Group No: 306 Bank Loan Status

First Name	Last Name	Email (hawk.iit.edu)	Student ID
Abhishek	Singh	asingh135@hawk.iit.edu	A20447125
Megha	Shrivastava	mshrivastava1@hawk.iit.edu	A20450886

Table of Contents

1. Introduction

Loan analysis is the method which helps in determining that the borrowers will be able to pay the loan or not. In this project we are taking 19 attributes, building a model and analyzing it further to check whether the person will be able to pay the loan or not. The dependent variable (Loan status) is a nominal variable we will convert that into numerical variable to apply linear regression model. This approach is also useful in understanding many things like which attribute will affect the loan delinquency. Also, the model analysis generally helps in determining and predicting the things which are related to measuring. The dataset contains the csv file credit_train.csv which contains the various independent and dependent variables. We will use this information to analyze whether the person who takes the loan is able to pay it back or not with the help various attributes present in the dataset. The dataset contains credit score, annual income and years of job which is helpful in predicting whether the person can pay the loan or not. We have applied KNN and logistic regression model. Also, we have formed a gradient boosting model found the accuracy and also stated which model is best.

2. Data

The data set was found on the Kaggle which has 99981 records in the data. The dataset has two tables credit_train.csv and credit_test.csv. The data contains 19 columns and is present in the credit_train.csv file. The data can be found from the link: https://www.kaggle.com/zaurbegiev/my-dataset

In the dataset except the loan status all are independent variable and the loan status is the dependent variable. By this data the analysis in between the dependent and independent variable is done and the further the graph is plot. There are only two values for the dependent variable which are "Fully paid" and "Charged off".

Data understanding:
The attributes present in the dataset are further divided into two categories: Qualitative and Quantitative.

Qualitative	Quantitative
Loan ID	Current Loan Amount
Customer ID	Credit Score
Loan Status	Annual Income
Term	Years in Current Job
Home Ownership	Monthly Debt
Purpose	Year of Credit History
	Month since last delinquent
	Number of Open Account
	Number of Credit Problems
	Current Credit Balance
	Maximum Open Credit
	Bankruptcies
	Tax Liens

Further the attributes are classified as:

- **Qualitative:**

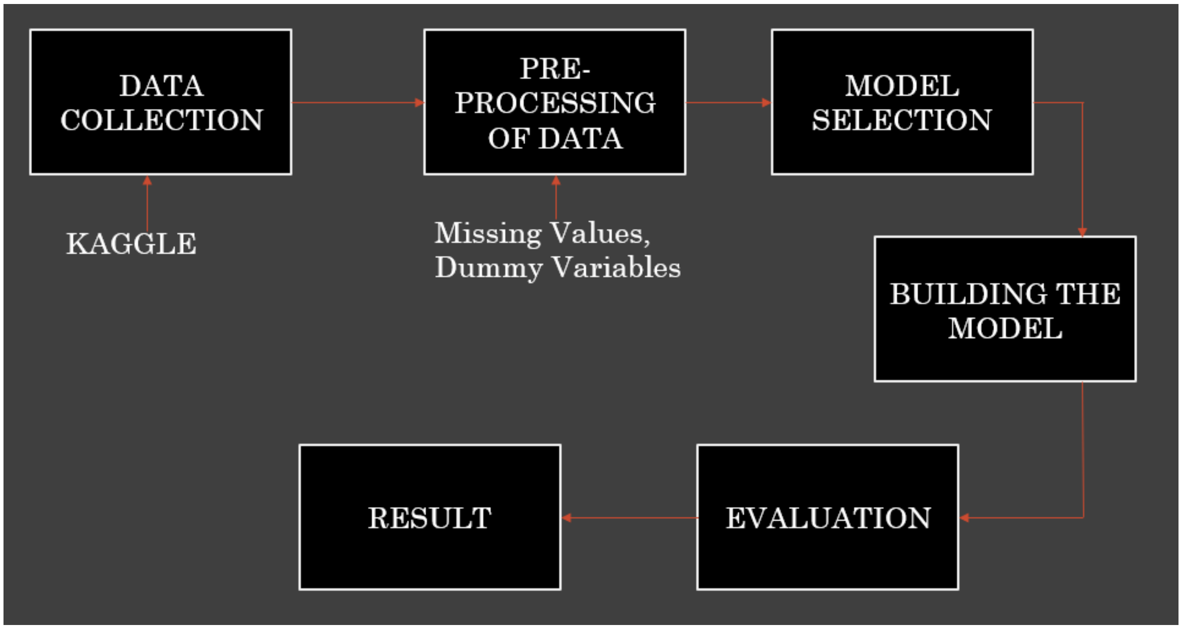
Loan ID: Nominal
Customer ID: Nominal
Loan Status: Nominal
Term: Ordinal
Home Ownership: Nominal
Purpose: Nominal

- **Quantitative:**

  Current Loan Amount: Discrete
  Credit Score: Discrete
  Annual Income: Discrete
  Years in Current Job: Discrete
  Monthly Debt: Discrete
  Year of Credit History: Discrete
  Month since last delinquent: Discrete
  Number of Open Account: Discrete
  Number of Credit Problems: Discrete
  Current Credit Balance: Discrete
  Maximum Open Credit: Discrete
  Bankruptcies: Discrete
  Tax Liens: Discrete

As we see according to the data that loan status is nominal variable, so we will change the data to binary form or the required form according to the analysis and then further analysis it. Like for logistic analysis we will make it to binary (0 or 1), for KNN we will change it to Yes or No. Also, all the remaining will also be converted according to the analysis.

3. System Flow of Project



4. Problem to be solved

By this the we get know about the attributes which will affect whether any person will pay the loan or not. Also, we will build different model which will help in analyzing and predicting that the person will pay the loan or not according to the attributes. With the help of independent and dependent attribute present in the dataset, the problem that who will pay the loan is solved to a level. The data include a credit score column which shows and is helpful in providing the loan and also in predicting further outcomes

5. Solutions

<u>Modeling:</u>

- After preprocessing and normalizing the data we will build different models based on Hold-Out evaluation since our dataset has more than 5000 records.
- We will split the dataset into training and test data set.

