LOAN APPROVAL PREDICTION USING MACHINE LEARNING

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by
Addepalli Sravani, 221910310003
Sowmya Nekkanti, 221910310036
Sameera Jampala, 221910310048
Bhargav Kiran Surapaneni, 221910310052

Under the esteemed guidance of
Dr. Pranayanath Reddy
Asst. professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GITAM
(Deemed to be University)
HYDERABAD

OCTOBER 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM (Deemed to be University)

DECLARATION

I/We, hereby declare that the project report entitled "LOAN APPROVAL PREDICTION USING MACHINE LEARNING" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. In Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No(s)	Name(s)	Signature(s)
221910310003	Addepalli Sravani	
221910310036	Sree Lakshmi Sowmya .N	
221910310048	Sameera Jampala	
221910310052	Bhargav Kiran Surapaneni	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM
(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled **"LOAN APPROVAL PREDICTION USING MACHINE LEARNING**" is a bonafide record of work carried out by **Addepalli Sravani (221910310003), Sree Lakshmi Sowmya .N (221910310036), Sameera Jampala (221910310048) and Bhargav Kiran Surapaneni (221910310052)** students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide	**Project coordinator**	**Head of the Department**
Dr. Pranayanath Reddy	**Dr.S.Aparna**	**Dr.S. Phani Kumar**
Assistant Professor Dept. of CSE	Assistant Professor Dept. of CSE	Professor & HOD Dept. of CSE

TABLE OF CONTENTS

1	Abstract	1
2	Introduction	2
3	Literature Review	3
4	Problem Identification And Solving	4
5	System Methodology	11
6	Overview of Technologies	12
7	Implementation	12
	7.1 Coding	12
8	Results And Discussions	17
9	Conclusion and Future Scope	27
10	References	27

1. ABSTRACT

In our banking system, banks have many products to sell but the main source of income of any bank is on its credit line. So they can earn from the interest of those loans. A bank's profit or a loss depends to a large extent on loans i.e. whether the customers are paying back the loan or defaulting. By predicting the loan defaulters, the bank can reduce its Non- Performing Assets. This makes the study of this phenomenon very important. Previous research in this era has shown that there are so many methods to study the problem of controlling loan default. But as the right predictions are very important for the maximization of profits, it is essential to study the nature of the different methods and their comparison. A very important approach in predictive analytics is used to study the problem of predicting loan defaulters: The Logistic regression model. The data is collected from the Kaggle for studying and prediction. Logistic Regression models have been performed and the different measures of performances are computed. The models are compared on the basis of the performance measures such as sensitivity and specificity. The final results have shown that the model produces different results. Model is marginally better because it includes variables (personal attributes of the customer like age, purpose, credit history, credit amount, credit duration, etc.) other than checking account information (which shows the wealth of a customer) that should be taken into account to calculate the probability of default on loan correctly. Therefore, by using a logistic regression approach, the right customers to be targeted for granting loan can be easily detected by evaluating their likelihood of default on loan. The model concludes that a bank should not only target the rich customers for granting loans but it should assess the other attributes of a customer as well which play a very important part in credit granting decisions and predicting the loan defaulters.

2. INTRODUCTION

This paper has taken the data of previous customers of various banks to whom on a set of parameters loans were approved. So the machine learning model is trained on that record to get accurate results. Our main objective of this research is to predict the safety of loans. To predict loan safety, the logistic regression algorithm is used. First the data is cleaned so as to avoid the missing values in the data set. To train our model data set of 1500 cases and 10 numerical and 8 categorical attributes has been taken. To credit a loan to a customer various parameters like CIBIL Score (Credit History), Business Value, Assets of Customer etc., has been considered.

Fig 2.1 Attributes of the dataset

Qualification	Categorical
In Service / Business Owner	Categorical
Individual income of Applicant	Qualitative
Individual income of Co-Applicant (if Any)	Qualitative
Amount of Loan required	Qualitative
Term for which loan Required	Qualitative
Credit History of Applicant	Qualitative
Area of Property	Categorical

3. LITERATURE REVIEW

3.1 DATASET EXTRACTION

Kaggle is one of the largest data source providers for the learning purpose. It allows users to find and publish data sets, explore and build models in a web-based data-science environment and hence the data is collected from Kaggle.

3.2 STUDY OF DATASETS

A Dataset is a set or collection of data. This set is normally presented in a tabular pattern. Every column describes a particular variable. And each row corresponds to a given member of the data set, as per the given user requirements, two data sets are considered, one for the training and another testing. The training dataset is used to train the model in which datasets are further divided into two parts such as 80:20 or 70:30 the major dataset is used for the training of the model and the minor dataset is used for the test model. Hence the accuracy of our developed model is calculated.

3.3 LITERATURE SURVEY

Logistic Regression is a popular and very useful algorithm of machine learning for classification problems. The advantage of logistic regression is that it is a predictive analysis. It is used for description of data and used to explain the relationship between a single binary variable and single or multiple nominal, ordinal and ratio level variables which are independent in nature. The considered dataset is divided into two, train dataset and test dataset. The dataset we considered in this problem is LOAN PREDICTION PROBLEM. The considered dataset is 614 rows and 13 columns. The included columns represent LOAN ID, GENDER, MARITAL STATUS, DEPENDENTS, EDUCATION, EMPLOYMENT, APPLICANT'S INCOME, CO-APPLICANT'S INCOME, LOAN AMOUNT, TERM OF LOAN, CREDIT HISTORY, PROPERTY STATUS AND LOAN STATUS RESPECTIVELY. Each row represents a unique loan id. The gender columns have three various types, female, male and other. The dependents column includes 0, 1 and other. The education columns have two types, namely, graduates and non- graduates. The employment column is basically about self-employment, that is whether the applicant is self-employed or not. The column of property area has three types, semi-urban, urban and others. The credit history ranges from 0.00-0.10 and 0.90-1.00.

The model development for the prediction is taken in account using the sigmoid function in logistic regression as the outcome is targeted binary either0 or 1. The dataset of bank customers has been divided into training and test data sets... The train dataset contains approximately 600+ rows and 13+ columns whereas the test dataset contains 300+ rows and 12+ columns, the test dataset does not contain the target variable. Both the datasets are having missing values in their rows, and the mean, median or mode is used to fill the missing values but not removing the rows completely because the datasets are already small. Using the Feature Engineering techniques, the project is further proceeded and moves towards exploratory data analysis, where the dependent and independent variable is studied through statistics concepts such normal distribution, Probability density function etc. Study of the univariate, bivariate and multivariate analysis will be eligible for loans and therefore the logistic regression is enabled using the sigmoid function as it divides the probability into binary output. Therefore the Prediction model can be developed.

4. PROBLEM IDENTIFICATION AND SOLVING

Banks, Housing Finance Companies and some NBFC deal in various types of loans like housing loan, personal loan, business loan etc., all over the country. These companies have existence in Rural, Semi Urban and Urban areas. After applying for a loan by a customer these companies validate the eligibility of customers to get the loan or not. This paper provides a solution to automate this process by employing machine learning algorithms. So the customer will fill an online loan application form. This form consist details like Se x, Marital Status, Qualification, Details of Dependents, Annual Income, Amount of Loan, Credit History of Applicant and others. To automate this process by using machine learning algorithm, First the algorithm will identify those segments of the customers who are eligible to get loan amounts so bank can focus on these customers [4][7]. Loan prediction is a very common real-life problem that every finance company faces in their lending operations. If the loan approval process is automated, it can save a lot of man hours and improve the speed of service to the customers. The increase in customer satisfaction and savings in operational costs are significant [9]. However, the benefits can only be reaped if the bank has a robust model to accurately predict which customer's loan it should approve and which to reject, in order to minimize the risk of loan default. This problem has been solved using machine learning techniques.

4.1 TYPES OF ML MODELS

Depending on the situation, machine learning algorithms function using more or less human intervention/reinforcement. The four major machine learning models are:

- supervised learning
- unsupervised learning
- semi-supervised learning reinforcement learning

4.1.1 MODELS USED

- Supervised Learning

With supervised learning, the computer is provided with a labeled set of data that enables it to learn how to do a human task. This is the least complex model, as it attempts to replicate human learning.

- Unsupervised Learning

With unsupervised learning, the computer is provided with unlabeled data and extracts previously unknown patterns/insights from it. There are many different ways machine learning algorithms do this, including:

Clustering, in which the computer finds similar data points within a data set and groups them accordingly (creating "clusters").

Density estimation, in which the computer discovers insights by looking at how a data set is distributed.

Anomaly detection, in which the computer identifies data points within a data set that are significantly different from the rest of the data.

Principal component analysis (PCA), in which the computer analyzes a data set and summarizes it so that it can be used to make accurate predictions.

