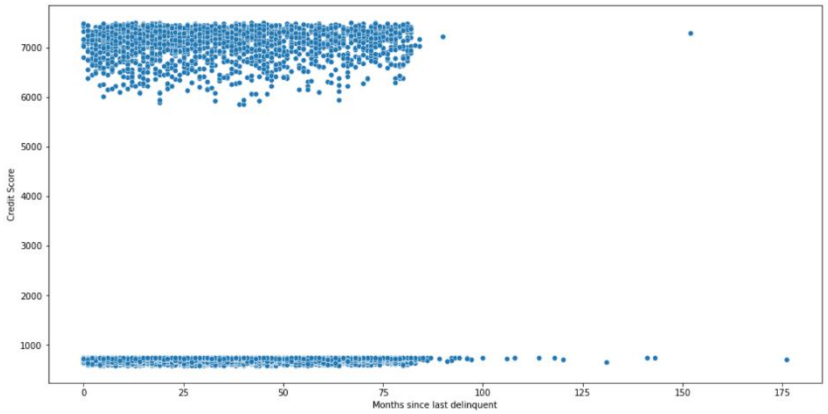
Credit Score Vs Annual Income
Figure 10



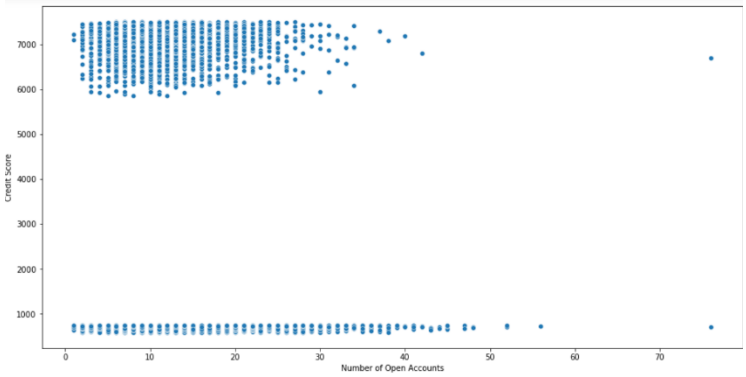
Credit Score Vs Years of Credit History
Figure 11



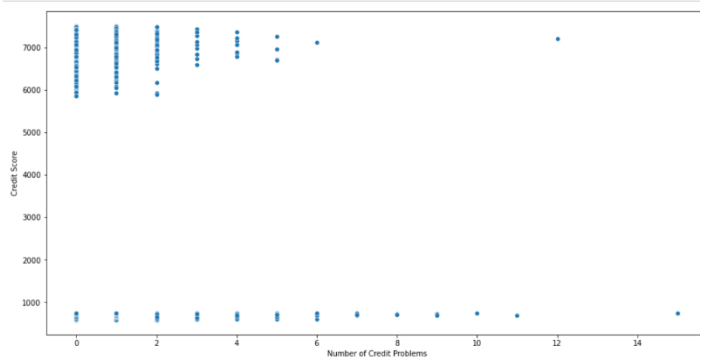
Credit Score Vs Monthly Debt
Figure 12



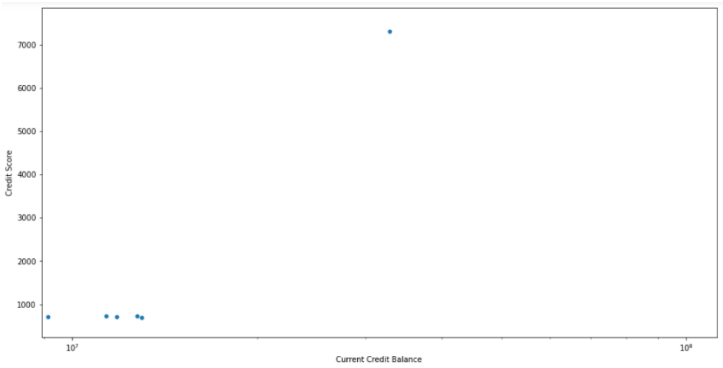
Credit Score Vs Months since last Delinquent
Figure 13



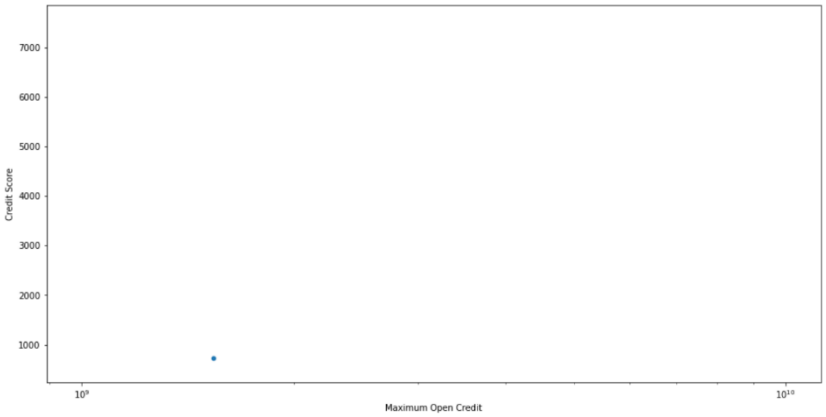
Credit Score Vs Number of open accounts
Figure 14



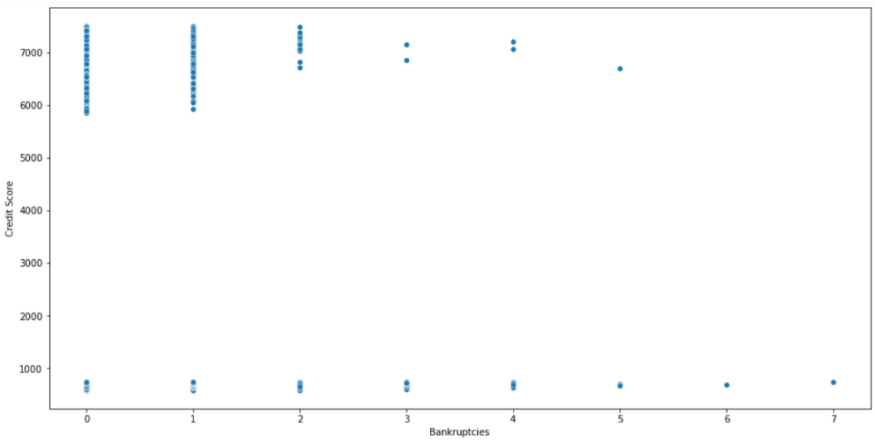
Credit Score Vs Number of credit problems
Figure 15



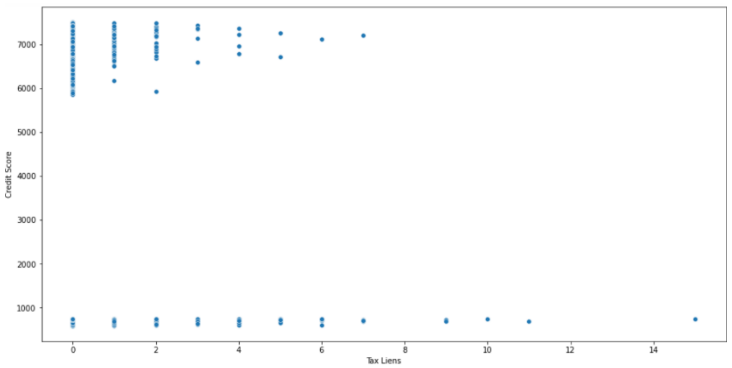
Credit Score Vs Current Credit Balance
Figure 16



Credit Score Vs Maximum open credit
Figure 17

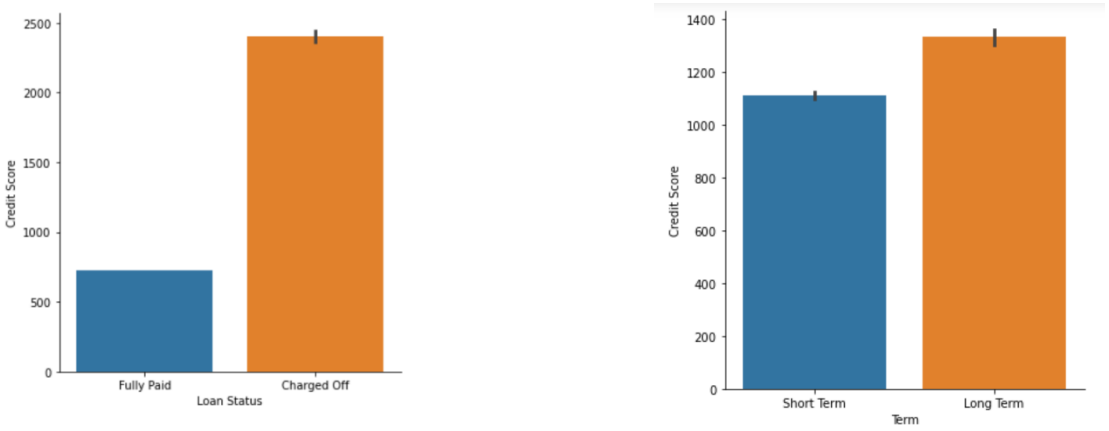


Credit Score Vs Bankruptcies
Figure 18



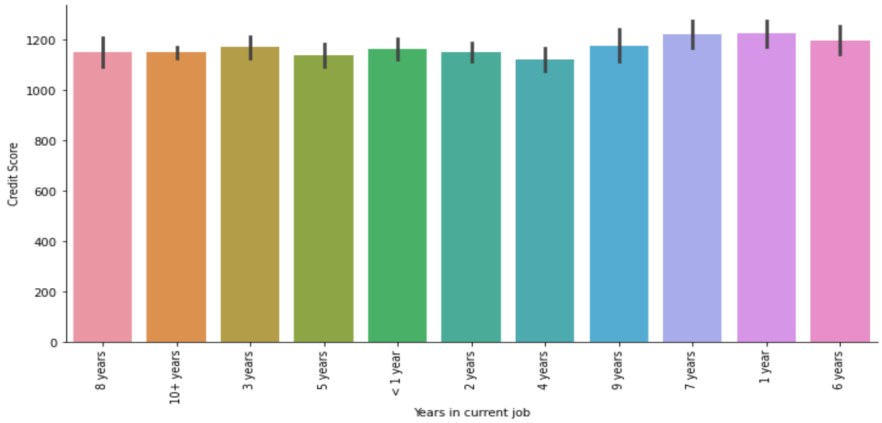
Credit Score Vs Tax Liens
Figure 19

- From graph its clear that most of features lies in range of 10000 to 1000000 with highest credit score and we can even observe the presence of outliers as credit score as been beyond range of 300-850



