LOAN STATUS PREDICTION USING MACHINE LEARING

Abstract

The aim of this project is to predict the loan approval of applicants. Banking Industry always needs a more accurate predictive modelling system for many issues. Predicting credit defaulters is a difficult task for the banking industry. The loan status is one of the quality indicators of the loan. It doesn't show everything immediately, but it is a first step of the loan lending process. The loan status is used for creating a credit scoring model. The credit scoring model is used for accurate analysis of credit data to find defaulters and valid customers. The objective of this paper is to create a credit scoring model for credit data. Various machine learning techniques are used to develop the financial credit scoring model. In this project, we will train and test the data using machine learning algorithms like Logistic Regression, Random forest and Decision tree and then choose the model which provides the important information with the highest accuracy. It is used to predict the loan status in commercial banks using machine learning classifiers.

List of Figures

INDEX

1. INTRODUCTION

Distribution of the loans is the core business part of almost every banks. The main portion the bank's assets are directly coming from the profit earned from the loans distributed by the banks. The prime objective in banking environment is to invest their assets in safe hands where it is. Today many banks/financial companies approve loan after a regress process of verification and validation but still there is no surety whether the chosen applicant is the deserving right applicant out of all applicants. Through this system we can predict whether that particular applicant is safe or not and the whole process of validation of features is automated by machine learning technique. Loan Prediction is very helpful for employee of banks as well as for the applicant also. The aim of this project is to provide quick, immediate and easy way to choose the deserving applicants. It can provide special advantages to the bank. The Loan Prediction System can automatically calculate the weight of each features taking part in loan processing and on new test data same features are processed with respect to their associated weight. A time limit can be set for the applicant to check whether his/her loan can be sanctioned or not. Loan Prediction System allows jumping to specific application so that it can be check on priority basis. This Project is exclusively for the managing authority of Bank/finance company, whole process of prediction is done privately no stakeholders would be able to alter the processing. Result against particular Loan Id can be send to various department of banks so that they can take appropriate action on application. This helps all others department to carried out other formalities.

Scope:

The scope of this project is to predict the loan status of loan applicants using machine learning models like :

Logistic Regression

Random Forest

Decision Tree classifier

Among these three models, we'll select the model which predicts and gives the outcomes with best accuracy.

1.1 MACHINE LEARNING

Machine learning is the scientific field dealing with the ways in which machines learn from experience. For many scientists, the term "machine learning" is identical to the term "artificial intelligence", given that the possibility of learning is the main characteristic of an entity called intelligent in the broadest sense of the word. The purpose of machine learning is the construction of computer systems that can adapt and learn from their. A more detailed and formal definition of machine learning is A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. With the rise of Machine Learning approaches we have the ability to find a solution to this issue, we have developed a system using data mining which has the ability to predict whether the message is spam or not. This research has focused on developing a system based on three classification methods namely, Support Vector Machine, Logistic regression and Artificial Neural Network algorithms.

1.1.1 Supervised learning

In supervised learning, the system must "learn" inductively a function called target function, which is an expression of a model describing the data. The objective function is used to predict the value of a variable, called dependent

variable or output variable, from a set of variables, called independent variables or input variables or characteristics or features. The set of possible input values of the function, i.e. its domain, are called instances. Each case is described by a set of characteristics (attributes or features). A subset of all cases, for which the output variable value is known, is called training data or examples. In order to infer the best target function, the learning system, given a training set, takes into consideration alternative functions, called hypothesis and denoted by h. In supervised learning, there are two kinds of learning tasks: classification and regression. Classification models try to predict distinct classes, such as e.g. blood groups, while regression models predict numerical values. Some of the most common techniques are Decision Trees (DT), Rule Learning, and Instance Based Learning (IBL), such as k-Nearest Neighbours (k-NN), Genetic Algorithms (GA), Artificial Neural Networks (ANN), and Support Vector Machines (SVM).

1.1.2 unsupervised learning

In unsupervised learning, the system tries to discover the hidden structure of data or associations between variables. In that case, training data consists of instances without any corresponding labels. Association Rule Mining appeared much later than machine learning and is subject to greater influence from the research area of databases. Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. IV. Reinforcement Learning The term Reinforcement Learning is a general term given to a family of techniques, in which the system attempts to learn through direct interaction with the

environment so as to maximize some notion of cumulative reward. It is important to mention that the system has no prior knowledge about the behaviour of the environment and the only way to find out is through trial and failure (trial and error). Reinforcement learning is mainly applied to autonomous systems, due to its independence in relation to its environment.

1.1.3 Logistic regression in statistics

Logistic regression is a regression model where the dependent variable is categorical, namely binary dependent variable-that is, where it can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed using logistic regression. Many other medical scales used to assess severity of a patient have been developed using logistic regression. The technique can also be used in engineering, especially for predicting the probability of failure of a given process, system or product. It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription. In economics it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application is about to predict the likelihood of a homeowner defaulting on a mortgage. Conditional random fields, an extension of logistic regression to sequential data, are used in natural language processing.

1.1.4 Random forest

Are an ensemble learning method for classification and regression and other task that operates by constructing a multitude of decision tree at training time and outputting the class that is the mode of the classes or mean prediction of individual trees. The first algorithm for random decision forests was created by Tin Kam Ho using random subspace method. Ho established that to gain the accuracy it should over train where it can randomly restrict sensitive selected features of the given data.

1.1.5 Decision tree

Decision tree are a type of supervised machine learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decision or the final outcome and the decision nodes are where the data is split. The leaf node is labeled by attribute and each attribute is assigned by a target value. The highest information gain of all attribute id calculated first. It is a method commonly used for data mining.

1.2. Existing System

Till now loans are processed by various banks through pen and paperwork. When the large no of customers apply for bank loan these bank take lot of time to approve their loan. After approval of loan by the banks, there is no surety that the chosen applicant is capable of paying loan or not. Many banks use their own softwares for the loan approval. In existing system we use datamining algorithms for the loan approval; this is the old technique for the approval of loan. Mutiple data sets are combined and form a generalised datasets, and different machine

learning algorithms are applied to generate results. But these techniques are not up to the mark. Due to this huge banks are suffering from financial crises. To resolve this issue we introduce a new way for approval of loans.

1.3 Proposed system

In this system we will train and test the data using machine learning algorithms like Logistic Regression, Random forest and Decision tree and then choose the model which provides the important information with the highest accuracy. Steps involved are:

- Data collection
- Data preparation
- Choose a model
- Train model
- Evaluate the model
- Parameter tuning
- Make predictions

2. SYSTEM ANALYSIS

This System Analysis is closely related to requirements analysis. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made. This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

2.1 Functional Requirement Specification

1. Data Collection

● The quantity & quality of your data dictate how accurate our model is
● The outcome of this step is generally a representation of data which we will use for training
● Using pre-collected data, by way of datasets from Kaggle, UCI, etc., still fits into this step.

2. Data Preparation

● Wrangle data and prepare it for training
● Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.)

● Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data.

3. Choose a Model

● Different algorithms are for different tasks; choose the right one

4. Train the Model

● The goal of training is to answer a question or make a prediction correctly as often as possible
● Linear regression example: algorithm would need to learn values for m (or W) and b (x is input, y is output)
● Each iteration of process is a training step.

5. Evaluate the Model

● Uses some metric or combination of metrics to "measure" objective performance of model ● Test the model against previously unseen data
● This unseen data is meant to be somewhat representative of model performance in the real world, but still helps tune the model (as opposed to test data, which does not)
● Good train/evaluate split 80/20, 70/30, or similar, depending on domain, data availability, dataset particulars, etc.