4.2 SUPERVISED LEARNING ALGORITHM

4.2.1 LINEAR REGRESSION

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

- Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.
- Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y (output).
- Hence, the name is Linear Regression.

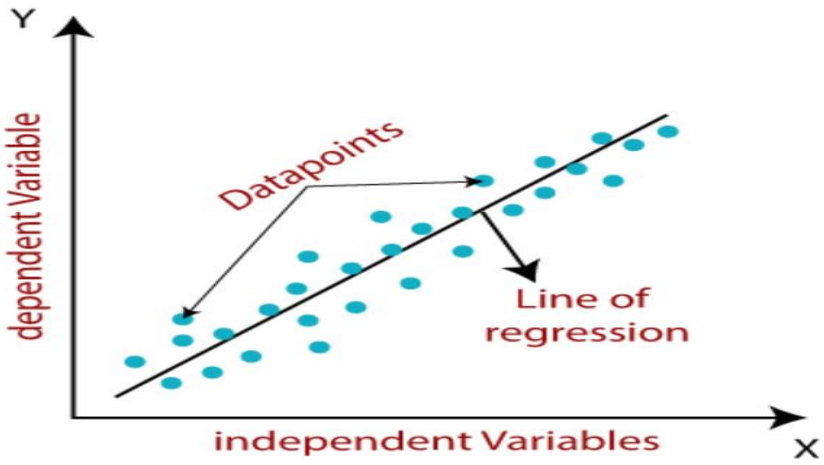


Fig 4.2.1 Linear regression

4.2.2 LOGISTIC REGRESSION

Logistic Regression is a popular and very useful algorithm of machine learning for classification problems. The advantage of logistic regression is that it is a predictive analysis.

● Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

● For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted or not to a particular college. These binary outcomes allow straightforward decisions between two alternatives.

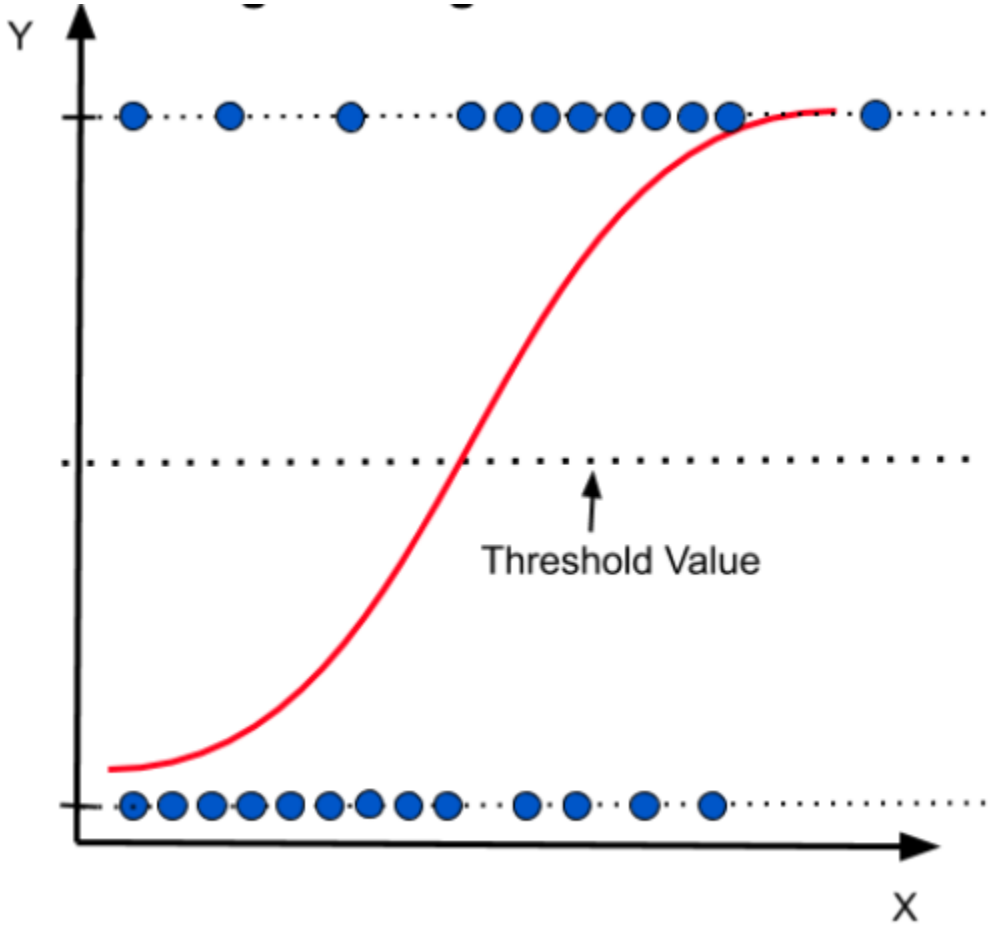


Fig 4.2.2 Logistic regression

4.2.3 SUPPORT VECTOR MACHINES

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

SVM algorithm can be used for Face detection, image classification, text categorization, etc.

SVM can be of two types:

- Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

4.2.4 RANDOM FOREST CLASSIFIER

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

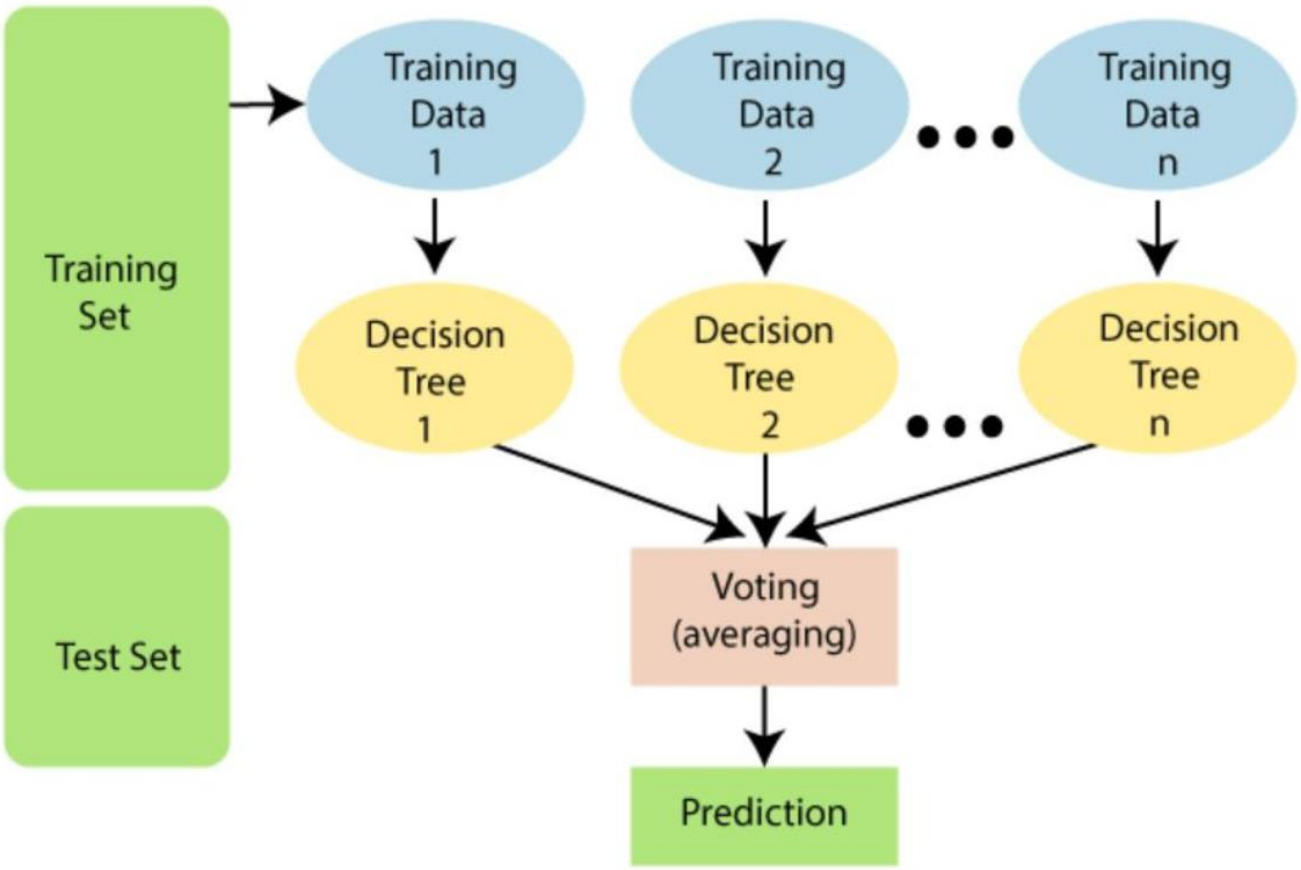


Fig 4.2.3 Working of the Random Forest algorithm

Advantages of Random Forest:

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the over fitting issue.
- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

Disadvantages of Random Forest:

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

There are mainly four sectors where Random forest mostly used:

- Banking: Banking sector mostly uses this algorithm for the identification of loan risk.
- Medicine: With the help of this algorithm, disease trends and risks of the disease can be identified.
- Land Use: We can identify the areas of similar land use by this algorithm.
- Marketing: Marketing trends can be identified using this algorithm.