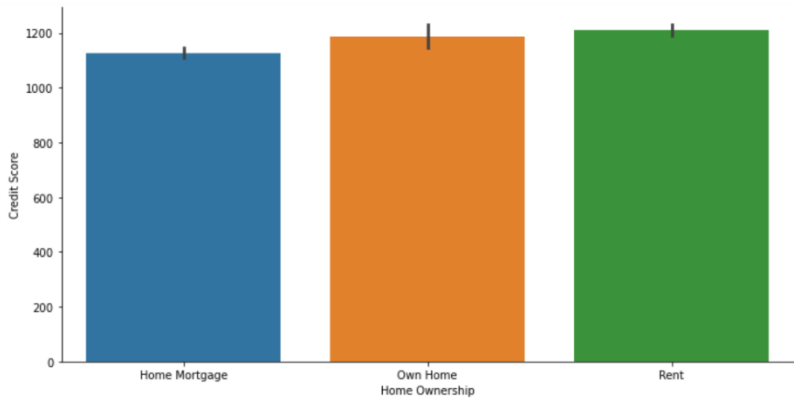
Credit Score Vs Loan status and Term
Figure 20

- Charged off have highest credit score compared to fully paid in loan status of data
- Credit score of long term is more with respect to short term
- Almost every years in current job has credit score in range of 1100 to 1200



Credit Score Vs Years in current job

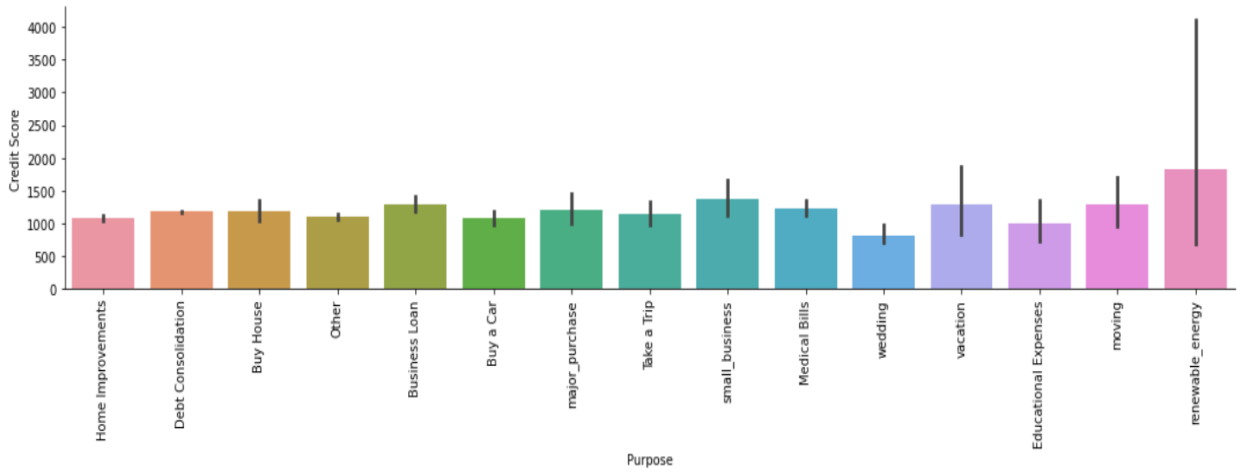
Figure 21



Credit Score Vs Home Ownership

Figure 22

- Rent type of home ownership have highest credit score

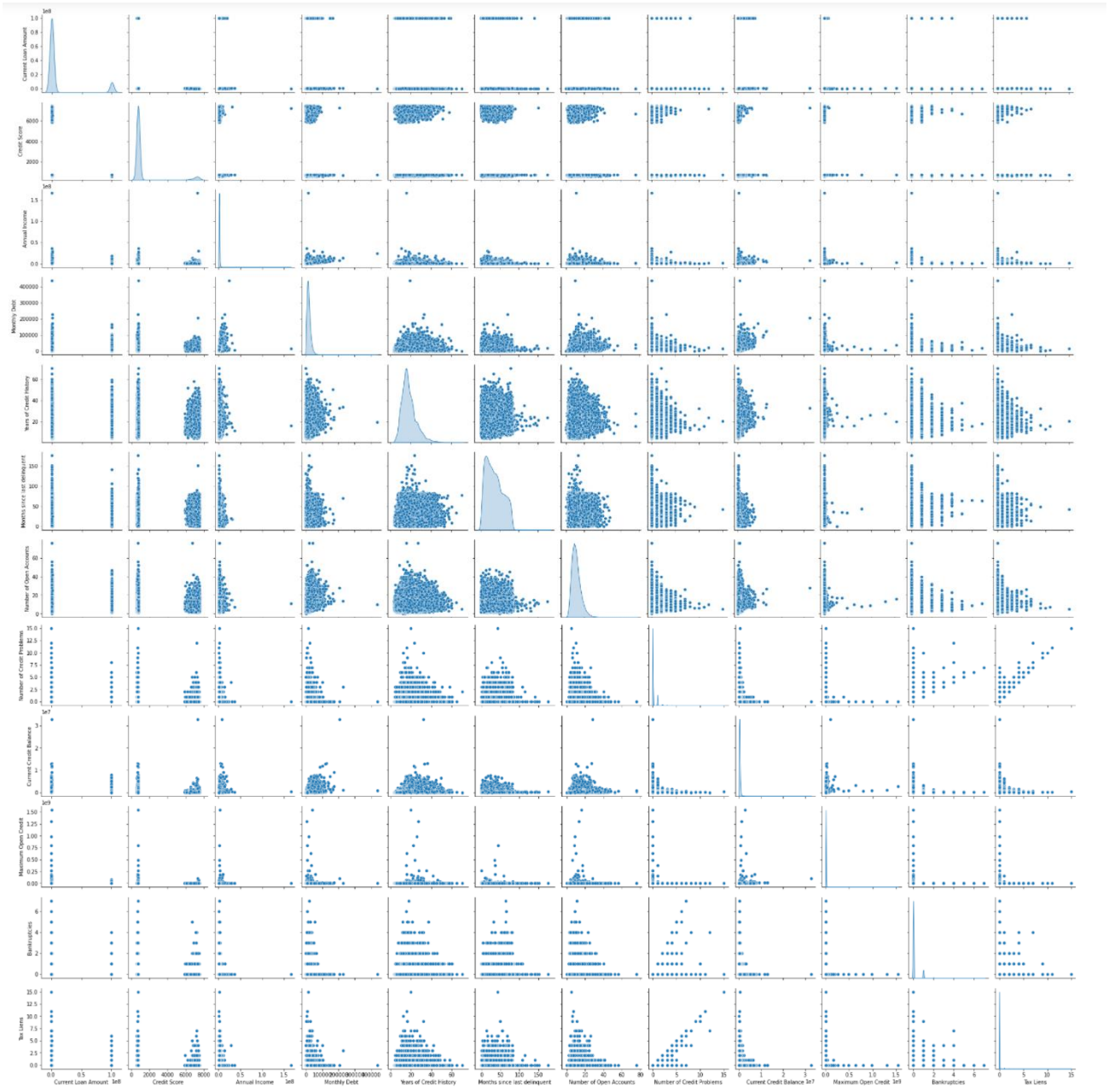


Credit Score Vs Purpose

Figure 23

- Renewable energy hast highest credit score

6.3 Multivariate Analysis



Multivariate Analysis

Figure 23

Pairplot is used to plot multiple pairwise bivariant distributions in dataset by displaying combinations of variables in data frame as a matrix of plots and the diagonals are univariant plots. We can observe the variations in each plot. The plots are in matrix format as displayed where the row name represents x-axis and column name represents y-axis.

6.4 Multicollinearity

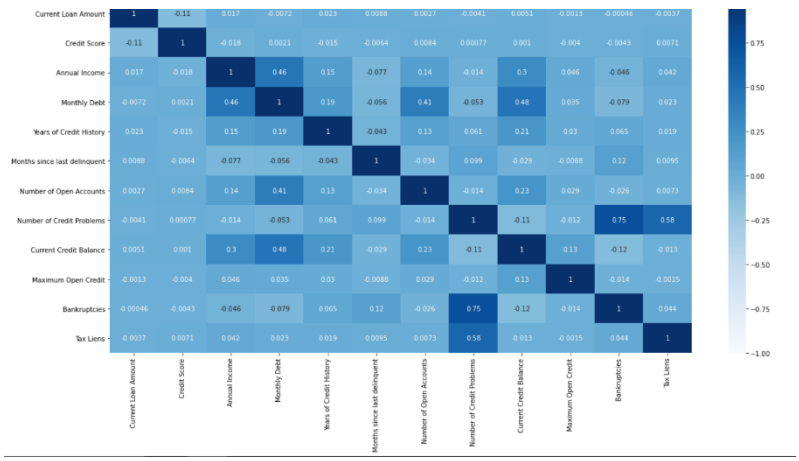
While estimating multiple regression models, quite often we obtain unsatisfactory results. This happens due to high variances and hence high values of standard errors of the estimated coefficients. This is possible when there is little variation in explanatory variables or high inter-correlations among the explanatory variables or both.
Multicollinearity represents a situation where the explanatory variables of the multiple regression model get highly correlated.

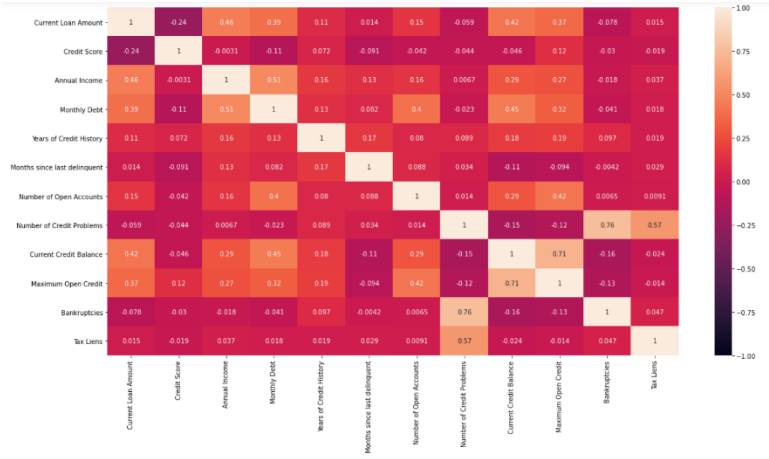
```
vif=pd.DataFrame()
vif['VIF_values']=[variance_inflation_factor(x.values,i) for i in range(x.shape[1])]
vif['features']=x.columns
vif.sort_values(by='VIF_values',ascending=False)
```

	VIF_values	features
1	7.746880	Annual Income
4	7.253759	Maximum Open Credit
3	7.237567	Current Credit Balance
2	6.702247	Monthly Debt
0	5.652784	Current Loan Amount

- VIF = 1, no correlation
- VIF > 10, High correlation
- Moderate correlation exists.
- Therefore, multicollinearity is present but no high correlation exists. Therefore, no need to drop any of the given features.

To check relation between the variables a correlation test is necessary value must lie between -1 to +1, here we can observe that diagonally that correlation is 1 because the correlation of variable itself gives 1. Their relationship between variables can clearly be seen by a heatmap which is given below.





- These are the correlations before and after treatment of outliers.

6.5 Presence of Outliers and its treatment

Few variables before and after the outlier treatment are shown as below:

Variables before outlier treatment

Figure 1

Type casting 'Number of Credit Problems', 'Bankruptcies', 'Tax Liens' for treating the outliers only in continuous numeric data. These variables consist of discrete numeric data within a smaller range. Hence, not performing outlier treatment on these variables.