We will create various models such as

A. KNN

B. Logistic Regression Model

C. Gradient Boosting Model

6. Experiments and Results

Data Cleaning

In the below data set, you can see there are two columns Loan Id & Customer Id which is just for identification purpose so we will remove those columns. You can see in the next screen shot that we removed those columns.

Loan Id & Customer Id in the Data set:

```
                               Loan.ID                          Customer.ID Loan.Status Current.Loan.Amount       Term Credit.Score
1 14dd8831-6af5-400b-83ec-68e61888a048 981165ec-3274-42f5-a3b4-d104041a9ca9  Fully Paid              445412 Short Term          709
2 4771cc26-131a-45db-b5aa-537ea4ba5342 2de017a3-2e01-49cb-a581-08169e83be29  Fully Paid              262328 Short Term           NA
3 4eed4e6a-aa2f-4c91-8651-ce984ee8fb26 5efb2b2b-bf11-4dfd-a572-3761a2694725  Fully Paid            99999999 Short Term          741
4 77598f7b-32e7-4e3b-a6e5-06ba0d98fe8a e777faab-98ae-45af-9a86-7ce5b33b1011  Fully Paid              347666  Long Term          721
5 d4062e70-befa-4995-8643-a0de73938182 81536ad9-5ccf-4eb8-befb-47a4d608658e  Fully Paid              176220 Short Term           NA
6 89d8cb0c-e5c2-4f54-b056-48a645c543dd 4ffe99d3-7f2a-44db-afc1-40943f1f9750 Charged Off              206602 Short Term         7290
  Annual.Income Years.in.current.job Home.Ownership            Purpose Monthly.Debt Years.of.Credit.History
1       1167493              8 years  Home Mortgage  Home Improvements      5214.74                    17.2
2            NA            10+ years  Home Mortgage Debt Consolidation     33295.98                    21.1
3       2231892              8 years       Own Home Debt Consolidation     29200.53                    14.9
4        806949              3 years       Own Home Debt Consolidation      8741.90                    12.0
5            NA              5 years           Rent Debt Consolidation     20639.70                     6.1
6        896857            10+ years  Home Mortgage Debt Consolidation     16367.74                    17.3
  Months.since.last.delinquent Number.of.Open.Accounts Number.of.Credit.Problems Current.Credit.Balance Maximum.Open.Credit
1                           NA                       6                         1                 228190              416746
2                            8                      35                         0                 229976              850784
3                           29                      18                         1                 297996              750090
4                           NA                       9                         0                 256329              386958
5                           NA                      15                         0                 253460              427174
6                           NA                       6                         0                 215308              272448
  Bankruptcies Tax.Liens
1            1         0
2            0         0
3            0         0
4            0         0
5            0         0
6            0         0
> |
```

Removing the Loan id & Customer Id column from the Data set:

```
> head(credit)
  Loan.Status Current.Loan.Amount       Term Credit.Score Annual.Income Years.in.current.job Home.Ownership            Purpose Monthly.Debt
1  Fully Paid              445412 Short Term          709       1167493              8 years  Home Mortgage  Home Improvements      5214.74
2  Fully Paid              262328 Short Term           NA            NA            10+ years  Home Mortgage Debt Consolidation     33295.98
3  Fully Paid            99999999 Short Term          741       2231892              8 years       Own Home Debt Consolidation     29200.53
4  Fully Paid              347666  Long Term          721        806949              3 years       Own Home Debt Consolidation      8741.90
5  Fully Paid              176220 Short Term           NA            NA              5 years           Rent Debt Consolidation     20639.70
6 Charged Off              206602 Short Term         7290        896857            10+ years  Home Mortgage Debt Consolidation     16367.74
  Years.of.Credit.History Months.since.last.delinquent Number.of.Open.Accounts Number.of.Credit.Problems Current.Credit.Balance
1                    17.2                           NA                       6                         1                 228190
2                    21.1                            8                      35                         0                 229976
3                    14.9                           29                      18                         1                 297996
4                    12.0                           NA                       9                         0                 256329
5                     6.1                           NA                      15                         0                 253460
6                    17.3                           NA                       6                         0                 215308
  Maximum.Open.Credit Bankruptcies Tax.Liens
1              416746            1         0
2              850784            0         0
3              750090            0         0
4              386958            0         0
5              427174            0         0
6              272448            0         0
> |
```

We have created a function to calculate the null values in the data set for all the columns, you can see the output below. As you can see there are 12 columns which consist N/A values. We can also see the number of N/A values too.

Function to calculate null values:

```
> # code for finding a N/A values in the data set.
> missing_data <- colSums(is.na(credit))[colSums(is.na(credit)) > 0] %>% sort(decreasing=T)
> missing_data
Months.since.last.delinquent                 Credit.Score                Annual.Income
                       53655                        19668                        19668
                 Bankruptcies                    Tax.Liens          Maximum.Open.Credit
                         718                          524                          516
          Current.Loan.Amount                 Monthly.Debt      Years.of.Credit.History
                         514                          514                          514
     Number.of.Open.Accounts    Number.of.Credit.Problems       Current.Credit.Balance
                         514                          514                          514
```

Since the Months.since.last delinquent column contains more than 50% of null values. We have removed that column in below screen shot:

```
> #removing Months.since.last.delinquent column as it contains more than 50% of N/A value
> credit=select(credit,-"Months.since.last.delinquent")
> head(credit)
  Loan.Status Current.Loan.Amount       Term Credit.Score Annual.Income Years.in.current.job Home.Ownership
1  Fully Paid              445412 Short Term          709       1167493              8 years  Home Mortgage
2  Fully Paid              262328 Short Term           NA            NA            10+ years  Home Mortgage
3  Fully Paid            99999999 Short Term          741       2231892              8 years       Own Home
4  Fully Paid              347666  Long Term          721        806949              3 years       Own Home
5  Fully Paid              176220 Short Term           NA            NA              5 years           Rent
6 Charged Off              206602 Short Term         7290        896857            10+ years  Home Mortgage
             Purpose Monthly.Debt Years.of.Credit.History Number.of.Open.Accounts Number.of.Credit.Problems
1  Home Improvements      5214.74                    17.2                       6                         1
2 Debt Consolidation     33295.98                    21.1                      35                         0
3 Debt Consolidation     29200.53                    14.9                      18                         1
4 Debt Consolidation      8741.90                    12.0                       9                         0
5 Debt Consolidation     20639.70                     6.1                      15                         0
6 Debt Consolidation     16367.74                    17.3                       6                         0
  Current.Credit.Balance Maximum.Open.Credit Bankruptcies Tax.Liens
1                 228190              416746            1         0
2                 229976              850784            0         0
3                 297996              750090            0         0
4                 256329              386958            0         0
5                 253460              427174            0         0
6                 215308              272448            0         0
>
```