Implementation Steps are given below:

1) Data Pre-processing step
2) Fitting the Random forest algorithm to the Training set
3) Predicting the test result
4) Test accuracy of the result (Creation of Confusion matrix)
5) Visualizing the test set result.

4.3 UNSUPERVISED LEARNING

4.3.1 K-MEANS

● K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.

● K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of predefined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

● It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

● It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

● The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

● The k-means clustering algorithm mainly performs two tasks:

1) Determines the best value for K center points or centroids by an iterative process.

2) Assign each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster. Hence each cluster has data points with some commonalities, and it is away from other clusters.

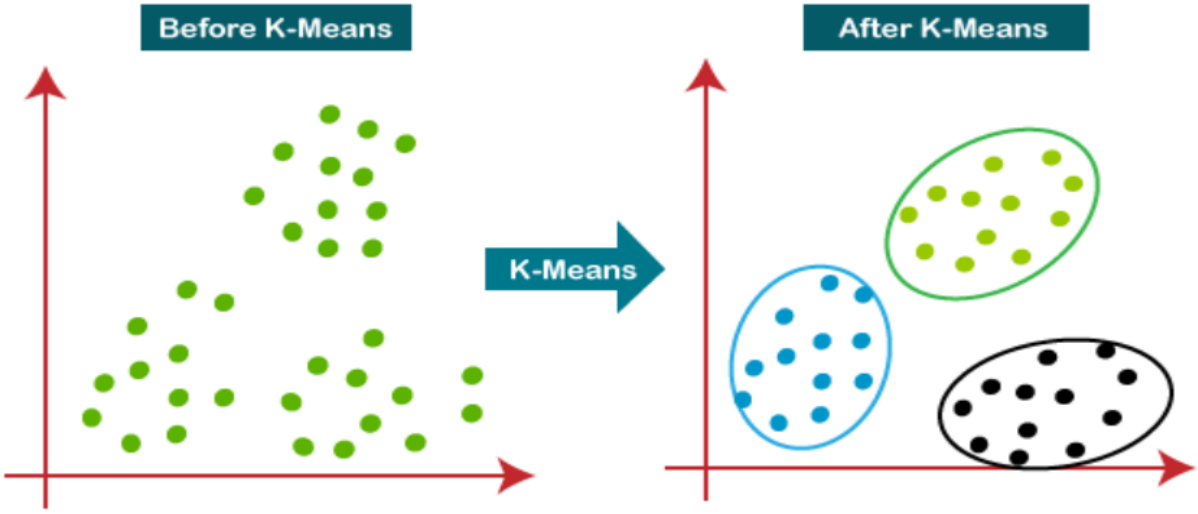


Fig 4.3.1 K-means clustering

5. SYSTEM METHODOLOGY

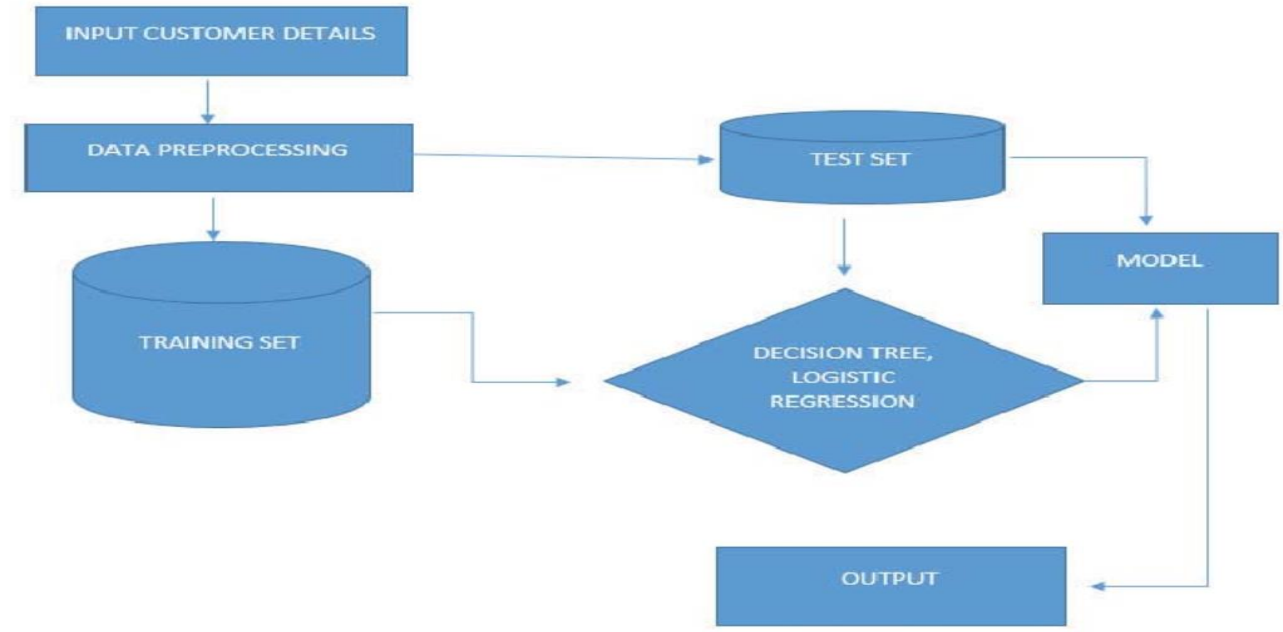


Fig 5.1 Flowchart

6. OVERVIEW OF TECHNOLOGIES

6.1 INTRODUCTION TO GOOGLE COLAB

Google Colab is a document that allows you to write, run, and share Python code within your browser. It is a version of the popular Jupyter Notebook within the Google suite of tools. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. It allows you to create a document containing executable code along with text, images, HTML, LaTeX, etc. which is then stored in your google drive and shareable to peers and colleagues for editing, commenting, and Viewing. Also it’s a free Jupyter notebook interactive development environment.

Fig 6.1 Google Colab logo

6.2 WHY GOOGLE COLAB

As our project is completely based on python and should be done together we've chosen google colab as the best working environment.

7. IMPLEMENTATION

7.1 CODING

1. Importing the libraries:
   import pandas as pd
   import matplotlib.pyplot as plot
   import seaborn as sb
2. Importing the dataset:
   train_df = pd.read_csv("/content/train_u6lujuX_CVtuZ9i.csv")
   print(train_df.info())
3. train_df = train_df.drop(columns= ["Loan_ID"])
4. Analyzing the dataset by graphs:
   By looking at the graphs, We will try to get some conclusions. This is a very important step. All having a data like this, We can
   1) generalise or remove the columns which doesnt have any significane on making the decision.
   2) remove the outliners. On large datasets, the computation can be decreased.
   - sb.countplot(x=train_df['Gender'], data=train_df, hue='Loan_Status')
   - sb.countplot(x=train_df['Married'], data=train_df, hue='Loan_Status')
   - sb.countplot(x=train_df['Dependents'], data=train_df, hue='Loan_Status')
   - sb.countplot(x=train_df['Education'], data=train_df, hue='Loan_Status')
   - sb.countplot(x=train_df['Self_Employed'], data=train_df, hue='Loan_Status')
   - sb.countplot(x=train_df['Credit_History'], data=train_df, hue='Loan_Status')
   - sb.countplot(x=train_df['Loan_Status'], data=train_df, hue='Loan_Status')
5. Preprocessing the data:
   The data needs to be pre processed to fit to the model. 2 reasons why this is mandatory is
   1) Missing values (Imputing)
   2) few columns have features in test format. these needs to be converted in number format (Encoding).
   - Convert categorical variable into dummy/indicator variables:
     train_df_encoded = pd.get_dummies(train_df,drop_first=True)
     print(train_df_encoded.head())
   - X = train_df_encoded.drop(columns='Loan_Status_Y')
     Y = train_df_encoded['Loan_Status_Y']
   - g = sb.lmplot(x='ApplicantIncome',y='LoanAmount',data= train_df_encoded , col='Self_Employed_Yes', hue='Gender_Male',palette= ["Red" , "Blue","Yellow"] ,aspect=1.2,size=6)
     g.set(ylim=(0, 800)
6. Correlation between all the numerical variables
   - import matplotlib.pyplot as plt
     matrix = train.corr()

```
f, ax = plt.subplots(figsize=(9,6))
sb.heatmap(matrix,vmax=.8,square=True,cmap='BuPu', annot = True)
```