We have used Inter Quartile Range to treat the outliers as it is used in measure variability by dividing a data set into quartiles. The data is sorted to in ascending order and split into 4 equal parts. Q1,Q2,Q3 called first ,second and third quartiles are the values which separate the 4 equal parts.

*#code:Q1 = data.quantile(0.25) Q3 = data.quantile(0.75) IQR = Q3 - Q1 data = data[~((data < (Q1 - 1.5 * IQR)) |(data > (Q3 + 1.5 * IQR)))]*

Variables after outlier treatment

Figure 2

As we can observe from above, most of the outliers are removed from these variables. After the outlier treatment there are 62083 records are present.

7.Assumptions

Most of the statistical tests we perform are based on a set of assumptions. When these assumptions are violated the results of the analysis can be misleading or completely erroneous.

Typical assumptions are:

- **Normality**: Data have a normal distribution (or at least is symmetric)
- **Homogeneity of variances**: Data from multiple groups have the same variance
- **Linearity**: Data have a linear relationship
- **Independence**: Data are independent

Since data was not normally distributed parametric tests were not suitable so we have performed non parametric test

7.1 Hypothesis Testing

Hypothesis testing or significance testing is a method for testing a claim or hypothesis about a parameter in a population, using data measured in a sample. In this method, we test some hypothesis by determining the likelihood that a sample statistic could have been selected, if the hypothesis regarding the population parameter were true.

Hypothesis Test	Null Hypothesis	Alternate Hypothesis	Inferences
Hypothesis test 1 - Mean Credit score of defaulters and non defaulters (MannWhitneyU test)	The mean Credit Score of Defaulters and non-defaulters are same	The mean Credit Score of Defaulters and non-defaulters are not same	As the p-value is less than 0.05, null hypothesis is rejected and hence it can be concluded that **mean Credit score of Defaulters and non-defaulters are not the same.**
Hypothesis test 2 - Mean Credit score of Long term and Short Term loans (MannWhitneyU test)	Mean Credit Score of people having Long term loan and short term loan are same.	Mean Credit Score of people having Long term loan and short term loan are not same.	As the p-value is less than 0.05, null hypothesis is rejected and hence it can be concluded that **mean Credit score of Long term and short term loans are not the same.**
Hypothesis test 3 - Credit Score of each category of Home Ownership (Kruskal Test)	Mean Credit Score of each category of Home Ownership are same	Mean Credit Score of each category of Home Ownership are not same.	As the p-value is less than 0.05, null hypothesis is rejected and hence it can be concluded that **mean Credit score of each category of Home Ownership are not same.**
Hypothesis test 4 -Average credit score of those with Fully paid loan and charged off loan. (MannWhitneyU test)	Average credit score whose loan status is Fully paid is less than or equal to the avg credit score of Customers whose loan status is charged off.	Average credit score whose loan status is Fully paid is greater than the avg credit score of Customers whose loan status is charged off.	The p-value is less than 0.05 and hence we to reject the null hypothesis. Hence we can conclude that the bank claim was right. **The average credit score whose loan status is Fully paid is greater than the avg credit score of Customers whose loan status is charged off.**
Hypothesis test 5 -Average Annual income for people having more than 700 credit score and less than 700 credit score (MannWhitneyU test)	Average Annual income for people having more than 700 credit score is equal to Average Annual income for people having less than 700 credit score	Average Annual income for people having more than 700 credit score is not equal to Average Annual income for people having less than 700 credit score.	As the p-value is less than 0.05, null hypothesis is rejected and hence it can be concluded that **Avg Annual income for people having more than 700 credit score is not equal to Average Annual income for people having less than 700 credit score**
Hypothesis test 6 - The average Credit Score for various categories of 'years in currebt job'. (Kruskal wallis test)	The average Credit Score is same irrespective of years in currebt job.	The average Credit Score is not same irrespective of years in currebt job.	As the p-value is less than 0.05, null hypothesis is rejected and hence it can be concluded that **Avg Credit Score is not same for all the different categories of 'Years in current job'.**
Hypothesis test 7 - Mean Credit Score of Bankrupts and non-Bankrupts (MannWhitneyU test)	Mean Credit Score of Bankrupts are less than that of non-Bankrupts	Mean Credit Score of Bankrupts are more than that of non-Bankrupts	As the p-value is less than 0.05, null hypothesis is rejected and hence it can be concluded that **mean Credit score of Bankrupts are more than that of non-Bankrupts**
Hypothesis test 8 -Average credit score of people with less than 16000 Monthly debt and more than 16000 Monthly debt. (MannWhitneyU test)	Average credit score of people with less than 16000 Monthly debt is less than or equal to the Average credit score of people with more than 16000 Monthly debt.	Average credit score of people with less than 16000 Monthly debt is more than the Average credit score of people with more than 16000 Monthly debt.	As the p-value is less than 0.05, null hypothesis is rejected and hence it can be concluded that **Average Credit Score of people with less than 16000 Monthly debt is more than that of the Average credit score of people with more than 16000 Monthly debt.**

8. Feature Engineering

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data. It also refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling.

Feature engineering consists of various process -

- Feature Creation
- Transformations
- Feature Extraction
- Exploratory Data
- Benchmark

8.1 Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

After Dropping columns as a result of vif results we are left with 8 variables:
'Current Loan Amount', 'Annual Income', 'Months since last delinquent','Number of Open Accounts', 'Current Credit Balance', 'Maximum Open Credit', 'Bankruptcies', 'Tax Liens'

Out[155]:

	Current Loan Amount	Credit Score	Annual Income	Monthly Debt	Years of Credit History	Months since last delinquent	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	Maximum Open Credit	Bankruptcies	Tax Liens
0	445412.0	709.0	1167493.0	5214.74	17.2	0.0	6.0	1.0	228190.0	416746.0	1.0	0.0
1	323444.0	741.0	2231892.0	29200.53	14.9	29.0	18.0	1.0	297996.0	750090.0	0.0	0.0
2	347666.0	721.0	806949.0	8741.90	12.0	0.0	9.0	0.0	256329.0	386958.0	0.0	0.0
3	206602.0	729.0	896857.0	16367.74	17.3	0.0	6.0	0.0	215308.0	272448.0	0.0	0.0
4	217646.0	730.0	1184194.0	10855.08	19.6	10.0	13.0	1.0	122170.0	272052.0	1.0	0.0

In [156]:
```
df.select_dtypes(np.number).columns
```

Out[156]:
```
Index(['Current Loan Amount', 'Credit Score', 'Annual Income', 'Monthly Debt',
       'Years of Credit History', 'Months since last delinquent',
       'Number of Open Accounts', 'Number of Credit Problems',
       'Current Credit Balance', 'Maximum Open Credit', 'Bankruptcies',
       'Tax Liens'],
      dtype='object')
```

In [157]:
```
df.describe()
```

Out[157]:

	Current Loan Amount	Credit Score	Annual Income	Monthly Debt	Years of Credit History	Months since last delinquent	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	Maximum Open Credit
count	64967.000000	64967.000000	6.496700e+04	64967.000000	64967.000000	64967.000000	64967.000000	64967.000000	6.496700e+04	6.496700e+04
mean	312062.913756	720.225607	1.376770e+06	18354.614787	18.294563	16.090354	11.125618	0.162913	2.937442e+05	7.641911e+05
std	169649.805202	27.790749	1.121549e+06	12181.904350	7.042457	22.951817	4.995636	0.478933	3.779431e+05	7.609786e+06
min	11242.000000	585.000000	7.662700e+04	0.000000	3.700000	0.000000	1.000000	0.000000	0.000000e+00	0.000000e+00
25%	186362.000000	708.000000	8.473050e+05	10151.700000	13.500000	0.000000	8.000000	0.000000	1.132780e+05	2.788720e+05
50%	323444.000000	728.000000	1.169545e+06	16089.960000	17.000000	0.000000	10.000000	0.000000	2.104250e+05	4.767180e+05
75%	392579.000000	741.000000	1.649409e+06	23827.330000	21.800000	29.000000	14.000000	0.000000	3.668615e+05	7.967190e+05
max	789250.000000	751.000000	1.655574e+08	435843.280000	70.500000	176.000000	76.000000	15.000000	3.287897e+07	1.539738e+09

Out of all 12 numerical columns there are 5 variable which are discrete is nature excluding the target variable. 'Months since last delinquent', 'Number of Open Accounts', 'Number of Credit Problems', 'Bankruptcies' and 'Tax Liens' these features are discrete in nature

For the above considered columns which are numerical, we have performed scaling so as to make all the features under the same range of values.

We have chosen Min – Max Scaler for scaling. The following image shows the scaled values of the chosen features.

```
min_max_scaler = MinMaxScaler()
df_num = pd.DataFrame(min_max_scaler.fit_transform(df_num), columns = df_num.columns)
data_num = pd.DataFrame(min_max_scaler.fit_transform(data_num), columns = data_num.columns)
#without outlier treatment
df_num
```

	Current Loan Amount	Annual Income	Monthly Debt	Years of Credit History	Months since last delinquent	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	Maximum Open Credit	Bankruptcies	Tax Liens
0	0.558053	0.006592	0.011965	0.202096	0.000000	0.066667	0.066667	0.006940	0.000271	0.142857	0.0
1	0.401284	0.013024	0.066998	0.167665	0.164773	0.226667	0.066667	0.009063	0.000487	0.000000	0.0
2	0.432417	0.004413	0.020057	0.124251	0.000000	0.106667	0.000000	0.007796	0.000251	0.000000	0.0
3	0.251103	0.004957	0.037554	0.203593	0.000000	0.066667	0.000000	0.006549	0.000177	0.000000	0.0
4	0.265298	0.006693	0.024906	0.238024	0.056818	0.160000	0.066667	0.003716	0.000177	0.142857	0.0
...	...	...	...	...	...	...	...	...	...	...	...
64962	0.401284	0.006728	0.027464	0.245509	0.090909	0.106667	0.000000	0.001137	0.000087	0.000000	0.0
64963	0.401284	0.007329	0.030077	0.085329	0.119318	0.280000	0.000000	0.004655	0.000331	0.000000	0.0
64964	0.118114	0.006490	0.016785	0.226048	0.102273	0.146667	0.066667	0.003332	0.000349	0.142857	0.0
64965	0.667204	0.009916	0.022692	0.169162	0.000000	0.093333	0.000000	0.012294	0.000479	0.000000	0.0
64966	0.401284	0.005188	0.020921	0.139222	0.000000	0.040000	0.066667	0.001387	0.000059	0.142857	0.0

```
#with outlier treatment
data_num
```

	Current Loan Amount	Annual Income	Monthly Debt	Years of Credit History	Months since last delinquent	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	Maximum Open Credit	Bankruptcies	Tax Liens
0	0.628643	0.412703	0.126784	0.202096	0.000000	0.066667	0.066667	0.343496	0.319072	0.142857	0.0
1	0.452043	0.815393	0.709943	0.167665	0.164773	0.226667	0.066667	0.448576	0.574290	0.000000	0.0
2	0.487115	0.276300	0.212539	0.124251	0.000000	0.106667	0.000000	0.385854	0.296266	0.000000	0.0
3	0.282866	0.310314	0.397943	0.203593	0.000000	0.066667	0.000000	0.324105	0.208594	0.000000	0.0
4	0.298856	0.419021	0.263916	0.238024	0.056818	0.160000	0.066667	0.183903	0.208291	0.142857	0.0
...	...	...	...	...	...	...	...	...	...	...	...
53391	0.452043	0.421235	0.291018	0.245509	0.090909	0.106667	0.000000	0.056286	0.102933	0.000000	0.0
53392	0.452043	0.458830	0.318716	0.085329	0.119318	0.280000	0.000000	0.230380	0.389884	0.000000	0.0
53393	0.133055	0.406291	0.177861	0.226048	0.102273	0.146667	0.066667	0.164912	0.411562	0.142857	0.0
53394	0.751601	0.620794	0.240454	0.169162	0.000000	0.093333	0.000000	0.608483	0.565228	0.000000	0.0
53395	0.452043	0.324813	0.221685	0.139222	0.000000	0.040000	0.066667	0.068642	0.069683	0.142857	0.0

53396 rows × 11 columns

9.Encoding of Categorical Features

Feature encoding is the process of turning categorical data in a dataset into numerical data. It is essential that we perform feature encoding because most machine learning models can only interpret numerical data and not data in text form.