6. Parameter Tuning

● This step refers to hyper-parameter tuning, which is an "art form" as opposed to a science ● Tune model parameters for improved performance
● Simple model hyper-parameters may include: number of training steps, learning rate, initialization values and distribution, etc.

7. Make Predictions

● Using further (test set) data which have, until this point, been withheld from the model (and for which class labels are known), are used to test the model; a better approximation of how the model will perform in the real world.

2.2 Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

2.3 Software Requirements:

Operating System : Microsoft Windows XP and later versions.
Languages : PYTHON,HTML,ML
IDE : Jupyter Notebook, VS Code

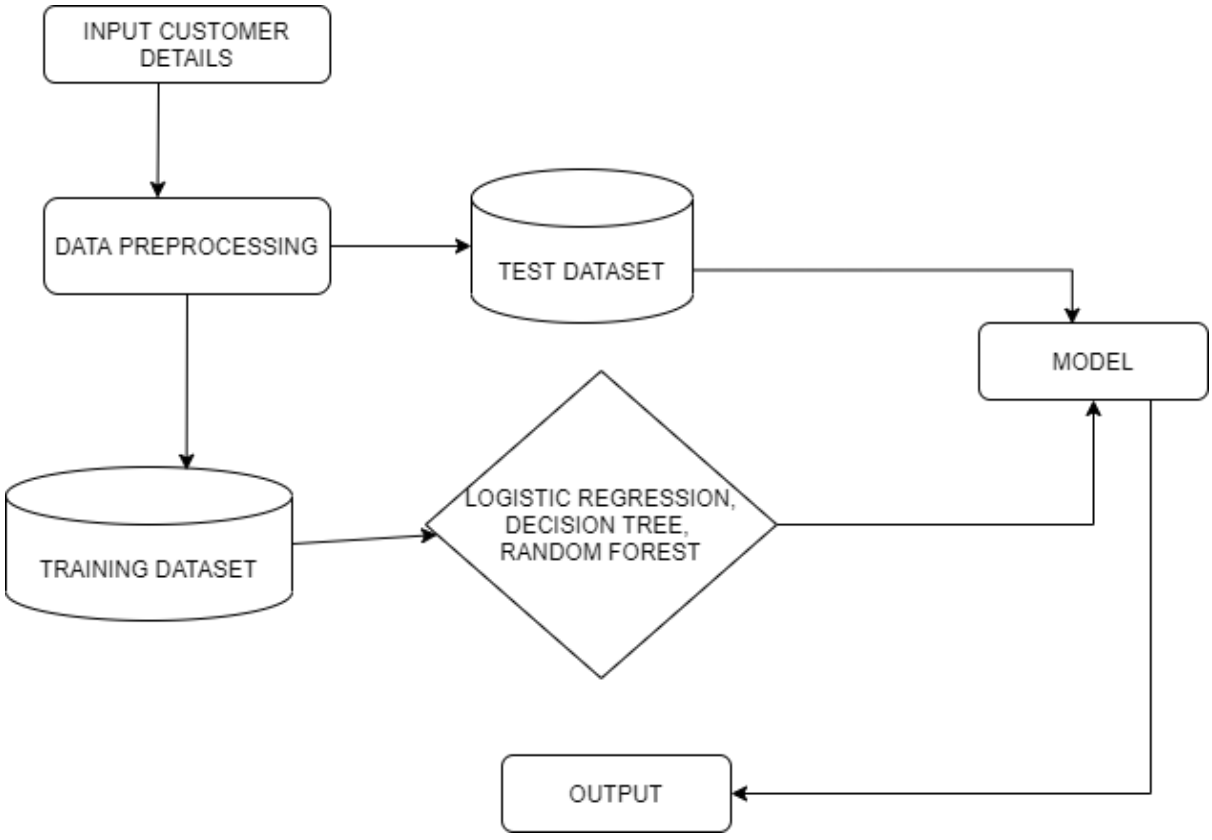
2.4 Hardware Requirements:

- Processor : Intel Dual Core Processor
- Hard Disk : 128 GB
- RAM : 4 GB or Higher

3. SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design.

3.1 Architectural Design



Implementation Details

Loan Dataset : Loan Dataset is very useful in our system for prediction of more accurate result. Using the loan Dataset the system will automatically predict which costumer's loan it should approve and which to reject. System will accept loan application form as an input. Justified format of application form should be given as an input to get processed.

Determine the training and testing data: Typically, Here the system separate a dataset into a training set and testing set, most of the data use for training, and a smaller portions of data is use for testing. after a system has been processed by using the training set, it makes the prediction against the test set.

Data cleaning and processing: In Data cleaning the system detect and correct corrupt or inaccurate records from database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying or detecting the dirty or coarse data. In Data processing the system convert data from a given form to a much more usable and desired form i.e. make it more meaningful and informative.

Models used :

1)Logistic Regression:

This is a classification algorithm which uses a logistic function to predict binary outcome (True/False, 0/1, Yes/No) given an independent variable. The aim of this model is to find a relationship between features and probability of particular outcome. The logistic function used is a logit function which is a log of odds in the favor of the event. Logit function develops a s-shaped curve with the probability estimate similar to a step function.

2)Decision Tree :

This is a supervised machine learning algorithm mostly used for classification problems. All features should be discretized in this model, so that the population can be split into two or more homogeneous sets or subsets. This model uses a different algorithm to split a node into two or more sub-nodes. With the creation of more sub-nodes, homogeneity and purity of the nodes increases with respect to the dependent variable.

3)Random Forest :

This is a tree based ensemble model which helps in improving the accuracy of the model . It combines a large number of Decision trees to build a powerful predicting model. It takes a random sample of rows and features of each individual tree to prepare a decision tree model. Final prediction class is either the mode of all the predictors or the mean of all the predictors.