7. Outliner Treatment:
    - Missing value
      ```
      train.isnull().sum()
      ```
    - ```
      train['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
      train['Married'].fillna(train['Married'].mode()[0], inplace=True)
      train['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
      train['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)
      train['LoanAmount'].fillna(train['LoanAmount'].mode()[0], inplace=True)
      train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0],inplace=True)
      train['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
      ```
    - ```
      train['Loan_Amount_Term'].value_counts()
      ```
    - ```
      train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace=True)
      ```
    - ```
      train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
      ```
    - ```
      train.isnull().sum()
      ```
    - ```
      import numpy as np
      train['LoanAmount_log']=np.log(train['LoanAmount'])
      train['LoanAmount_log'].hist(bins=20)
      test['LoanAmount_log']=np.log(test['LoanAmount'])
      ```
8. Train and Test split:
    - ```
      from sklearn.model_selection import train_test_split
      X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.20,stratify=Y)
      ```
    - ```
      from sklearn.impute import SimpleImputer
      temp = SimpleImputer(strategy='mean')
      X_train = temp.fit_transform(X_train)
      X_test = temp.fit_transform(X_test)
      ```
9. Linear Regression for Classification:
    - ```
      from sklearn.linear_model import LinearRegression
      lrc = LinearRegression()
      lrc.fit(X_train, Y_train)
      temp= lrc.predict(X_test)
      Y_pred_lrc = []
      ```
    - ```
      for i in temp:
        if i>0.5:
          Y_pred_lrc.append(1)
        else:
          Y_pred_lrc.append(0)
      ```
    - ```
      from sklearn.metrics import accuracy_score
      ```
      ```

result2 = accuracy_score(Y_test, Y_pred_lrc)
print("\nAccuracy:",result2)

10. Logistic Regression:
    - from sklearn.linear_model import LogisticRegression
      LR = LogisticRegression(solver='lbfgs')
      LR.fit(X_train, Y_train)
      Y_pred_LR = LR.predict(X_test)
    - from sklearn.metrics import accuracy_score
      result2 = accuracy_score(Y_test, Y_pred_LR)
      print("\nAccuracy:",result2)
11. Support Vector Machines(SVM):
    - from sklearn.svm import SVC
      SVM = SVC(kernel='linear')
      SVM.fit(X_train, Y_train)
      Y_pred_SVM = SVM.predict(X_test)
    - result2 = accuracy_score(Y_test, Y_pred_SVM)
      print("\nAccuracy:",result2)
12. Random Forest Classifier:
    - from sklearn.ensemble import RandomForestClassifier
      random_forest = RandomForestClassifier(n_estimators= 100)
      random_forest.fit(X_train, Y_train)
      Y_pred_rf = random_forest.predict(X_test)
    - result2 = accuracy_score(Y_test, Y_pred_rf)
      print("\nAccuracy:",result2)
13. Neural Network:
    - from keras.layers import Dropout
      from keras import regularizers
      from keras.callbacks import EarlyStopping
      from keras.models import Sequential
      from keras.layers import Dense
      model = Sequential()
      model.add(Dense(16 ,activation='relu',input_shape=(14,)))
      model.add(Dense(16,activation='relu'))
      model.add(Dropout(0.3))
      model.add(Dense(1,activation='sigmoid'))
      model.compile(loss='binary_crossentropy',optimizer='rmsprop',
      metrics=['accuracy'])
      hist = model.fit(X_train, Y_train,epochs=300,verbose=1, validation_split=0.1)
      accuracy = model.evaluate(X_test,Y_test)[1]
    - plot.plot(hist.history['loss'])
      plot.plot(hist.history['val_loss'])
      plot.title('Model loss')
      plot.ylabel('Loss')

```
plot.xlabel('Epoch')
plot.legend(['Train', 'Val'], loc='upper right')
plot.show()
plot.plot(hist.history['accuracy'])
plot.plot(hist.history['val_accuracy'])
plot.title('Model acc')
plot.ylabel('acc')
plot.xlabel('Epoch')
plot.legend(['Train', 'Val'], loc='upper right')
plot.show()
```

14. Unsupervised Learning - K means clustering:

- ```
  from sklearn.cluster import Kmeans
  wcss = []
  for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,random_state=0)
    kmeans.fit(X_train)
    wcss.append(kmeans.inertia_)
  plot.plot(range(1, 11), wcss)
  plot.title('Elbow Method')
  plot.xlabel('Number of clusters')
  plot.ylabel('WCSS')
  plot.show()
  kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=100, n_init=10,random_state=0)
  pred_y_kn = kmeans.fit_predict(X_train)
  result2 = accuracy_score(Y_train, pred_y_kn)
  print("\nAccuracy:",result2)
  ```

15. On Unseen Data By Logistic Regression:

- ```
  data1 = pd.read_csv('/content/test_Y3wMUE5_7gLdaTN.csv')
  data1 = data1.drop(columns=['Loan_ID'])
  data1_df_encoded = pd.get_dummies(data1,drop_first=True)
  data1_df_encoded.head()
  imp1 = SimpleImputer(strategy='mean')
  imp1_train = imp1.fit(data1_df_encoded)
  data1_df_encoded = imp1_train.transform(data1_df_encoded)
  predict= LR.predict(data1_df_encoded)
  data1['Y/N']= predict
  data1.to_csv('predicted.csv')
  ```