You can see in the below screen shot there are 514 rows at the tail of the data set that contains all the N/A values. So, we have removed all those rows.

```
Purpose Monthly.Debt Years.of.Credit.History Number.of.Open.Accounts Number.of.Credit.Problems
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
                  NA                      NA                      NA                        NA
```

We can see now at the bottom of the data set that there no N/A Rows:

```
> credit<- credit[-seq(nrow(credit),nrow(credit)-514),]
> tail(credit)
      Loan.Status Current.Loan.Amount       Term Credit.Score Annual.Income Years.in.current.job Home.Ownership
99994  Fully Paid               44484 Short Term          717       1152426            10+ years  Home Mortgage
99995  Fully Paid              210584 Short Term          719        783389               1 year  Home Mortgage
99996  Fully Paid              147070 Short Term          725        475437              7 years       Own Home
99997  Fully Paid            99999999 Short Term          732       1289416               1 year           Rent
99998  Fully Paid              103136 Short Term          742       1150545              6 years           Rent
99999  Fully Paid              530332 Short Term          746       1717524              9 years           Rent
                 Purpose Monthly.Debt Years.of.Credit.History Number.of.Open.Accounts Number.of.Credit.Problems
99994     small_business      6280.64                    21.0                       6                         0
99995              other      3727.61                    17.4                       6                         0
99996              other      2202.86                    22.3                       5                         0
99997 Debt Consolidation     13109.05                     9.4                      22                         0
99998 Debt Consolidation      7315.57                    18.8                      12                         1
99999 Debt Consolidation      9890.07                    15.0                       8                         0
      Current.Credit.Balance Maximum.Open.Credit Bankruptcies Tax.Liens
99994                 961932                   0            0         0
99995                    456              259160            0         0
99996                  47766              658548            0         0
99997                 153045              509234            0         0
99998                 109554              537548            1         0
99999                 404225              738254            0         0
>
```

Now, we are left with the 5 columns which contains N/A values. We can see that in the below screen shot:

```
> missing_data <- colSums(is.na(credit))[colSums(is.na(credit)) > 0] %>% sort(decreasing=T)
> missing_data
    Credit.Score    Annual.Income     Bankruptcies        Tax.Liens Maximum.Open.Credit
           19154            19154              204               10                   2
>
```

To remove those N/A values from the 5 columns. We replaced those values with the mean value of the dataset. The code for that is given below.

```
#swaping the N/A values with the mean value
credit$Credit.Score=ifelse(is.na(credit$Credit.Score),ave(credit$Credit.Score, FUN= function(x) mean(x,na.rm=TRUE)),credit$Credit.Score)
credit$Annual.Income=ifelse(is.na(credit$Annual.Income),ave(credit$Annual.Income, FUN= function(x) mean(x,na.rm=TRUE)),credit$Annual.Income)
credit$Bankruptcies=ifelse(is.na(credit$Bankruptcies),ave(credit$Bankruptcies, FUN= function(x) mean(x,na.rm=TRUE)),credit$Bankruptcies)
credit$Tax.Liens=ifelse(is.na(credit$Tax.Liens),ave(credit$Tax.Liens, FUN= function(x) mean(x,na.rm=TRUE)),credit$Tax.Liens)
credit$Maximum.Open.Credit=ifelse(is.na(credit$Maximum.Open.Credit),ave(credit$Maximum.Open.Credit, FUN= function(x) mean(x,na.rm=TRUE)),credit$Maximum.Open.Credit)
```

No N/A values present:

```
> missing_data <- colSums(is.na(credit))[colSums(is.na(credit)) > 0] %>% sort(decreasing=T)
> missing_data
named numeric(0)
```

Dataset with No N/A values:

```
> head(credit,n=10)
   Loan.Status Current.Loan.Amount       Term Credit.Score Annual.Income Years.in.current.job Home.Ownership            Purpose Monthly.Debt
1   Fully Paid              445412 Short Term       709.00       1167493              8 years  Home Mortgage  Home Improvements      5214.74
2   Fully Paid              262328 Short Term      1076.46       1378282            10+ years  Home Mortgage Debt Consolidation     33295.98
3   Fully Paid            99999999 Short Term       741.00       2231892              8 years       Own Home Debt Consolidation     29200.53
4   Fully Paid              347666  Long Term       721.00        806949              3 years       Own Home Debt Consolidation      8741.90
5   Fully Paid              176220 Short Term      1076.46       1378282              5 years           Rent Debt Consolidation     20639.70
6  Charged Off              206602 Short Term      7290.00        896857            10+ years  Home Mortgage Debt Consolidation     16367.74
7   Fully Paid              217646 Short Term       730.00       1184194             < 1 year  Home Mortgage Debt Consolidation     10855.08
8  Charged Off              648714  Long Term      1076.46       1378282             < 1 year  Home Mortgage          Buy House     14806.13
9   Fully Paid              548746 Short Term       678.00       2559110              2 years           Rent Debt Consolidation     18660.28
10  Fully Paid              215952 Short Term       739.00       1454735             < 1 year           Rent Debt Consolidation     39277.75
   Years.of.Credit.History Number.of.Open.Accounts Number.of.Credit.Problems Current.Credit.Balance Maximum.Open.Credit Bankruptcies Tax.Liens
1                     17.2                       6                         1                 228190              416746            1         0
2                     21.1                      35                         0                 229976              850784            0         0
3                     14.9                      18                         1                 297996              750090            0         0
4                     12.0                       9                         0                 256329              386958            0         0
5                      6.1                      15                         0                 253460              427174            0         0
6                     17.3                       6                         0                 215308              272448            0         0
7                     19.6                      13                         1                 122170              272052            1         0
8                      8.2                      15                         0                 193306              864204            0         0
```

A. KNN

- KNN is a supervised learning model. We must have predefined labels for this model. In the given data set we have predefined labels for all the attributes.
- We also should have knowledge from the past data for this model.