3.2 Modules

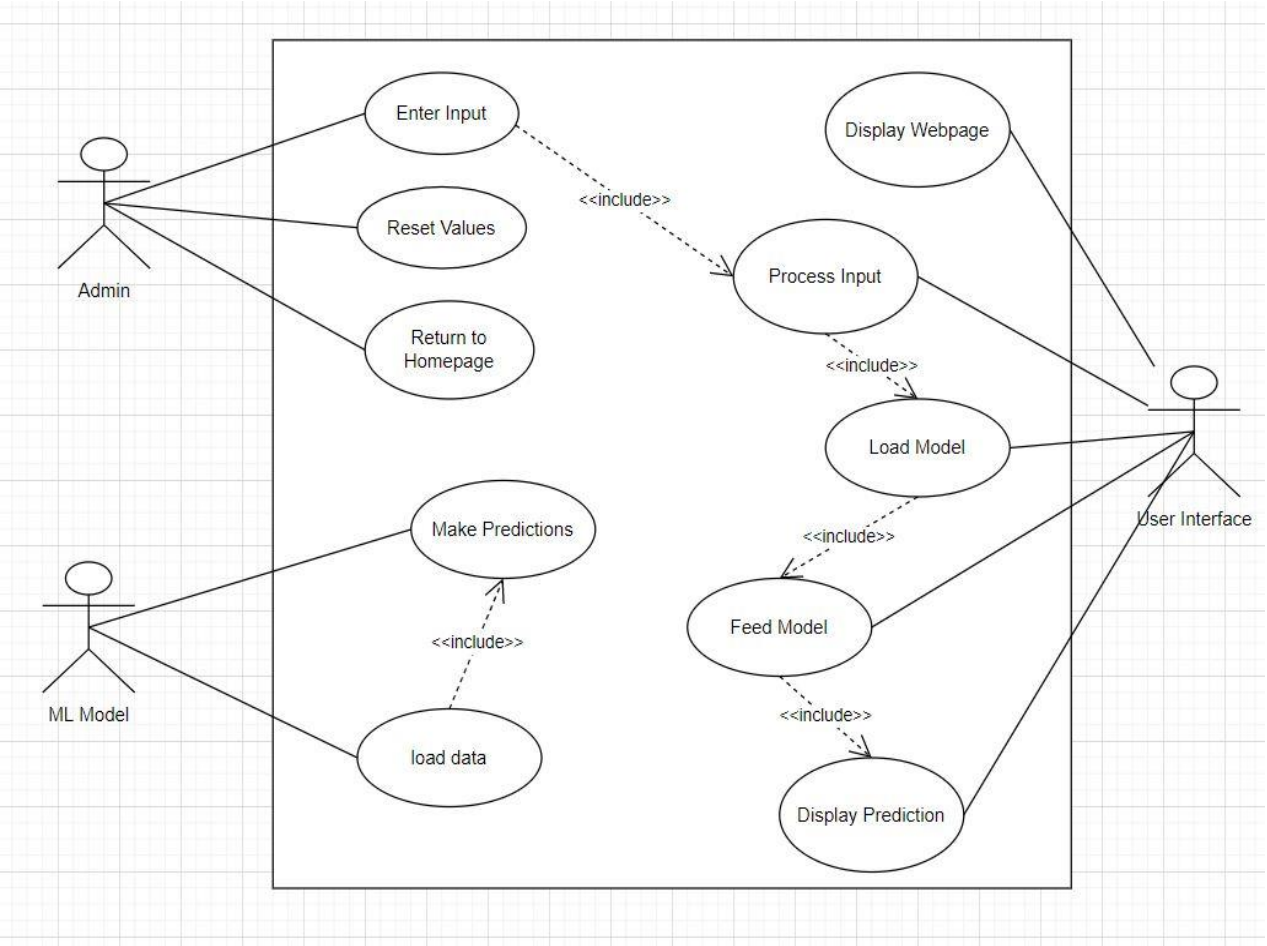
Numpy: NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

Pandas: Pandas is a Python package to work with structured and time series data.The data from various file formats such as csv, json, sql etc can be imported using Pandas. It is a powerful open source tool used for data analysis and data manipulation operations such as data cleaning, merging, selecting as well wrangling.

Seaborn: Seaborn is a python library for building graphs to visualise data. It provides integration with pandas. This open source tool helps in defining the data by mapping the data on the informative and interactive plots. Each element of the plots gives meaningful information about the data.

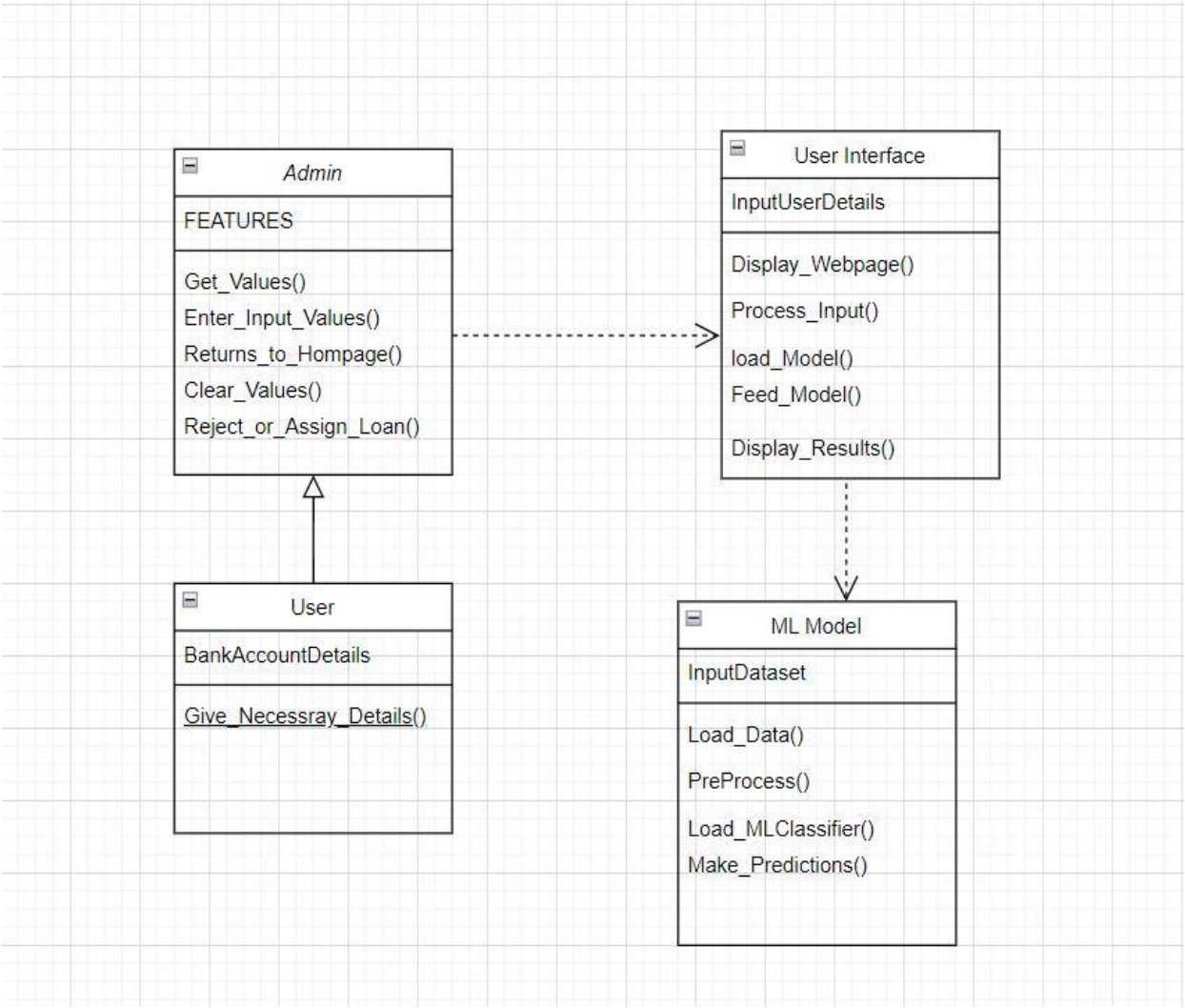
3.3 UML Diagrams

3.3.1. UseCase Diagram :



A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cass are represented by either circles or ellipses. Here the actors are Admin, ML Model and User Interface. The usecases depicts the functionalites of each actor. There will be relationships between the usecases, between the actors and between the usecases and actors.