Dummy encoding for categorical columns

```
df_cat = df.select_dtypes('O')
data_cat = data.select_dtypes('O')
```

```
df_cat.head()
```

52]:

	Loan Status	Term	Years in current job	Home Ownership	Purpose
0	Fully Paid	Short Term	8 years	Home Mortgage	Home Improvements
1	Fully Paid	Short Term	8 years	Own Home	Debt Consolidation
2	Fully Paid	Long Term	3 years	Own Home	Debt Consolidation
3	Charged Off	Short Term	10+ years	Home Mortgage	Debt Consolidation
4	Fully Paid	Short Term	< 1 year	Home Mortgage	Debt Consolidation

```
df_cat = pd.get_dummies(df_cat, drop_first = True)
```

```
data_cat = pd.get_dummies(data_cat, drop_first = True)
```

```
df_cat.shape
```

55]: (64967, 29)

```
data_cat.shape
```

56]: (53396, 29)

For encoding, we chose Dummy encoding for all the categorical features. The number of categorical features has increased to 29. After encoding the categorical feature, we need to concatenate both categorical as well as scaled numerical features.

```
#without outlier treatment
df_final = pd.concat([df_num, df_cat], axis=1)
#with outlier treatment
data_final = pd.concat([data_num, data_cat], axis=1)
```

```
print(df_final.shape,data_final.shape)
```

```
(64967, 40) (53396, 40)
```

```
data_final.isnull().sum()
```

```
Years of Credit History              0
Months since last delinquent         0
Number of Open Accounts              0
Number of Credit Problems            0
Current Credit Balance               0
Maximum Open Credit                  0
Bankruptcies                         0
Tax Liens                            0
Loan Status_Fully Paid               0
Term_Short Term                      0
Years in current job_10+ years       0
Years in current job_2 years         0
Years in current job_3 years         0
```

After concatenating the total number of features have increased to 40. We have also concatenated the discrete numerical features as well.

8.Train Test Split

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets.

```
#without outlier treated data
#X = df_final.drop('Credit Score',axis=1)
X = df_final
y = df['Credit Score']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=0)
print('The shape of X_train is:',X_train.shape)
print('The shape of X_test is:',X_test.shape)
print('The shape of y_train is:',y_train.shape)
print('The shape of y_test is:',y_test.shape)
```

```
The shape of X_train is: (45476, 40)
The shape of X_test is: (19491, 40)
The shape of y_train is: (45476,)
The shape of y_test is: (19491,)
```

```
#with outlier treatment
X_out = data_final
y_out = data['Credit Score']
X_train_out, X_test_out, y_train_out, y_test_out = train_test_split(X_out, y_out, test_size = 0.3, random_state=0)
print('The shape of X_train is:',X_train_out.shape)
print('The shape of X_test is:',X_test_out.shape)
print('The shape of y_train is:',y_train_out.shape)
print('The shape of y_test is:',y_test_out.shape)
```

```
The shape of X_train is: (37377, 40)
The shape of X_test is: (16019, 40)
The shape of y_train is: (37377,)
The shape of y_test is: (16019,)
```

9. OLS Model

For model building, we chose a OLS model. As the target variable i.e., Credit Score does not follow normality, the basic assumptions of a linear regression are violated, we cannot proceed with a linear regression model.

For outlier treated data

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Credit Score   R-squared:                       0.346
Model:                            OLS   Adj. R-squared:                  0.345
Method:                 Least Squares   F-statistic:                     493.7
Date:                Thu, 17 Mar 2022   Prob (F-statistic):               0.00
Time:                        01:06:47   Log-Likelihood:            -1.6856e+05
No. Observations:               37377   AIC:                         3.372e+05
Df Residuals:                   37336   BIC:                         3.376e+05
Df Model:                          40
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                          680.1559      1.180    576.583      0.000     677.844     682.468
Current Loan Amount             -9.6270      0.707    -13.613      0.000     -11.013      -8.241
Annual Income                   12.8087      0.783     16.350      0.000      11.273      14.344
Monthly Debt                    -5.9760      0.747     -8.005      0.000      -7.439      -4.513
Years of Credit History         20.3961      1.216     16.777      0.000      18.013      22.779
Months since last delinquent    -9.5590      0.880    -10.859      0.000     -11.284      -7.834
Number of Open Accounts        -40.8382      2.211    -18.472      0.000     -45.172     -36.505
Number of Credit Problems      -72.7154      9.575     -7.595      0.000     -91.482     -53.949
Current Credit Balance         -25.3764      0.801    -31.671      0.000     -26.947     -23.806
Maximum Open Credit             41.4192      0.810     51.127      0.000      39.831      43.007
Bankruptcies                     9.4355      4.958      1.903      0.057      -0.282      19.153
Tax Liens                       43.4368     11.837      3.670      0.000      20.236      66.638
Loan Status_Fully Paid           5.4535      0.263     20.707      0.000       4.937       5.970
Term_Short Term                 29.0566      0.297     97.947      0.000      28.475      29.638
Years in current job_10+ years  -2.0423      0.500     -4.081      0.000      -3.023      -1.062
Years in current job_2 years    -0.5935      0.579     -1.025      0.305      -1.728       0.541
Years in current job_3 years    -0.5662      0.595     -0.952      0.341      -1.731       0.599
Years in current job_4 years    -0.3917      0.632     -0.620      0.535      -1.630       0.846
Years in current job_5 years    -0.7095      0.618     -1.148      0.251      -1.921       0.502
Years in current job_6 years    -1.2144      0.643     -1.890      0.059      -2.474       0.045
Years in current job_7 years    -1.3122      0.653     -2.010      0.044      -2.592      -0.032
Years in current job_8 years    -1.4221      0.696     -2.045      0.041      -2.785      -0.059
Years in current job_9 years    -1.8717      0.735     -2.545      0.011      -3.313      -0.430
Years in current job_< 1 year   -0.2973      0.592     -0.502      0.616      -1.458       0.864
Years in current job_Unavailable -5.0318     0.706     -7.129      0.000      -6.415      -3.648
Home Ownership_Own Home         -4.1166      0.418     -9.847      0.000      -4.936      -3.297
Home Ownership_Rent             -3.7517      0.257    -14.598      0.000      -4.255      -3.248
Purpose_Buy House                2.1465      1.689      1.271      0.204      -1.163       5.456
Purpose_Buy a Car               17.0007      1.376     12.357      0.000      14.304      19.697
Purpose_Debt Consolidation      15.7211      0.970     16.213      0.000      13.821      17.622
Purpose_Educational Expenses    14.2681      3.699      3.857      0.000       7.018      21.518
Purpose_Home Improvements       12.9952      1.081     12.020      0.000      10.876      15.114
Purpose_Medical Bills            1.6453      1.426      1.154      0.248      -1.149       4.440
Purpose_Other                    4.4575      1.030      4.327      0.000       2.438       6.477
Purpose_Take a Trip              3.9618      1.770      2.238      0.025       0.492       7.431
Purpose_major_purchase          14.2329      2.136      6.663      0.000      10.046      18.420
Purpose_moving                  -2.1452      3.147     -0.682      0.495      -8.314       4.023
Purpose_renewable_energy         8.4246     11.049      0.762      0.446     -13.231      30.080
Purpose_small_business          -5.3238      2.442     -2.180      0.029     -10.111      -0.537
Purpose_vacation                 7.4741      4.071      1.836      0.066      -0.505      15.454
Purpose_wedding                 10.3616      3.570      2.903      0.004       3.365      17.358
==============================================================================
Omnibus:                     7891.273   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            18154.205
Skew:                          -1.199   Prob(JB):                         0.00
Kurtosis:                       5.431   Cond. No.                         266.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

For data without outlier treated data

OLS Regression Results

Dep. Variable:	Credit Score	R-squared:	0.339
Model:	OLS	Adj. R-squared:	0.338
Method:	Least Squares	F-statistic:	517.5
Date:	Thu, 17 Mar 2022	Prob (F-statistic):	0.00
Time:	01:06:47	Log-Likelihood:	-1.6876e+05
No. Observations:	37377	AIC:	3.376e+05
Df Residuals:	37339	BIC:	3.379e+05
Df Model:	37		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	683.0709	1.173	582.387	0.000	680.772	685.370
Current Loan Amount	-10.0376	0.709	-14.157	0.000	-11.427	-8.648
Annual Income	11.2795	0.721	15.648	0.000	9.867	12.692
Months since last delinquent	-8.2484	0.878	-9.398	0.000	-9.969	-6.528
Number of Open Accounts	-48.1220	2.083	-23.100	0.000	-52.205	-44.039
Current Credit Balance	-26.2834	0.770	-34.126	0.000	-27.793	-24.774
Maximum Open Credit	43.7030	0.802	54.524	0.000	42.132	45.274
Bankruptcies	-20.1671	2.237	-9.016	0.000	-24.551	-15.783
Tax Liens	-30.0136	6.673	-4.498	0.000	-43.093	-16.934
Loan Status_Fully Paid	5.6342	0.264	21.309	0.000	5.116	6.152
Term_Short Term	29.2259	0.298	98.080	0.000	28.642	29.810
Years in current job_10+ years	-1.4587	0.500	-2.915	0.004	-2.439	-0.478
Years in current job_2 years	-0.6235	0.582	-1.072	0.284	-1.764	0.517
Years in current job_3 years	-0.6898	0.598	-1.154	0.248	-1.861	0.482
Years in current job_4 years	-0.7038	0.635	-1.109	0.267	-1.948	0.540
Years in current job_5 years	-0.8461	0.621	-1.362	0.173	-2.063	0.371
Years in current job_6 years	-1.2396	0.646	-1.920	0.055	-2.505	0.026
Years in current job_7 years	-1.3180	0.656	-2.009	0.045	-2.604	-0.032
Years in current job_8 years	-1.3759	0.699	-1.968	0.049	-2.746	-0.006
Years in current job_9 years	-1.7974	0.739	-2.433	0.015	-3.246	-0.349
Years in current job_< 1 year	-0.3070	0.595	-0.516	0.606	-1.474	0.860
Years in current job_Unavailable	-3.2800	0.701	-4.681	0.000	-4.653	-1.907
Home Ownership_Own Home	-4.0944	0.420	-9.743	0.000	-4.918	-3.271
Home Ownership_Rent	-4.0795	0.257	-15.860	0.000	-4.584	-3.575
Purpose_Buy House	1.9780	1.697	1.165	0.244	-1.349	5.305
Purpose_Buy a Car	16.8754	1.383	12.202	0.000	14.165	19.586
Purpose_Debt Consolidation	15.2866	0.974	15.694	0.000	13.377	17.196
Purpose_Educational Expenses	15.2977	3.718	4.115	0.000	8.011	22.585
Purpose_Home Improvements	12.7211	1.087	11.706	0.000	10.591	14.851
Purpose_Medical Bills	1.4055	1.432	0.981	0.326	-1.402	4.213
Purpose_Other	4.1547	1.035	4.014	0.000	2.126	6.184
Purpose_Take a Trip	3.4049	1.779	1.914	0.056	-0.082	6.892
Purpose_major_purchase	14.1524	2.147	6.590	0.000	9.943	18.361
Purpose_moving	-2.5129	3.164	-0.794	0.427	-8.714	3.688
Purpose_renewable_energy	10.3620	11.106	0.933	0.351	-11.407	32.131
Purpose_small_business	-5.3887	2.455	-2.195	0.028	-10.201	-0.577
Purpose_vacation	6.7920	4.092	1.660	0.097	-1.229	14.813
Purpose_wedding	10.2205	3.588	2.848	0.004	3.187	17.254