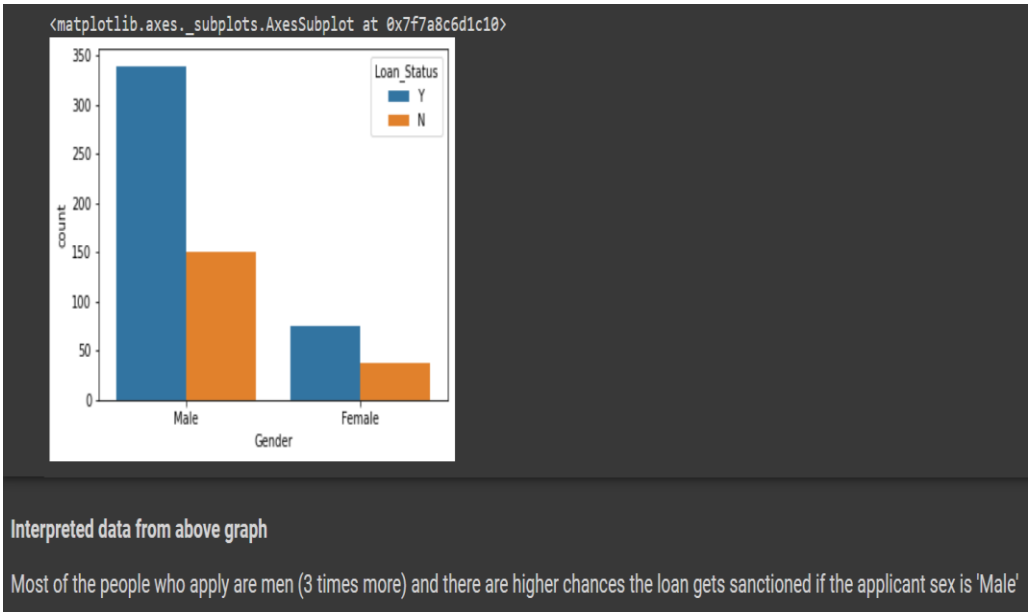
8. RESULTS AND DISCUSSION

1. Importing datasets:

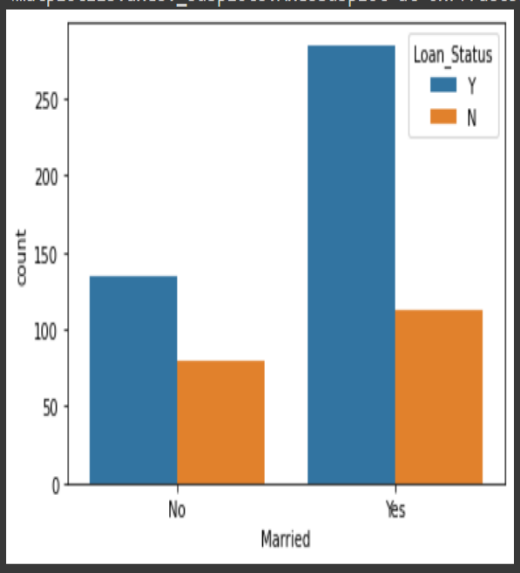
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
None
```

We can clearly see there are 13 columns and there are missing data points in the dataset.

2. Analyzing Dataset by Graph:



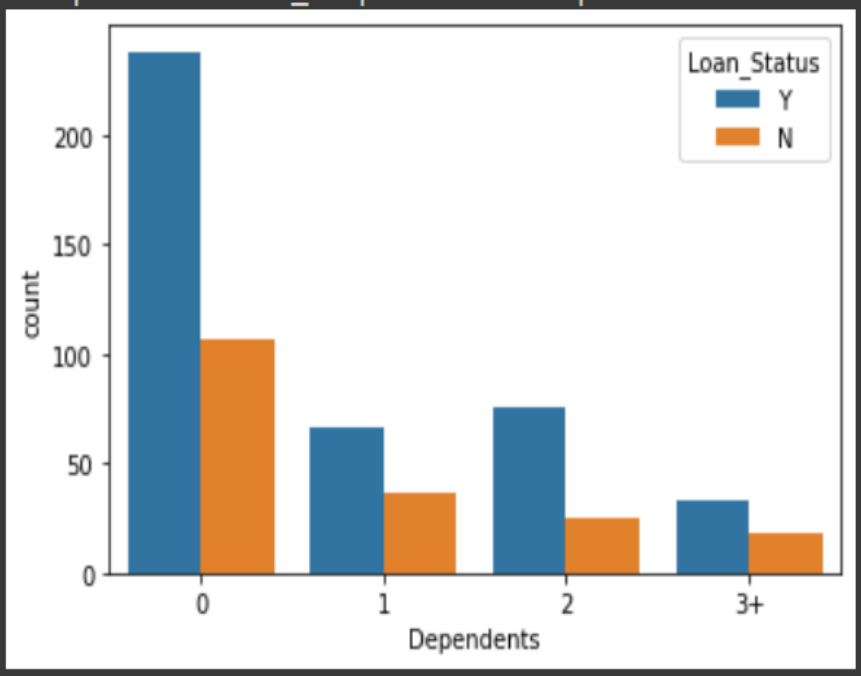
- <matplotlib.axes._subplots.AxesSubplot at 0x7f7a8c5b2e10>



Interpreted data from above graph

2/3 rd of the people who applied for loan are married and married applicants are more likely to get loan sanctioned

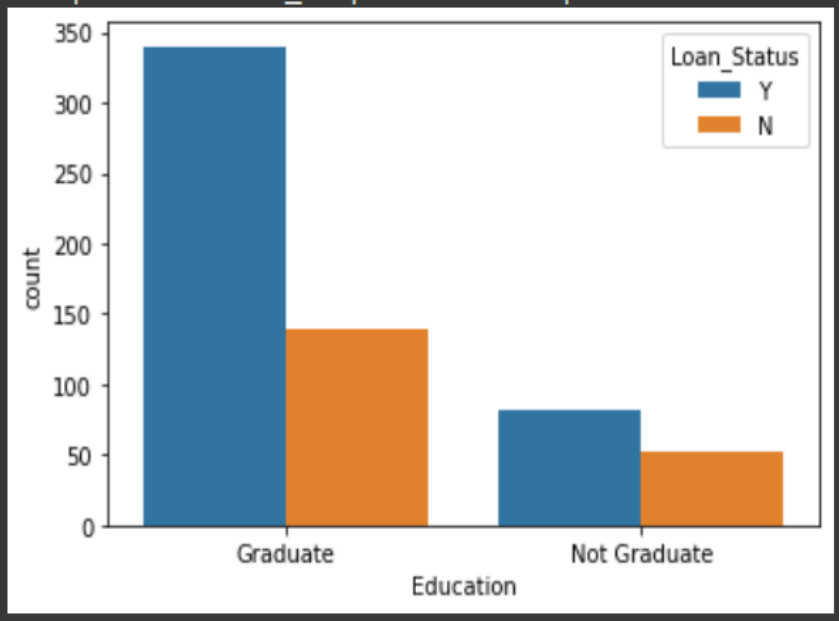
- <matplotlib.axes._subplots.AxesSubplot at 0x7f7a8c0fb6d0>



Interpreted data from above graph

If the dependants are 0, there is highly likely that loan is approved

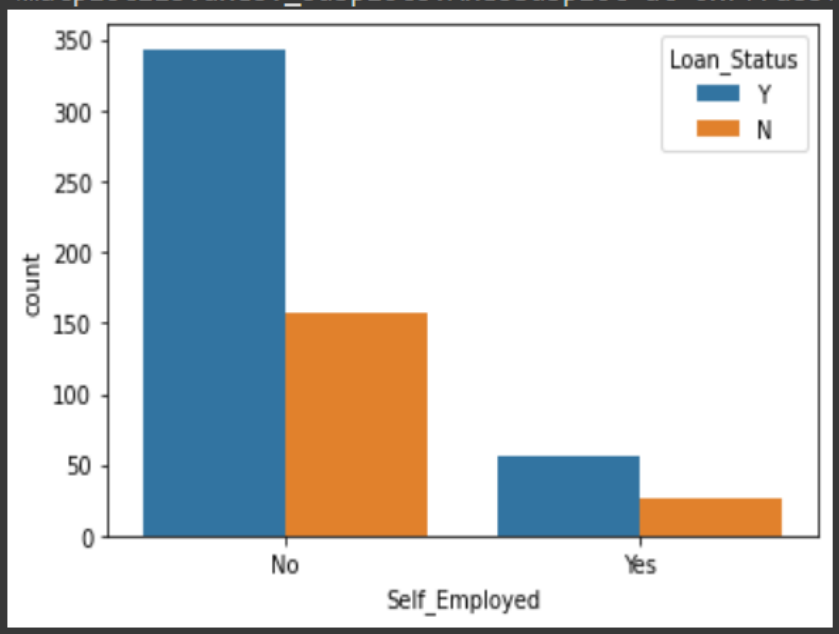
- ```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7a8c0d72d0>
```




**Interpreted data from above graph**

Graduated population are more likely to get the loan

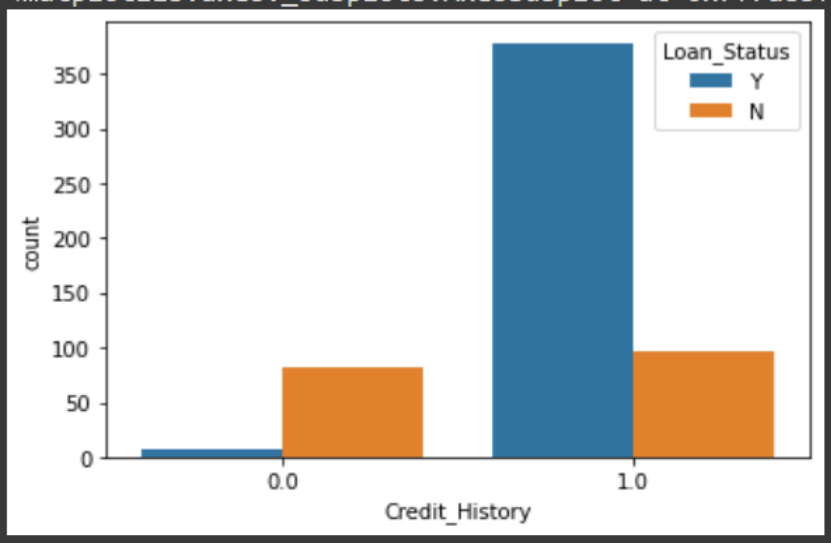
- ```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7a8bfce9d0>
```



Interpreted data from above graph

5/6th of population is not self employed

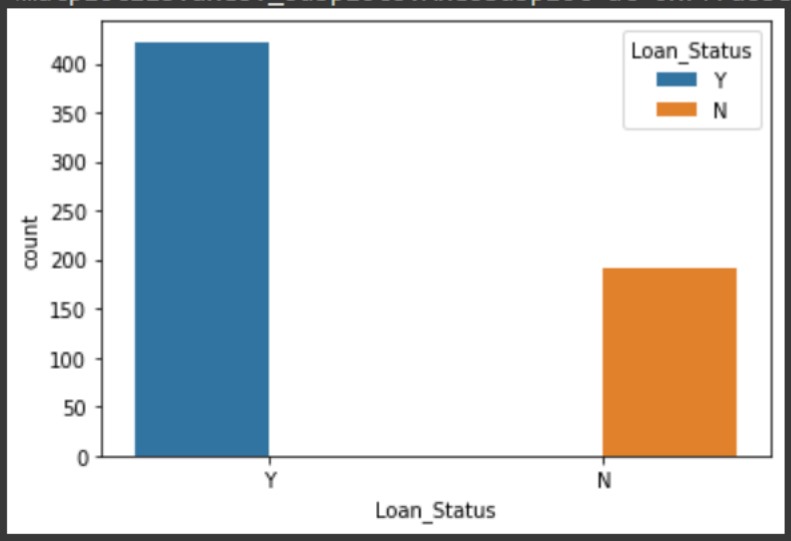
<matplotlib.axes._subplots.AxesSubplot at 0x7f7a8bf5c310>



Interpreted data from above graph

- Applicant with credit history are far more likely to be accepted.

<matplotlib.axes._subplots.AxesSubplot at 0x7f7a8bed02d0>



Interpreted data from above graph

- We can conclude, 2/3 of people who applied got there loan sanctioned.

3. Preprocessing Data:

-

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	\
0	5849	0.0	NaN	360.0	
1	4583	1508.0	128.0	360.0	
2	3000	0.0	66.0	360.0	
3	2583	2358.0	120.0	360.0	
4	6000	0.0	141.0	360.0	

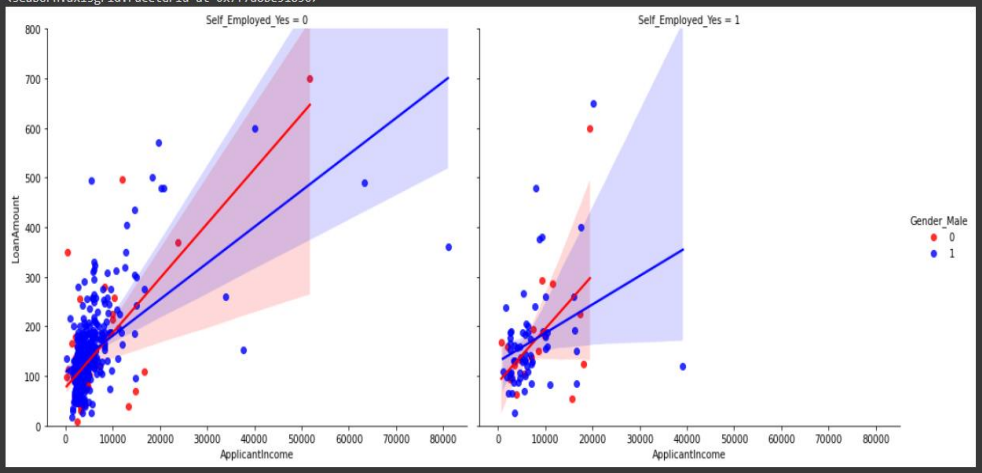
	Credit_History	Gender_Male	Married_Yes	Dependents_1	Dependents_2	\
0	1.0	1	0	0	0	
1	1.0	1	1	1	0	
2	1.0	1	1	0	0	
3	1.0	1	1	0	0	
4	1.0	1	0	0	0	

	Dependents_3+	Education_Not Graduate	Self_Employed_Yes	\
0	0	0	0	
1	0	0	0	
2	0	0	1	
3	0	1	0	
4	0	0	0	

	Property_Area_Semiurban	Property_Area_Urban	Loan_Status_Y
0	0	1	1
1	0	0	0
2	0	1	1
3	0	1	1
4	0	1	1

-

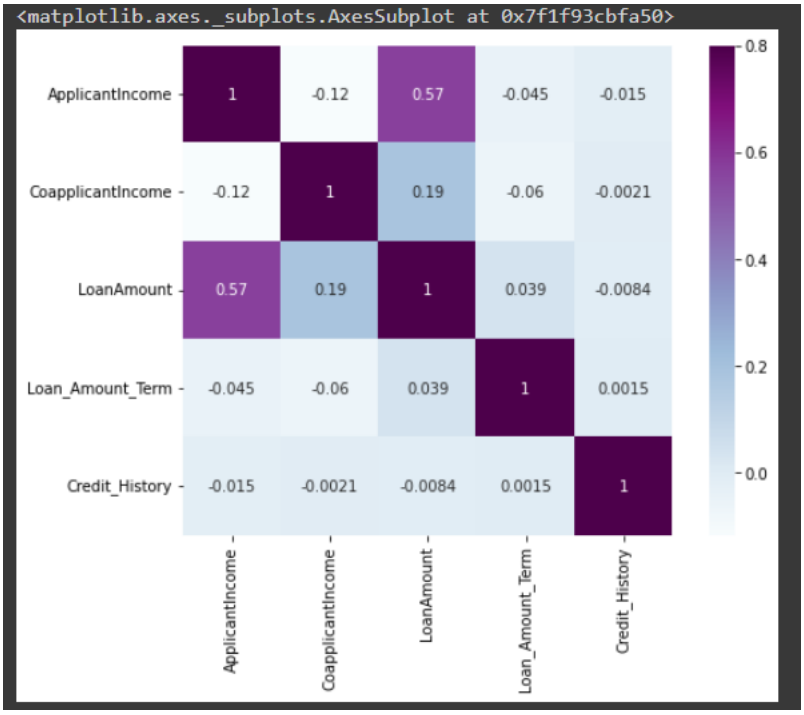
```
[ ] /usr/local/lib/python3.7/dist-packages/seaborn/regression.py:581: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.FacetGrid at 0x7f7a8be51850>
```



Above graph tells:

- The male applicants take more amount of loan than female.
- The males are higher in number of "NOT self employed" category.
- Majority of applicants are NOT self employed.
- The majority of income taken is about 0-200 with income in the range 0-20000.

4. Correlation between all the numerical variables:



5. Outlier Treatment:
   - Missing value

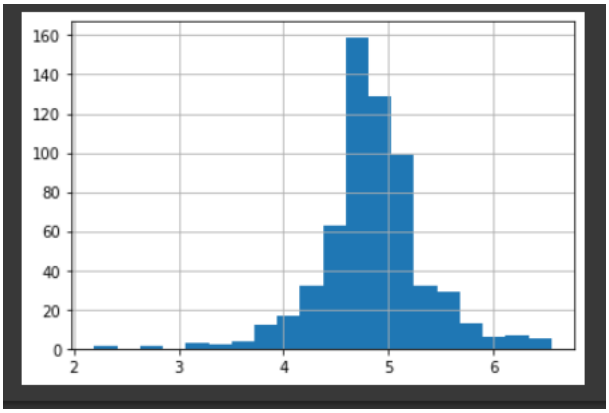
```
Loan_ID                0
Gender                13
Married                3
Dependents            15
Education              0
Self_Employed         32
ApplicantIncome        0
CoapplicantIncome      0
LoanAmount            22
Loan_Amount_Term      14
Credit_History        50
Property_Area          0
Loan_Status            0
LoanAmount_log        22
dtype: int64
```

   - Value counts