3.3.2 Class Diagram:

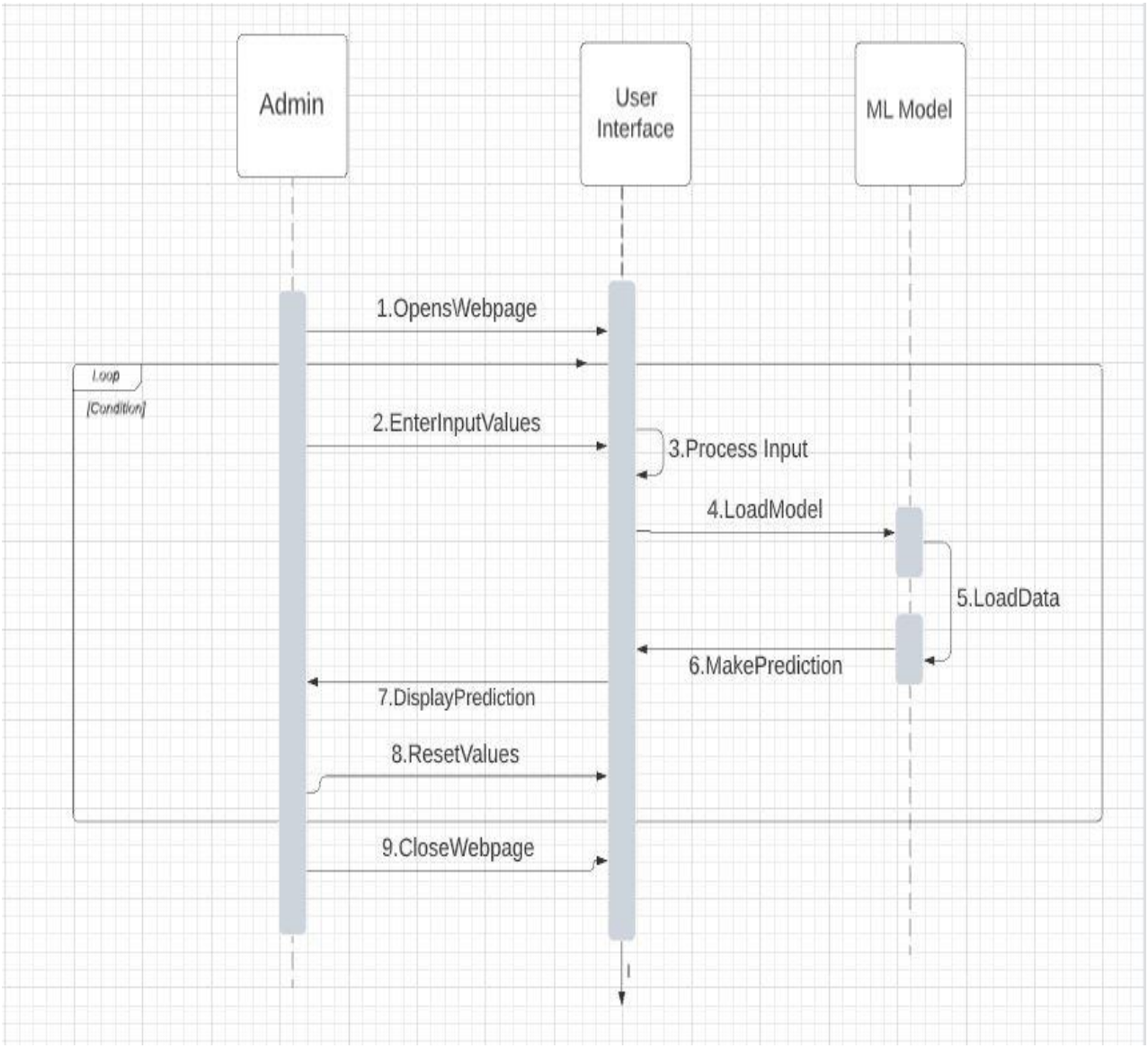


Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

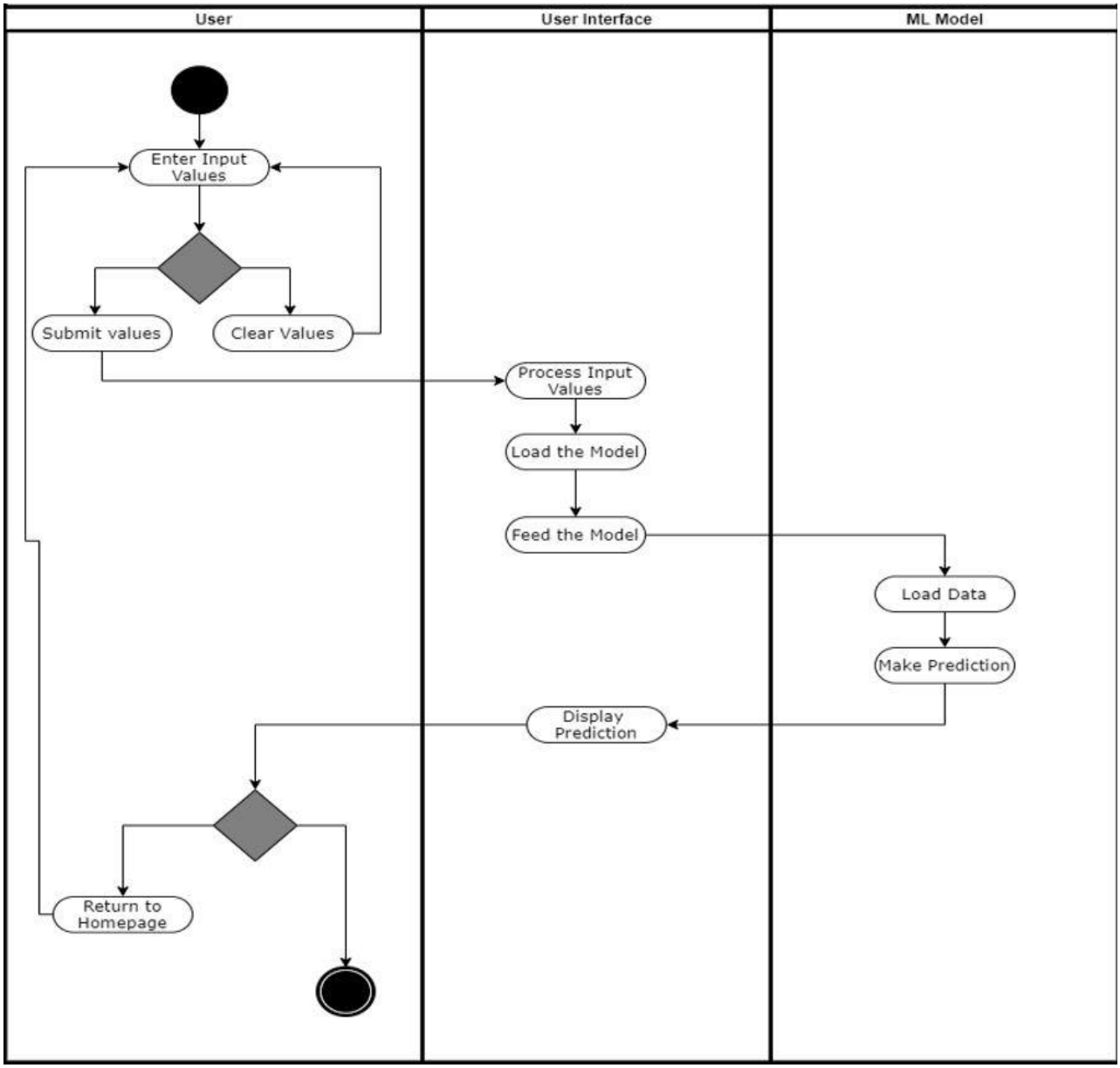
Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

3.3.3 Sequence Diagram:



A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

3.3.4 Activity Diagrams:



Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join. etc

4. SYSTEM IMPLEMENTATION

The implementation stage of any project is a true display of the defining moments that make a project a success or a failure. The implementation stage is defined as the system or system modifications being installed and made operational in a production environment. This phase is initiated after the system has been tested and accepted by the user. This phase continues until the system is operating in production in accordance with the defined user requirements.