Omnibus:	7861.284	Durbin-Watson:	2.004
Prob(Omnibus):	0.000	Jarque-Bera (JB):	18040.464
Skew:	-1.195	Prob(JB):	0.00
Kurtosis:	5.422	Cond. No.	189.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- Since Durbin-Watson value is nearly 2 i.e; 2.004 almost no autocorrelation.
- Jarque-Bera (JB) value is very high so this shows data is not normal.
- The model shows 33.9% Accuracy. Hence the model is poor and it does not follow the assumptions of linearity.
- Since the condition number is very high. This shows the presence of multicollinearity.
- Explained variation is around 33.9%. Hence the model is Moderate.
- Explained variation is around 34.6% for outlier treated data. Hence the model is Moderate.

From the basic OLS model, we can infer that the model has almost no autocorrelation and presence of high multicollinearity which are the basic assumptions of a linear regression model. Therefore, we will be choosing other options for model building.

10. Decision Tree Regressor

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches, each representing values for the attribute tested. Leaf node represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

For data without outlier

```
#Outlier treated data
x=X_train_out
y=y_train_out

dtr=DecisionTreeRegressor(random_state=0)
dtr.fit(x,y)
```

```
]: DecisionTreeRegressor(random_state=0)
```

```
dtr.get_params()
```

```
]: {'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': 0,
 'splitter': 'best'}
```

```
dtr.score(x,y)
```

```
]: 1.0
```

```
dtr.score(X_test_out,y_test_out)
```

```
]: -0.3182089947774236
```

```
imp_feature = pd.DataFrame({'feature': X_train_out.columns,
                            'Importance': dtr.feature_importances_})
imp_feature.sort_values('Importance', ascending=False)
```

]:

	feature	Importance
12	Term_Short Term	2.289978e-01
8	Maximum Open Credit	1.184758e-01
7	Current Credit Balance	1.026344e-01
0	Current Loan Amount	8.909215e-02
1	Annual Income	8.734199e-02
3	Years of Credit History	7.774271e-02
2	Monthly Debt	7.264824e-02
5	Number of Open Accounts	5.107976e-02
4	Months since last delinquent	4.035256e-02
11	Loan Status_Fully Paid	1.474645e-02
28	Purpose_Debt Consolidation	1.173410e-02
25	Home Ownership_Rent	9.921115e-03
13	Years in current job_10+ years	9.074642e-03
6	Number of Credit Problems	8.921666e-03
32	Purpose_Other	7.147163e-03
9	Bankruptcies	6.490886e-03
24	Home Ownership_Own Home	6.060265e-03
22	Years in current job_< 1 year	5.666315e-03
15	Years in current job_3 years	5.452741e-03
19	Years in current job_7 years	4.737878e-03
14	Years in current job_2 years	4.570995e-03
17	Years in current job_5 years	4.437513e-03
16	Years in current job_4 years	4.333718e-03
21	Years in current job_9 years	4.259530e-03
23	Years in current job_Unavailable	3.985394e-03
18	Years in current job_6 years	3.796199e-03
30	Purpose_Home Improvements	3.518868e-03
20	Years in current job_8 years	3.143481e-03
26	Purpose_Buy House	2.457288e-03
37	Purpose_small_business	2.262798e-03
10	Tax Liens	1.434155e-03
31	Purpose_Medical Bills	1.185533e-03
27	Purpose_Buy a Car	9.170386e-04
33	Purpose_Take a Trip	5.065961e-04
35	Purpose_moving	4.646094e-04
29	Purpose_Educational Expenses	2.339011e-04
38	Purpose_vacation	7.106488e-05
39	Purpose_wedding	5.761770e-05
34	Purpose_major_purchase	4.479754e-05
36	Purpose_renewable_energy	2.953808e-07

```
y_outlier_pred=pd.DataFrame(dtr.predict(X_test_out), columns=['Credit Score'])
```

```
act = y_test_out
act=act.reset_index(drop=True)
```

```
pd.concat([act,y_outlier_pred],axis=1)
```

]:

	Credit Score	Credit Score
0	715.0	745.0
1	689.0	720.0
2	741.0	742.0
3	742.0	739.0
4	738.0	740.0
...	...	...
16014	741.0	716.0
16015	682.0	744.0
16016	748.0	748.0
16017	744.0	736.0
16018	726.0	742.0

16019 rows × 2 columns

```
from sklearn.feature_selection import RFE
x=X_train_out
y=y_train_out
rfe=RFE(estimator=DecisionTreeRegressor(),n_features_to_select=10)
rfe.fit(x,y)
f_index=pd.Series(rfe.ranking_,index=x.columns)
f_index[f_index==1].index
```

]:
```
Index(['Current Loan Amount', 'Annual Income', 'Monthly Debt',
       'Years of Credit History', 'Months since last delinquent',
       'Number of Open Accounts', 'Current Credit Balance',
       'Maximum Open Credit', 'Term_Short Term', 'Purpose_Debt Consolidation'],
      dtype='object')
```

Grid Search

Grid-searching is the process of scanning the data to configure optimal parameters for a given model. Depending on the type of model utilized, certain parameters are necessary. Grid-searching does NOT only apply to one model type. Grid-searching can be applied across machine learning to calculate the best parameters to use for any given model. It is important to note that Grid-searching can be extremely computationally expensive and may take your machine quite a long time to run. Grid-Search will build a model on each parameter combination possible. It iterates through every parameter combination and stores a model for each combination.

```
from sklearn.model_selection import GridSearchCV
parameters={"splitter":["best","random"],
            "max_depth" : [1,3,5,7,9,11,12],
           "min_samples_leaf":[1,2,3,4,5,6,7,8,9,10],
           "min_weight_fraction_leaf":[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
           "max_features":["auto","log2","sqrt"],
           "max_leaf_nodes":[10,20,30,40,50,60,70,80,90]}

dtr = DecisionTreeRegressor(random_state = 42)

tree_grid = GridSearchCV(estimator = dtr,param_grid=parameters,scoring='neg_mean_squared_error',cv=5,verbose=3)

tree_grid_model = tree_grid.fit(x, y)
print('Best parameters for decision tree classifier: ', tree_grid_model.best_params_, '\n')
```

- Results from grid search cv
- Best parameters for decision tree classifier: {'max_depth': 15, 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_leaf': 1, 'min_weight_fraction_leaf': 0.1, 'splitter': 'best'}

```
rf_model = DecisionTreeRegressor(max_depth= 5, max_features= 'auto', max_leaf_nodes= None,
                                 min_samples_leaf= 1, min_weight_fraction_leaf= 0.1, splitter= 'best')

x=X_train_out[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Annual Income', 'Months since last delinquent']]
y=y_train_out
# use fit() to fit the model on the train set
rf_model = rf_model.fit(x, y)

print('score on train',rf_model.score(x,y))
```

```
score on train 0.2646585206315696
```

```
x=X_test_out[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Annual Income', 'Months since last delinquent']]
y=y_test_out
print('score on test',rf_model.score(x,y))
```

```
score on test 0.255860184637544
```

So, for the data without outliers
Train score: 0.2646
Test score: 0.2558

For the data with outliers

```
# without outlier removal
x=X_train
y=y_train
dtr=DecisionTreeRegressor(random_state=0)
dtr.fit(x,y)
```

```
DecisionTreeRegressor(random_state=0)
```

```
dtr.get_params()
```

```
{'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': 0,
 'splitter': 'best'}
```

```
dtr.score(x,y)
```

```
1.0
```

```
dtr.score(X_test,y_test)
```

```
-0.25702864725043484
```

```
y_pred=pd.DataFrame(dtr.predict(X_test),columns=['Credit Score'])
```

```
y_pred.head()
```

	Credit Score
0	742.0
1	720.0
2	684.0
3	746.0
4	746.0

```
act = pd.DataFrame(y_test, columns=['Credit Score'])
act.reset_index(drop=True, inplace=True)
```

```
pd.concat([act,y_pred],axis=1)
```

	Credit Score	Credit Score
0	745.0	742.0
1	744.0	720.0
2	745.0	684.0
3	739.0	746.0
4	671.0	746.0
...	...	...
19486	733.0	704.0
19487	748.0	647.0
19488	724.0	734.0
19489	730.0	725.0
19490	711.0	721.0

19491 rows × 2 columns

```
rf_model = DecisionTreeRegressor(max_depth = 5,
                                 min_samples_split = 350,
                                 random_state = 10)

# use fit() to fit the model on the train set
rf_model = rf_model.fit(x, y)

print('score on train',rf_model.score(x,y))
```

```
score on train 0.3168153275396989
```

```
x=X_test[['Current Loan Amount', 'Annual Income', 'Monthly Debt',
       'Years of Credit History', 'Current Credit Balance',
       'Maximum Open Credit', 'Loan Status_Fully Paid', 'Term_Short Term',
       'Years in current job_10+ years', 'Years in current job_2 years',
       'Years in current job_3 years', 'Years in current job_4 years',
       'Years in current job_5 years', 'Years in current job_< 1 year',
       'Home Ownership_Own Home', 'Home Ownership_Rent',
       'Purpose_Debt Consolidation', 'Purpose_Other',
       'Months since last delinquent', 'Bankruptcies']]
y=y_test
print('score on test',rf_model.score(x,y))
```

```
score on test 0.2952608851980173
```

So, for the data with outliers
Train score: 0.31681
Test score: 0.29526

11. Random Forest Regressor

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. The averaging makes a Random Forest better than a single Decision Tree hence improves its accuracy and reduces overfitting. A prediction from the Random Forest Regressor is an average of the predictions produced by the trees in the forest.