```
360.0    526
180.0     44
480.0     15
300.0     13
240.0      4
84.0       4
120.0      3
60.0       2
36.0       2
12.0       1
Name: Loan_Amount_Term, dtype: int64
```

   - Sum

```
Loan_ID                0
Gender                 0
Married                0
Dependents             0
Education              0
Self_Employed          0
ApplicantIncome        0
CoapplicantIncome      0
LoanAmount             0
Loan_Amount_Term       0
Credit_History         0
Property_Area          0
Loan_Status            0
LoanAmount_log        22
dtype: int64
```

- 

6. Linear Regression:
    - Accuracy score

```
Accuracy: 0.7967479674796748
```

7. Logistic Regression:
    - Accuracy score

```
Accuracy: 0.7804878048780488
```

8. Support Vector Machines:
    - Accuracy Score

```
Accuracy: 0.8130081300813008
```

9. Random Forest Classifier:
    - Accuracy Score

```
Accuracy: 0.8292682926829268
```

10. Neural Network:

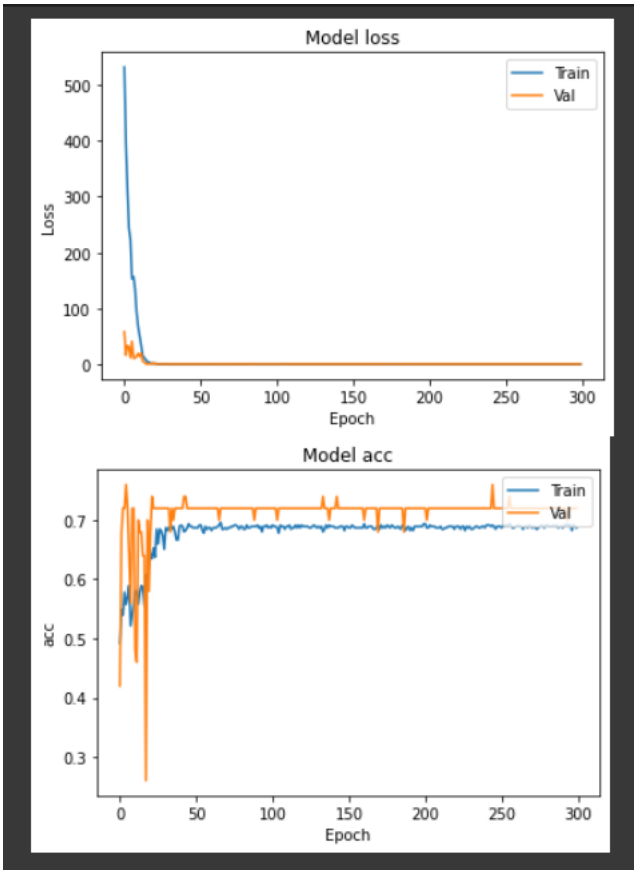
```
Epoch 1/300
14/14 [==============================] - 1s 33ms/step - loss: 644.9342 - accuracy: 0.4702 - val_loss: 58.2916 - val_accuracy: 0.4200
Epoch 2/300
14/14 [==============================] - 0s 5ms/step - loss: 404.0068 - accuracy: 0.5505 - val_loss: 16.4476 - val_accuracy: 0.6800
Epoch 3/300
14/14 [==============================] - 0s 4ms/step - loss: 346.9683 - accuracy: 0.5190 - val_loss: 33.3986 - val_accuracy: 0.7200
Epoch 4/300
14/14 [==============================] - 0s 5ms/step - loss: 220.5878 - accuracy: 0.5785 - val_loss: 31.5050 - val_accuracy: 0.7200
Epoch 5/300
14/14 [==============================] - 0s 5ms/step - loss: 237.9586 - accuracy: 0.5766 - val_loss: 12.1917 - val_accuracy: 0.7600
Epoch 6/300
14/14 [==============================] - 0s 5ms/step - loss: 162.9829 - accuracy: 0.5846 - val_loss: 41.3104 - val_accuracy: 0.7200
Epoch 7/300
14/14 [==============================] - 0s 5ms/step - loss: 220.7441 - accuracy: 0.5648 - val_loss: 11.1583 - val_accuracy: 0.6400
Epoch 8/300
14/14 [==============================] - 0s 5ms/step - loss: 140.3237 - accuracy: 0.4872 - val_loss: 11.3351 - val_accuracy: 0.5600
Epoch 9/300
14/14 [==============================] - 0s 5ms/step - loss: 93.8898 - accuracy: 0.5414 - val_loss: 14.8922 - val_accuracy: 0.7200
Epoch 10/300
14/14 [==============================] - 0s 5ms/step - loss: 69.0973 - accuracy: 0.5542 - val_loss: 19.6087 - val_accuracy: 0.7200
Epoch 11/300
14/14 [==============================] - 0s 4ms/step - loss: 53.5854 - accuracy: 0.5877 - val_loss: 14.0410 - val_accuracy: 0.4800
Epoch 12/300
14/14 [==============================] - 0s 5ms/step - loss: 41.5712 - accuracy: 0.5376 - val_loss: 20.0064 - val_accuracy: 0.4600
Epoch 13/300
14/14 [==============================] - 0s 5ms/step - loss: 16.7584 - accuracy: 0.5486 - val_loss: 5.8612 - val_accuracy: 0.7000
Epoch 14/300
14/14 [==============================] - 0s 6ms/step - loss: 15.7708 - accuracy: 0.5575 - val_loss: 4.4142 - val_accuracy: 0.6800
Epoch 15/300
14/14 [==============================] - 0s 5ms/step - loss: 9.6603 - accuracy: 0.6198 - val_loss: 1.1583 - val_accuracy: 0.6800
Epoch 16/300
14/14 [==============================] - 0s 5ms/step - loss: 6.0415 - accuracy: 0.6007 - val_loss: 0.7973 - val_accuracy: 0.6400
Epoch 17/300
14/14 [==============================] - 0s 5ms/step - loss: 4.6859 - accuracy: 0.5343 - val_loss: 0.7763 - val_accuracy: 0.6400
Epoch 18/300
14/14 [==============================] - 0s 5ms/step - loss: 2.0930 - accuracy: 0.6140 - val_loss: 1.1261 - val_accuracy: 0.2600
Epoch 19/300
14/14 [==============================] - 0s 4ms/step - loss: 1.8011 - accuracy: 0.5672 - val_loss: 0.7443 - val_accuracy: 0.7000
```

-

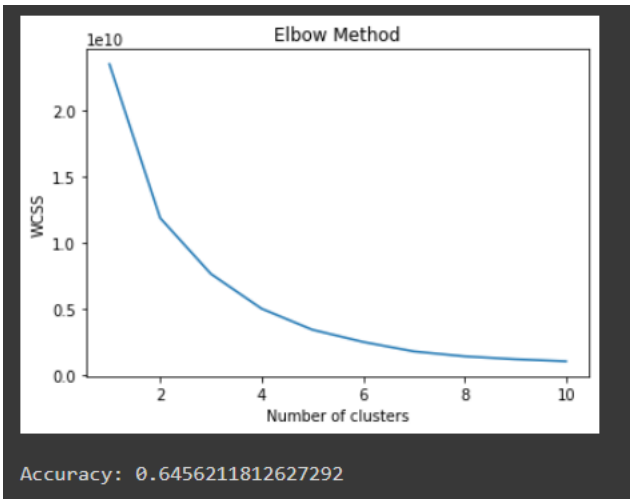
```
Epoch 21/300
14/14 [==============================] - 0s 5ms/step - loss: 1.0504 - accuracy: 0.6945 - val_loss: 0.8381 - val_accuracy: 0.6800
Epoch 22/300
14/14 [==============================] - 0s 5ms/step - loss: 1.3423 - accuracy: 0.6185 - val_loss: 0.6476 - val_accuracy: 0.7400
Epoch 23/300
14/14 [==============================] - 0s 5ms/step - loss: 0.7064 - accuracy: 0.6756 - val_loss: 0.5721 - val_accuracy: 0.7200
Epoch 24/300
14/14 [==============================] - 0s 5ms/step - loss: 0.7778 - accuracy: 0.6294 - val_loss: 0.5952 - val_accuracy: 0.7200
Epoch 25/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6460 - accuracy: 0.7008 - val_loss: 0.6127 - val_accuracy: 0.7200
Epoch 26/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6938 - accuracy: 0.6740 - val_loss: 0.6362 - val_accuracy: 0.7200
Epoch 27/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6480 - accuracy: 0.7034 - val_loss: 0.6192 - val_accuracy: 0.7200
Epoch 28/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6972 - accuracy: 0.6610 - val_loss: 0.6254 - val_accuracy: 0.7200
Epoch 29/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6556 - accuracy: 0.6849 - val_loss: 0.6353 - val_accuracy: 0.7200
Epoch 30/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6995 - accuracy: 0.6832 - val_loss: 0.6314 - val_accuracy: 0.7200
Epoch 31/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6586 - accuracy: 0.6883 - val_loss: 0.6360 - val_accuracy: 0.7200
Epoch 32/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6494 - accuracy: 0.6942 - val_loss: 0.6377 - val_accuracy: 0.7200
Epoch 33/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6511 - accuracy: 0.7105 - val_loss: 0.6438 - val_accuracy: 0.7200
Epoch 34/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6542 - accuracy: 0.6773 - val_loss: 0.6619 - val_accuracy: 0.6800
Epoch 35/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6616 - accuracy: 0.6743 - val_loss: 0.6522 - val_accuracy: 0.7200
Epoch 36/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6625 - accuracy: 0.6656 - val_loss: 0.6558 - val_accuracy: 0.7000
Epoch 37/300
14/14 [==============================] - 0s 4ms/step - loss: 0.6709 - accuracy: 0.6669 - val_loss: 0.6359 - val_accuracy: 0.7200
Epoch 38/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6661 - accuracy: 0.6774 - val_loss: 0.6396 - val_accuracy: 0.7200
Epoch 39/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6670 - accuracy: 0.6657 - val_loss: 0.6422 - val_accuracy: 0.7200
```