4.1 DATASETS

● Here we have two datasets. First is train_dataset.csv, test_dataset.csv.
● These are datasets of loan approval applications which are featured with annual income, married or not, dependents are there or not, educated or not, credit history present or not, loan amount etc.
● The outcome of the dataset is represented by loan_status in the train dataset.
● This column is absent in test_dataset.csv as we need to assign loan status with the help of training dataset.

4.2 FEATURES PRESENT IN LOAN PREDICTION

● Loan_ID – The ID number generated by the bank which is giving loan
. ● Gender – Whether the person taking loan is male or female.
● Married – Whether the person is married or unmarried.
● Dependents – Family members who stay with the person.
● Education – Educational qualification of the person taking loan.

● Self_Employed – Whether the person is self-employed or not.

● ApplicantIncome – The basic salary or income of the applicant per month.

● CoapplicantIncome – The basic income or family members.

● LoanAmount – The amount of loan for which loan is applied.

● Loan_Amount_Term – How much time does the loan applicant take to pay the loan.

● Credit_History – Whether the loan applicant has taken loan previously from same bank.

● Property_Area – This is about the area where the person stays (Rural/Urban).

LABELS

● LOAN_STATUS – Based on the mentioned features, the machine learning algorithm decides whether the person should be give loan or not.

4.3 Visualizing data using google Colab

Code and output

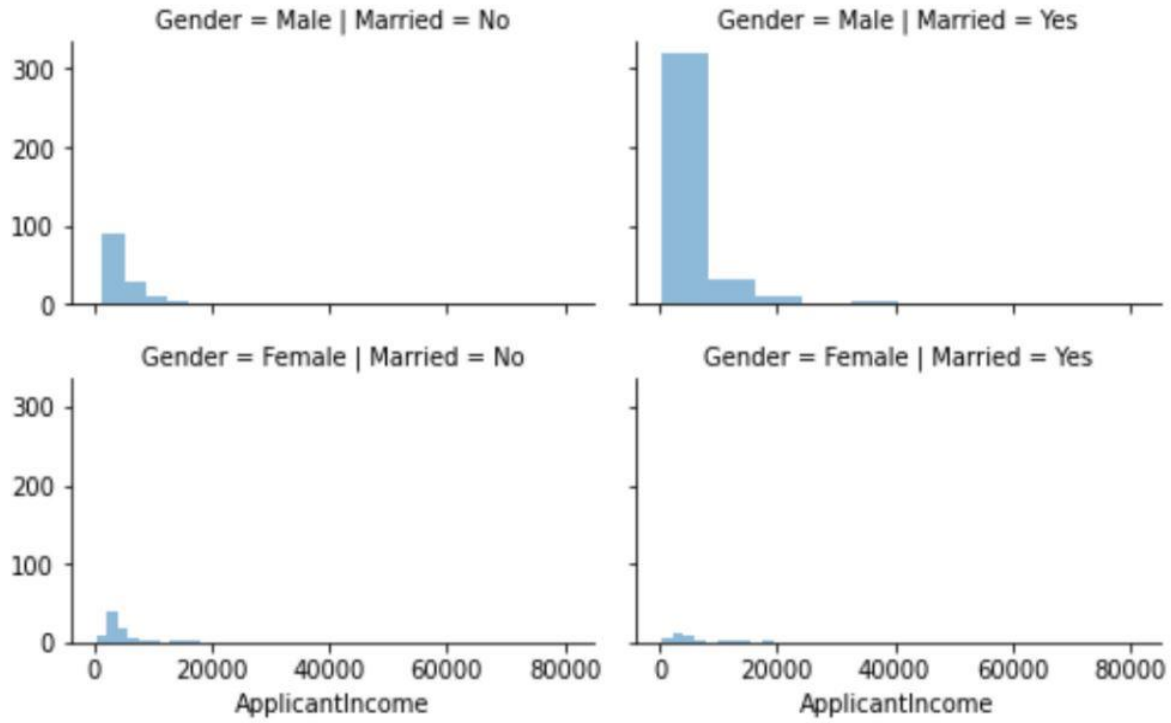
```
#Importing required libraries
import pandas as pd
import matplotlib.pyplot as plt import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

df_train = pd.read_csv('train_dataset.csv')
```

4.3.1 # take a look at the top 5 rows of the train set, notice the column "Loan_Status" df_train.head()

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

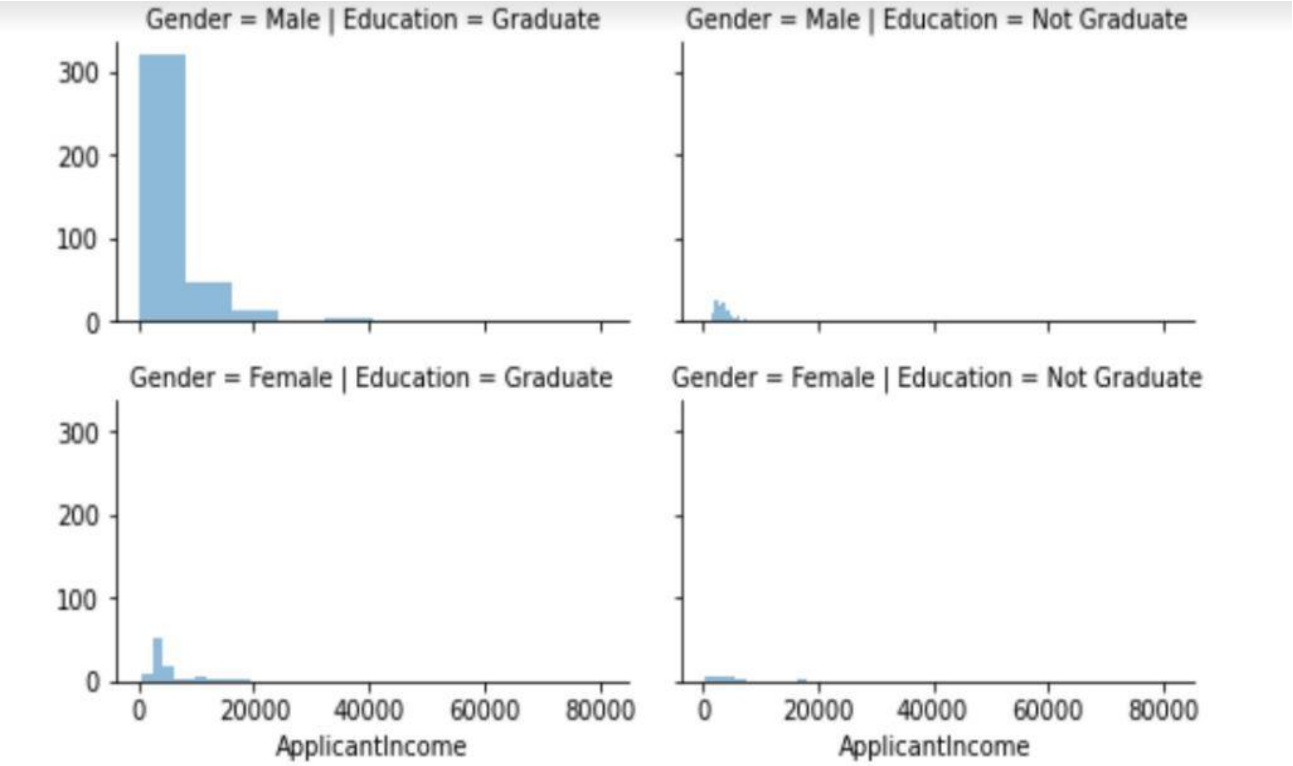
4.3.2 # This code visualizes the people applying for loan who are categorized based on gender and marriage

grid = sns.FacetGrid(df_train, row='Gender', col='Married', size=2.2, aspect=1.6) grid.map(plt.hist, 'ApplicantIncome', alpha=.5, bins=10) grid.add_legend()