For data with outlier treatment

```
#Outlier treated data

x=X_train_out
y=y_train_out

rfr=RandomForestRegressor(random_state=0)
rfr.fit(x,y)
```

```
RandomForestRegressor(random_state=0)
```

```
rfr.get_params()
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 0,
 'verbose': 0,
 'warm_start': False}
```

```
rfr.score(x,y)
```

```
0.9104168025089541
```

```
rfr.score(X_test_out,y_test_out)
```

```
0.3582537503281077
```

```
y_pred_out=pd.DataFrame(rfr.predict(X_test_out),columns=['Credit Score'])
```

```
y_pred_out.head()
```

	Credit Score
0	729.30
1	704.39
2	731.22
3	722.45
4	733.01

```
y_test.shape
```

```
(19491,)
```

```
act_out = pd.DataFrame(y_test_out, columns=['Credit Score'])
act_out.reset_index(drop=True, inplace=True)
```

```
pd.concat([act_out,y_pred_out],axis=1)
```

	Credit Score	Credit Score
0	715.0	729.30
1	689.0	704.39
2	741.0	731.22
3	742.0	722.45
4	738.0	733.01
...	...	...
16014	741.0	729.31
16015	682.0	730.85
16016	748.0	735.64
16017	744.0	712.58
16018	726.0	728.74

16019 rows × 2 columns

These 16 features are selected by extra tree regressor

'Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance', 'Current Loan Amount', 'Years of Credit History', 'Number of Open Accounts', 'Annual Income', 'Monthly Debt', 'Months since last delinquent', 'Years in current job_10+ years', 'Loan Status_Fully Paid', 'Home Ownership_Rent', 'Years in current job_2 years', 'Years in current job_< 1 year', 'Number of Credit Problems', 'Years in current job_3 years'

```
from sklearn.model_selection import GridSearchCV
tuned_paramaters = [{'max_depth': [3,5,7,9],
                     'min_samples_split': [100,150,200,250,300,350], 'max_features':['auto', 'sqrt', 'log2', 'int', 'float'],
                    'n_estimators':[100,150,200,250,300,350], 'min_samples_leaf':[100,150,200,250,300,350]}]

rfr = RandomForestRegressor(random_state = 10)

tree_grid = GridSearchCV(estimator = rfr,
                         param_grid = tuned_paramaters,
                         cv = 5)

tree_grid_model = tree_grid.fit(x, y)
print('Best parameters for decision tree classifier: ', tree_grid_model.best_params_, '\n')
```

- Results from grid search cv
- Best parameters forrandom forest regressor: {'max_depth': 40, 'max_features': 'auto', 'max_leaf_nodes': 25, 'min_samples_split': 20, 'min_weight_fraction_leaf': 0.1, 'splitter': 'best'}

```
rf_model = RandomForestRegressor(max_depth =40, max_features='auto',
                                 min_samples_split = 20,min_samples_leaf=1,max_leaf_nodes=25,
                                 random_state = 5)

x=X_train_out[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Annual Income', 'Monthly Debt',
       'Months since last delinquent', 'Years in current job_10+ years',
       'Loan Status_Fully Paid', 'Home Ownership_Rent',
       'Years in current job_2 years', 'Years in current job_< 1 year',
       'Number of Credit Problems', 'Years in current job_3 years']]
y=y_train_out
# use fit() to fit the model on the train set
rf_model = rf_model.fit(x, y)

print('score on train',rf_model.score(x,y))
```

```
score on train 0.3161856668596953
```

```
x=X_test_out[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Annual Income', 'Monthly Debt',
       'Months since last delinquent', 'Years in current job_10+ years',
       'Loan Status_Fully Paid', 'Home Ownership_Rent',
       'Years in current job_2 years', 'Years in current job_< 1 year',
       'Number of Credit Problems', 'Years in current job_3 years']]
y=y_test_out
print('score on test',rf_model.score(x,y))
```

```
score on test 0.3036631217038531
```

```
#outlier treated data
rf_model = RandomForestRegressor(max_depth = 9,
                                 min_samples_split = 100,
                                 random_state = 10)
rf_model = rf_model.fit(X_train_out, y_train_out)

print('score on train',rf_model.score(X_train_out, y_train_out))
print('score on test',rf_model.score(X_test_out, y_test_out))
```

```
score on train 0.4089568193356429
score on test 0.3456285181631692
```

So, for the data without outliers
Train score: 0.40895
Test score: 0.34562

For the data with outliers

```
rf_model = RandomForestRegressor(max_depth = 9,
                                 min_samples_split = 100,
                                 random_state = 10)

# use fit() to fit the model on the train set
rf_model = rf_model.fit(x, y)

print('score on train',rf_model.score(x, y))
```

```
score on train 0.4127841668840353
```

```
x=X_test[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Annual Income', 'Monthly Debt',
       'Months since last delinquent', 'Years in current job_10+ years',
       'Loan Status_Fully Paid', 'Home Ownership_Rent',
       'Years in current job_2 years', 'Years in current job_< 1 year',
       'Number of Credit Problems', 'Years in current job_3 years']]
y=y_test
print('score on test',rf_model.score(x, y))
```

```
score on test 0.35068507619612854
```

```
rf_model = RandomForestRegressor(max_depth = 9,
                                 min_samples_split = 100,
                                 random_state = 10)
rf_model = rf_model.fit(X_train, y_train)

print('score on train',rf_model.score(X_train, y_train))
print('score on test',rf_model.score(X_test, y_test))
```

```
score on train 0.4239110755717542
score on test 0.35952060970231403
```

So, for the data with outliers
Train score: 0.42391
Test score: 0.35952

12. Boosting

Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. In boosting, a random sample of data is selected, fitted with a model and then trained sequentially—that is, each model tries to compensate for the weaknesses of its predecessor. Boosting grants power to machine learning models to improve their accuracy of prediction. Boosting algorithms are one of the most widely used algorithm in data science competitions. The winners of our last hackathons agree that they try boosting algorithm to improve accuracy of their models.

```
from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor
from xgboost import XGBRegressor
```

```
print(df_final.shape,data_final.shape)
```

```
(64967, 40) (53396, 40)
```

```
X = df_final
y = df['Credit Score']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=0)
print('The shape of X_train is:',X_train.shape)
print('The shape of X_test is:',X_test.shape)
print('The shape of y_train is:',y_train.shape)
print('The shape of y_test is:',y_test.shape)
```

```
The shape of X_train is: (45476, 40)
The shape of X_test is: (19491, 40)
The shape of y_train is: (45476,)
The shape of y_test is: (19491,)
```

13. AdaBoost

AdaBoost, short for Adaptive Boosting, is a statistical classification meta-algorithm. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms.

For data with outliers

```
ada=AdaBoostRegressor()
search_grid={'n_estimators':range(10,100,10),'learning_rate':[.001,0.01,.1,1],'random_state':[1]}
search=GridSearchCV(estimator=ada,param_grid=search_grid,n_jobs=2,cv=5)

#n_jobsint=Number of jobs to run in parallel. None means 1 unless in a joblib.
# parallel_backend context. -1 means using all processors.

search.fit(X_train, y_train)
search.best_params_
```

```
ada_model = AdaBoostRegressor(n_estimators = 10, random_state = 10,learning_rate=0.001)

ada_model.fit(X_train, y_train)
```

```
AdaBoostRegressor(learning_rate=0.001, n_estimators=10, random_state=10)
```

```
ada_model.score(X_train, y_train)
```

```
0.2891878826181943
```

```
ada_model.score(X_test, y_test)
```

```
0.27665444897436886
```

AdaBoost score for the data with outliers
Train score: 0.28918
Test score: 0.27665

For data without outliers

```
X_out = data_final
y_out = data['Credit Score']
X_train_out, X_test_out, y_train_out, y_test_out = train_test_split(X_out, y_out, test_size = 0.3, random_state=0)
print('The shape of X_train is:',X_train_out.shape)
print('The shape of X_test is:',X_test_out.shape)
print('The shape of y_train is:',y_train_out.shape)
print('The shape of y_test is:',y_test_out.shape)
```

```
The shape of X_train is: (37377, 40)
The shape of X_test is: (16019, 40)
The shape of y_train is: (37377,)
The shape of y_test is: (16019,)
```

```
ada_model_out = AdaBoostRegressor(n_estimators = 10, random_state = 10,learning_rate=0.001)
# the above values are taken from the gridsearch on the data with outliers

ada_model_out.fit(X_train_out, y_train_out)
```

```
]: AdaBoostRegressor(learning_rate=0.001, n_estimators=10, random_state=10)
```

```
print(ada_model_out.score(X_train_out, y_train_out))
print(ada_model_out.score(X_test_out, y_test_out))
```

```
0.2758853696026208
0.2691707012822854
```

AdaBoost score for the data without outliers
Train score: 0.275885
Test score: 0.269170

14. Recursive Feature Elimination (RFE)

Recursive Feature Elimination, or RFE for short, is a popular feature selection algorithm. RFE is popular because it is easy to configure and use and because it is effective at selecting those features (columns) in a training dataset that are more or most relevant in predicting the target variable.This is a popular feature selection algorithm. RFE is popular because it is easy to configure and use and because it is effective at selecting those features (columns) in a training dataset that are more or most relevant in predicting the target variable.

For the data with outliers

```
rfe_X_train=X_train[['Current Loan Amount', 'Annual Income', 'Monthly Debt',
       'Years of Credit History', 'Months since last delinquent',
       'Number of Open Accounts', 'Current Credit Balance',
       'Maximum Open Credit', 'Loan Status_Fully Paid', 'Term_Short Term',
       'Years in current job_10+ years', 'Home Ownership_Rent',
       'Purpose_Buy House', 'Purpose_Debt Consolidation',
       'Purpose_small_business']]

rfe_X_test=X_test[['Current Loan Amount', 'Annual Income', 'Monthly Debt',
       'Years of Credit History', 'Months since last delinquent',
       'Number of Open Accounts', 'Current Credit Balance',
       'Maximum Open Credit', 'Loan Status_Fully Paid', 'Term_Short Term',
       'Years in current job_10+ years', 'Home Ownership_Rent',
       'Purpose_Buy House', 'Purpose_Debt Consolidation',
       'Purpose_small_business']]
print(rfe_X_train.shape,rfe_X_test.shape)
```

```
(45476, 15) (19491, 15)
```

```
rfe_ada_model = AdaBoostRegressor(n_estimators = 10, random_state = 10,learning_rate=0.001)


rfe_ada_model.fit(rfe_X_train, y_train)
```

```
]: AdaBoostRegressor(learning_rate=0.001, n_estimators=10, random_state=10)
```

```
print(rfe_ada_model.score(rfe_X_train, y_train))
print(rfe_ada_model.score(rfe_X_test, y_test))
```

```
0.28984456665638525
0.27725426171470613
```

RFE score for the data with outliers
Train score: 0.2898
Test score: 0.2772

These are the features of RFE model with outliers

```
a=rfe_ada_model.feature_importances_
f=pd.DataFrame()
f['values']=a
f['features']=rfe_X_train.columns
f.sort_values('values',ascending=False)
```

```
]:
```

	values	features
9	0.819300	Term_Short Term
7	0.102454	Maximum Open Credit
0	0.057985	Current Loan Amount
8	0.014093	Loan Status_Fully Paid
13	0.006169	Purpose_Debt Consolidation
1	0.000000	Annual Income
2	0.000000	Monthly Debt
3	0.000000	Years of Credit History
4	0.000000	Months since last delinquent
5	0.000000	Number of Open Accounts
6	0.000000	Current Credit Balance
10	0.000000	Years in current job_10+ years
11	0.000000	Home Ownership_Rent
12	0.000000	Purpose_Buy House
14	0.000000	Purpose_small_business