```
Epoch 41/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6513 - accuracy: 0.7097 - val_loss: 0.6378 - val_accuracy: 0.7200
Epoch 42/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6408 - accuracy: 0.6997 - val_loss: 0.6360 - val_accuracy: 0.7200
Epoch 43/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6583 - accuracy: 0.6923 - val_loss: 0.6296 - val_accuracy: 0.7400
Epoch 44/300
14/14 [==============================] - 0s 5ms/step - loss: 1.0062 - accuracy: 0.6924 - val_loss: 0.6322 - val_accuracy: 0.7400
Epoch 45/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6611 - accuracy: 0.6507 - val_loss: 0.6335 - val_accuracy: 0.7200
Epoch 46/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6384 - accuracy: 0.6773 - val_loss: 0.6316 - val_accuracy: 0.7200
Epoch 47/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6418 - accuracy: 0.6795 - val_loss: 0.6311 - val_accuracy: 0.7200
Epoch 48/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6518 - accuracy: 0.6896 - val_loss: 0.6294 - val_accuracy: 0.7200
Epoch 49/300
14/14 [==============================] - 0s 4ms/step - loss: 0.6399 - accuracy: 0.6943 - val_loss: 0.6273 - val_accuracy: 0.7200
Epoch 50/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6693 - accuracy: 0.6863 - val_loss: 0.6270 - val_accuracy: 0.7200
Epoch 51/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6669 - accuracy: 0.6383 - val_loss: 0.6271 - val_accuracy: 0.7200
Epoch 52/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6301 - accuracy: 0.6908 - val_loss: 0.6247 - val_accuracy: 0.7200
Epoch 53/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6253 - accuracy: 0.7123 - val_loss: 0.6233 - val_accuracy: 0.7200
Epoch 54/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6346 - accuracy: 0.6878 - val_loss: 0.6205 - val_accuracy: 0.7200
Epoch 55/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6530 - accuracy: 0.6865 - val_loss: 0.6217 - val_accuracy: 0.7200
Epoch 56/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6542 - accuracy: 0.6941 - val_loss: 0.6208 - val_accuracy: 0.7200
Epoch 57/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6239 - accuracy: 0.6997 - val_loss: 0.6207 - val_accuracy: 0.7200
Epoch 58/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6800 - accuracy: 0.6504 - val_loss: 0.6191 - val_accuracy: 0.7200
Epoch 59/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6365 - accuracy: 0.6729 - val_loss: 0.6183 - val_accuracy: 0.7200
Epoch 60/300
```

```
Epoch 282/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6158 - accuracy: 0.6879 - val_loss: 0.6061 - val_accuracy: 0.7200
Epoch 283/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6155 - accuracy: 0.6871 - val_loss: 0.6033 - val_accuracy: 0.7200
Epoch 284/300
14/14 [==============================] - 0s 6ms/step - loss: 0.5874 - accuracy: 0.7042 - val_loss: 0.6050 - val_accuracy: 0.7200
Epoch 285/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6200 - accuracy: 0.6759 - val_loss: 0.6051 - val_accuracy: 0.7200
Epoch 286/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6233 - accuracy: 0.6871 - val_loss: 0.6039 - val_accuracy: 0.7200
Epoch 287/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6204 - accuracy: 0.6764 - val_loss: 0.6066 - val_accuracy: 0.7200
Epoch 288/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6213 - accuracy: 0.7009 - val_loss: 0.6056 - val_accuracy: 0.7200
Epoch 289/300
14/14 [==============================] - 0s 5ms/step - loss: 0.5690 - accuracy: 0.7238 - val_loss: 0.6050 - val_accuracy: 0.7200
Epoch 290/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6251 - accuracy: 0.6762 - val_loss: 0.6072 - val_accuracy: 0.7200
Epoch 291/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6079 - accuracy: 0.6877 - val_loss: 0.6049 - val_accuracy: 0.7200
Epoch 292/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6156 - accuracy: 0.6826 - val_loss: 0.6050 - val_accuracy: 0.7200
Epoch 293/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6006 - accuracy: 0.7004 - val_loss: 0.6151 - val_accuracy: 0.7200
Epoch 294/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6243 - accuracy: 0.6904 - val_loss: 0.7675 - val_accuracy: 0.7000
Epoch 295/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6176 - accuracy: 0.6894 - val_loss: 0.6046 - val_accuracy: 0.7200
Epoch 296/300
14/14 [==============================] - 0s 6ms/step - loss: 0.5998 - accuracy: 0.6997 - val_loss: 0.6057 - val_accuracy: 0.7200
Epoch 297/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6050 - accuracy: 0.6892 - val_loss: 0.6052 - val_accuracy: 0.7200
Epoch 298/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6226 - accuracy: 0.6617 - val_loss: 0.6322 - val_accuracy: 0.7200
Epoch 299/300
14/14 [==============================] - 0s 5ms/step - loss: 0.6366 - accuracy: 0.6823 - val_loss: 0.6047 - val_accuracy: 0.7200
Epoch 300/300
14/14 [==============================] - 0s 6ms/step - loss: 0.6151 - accuracy: 0.6807 - val_loss: 0.6063 - val_accuracy: 0.7200
4/4 [==============================] - 0s 4ms/step - loss: 0.6256 - accuracy: 0.6911
```

- 

11. Un-Supervised Learning-K Means Clustering:
    - Accuracy



9. CONCLUSION AND FUTURE SCOPE

The process of prediction starts from cleaning and processing of data, imputation of missing values, experimental analysis of data set and then model building to evaluation of model and testing on test data. On the Data set, the best case accuracy obtained on the original data set is 0.811. The following conclusions are reached after analysis that those applicants whose credit score was worst will fail to get loan approval, due to a higher probability of not paying back the loan amount. Most of the time, those applicants who have high income and demands for a lower amount of loan are more likely to get approved which makes sense, more likely to pay back their loans. Some other characteristics like gender and marital status seem not to be taken into consideration by the company.

10. REFERENCES

A. JOURNALS

1. Nikhil Madane, SiddharthNanda,"LoanPredictionusingDecisiontree", Journal of the Gujarat Research History**, Volume 21 Issue 14s, December 2019.**

2. X.Frencis Jensy, V.P.Sumathi, Janani Shiva Shri, "An exploratory Data Analysis for Loan Prediction based on nature of clients", International Journal of Recent Technology and Engineering (IJRT E)**, Volume-7 Issue-4S, November2018**

3. Amit Kumar Goel, Kalpana Batra, Poonam Phogat," Manage big data using optical networks", Journal of Statistics and Management Systems **"Volume 23,2020,Issue2, Taylor & Francis.**

4. Raj, J. S., & Ananthi, J. V., "Recurrent neural networks and nonlinear prediction in support vector machine" Journal of Soft Computing Paradigm (JSCP), **1(01), 33-40, 2019.**

5. Pidikiti Supriya, Myneedi Pavani, Nagarapu Saisushma,Namburi VimalaKumari, k Vikash,"LoanPrediction byusingMachine Learning Models",InternationalJournalofEngineeringand Techniques**.Volume 5 Issue 2, Mar-Apr2019**.

6. Computing (BIGCOMP), **pp.39–42. IEEE, 2015.**

B. CONFERENCES

7. PhilHyo Jin Do ,Ho-Jin Choi,"Sentiment analysis of real-life situations using loca- tion, people and time as contextual features," International Conference On BigData andSmart Computing (BIGCOMP), **pp.39–42. IEEE, 2015.**

8. K I Rahmani, M.A. Ansari, Amit Kumar Goel, " An Efficient Indexing Algorithm for CBIR,"IEEE-International Conference On Computational Intelligence & Communication Technology,**13-14 Feb2015.**

9. Aakanksha Saha, Tamara Denning, VivekSrikumar, Sneha Kumar Kasera."Secrets inSourceCode:ReducingFalsePositives usingMachine Learning", **2020 InternationalConference on Communication Systems & Networks (COMSNETS),2020.**

10. Gurlove Singh, Amit Kumar Goel ,"Face Detection and Recognition System using Digital Image Processing" , 2nd International conference on Innovative Mechanism for Industry Application **ICMIA 2020, 5 -7 March 2020, IEEEPublisher.**

C. BOOKS

11. Toby Segaran, “Programming Collective Intelligence: Building Smart Web2.0 Applications.”O’Reilly Media.
12. Drew Conway and John Myles White,” Machine Learning for Hackers: CaseStudies andAlgorithmstoGet youStarted,”O’Reilly Media.
13. Trevor Hastie, Robert Tibshirani, and Jerome Friedman ,”The Elements of Statistical Learning: Data Mining, Inference, and Prediction,” Springer ,Kindle
14. Bing Liu, “ Sentiment Analysis and Opinion Mining,” Morgan & Claypool Publishers, **May 2012.**
15. Shiyang Liao, Junbo Wang, Ruiyun Yu, Koichi Sato, and Zixue Cheng, “ CNN for situation understanding based on sentiment analysis of twitter data,”Procedia Computer Science,**111:376–381, 2017.CrossRef.**
16. Bing Liu, “ Sentiment Analysis: Mining Opinions, Sentiments, and Emotions,”CambridgeUniversityPress, **ISBN:978-1-107-01789-4.**