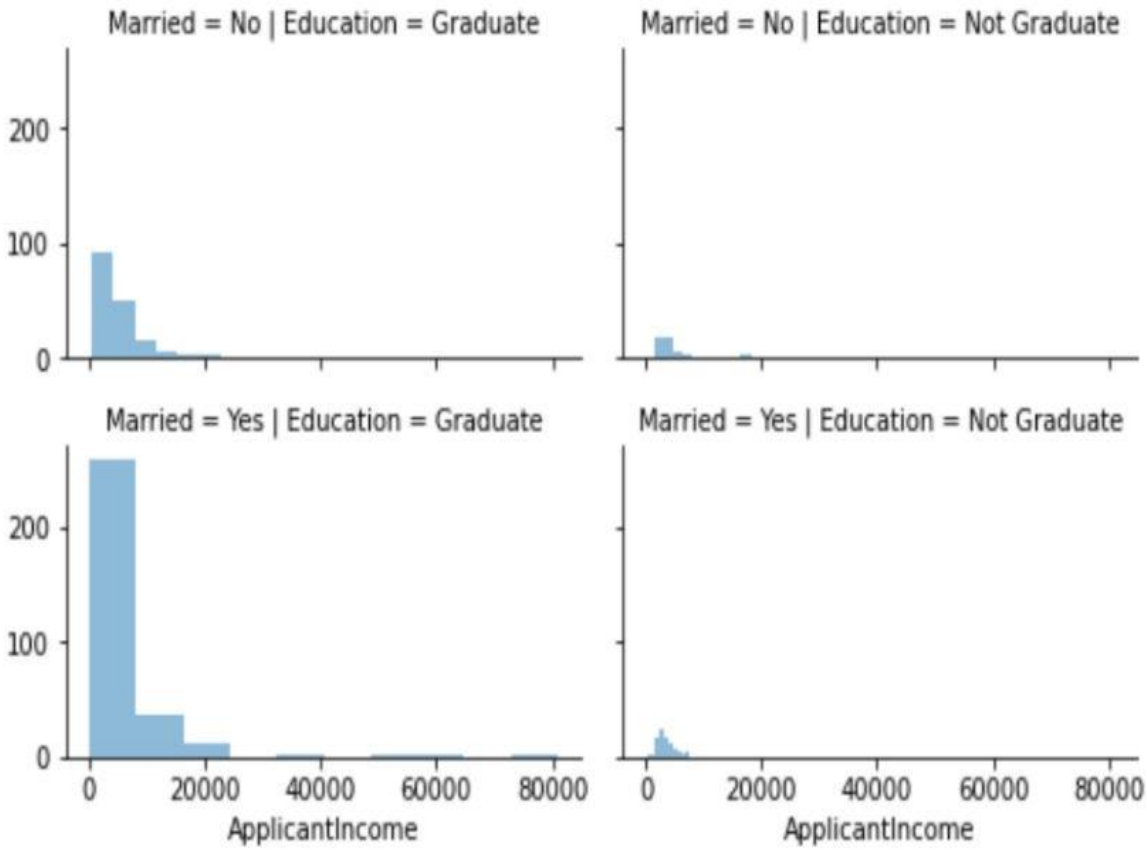
4.3.3 # Graphs plotted based on categories gender and education

```
grid = sns.FacetGrid(df_train, row='Gender', col='Education', size=2.2,
aspect=1.6)
grid.map(plt.hist, 'ApplicantIncome', alpha=.5, bins=10)
grid.add_legend()
```



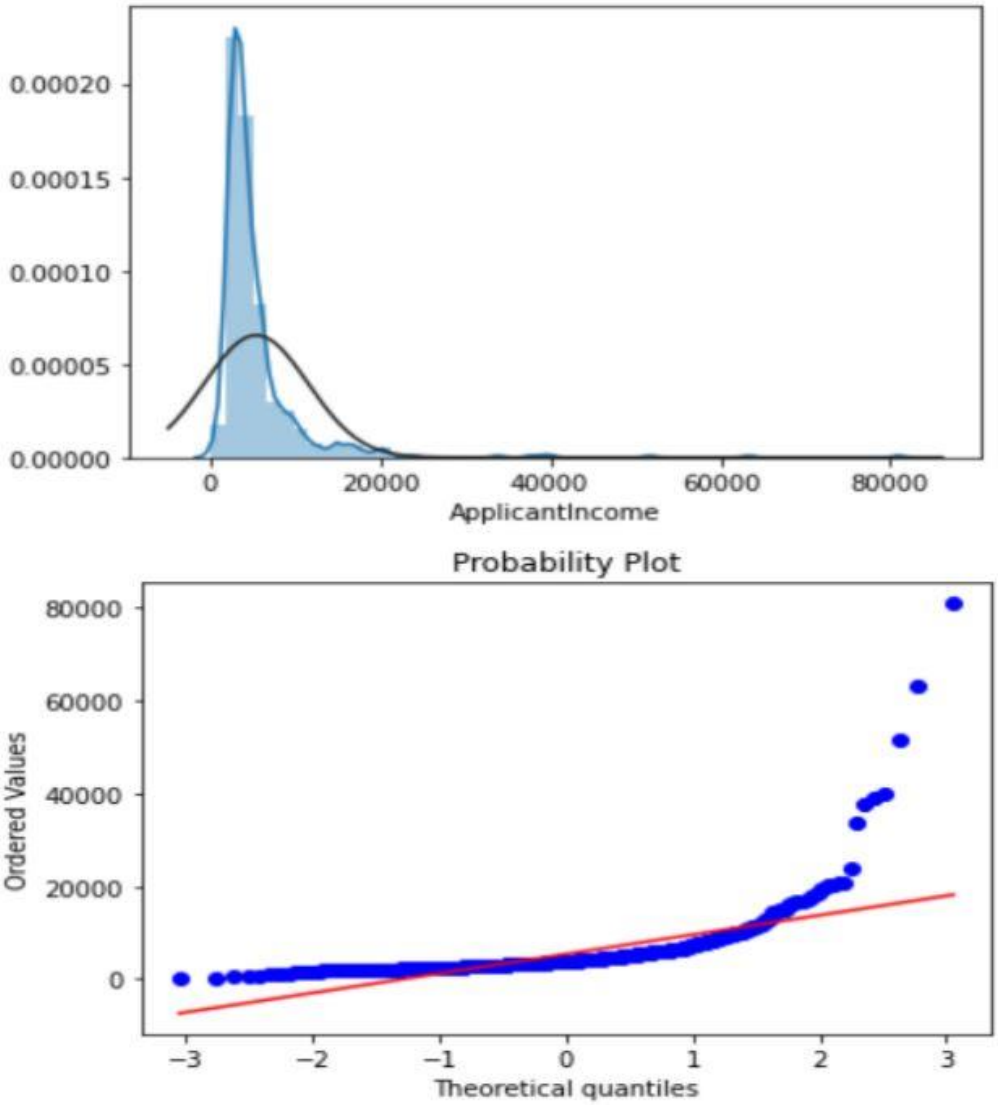
4.3.4 # Graphs plotted based on categories marriage and education

```
grid = sns.FacetGrid(df_train, row='Married', col='Education', size=2.2,
aspect=1.6) grid.map(plt.hist, 'ApplicantIncome', alpha=.5, bins=10)
grid.add_legend()
```