For the data without outliers

```
rfe_X_train_out=X_train_out[['Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Current Credit Balance',
       'Maximum Open Credit', 'Loan Status_Fully Paid', 'Term_Short Term',
       'Purpose_Buy House', 'Purpose_Debt Consolidation',
       'Purpose_small_business']]

rfe_X_test_out=X_test_out[['Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Current Credit Balance',
       'Maximum Open Credit', 'Loan Status_Fully Paid', 'Term_Short Term',
       'Purpose_Buy House', 'Purpose_Debt Consolidation',
       'Purpose_small_business']]
print(rfe_X_train_out.shape,rfe_X_test_out.shape)
```

```
(37377, 10) (16019, 10)
```

```
rfe_ada_model_out = AdaBoostRegressor(n_estimators = 10, random_state = 10,learning_rate=0.001)

rfe_ada_model_out.fit(rfe_X_train_out, y_train_out)
```

```
]: AdaBoostRegressor(learning_rate=0.001, n_estimators=10, random_state=10)
```

```
print(rfe_ada_model_out.score(rfe_X_train_out, y_train_out))
print(rfe_ada_model_out.score(rfe_X_test_out, y_test_out))
```

```
0.2745297343501665
0.2681544034429739
```

RFE score for the data without outliers
Train score: 0.2745
Test score: 0.2681

These are the features of RFE model without outliers

```
a=rfe_ada_model_out.feature_importances_
f=pd.DataFrame()
f['values']=a
f['features']=rfe_X_train_out.columns
f.sort_values('values',ascending=False)
```

```
]:
```

	values	features
6	0.831484	Term_Short Term
4	0.103764	Maximum Open Credit
5	0.029123	Loan Status_Fully Paid
0	0.021906	Current Loan Amount
8	0.008436	Purpose_Debt Consolidation
3	0.003974	Current Credit Balance
1	0.001314	Years of Credit History
2	0.000000	Number of Open Accounts
7	0.000000	Purpose_Buy House
9	0.000000	Purpose_small_business

15. Gradient Boosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.[1][2] When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

For the data with outliers

```
gb_model = GradientBoostingRegressor(n_estimators = 10, random_state = 10,learning_rate=0.001)

# fit the model using fit() on train data
gb_model.fit(X_train, y_train)
```

```
]: GradientBoostingRegressor(learning_rate=0.001, n_estimators=10, random_state=10)
```

```
print(gb_model.score(X_train, y_train))
print(gb_model.score(X_test, y_test))
```

```
0.005674824296758141
0.005530893251812441
```

Gradient boosting score for the data with outliers
Train score: 0.00567
Test score: 0.00553

For the data without outliers

```
gb_model_out= GradientBoostingRegressor(n_estimators = 10, random_state = 10,learning_rate=0.001)


gb_model_out.fit(X_train_out, y_train_out)
```

```
]: GradientBoostingRegressor(learning_rate=0.001, n_estimators=10, random_state=10)
```

```
print(gb_model_out.score(X_train_out, y_train_out))
print(gb_model_out.score(X_test_out, y_test_out))
```

```
0.005391932543006961
0.005370273906992407
```

Gradient boosting score for the data without outliers
Train score: 0.005391
Test score: 0.005370

16. RFE XGBoosting

XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems. XGBoost minimizes a regularized objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity.

For data with outliers

These parameters were selected from grid search ,max_depth = 8, gamma = 4,learning_rate=0.1

```
xgb_model = XGBRegressor(max_depth = 8, gamma = 4,learning_rate=0.1)
xgb_model.fit(rfe_X_train_out, y_train_out)

print(xgb_model.score(rfe_X_train_out, y_train_out))
print(xgb_model.score(rfe_X_test_out, y_test_out))
```

```
0.5295232529679286
0.3410996489199516
```

```
a=xgb_model.feature_importances_
f=pd.DataFrame()
f['values']=a
f['features']=rfe_X_train_out.columns
f.sort_values('values',ascending=False)
```

	values	features
6	0.732174	Term_Short Term
8	0.052733	Purpose_Debt Consolidation
5	0.052448	Loan Status_Fully Paid
4	0.029698	Maximum Open Credit
7	0.026524	Purpose_Buy House
3	0.025904	Current Credit Balance
9	0.023417	Purpose_small_business
2	0.019379	Number of Open Accounts
0	0.019374	Current Loan Amount
1	0.018347	Years of Credit History

```
b=f['features'][:20].values
```

```
features_X_train=X_train[b]

features_X_test=X_test[b]
print(features_X_test.shape,features_X_test.shape)
```

```
(19491, 10) (19491, 10)
```

```
feautures_xgb_model = XGBRegressor(max_depth = 3, gamma = 0.2,learning_rate=0.2)


feautures_xgb_model.fit(rfe_X_train, y_train)

print(feautures_xgb_model.score(rfe_X_train, y_train))
print(feautures_xgb_model.score(rfe_X_test, y_test))
```

```
0.4267187093534538
0.39241660042071513
```

RFE XGboosting score for the data with outliers
Train score: 0.4267
Test score: 0.3924

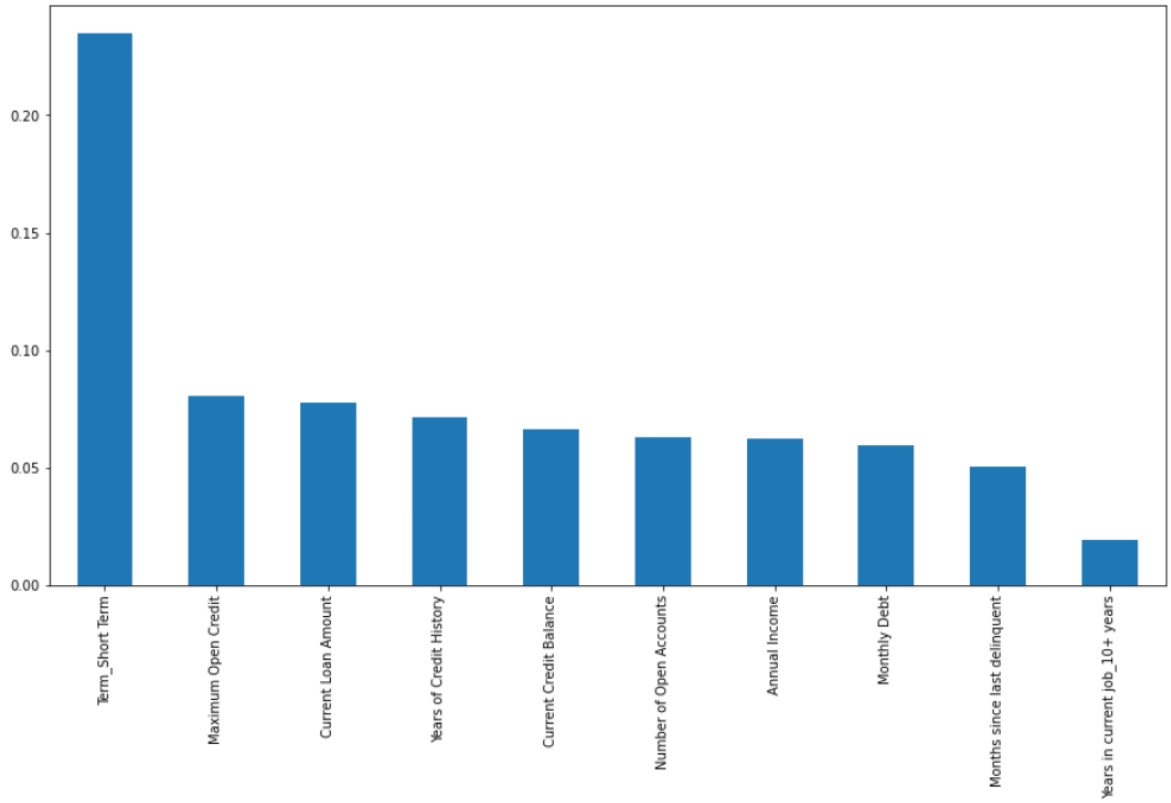


Fig: Extra tree regressor with outliers

For the data without outlier

```
xgb_model_out = XGBRegressor(max_depth = 3, gamma = 0,learning_rate=0.2)
# the above parameters used are taken from GridSearchCV on the data with outliers


xgb_model_out.fit(rfe_X_train_out, y_train_out)

print(xgb_model_out.score(rfe_X_train_out, y_train_out))
print(xgb_model_out.score(rfe_X_test_out, y_test_out))
```

```
0.41893698022164594
0.3851973053342892
```

RFE XGboosting score for the data without outliers
Train score: 0.4189
Test score: 0.3851

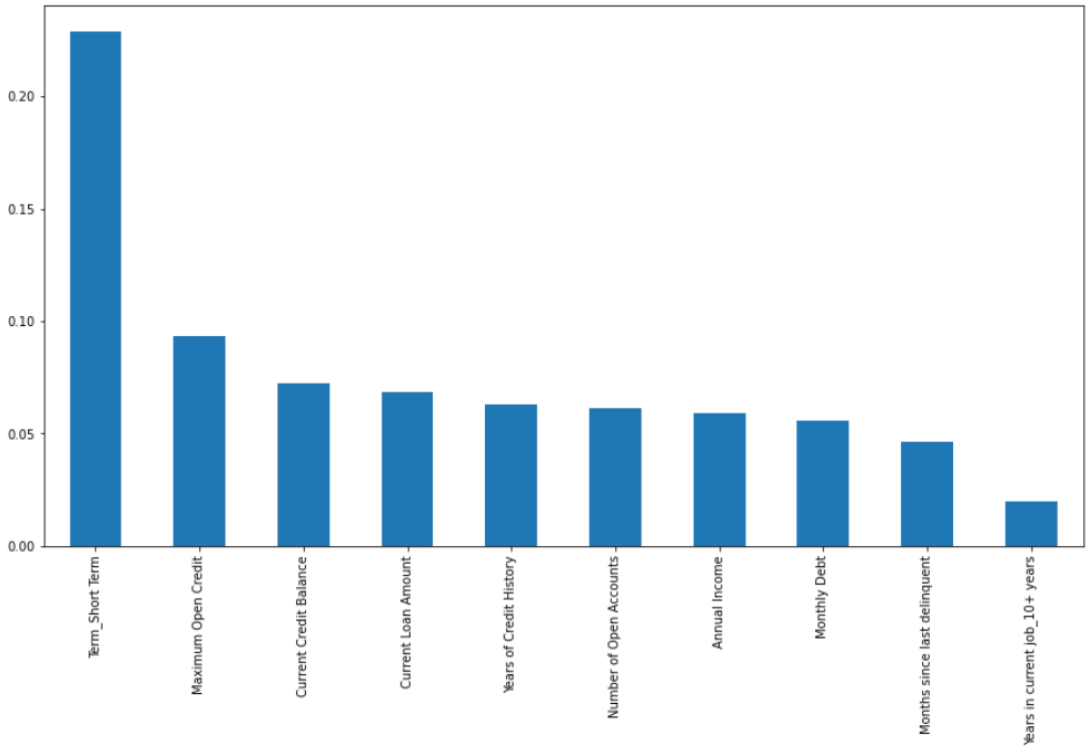


Fig: Extra tree regressor without outliers

17. Elastic Net Regressor

Elastic Net regression is a classification algorithm that overcomes the limitations of the lasso(least absolute shrinkage and selection operator) method which uses a penalty function in its L1 regularization.
Elastic Net is an extension of linear regression that adds regularization penalties to the loss function during training.

```
enet = ElasticNet(alpha = 0.00001, l1_ratio = 0.2, max_iter = 1000, random_state=42,positive=False,selection='random')

# fit the model on train data
enet.fit(rfe_X_train_out, y_train_out)
print(enet.score(rfe_X_train_out, y_train_out))
print(enet.score(rfe_X_test_out, y_test_out))
```

```
0.3289480384217378
0.31837169859132786
```

Elastic Net Regressor score for the data without outliers
Train score: 0.3289
Test score: 0.3183

18. MLP Regressor

Regression. Class MLP Regressor implements a multi-layer perceptron (MLP) that trains using backpropagation with no activation function in the output layer, which can also be seen as using the identity function as activation function. MLP Regressor trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting.