- There are three types of classification in this model. We will use binary classification because our dependent variable (loan status) has only two values which are “Fully paid” or “charged off”.

  KNN classifier:

  - In this algorithm we calculate the distance between the target (Loan Status) and various instances.
  - Then we will identify the k-nearest neighbor.
  - Then we will predict the labels and validate with the truth. We can use Manhattan distance or Euclidian distance to calculate the distance between the target (Loan Status) and various instances

Normalization

To calculate the distance between the dependent variable and each independent variable. We need to first to normalize all the numeric values. So that we can correctly calculate the distance. We used min-max function to calculate the relative distance the minimum and maximum value of the column.

```
#normalize selected data using function scale
credit[num.vars] <-lapply(credit[num.vars], scale)
credit[num.vars]


head(credit)
credit[num.vars] <-apply(credit[num.vars], 2, FUN = function(x) (x - min(x))/(max(x)-min(x)))
head(credit[num.vars])

head(credit)
```

Normalized data set:

```
head(credit)
 Loan.Status Current.Loan.Amount       Term Credit.Score Annual.Income Years.in.current.job Home.Ownership              Purpose Monthly.Debt
  Fully Paid         0.004346570 Short Term   0.01790614   0.006592101              8 years  Home Mortgage    Home Improvements   0.01196471
  Fully Paid         0.002515532 Short Term   0.07096898   0.007865899            10+ years  Home Mortgage   Debt Consolidation   0.07639439
  Fully Paid         1.000000000 Short Term   0.02252708   0.013024263              8 years       Own Home   Debt Consolidation   0.06699777
  Fully Paid         0.003369004  Long Term   0.01963899   0.004413335              3 years       Own Home   Debt Consolidation   0.02005744
  Fully Paid         0.001654359 Short Term   0.07096898   0.007865899              5 years           Rent   Debt Consolidation   0.04735578
 Charged Off         0.001958212 Short Term   0.96823105   0.004956649            10+ years  Home Mortgage   Debt Consolidation   0.03755419
 Years.of.Credit.History Number.of.Open.Accounts Number.of.Credit.Problems Current.Credit.Balance Maximum.Open.Credit Bankruptcies Tax.Liens
              0.20328849              0.07894737                0.06666667            0.006940303        0.0002706604    0.1428571         0
              0.26158445              0.46052632                0.00000000            0.006994623        0.0005525512    0.0000000         0
              0.16890882              0.23684211                0.06666667            0.009063423        0.0004871543    0.0000000         0
              0.12556054              0.11842105                0.00000000            0.007796139        0.0002513142    0.0000000         0
              0.03736921              0.19736842                0.00000000            0.007708879        0.0002774329    0.0000000         0
              0.20478326              0.07894737                0.00000000            0.006548502        0.0001769444    0.0000000         0
```

Creating Dummies:

As we know we can normalize the numeric columns using the min-max function. Although, we cannot do this for categorical columns. So, we create dummy variables for categorical data values.

```
#dummies
install.packages("dummies")
library(dummies)

credit.dummy=dummy.data.frame(credit,names=c("Term","Years.in.current.job","Home.Ownership","Purpose"))
head(credit.dummy)

#Removing the extra column
```

Data set with dummy variables:

```
head(credit.dummy)
Loan.Status Current.Loan.Amount TermLong Term TermShort Term Credit.Score Annual.Income Years.in.current.job< 1 year Years.in.current.job1 year
 Fully Paid         0.004346570             0              1   0.01790614   0.006592101                            0                          0
 Fully Paid         0.002515532             0              1   0.07096898   0.007865899                            0                          0
 Fully Paid         1.000000000             0              1   0.02252708   0.013024263                            0                          0
 Fully Paid         0.003369004             1              0   0.01963899   0.004413335                            0                          0
 Fully Paid         0.001654359             0              1   0.07096898   0.007865899                            0                          0
Charged Off         0.001958212             0              1   0.96823105   0.004956649                            0                          0
Years.in.current.job10+ years Years.in.current.job2 years Years.in.current.job3 years Years.in.current.job4 years Years.in.current.job5 years
                            0                           0                           0                           0                           0
                            1                           0                           0                           0                           0
                            0                           0                           0                           0                           0
                            0                           0                           1                           0                           0
                            0                           0                           0                           0                           1
                            1                           0                           0                           0                           0
Years.in.current.job6 years Years.in.current.job7 years Years.in.current.job8 years Years.in.current.job9 years Years.in.current.jobn/a
                          0                           0                           1                           0                       0
                          0                           0                           0                           0                       0
                          0                           0                           1                           0                       0
                          0                           0                           0                           0                       0
                          0                           0                           0                           0                       0
                          0                           0                           0                           0                       0
```

Removing the extra column:

We only need n-1 levels for each column. So, we will remove the extra dummy variable column from the data set.

```
#Removing the extra column
credit.final=select(credit.dummy,-c("TermLong Term","Years.in.current.jobn/a","Home.OwnershipHaveMortgage","PurposeBusiness Loan"))
head(credit.final)
```

```
  Loan.Status Current.Loan.Amount TermShort Term Credit.Score Annual.Income Years.in.current.job< 1 year Years.in.current.job1 year Years.in.current.job10+ years
1  Fully Paid         0.004346570              1   0.01790614   0.006592101                            0                          0                             0
2  Fully Paid         0.002515532              1   0.07096898   0.007865899                            0                          0                             1
3  Fully Paid         1.000000000              1   0.02252708   0.013024263                            0                          0                             0
4  Fully Paid         0.003369004              0   0.01963899   0.004413335                            0                          0                             0
5  Fully Paid         0.001654359              1   0.07096898   0.007865899                            0                          0                             0
6 Charged Off         0.001958212              1   0.96823105   0.004956649                            0                          0                             1
  Years.in.current.job2 years Years.in.current.job3 years Years.in.current.job4 years Years.in.current.job5 years Years.in.current.job6 years
1                           0                           0                           0                           0                           0
2                           0                           0                           0                           0                           0
3                           0                           0                           0                           0                           0
4                           0                           1                           0                           0                           0
5                           0                           0                           0                           1                           0
6                           0                           0                           0                           0                           0
  Years.in.current.job7 years Years.in.current.job8 years Years.in.current.job9 years Home.OwnershipHome Mortgage Home.OwnershipOwn Home Home.OwnershipRent
1                           0                           1                           0                           1                      0                  0
2                           0                           0                           0                           1                      0                  0
3                           0                           1                           0                           0                      1                  0
4                           0                           0                           0                           0                      1                  0
5                           0                           0                           0                           0                      0                  1
6                           0                           0                           0                           1                      0                  0
```