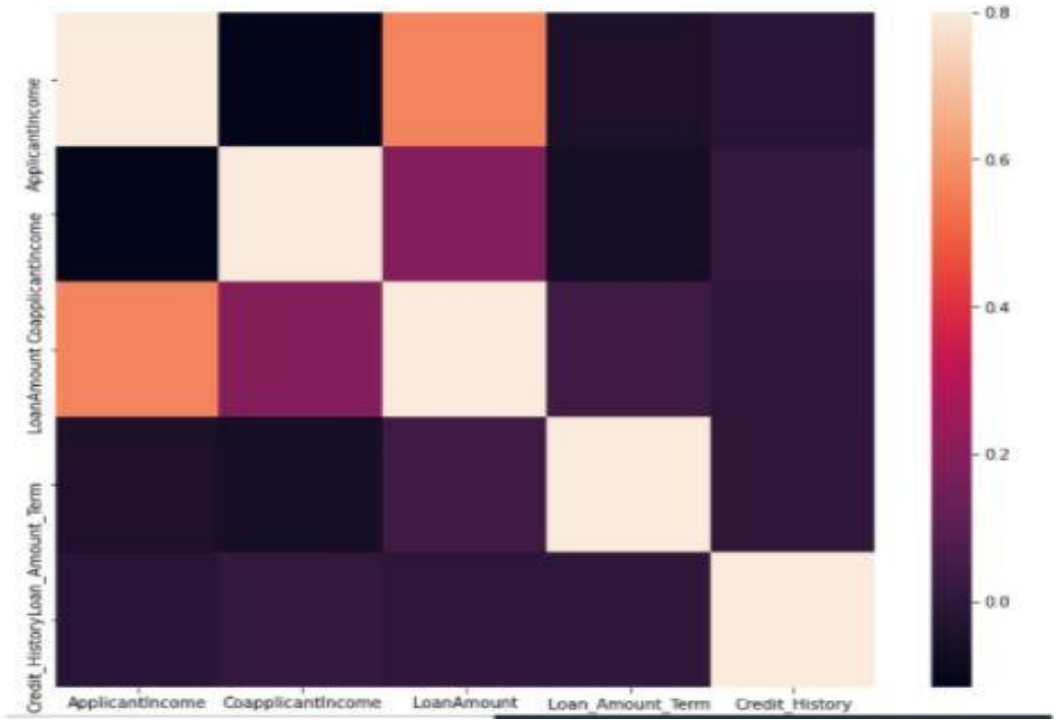
4.3.5 #histogram and normal probability plot

```
sns.distplot(df_train['ApplicantIncome'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['ApplicantIncome'], plot=plt)
```



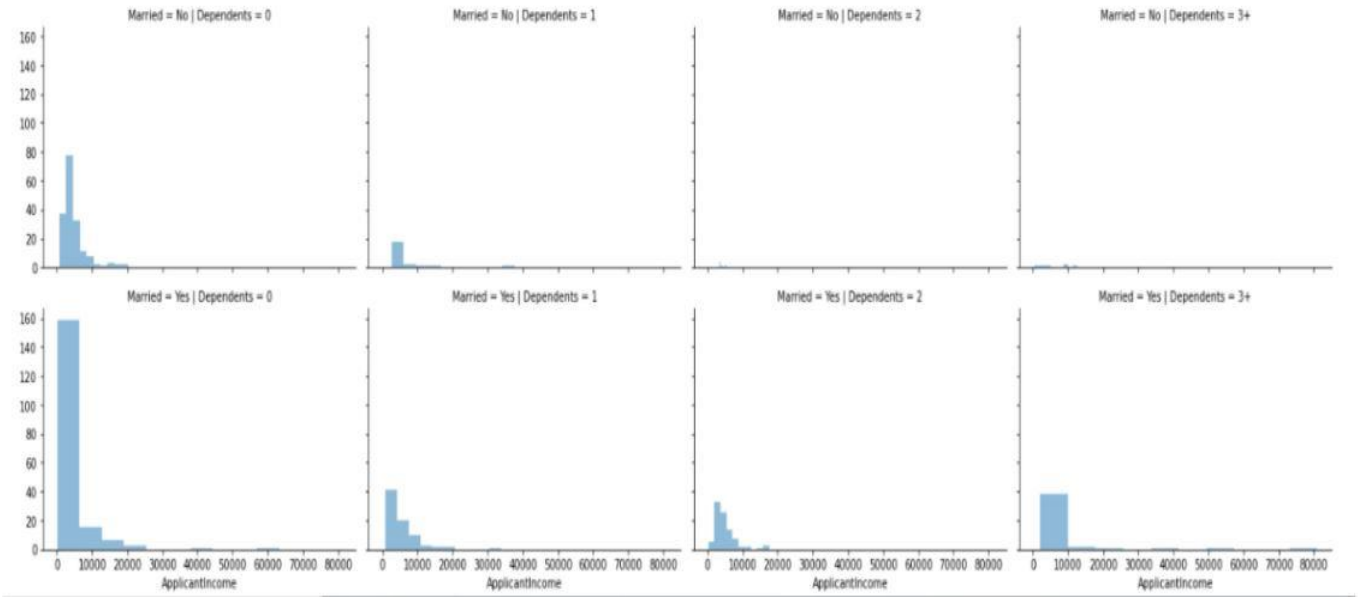
4.3.6 #correlation matrix

```
corrmat = df_train.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```



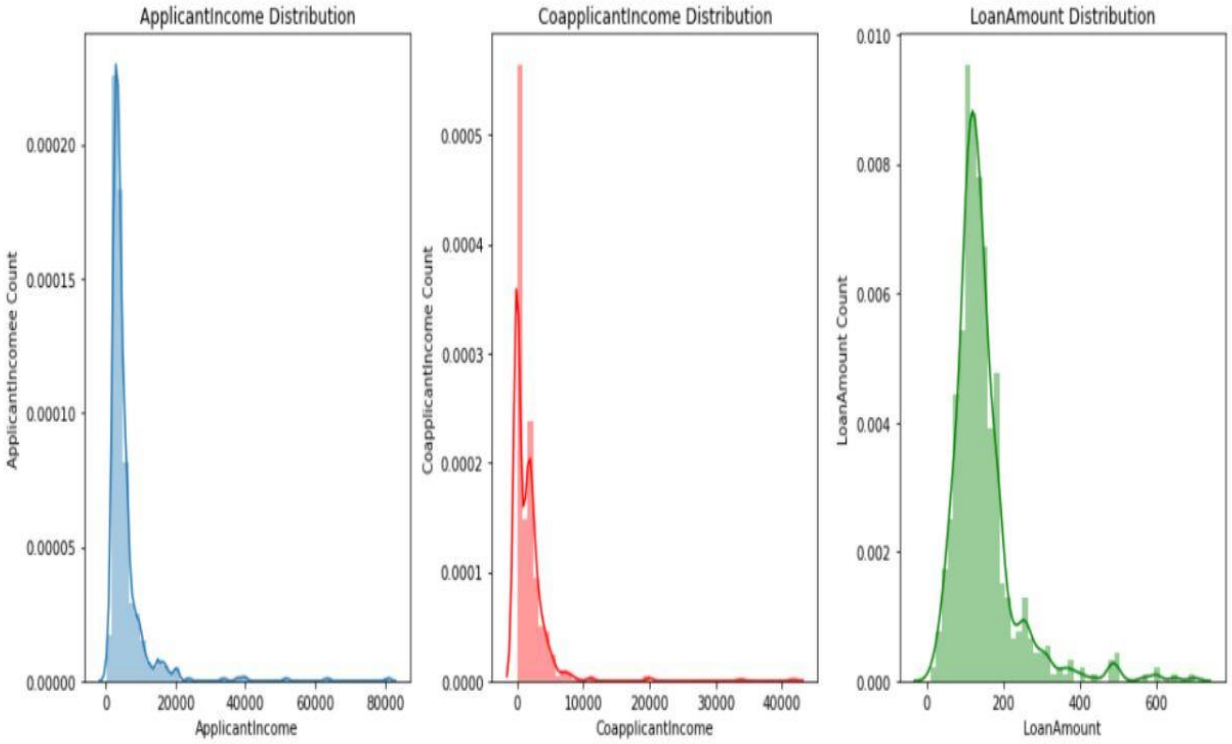
4.3.7 # This graph depicts the combination of applicant income, married people and dependent people in a family