For the data with outliers

```
mlpmodel.score(X_train[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Annual Income', 'Monthly Debt',
       'Months since last delinquent', 'Years in current job_10+ years',
       'Loan Status_Fully Paid', 'Home Ownership_Rent',
       'Years in current job_2 years', 'Years in current job_< 1 year',
       'Number of Credit Problems', 'Years in current job_3 years',
       'Purpose_Debt Consolidation', 'Years in current job_6 years',
       'Years in current job_7 years', 'Years in current job_5 years']],y_train)
```

```
0.03596651044025989
```

```
mlpmodel.score(X_test[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Annual Income', 'Monthly Debt',
       'Months since last delinquent', 'Years in current job_10+ years',
       'Loan Status_Fully Paid', 'Home Ownership_Rent',
       'Years in current job_2 years', 'Years in current job_< 1 year',
       'Number of Credit Problems', 'Years in current job_3 years',
       'Purpose_Debt Consolidation', 'Years in current job_6 years',
       'Years in current job_7 years', 'Years in current job_5 years']],y_test)
```

```
-0.0006258343568805724
```

MLP Regressor score for the data with outliers
Train score: 0.0359
Test score: -0.000625

For the data without outlier

```
mlpmodel = MLPRegressor(activation='relu', solver='adam',
                        alpha=0.0001, learning_rate='constant',
                        power_t=0.5, max_iter=200,random_state=42)

mlpmodel.score(X_train_out[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Annual Income', 'Monthly Debt',
       'Months since last delinquent', 'Years in current job_10+ years',
       'Loan Status_Fully Paid', 'Home Ownership_Rent',
       'Years in current job_2 years', 'Years in current job_< 1 year',
       'Number of Credit Problems', 'Years in current job_3 years',
       'Purpose_Debt Consolidation', 'Years in current job_6 years',
       'Years in current job_7 years', 'Years in current job_5 years']],y_train_out)
```

```
0.3582691642045017
```

```
mlpmodel.score(X_test_out[['Term_Short Term', 'Maximum Open Credit', 'Current Credit Balance',
       'Current Loan Amount', 'Years of Credit History',
       'Number of Open Accounts', 'Annual Income', 'Monthly Debt',
       'Months since last delinquent', 'Years in current job_10+ years',
       'Loan Status_Fully Paid', 'Home Ownership_Rent',
       'Years in current job_2 years', 'Years in current job_< 1 year',
       'Number of Credit Problems', 'Years in current job_3 years',
       'Purpose_Debt Consolidation', 'Years in current job_6 years',
       'Years in current job_7 years', 'Years in current job_5 years']],y_test_out)
```

```
0.3504891379168237
```

MLP Regressor score for the data without outliers
Train score: 0.03582
Test score: 0.35048

19. Model Comparison

```
X_train_out1 = X_train_out.rename({'Years in current job_< 1 year':'Years in current job__ 1 year'},axis=1)
# user variables to tune
folds   = 5
metric  = "neg_mean_absolute_error"

# hold different regression models in a single dictionary
models = {}
models["Linear"]        = LinearRegression()
models["Lasso"]         = Lasso()
models["Ridge"]         = Ridge()
models["ElasticNet"]    = ElasticNet()
models["DecisionTree"]  = DecisionTreeRegressor()
models["KNN"]           = KNeighborsRegressor()
models["RandomForest"]  = RandomForestRegressor()
models["AdaBoost"]      = AdaBoostRegressor()
models["GradientBoost"] = GradientBoostingRegressor()
models["MLPRegressor"] = MLPRegressor()
# 10-fold cross validation for each model
model_results = []
model_names   = []
for model_name in models:
    model   = models[model_name]
    k_fold  = KFold(n_splits=folds, random_state=10, shuffle=True)
    results = cross_val_score(model, X_train, y_train, cv=k_fold, scoring=metric)
    results1 = cross_val_score(XGBRegressor(), X_train_out1, y_train_out, cv=k_fold, scoring=metric)
    model_results.extend([results,results1])
    model_names.append(model_name)
    print("{}: {}, {}".format(model_name, round(results.mean(), 3), round(results.std(), 3)))
results1 = cross_val_score(XGBRegressor(), X_train_out1, y_train_out, cv=k_fold, scoring=metric)
print("XGBoost: {}, {}".format(round(results1.mean(), 3), round(results1.std(), 3)))
```

```
Linear: -17.528, 0.096
Lasso: -18.553, 0.078
Ridge: -17.533, 0.105
ElasticNet: -20.004, 0.117
DecisionTree: -22.034, 0.206
KNN: -18.448, 0.146
RandomForest: -16.127, 0.141
AdaBoost: -25.904, 2.311
GradientBoost: -16.04, 0.092
MLPRegressor: -17.136, 0.217
XGBoost: -15.865, 0.188
```

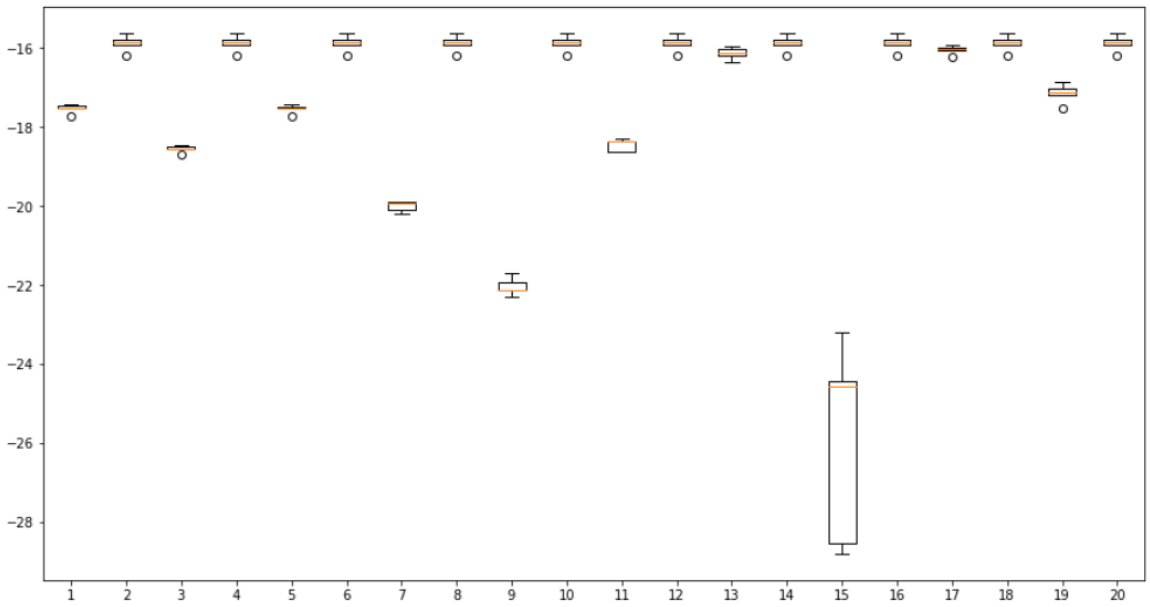


Fig: Regression models comparison

20. Implications

The normal operation and improvement of the rating system is an obvious subject of concern to banks, customers and regulators, such as the latest developments in the world economy.

A scorecard is a digital scale used to assign levels to customer characteristics to obtain a value that represents the risk that the customer is considering you have not fulfilled your financial obligations to other customers.

The scorecard development process is divided into three main stages:
(i) creating a data set;
(ii) modeling;
(iii) documentation.

This introduces various methods and methods used in the creation and modeling of data sets. The development of a scorecard is a detailed process and many aspects need to be considered. Demographic trends and economic events can cause various situations. Which standards and accepted dashboard design methods are not suitable.

One of these problems arises when the sample contains few default values, which makes it difficult to construct a reliable scorecard. After all, due to the sensitivity of privacy laws and companies, obtaining actual credit information is a challenge for many scholars. With the help of artificial data, scientists can overcome these limitations and create special conditions for studying specific problems.

21. Limitations

Although credit scoring systems are being implemented and used by most banks nowadays, they do face a number of limitations.

- A first limitation concerns the data that is used to estimate credit scoring models.
- Since data are the major, and in most cases the only, ingredient to build these models, its quality and predictive ability is key to their success.
- Data quality issues can be difficult to detect without specific domain knowledge, but have an important impact on the scorecard development and resulting risk measures.
- A properly designed credit scoring system allows creditors to evaluate thousands of applications consistently, impartially and quickly. If this is true, then the opposite must also be true. A poorly designed credit scoring system can evaluate thousands of applicants and can make the wrong recommendation every time.
- Credit risk can never be measured precisely, and any model that says it can is wrong.
- Credit managers should be able to override the credit score and its credit recommendation. However, doing so is difficult to justify if there is a serious payment problem, or worse a bad debt loss. For this reason, credit professionals are reticent to override the scoring model even when they believe the "recommendation" is wrong.
- Internally developed credit scoring models often lack sophistication and usually have not been subjected to critical analysis of the statistical significance of the factors used to develop a credit score and a credit recommendation...but

References and Bibliography

1. https://www.lendingtree.com/credit-repair/what-are-the-different-types-of-credit-scores/
2. https://www.credit.com/loans/loan-articles/high-risk-loans/
3. https://www.researchgate.net/publication/220613924_Credit_Scoring_Statistical_Techniques_and_Evaluation_Criteria_A_Review_of_the_Literature
4. https://www.researchgate.net/publication/312462885_Credit_scoring_in_banks_and_financial_institutions_via_data_mining_techniques_A_literature_review
5. https://www.omnisci.com/technical-glossary/feature-engineering
6. https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/
7. https://iopscience.iop.org/article/10.1088/1757-899X/1022/1/012042/pdf