KNN Model

We have created KNN model using N fold cross validation to determine the best K value for our data set. After creating the model, we can see that we got a best accuracy on K=24 which is 0.8119581= 81.19%

```
> model.nfold1 = train(credit.final[,2:42],credit.final$Loan.Status,'knn'
id(k = 20:30))
> model.nfold1
k-Nearest Neighbors

99999 samples
   41 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 89999, 89999, 89999, 90000, 89999, 89999, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
  20  0.8106481  0.2553137
  21  0.8113581  0.2553006
  22  0.8110681  0.2533087
  23  0.8113781  0.2526653
  24  0.8119581  0.2548797
  25  0.8116681  0.2513339
  26  0.8114181  0.2500168
  27  0.8115381  0.2489316
  28  0.8116081  0.2487669
  29  0.8115781  0.2469254
  30  0.8116581  0.2467509

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 24.
>
```

Model for K=24:

```
library(class)
#final model KNN
# training and testing data
credit.final=credit.final[sample(nrow(credit.final)),]
select.data = sample (1:nrow(credit.final), 0.80*nrow(credit.final))
train.data.final=credit.final[select.data,]
test.data.final = credit.final[-select.data,]
head(train.data.final)
head(test.data.final)

#creating labels for training data and testing data
train.Loan.status.final=train.data.final$Loan.Status
head(train.Loan.status.final)
train.data.final=select(train.data.final,-1)
head(train.data.final)

test.Loan.status.final=test.data.final$Loan.Status
head(test.Loan.status.final)
test.data.final=select(test.data.final,-1)
head(test.data.final)

#KNN model
model.final=knn(train = train.data.final,test = test.data.final,cl=train.Loan.status.final,k=24)
summary(model.final)
```

Confusion Metrics:

```
> table(model.final,test.Loan.status.final)
              test.Loan.status.final
model.final           Charged Off Fully Paid
                  0         0          0
  Charged Off     0      1468       1420
  Fully Paid      0      2986      14126
>
```

After creating confusion metrics. We will calculate Precision and Recall:

Precision: True positive/(True positive + False Positive)=1468/(1468+1420)

Precision = 0.5083=50.83%

Recall: True Positive/(True Positive + False Negative)= 1468/(1468+2986)

Recall= 0.3295=32.95%

B. Logistic Regression Model

- Logistic Regression can be applied for binary dependent variable and various independent variables.
- In the dataset, dependent variable is loan Status which has value “Fully paid” or “Charged off”. So, we can apply the logistic regression model because we have only two values for it.
- We will use different feature selection methods such as backward, forward, stepwise etc. We will select the best model based on the AIC value.

We will perform all the data cleaning techniques which we performed the for the KNN. We can see the data below.

Data Set:

```
> tail(credit.log)
      Loan.Status Current.Loan.Amount       Term Credit.Score Annual.Income Years.in.current.job Home.Ownership             Purpose Monthly.Debt
99994  Fully Paid               44484 Short Term          717       1152426            10+ years  Home Mortgage      small_business      6280.64
99995  Fully Paid              210584 Short Term          719        783389               1 year  Home Mortgage               other      3727.61
99996  Fully Paid              147070 Short Term          725        475437              7 years       Own Home               other      2202.86
99997  Fully Paid            99999999 Short Term          732       1289416               1 year           Rent Debt Consolidation     13109.05
99998  Fully Paid              103136 Short Term          742       1150545              6 years           Rent Debt Consolidation      7315.57
99999  Fully Paid              530332 Short Term          746       1717524              9 years           Rent Debt Consolidation      9890.07
      Years.of.Credit.History Number.of.Open.Accounts Number.of.Credit.Problems Current.Credit.Balance Maximum.Open.Credit Bankruptcies Tax.Liens
99994                    21.0                       6                         0                 961932                   0            0         0
99995                    17.4                       6                         0                    456              259160            0         0
99996                    22.3                       5                         0                  47766              658548            0         0
99997                     9.4                      22                         0                 153045              509234            0         0
99998                    18.8                      12                         1                 109554              537548            1         0
99999                    15.0                       8                         0                 404225              738254            0         0
> missing_data2 <- colSums(is.na(credit.log))[colSums(is.na(credit.log)) > 0] %>% sort(decreasing=T)
```

Scaling:

The additional thing which we are going to do over here is Vector scaling. In many machine learning algorithms, to bring all features in the same standing, we need to do scaling so that one significant number doesn't impact the model just because of their large magnitude. scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one. We are going to scale all are numeric columns.

Data set after scaling:

```
 Loan.Status Current.Loan.Amount       Term Credit.Score Annual.Income Years.in.current.job Home.Ownership            Purpose Monthly.Debt
  Fully Paid          -0.3559827 Short Term   -0.2769929    -0.2167940              8 years  Home Mortgage  Home Improvements   -1.0889320
  Fully Paid          -0.3617431 Short Term    0.0000000     0.0000000            10+ years  Home Mortgage Debt Consolidation    1.2175321
  Fully Paid           2.7763514 Short Term   -0.2528712     0.8779274              8 years       Own Home Debt Consolidation    0.8811506
  Fully Paid          -0.3590581  Long Term   -0.2679473    -0.5876091              3 years       Own Home Debt Consolidation   -0.7992273
  Fully Paid          -0.3644524 Short Term    0.0000000     0.0000000              5 years           Rent Debt Consolidation    0.1780034
 Charged Off          -0.3634965 Short Term    4.6837898    -0.4951398            10+ years  Home Mortgage Debt Consolidation   -0.1728758
 Years.of.Credit.History Number.of.Open.Accounts Number.of.Credit.Problems Current.Credit.Balance Maximum.Open.Credit Bankruptcies Tax.Liens
              -0.1424297              -1.0237046                         1           -0.176647582        -0.041035323            1         0
               0.4134949               4.7648980                         0           -0.171899754         0.010731628            0         0
              -0.4702827               1.3715792                         1            0.008921789        -0.001277969            0         0
              -0.8836626              -0.4248837                         0           -0.101844033        -0.044588086            0         0
              -1.7246768               0.7727582                         0           -0.109470864        -0.039791594            0         0
              -0.1281753              -1.0237046                         0           -0.210892556        -0.058245493            0         0
```

Training and Testing the data:

We divide the data set using Hold out evaluation. We will divide the data set into 80% training data and 20% test data. The code can be seen below.

```
#test and train
credit.log=credit.log[sample(nrow(credit.log)),]
select.data=sample(1:nrow(credit.log),0.8*nrow(credit.log))
train.data.log=credit.log[select.data,]
test.data.log=credit.log[-select.data,]
head(train.data.log)

train.log.label=train.data.log$Loan.Status
test.log.label=test.data.log$Loan.Status
head(train.log.label)
head(test.log.label)

train.data.log=select(train.data.log,-1)
test.data.log=select(test.data.log,-1)
head(train.data.log)
head(test.data.log)

head(train.log.label)
head(test.log.label)
```

```
> head(test.data.log)
      Current.Loan.Amount       Term Credit.Score Annual.Income Years.in.current.job Home.Ownership            Purpose Monthly.Debt Years.of.Credit.History
59577          -0.3533046  Long Term   -0.3078989    0.20664578            10+ years  Home Mortgage Debt Consolidation   0.98828392               2.0670143
39291           2.7763514 Short Term   -0.2528712   -0.74233683              3 years           Rent Debt Consolidation  -0.53316522              -0.9264260
92098          -0.3638522 Short Term   -0.2724701    0.02797997              5 years           Rent Debt Consolidation   0.09891354              -0.8694081
57940          -0.3584697 Short Term    0.0000000    0.00000000              3 years  Home Mortgage Debt Consolidation   0.79216675               0.2424412
67977          -0.3598507 Short Term   -0.2611630   -0.47231558            10+ years  Home Mortgage Debt Consolidation  -0.27170687               2.1097777
62494          -0.3628631 Short Term   -0.2739777   -0.78757486              4 years       Own Home Debt Consolidation  -0.16309101               0.2139322
      Number.of.Open.Accounts Number.of.Credit.Problems Current.Credit.Balance Maximum.Open.Credit Bankruptcies Tax.Liens
59577               1.3715792                         0              1.9840688         0.114341624            0         0
39291              -1.4229186                         0             -0.7056264        -0.085528820            0         0
92098              -0.2252767                         0             -0.1330585        -0.053210226            0         0
57940               1.1719722                         0             -0.2436223        -0.006048222            0         0
67977               0.5731512                         0              0.2636882        -0.016234207            0         0
62494              -0.2252767                         0             -0.1622526        -0.036879063            0         0
> head(train.log.label)
[1] Fully Paid  Charged Off Fully Paid  Fully Paid  Fully Paid  Charged Off
Levels:  Charged Off Fully Paid
> head(test.log.label)
[1] Charged Off Fully Paid  Fully Paid  Fully Paid  Charged Off Fully Paid
Levels:  Charged Off Fully Paid
```

Converting the dependent variable into binary

We converted our dependent variable into binary value so that we can calculate res value easily and we can classify the probability value close to 1 as “fully paid” and close to 0 as “Charged off”.

```
#converting dependent variable into binary for train data set
train.log.label=as.character(train.log.label)
head(train.log.label)
is.character(train.log.label)
str(train.log.label)
train.log.label[train.log.label == "Fully Paid"]=1
train.log.label[train.log.label == "Charged Off"]=0
train.log.label=as.integer(train.log.label)
head(train.log.label)

#converting dependent variable into binary for test data set

test.log.label=as.character(test.log.label)
head(test.log.label)
is.character(test.log.label)
str(train.log.label)
test.log.label[test.log.label == "Fully Paid"]=1
test.log.label[test.log.label == "Charged Off"]=0
test.log.label=as.integer(test.log.label)
head(test.log.label)
```

```
> train.log.label=as.character(train.log.label)
> head(train.log.label)
[1] "Fully Paid"  "Charged Off" "Fully Paid"  "Fully Paid"  "Fully Paid"  "Charged Off"
> is.character(train.log.label)
[1] TRUE
> str(train.log.label)
 chr [1:79999] "Fully Paid" "Charged Off" "Fully Paid" "Fully Paid" "Fully Paid" "Charged Off" "Fully Paid" "Fully Paid" "Fully Paid" "Fully Paid" ..
> train.log.label[train.log.label == "Fully Paid"]=1
> train.log.label[train.log.label == "Charged Off"]=0
> train.log.label=as.integer(train.log.label)
> head(train.log.label)
[1] 1 0 1 1 1 0
> test.log.label=as.character(test.log.label)
> head(test.log.label)
[1] "Charged Off" "Fully Paid"  "Fully Paid"  "Fully Paid"  "Charged Off" "Fully Paid"
> is.character(test.log.label)
[1] TRUE
> str(train.log.label)
 int [1:79999] 1 0 1 1 1 0 1 1 1 1 ...
> test.log.label[test.log.label == "Fully Paid"]=1
> test.log.label[test.log.label == "Charged Off"]=0
> test.log.label=as.integer(test.log.label)
> head(test.log.label)
[1] 0 1 1 1 0 1
```

Logistic Model

```
#logistic model
model.log.final=glm(train.log.label~.,data = train.data.log,family="binomial")
summary(model.log.final)
```

We calculated different cut off value and predicted accuracy of each model. Then we selected best Cut off value.

Res & Accuracy Calculation for 0.4

```
> for(i in 1:length(res)){
+   if(res[i]>0.4){
+     res[i]=1
+
+   }else{
+     res[i]=0
+   }
+ }
> library(Metrics)
Warning message:
package 'Metrics' was built under R version 3.6.3
> confmatrix=table(Actual_Value=test.log.label,predicted_value= res)
> confmatrix
            predicted_value
Actual_Value     0     1
           0   903  3753
           1     0 15344
> accuracy(test.log.label,res)
[1] 0.81235
```

Res & Accuracy Calculation for 0.2

```
[ reached getOption("max.print" 
> for(i in 1:length(res)){
+   if(res[i]>0.2){
+     res[i]=1
+
+   }else{
+     res[i]=0
+   }
+ }
> accuracy(test.log.label,res)
[1] 0.8192
>
```

Res & Accuracy Calculation for 0.3

```
> res=predict(model.log.final,test.data.log,type = "response")
> for(i in 1:length(res)){
+   if(res[i]>0.3){
+     res[i]=1
+
+   }else{
+     res[i]=0
+   }
+ }
> accuracy(test.log.label,res)
[1] 0.8153
>
```

Res & Accuracy Calculation for 0.5

```
glm.fit: fitted probabilities numerically 0 or 1 occurred
> # calculating res
> res=predict(model.log.final,test.data.log,type = "response")
> for(i in 1:length(res)){
+   if(res[i]>0.5){
+     res[i]=1
+
+   }else{
+     res[i]=0
+   }
+ }
> accuracy(test.log.label,res)
[1] 0.8222
>
```

Res & Accuracy Calculation for 0.6

```
# calculating res
res=predict(model.log.final,test.data.log,type = "response")
for(i in 1:length(res)){
  if(res[i]>0.6){
    res[i]=1

  }else{
    res[i]=0
  }
}
accuracy(test.log.label,res)
1] 0.8154
```

Res & Accuracy Calculation for 0.7

```
glm.fit: fitted probabilities numerically 0 or 1 o
> for(i in 1:length(res)){
+   if(res[i]>0.7){
+     res[i]=1
+
+   }else{
+     res[i]=0
+   }
+ }
> accuracy(test.log.label,res)
[1] 0.7358
>
```

As we can see above the best cut off value for this model is **0.5**. This can be concluded by observing the accuracy for that cut off model. If we **increase** the cut off value, then the **accuracy decreases** and if we **decrease** the cut off value then also the **accuracy decreases**.

Confusion Metrics for res 0.5:

```
> table(Actual_Value=test.log.label,predicted_value= res)
            predicted_value
Actual_Value     0     1
           0   930  3561
           1    19 15490
```

Precision: True positive/(True positive + False Positive)=930/(930+3561)

Precision = 0.2070=20.70%

Recall: True Positive/(True Positive + False Negative)= 930/(930+19)

Recall= 0.9799=97.99%

C. Gradient Boosting Model

After performing data cleaning and scaling the data set we will divide the data set into training and testing as shown below. We will also convert our dependent variable into binary.

Training and Testing

```
#test and train
credit.dummy1=credit.dummy1[sample(nrow(credit.dummy1)),]
select.data=sample(1:nrow(credit.dummy1),0.8*nrow(credit.dummy1))
train.data.grad=credit.dummy1[select.data,]
test.data.grad=credit.dummy1[-select.data,]
head(train.data.grad)

train.grad.label=train.data.grad$Loan.Status
test.grad.label=test.data.grad$Loan.Status
head(train.grad.label)
head(test.grad.label)

train.data.grad=select(train.data.grad,-1)
test.data.grad=select(test.data.grad,-1)
head(train.data.grad)
head(test.data.grad)

head(train.grad.label)
unique(test.grad.label)
```

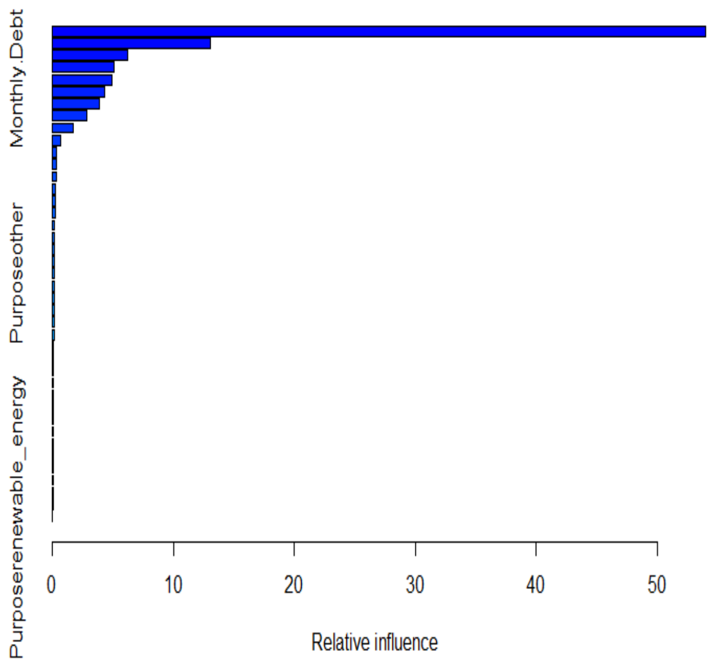
Gradient Boosting Model:

As we can see in the below screenshot, Credit score plays very important role whether the person will be able to pay the loan or not. Credit score has maximum rel.inference after that Current loan amount plays very important role and so on.

```
> Gradient.boost=gbm(train.grad.label~.,data = train.data.grad,distribution = "gaussian",n.tre
> summary(Gradient.boost)
                                                                 var      rel.inf
Credit.Score                                            Credit.Score 53.99929468
Current.Loan.Amount                              Current.Loan.Amount 13.01901695
Annual.Income                                          Annual.Income  6.17598785
Maximum.Open.Credit                              Maximum.Open.Credit  5.08893058
Monthly.Debt                                            Monthly.Debt  4.87340883
Current.Credit.Balance                        Current.Credit.Balance  4.25571737
Years.of.Credit.History                      Years.of.Credit.History  3.87923690
`TermShort Term`                                    `TermShort Term`  2.84416431
Number.of.Open.Accounts                      Number.of.Open.Accounts  1.66851772
Home.OwnershipRent                                Home.OwnershipRent  0.62553748
`Home.OwnershipHome Mortgage`          `Home.OwnershipHome Mortgage`  0.34281448
Purposesmall_business                          Purposesmall_business  0.31983583
`PurposeDebt Consolidation`              `PurposeDebt Consolidation`  0.29805864
Number.of.Credit.Problems                  Number.of.Credit.Problems  0.23161899
`Years.in.current.job10+ years` `Years.in.current.job10+ years`  0.22758145
Tax.Liens                                                  Tax.Liens  0.20592560
Bankruptcies                                            Bankruptcies  0.17202490
`Years.in.current.job3 years`          `Years.in.current.job3 years`  0.14591317
`Years.in.current.job< 1 year`        `Years.in.current.job< 1 year`  0.13258425
`Years.in.current.job2 years`          `Years.in.current.job2 years`  0.12768791
Purposeother                                            Purposeother  0.11448109
`Years.in.current.job7 years`          `Years.in.current.job7 years`  0.11049969
`Years.in.current.job4 years`          `Years.in.current.job4 years`  0.10736516
`Home.OwnershipOwn Home`                    `Home.OwnershipOwn Home`  0.09974454
`PurposeBuy a Car`                                `PurposeBuy a Car`  0.09528329
`Years.in.current.job5 years`          `Years.in.current.job5 years`  0.09099810
`Years.in.current.job9 years`          `Years.in.current.job9 years`  0.08425178
`Years.in.current.job1 year`            `Years.in.current.job1 year`  0.08208809
PurposeOther                                            PurposeOther  0.07925212
`Years.in.current.job6 years`          `Years.in.current.job6 years`  0.06871946
`PurposeHome Improvements`                `PurposeHome Improvements`  0.06825031
`Years.in.current.job8 years`          `Years.in.current.job8 years`  0.06387096
`PurposeMedical Bills`                        `PurposeMedical Bills`  0.06375668
Purposemajor_purchase                          Purposemajor_purchase  0.05153879
Purposevacation                                      Purposevacation  0.05016276
`PurposeBuy House`                                `PurposeBuy House`  0.04983904
Purposemoving                                          Purposemoving  0.03437133
Purposewedding                                        Purposewedding  0.01886387
`PurposeTake a Trip`                            `PurposeTake a Trip`  0.01823612
`PurposeEducational Expenses`          `PurposeEducational Expenses`  0.01456893
Purposerenewable_energy                      Purposerenewable_energy  0.00000000
```

Graph:

We have created a bar graph for the above model. The bar graph is based on the rel.inference value. As you can se Credit score has a maximum height compared to any other feature.



Prediction Metrics:

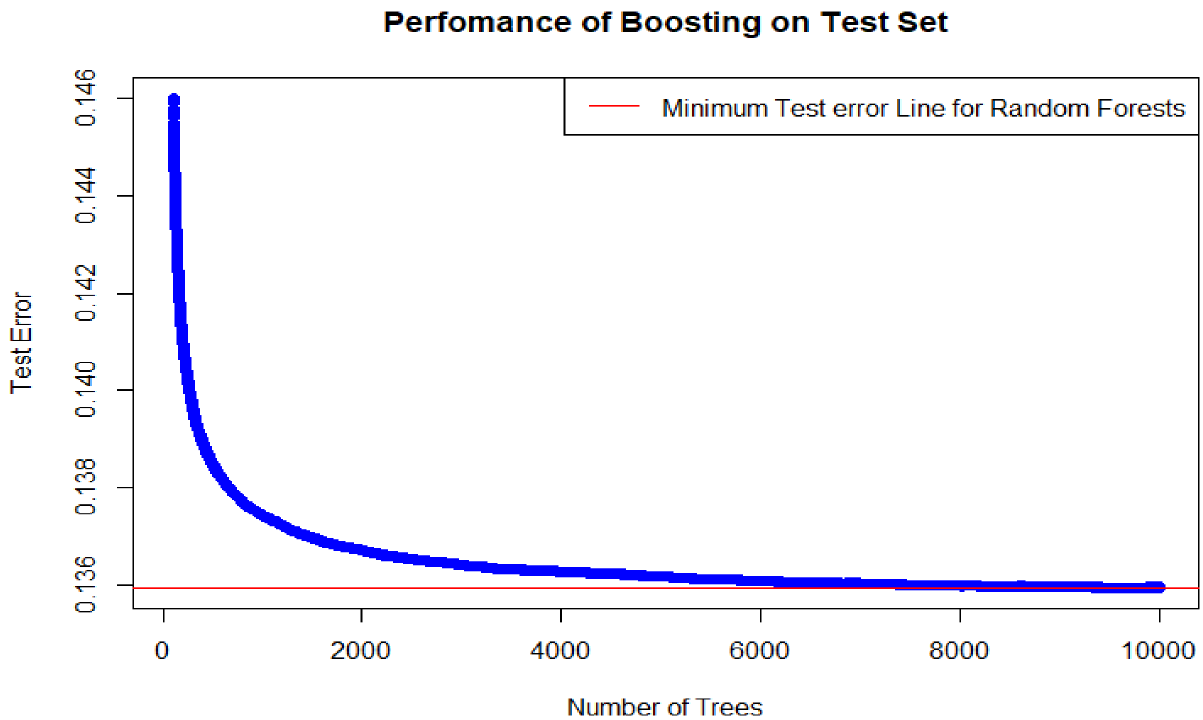
```
> n.trees = seq(from=100 ,to=10000, by=1) #no of trees-a vector of 100 values
> #Generating a Prediction matrix for each Tree
> predmatrix<-predict(Gradient.boost,test.data.grad,n.trees = n.trees)
> dim(predmatrix) #dimentions of the Prediction Matrix
[1] 20000  9901
>
```

Test Errors:

```
> test.error<-with(test.data.grad,apply( (predmatrix-test.grad.label)^2,2,mean))
> head(test.error) #contains the Mean squared test error for each of the 100 trees averaged
      100       101       102       103       104       105
0.1460116 0.1459054 0.1458006 0.1456993 0.1456003 0.1455015
>
```

Graph:

We have created a graph for the test errors done by the gradient boosting model. As you can see in the graph as the number of trees increases the number of test errors decreases. At 10,000 trees you can see the test error is near to 0.135



Accuracy

As the test error value for gradient boosting is 0.135. The accuracy of the model is 1- test error. So, the accuracy is 0.865. Therefore, the accuracy is 86.5%.

7. Conclusion:

- After comparing all the model, we found that Accuracy of KNN Model is 81.22%. Accuracy of Logistic Regression Model is 82.222%. Accuracy of Gradient Boosting Algorithm is 86.5%. So, the best model for data set is Gradient Boosting.
- We also calculated Precision and recall to find out the percentage of true value with the false positive and false negative.
- We used various data cleaning techniques for cleaning the data. We also understood the concept of scaling