```
grid = sns.FacetGrid(df_train, row='Married', col='Dependents', size=3.2, aspect=1.6) grid.map(plt.hist, 'ApplicantIncome', alpha=.5, bins=10) grid.add_legend()
```



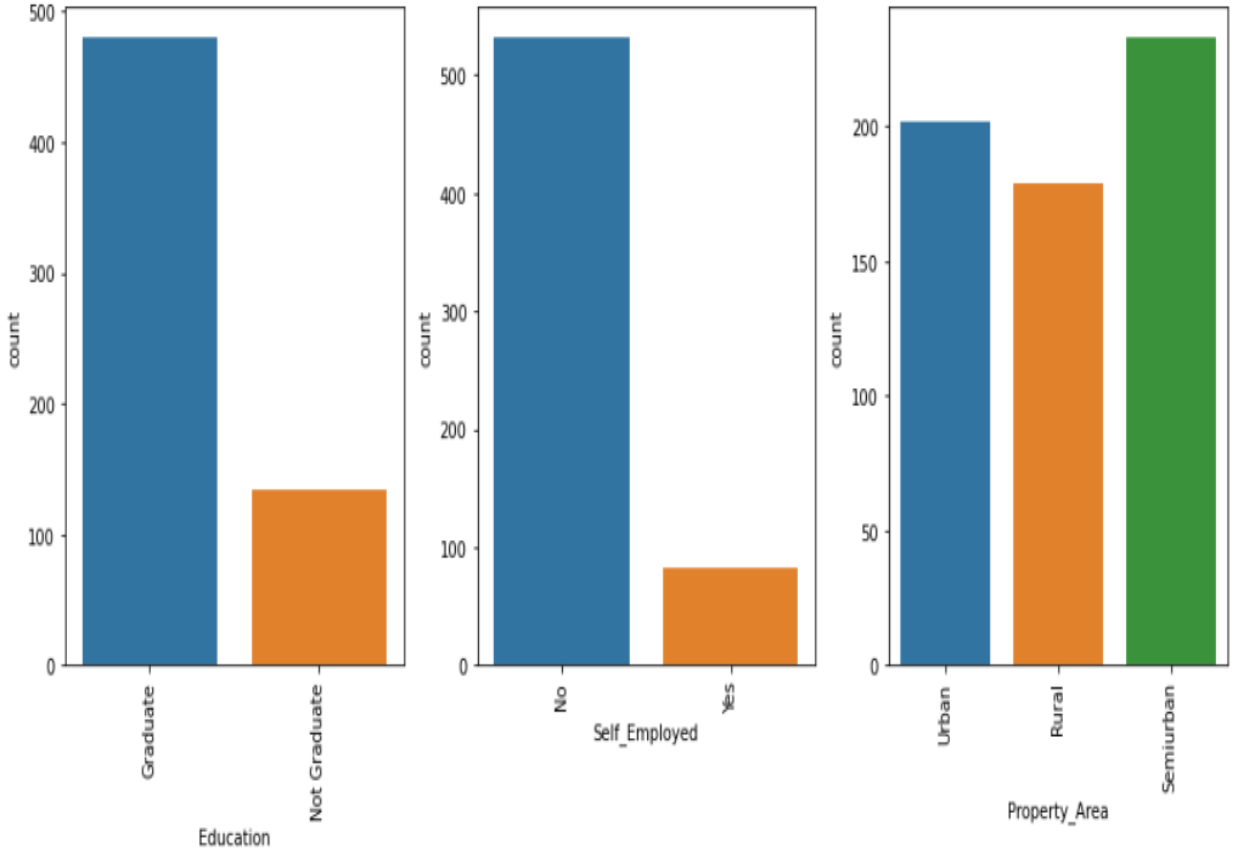
4.3.8 # The graph which differentiates the applicant income distribution, Coapplicant income distribution, loan amount distribution

```
flg, axes = plt.subplots(nrows = 1, ncols = 3, figsize = (14,6))

sns.distplot(df_train['ApplicantIncome'], ax=axes[0]).set_title('ApplicantIncome Distribution')
axes[0].set_ylabel('ApplicantIncomee Count')

sns.distplot(df_train['CoapplicantIncome'], color = "r", ax = axes[1]).set_title('CoapplicantIncome Distribution')
axes[1].set_ylabel('CoapplicantIncome Count')

sns.distplot(df_train['LoanAmount'],color="g",ax=axes[2]).set_title('LoanAmount Distribution')
axes[2].set_ylabel('LoanAmount Count')

plt.tight_layout()
plt.show()
plt.gcf().clear()
```



4.3.9 # This figure shows the count of people differentiated based on education, self_employed, and property_area

```
fig, axes = plt.subplots(ncols=3,figsize=(12,6))
g = sns.countplot(df_train["Education"], ax=axes[0])
plt.setp(g.get_xticklabels(), rotation=90)
g = sns.countplot(df_train["Self_Employed"], ax=axes[1])
plt.setp(g.get_xticklabels(), rotation=90)
g = sns.countplot(df_train["Property_Area"], ax=axes[2])
plt.setp(g.get_xticklabels(), rotation=90)

plt.tight_layout()
plt.show()
plt.gcf().clear()
```



4.4
Explanation of the Main Code using Visual Studio Code

1. Using Logistic Regression Model

Importing required Libraries

```
import pandas as pd
import numpy as np                  # For mathematical calculations
import seaborn as sns               # For data visualization
import matplotlib.pyplot as plt     # For plotting graphs
```

Importing dataset

```
train = pd.read_csv('train_dataset.csv')
test = pd.read_csv('test_dataset.csv')
```

```
 # Converting the values to number

train['Dependents'].replace('3+', 3,inplace=True)
test['Dependents'].replace('3+', 3,inplace=True)
```

```
# take a look at the top 5 rows of the train set, notice the column "Loan_Status"

train.head()
```

	Loan_ID	Gender	Married	Dependen	Education	Self_Empl	ApplicantIı	Coapplicar	LoanAmou	Loan_Amc	Credit_His	Property_	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
5	LP001006	Male	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y

```
# take a look at the top 5 rows of the test set, notice the absense of
"Loan_Status" that we will predict

test.head()
```

	Loan_ID	Gender	Married	Dependen	Education	Self_Empl	ApplicantIı	Coapplicar	LoanAmou	Loan_Amc	Credit_His	Property_Area
2	LP001015	Male	Yes	0	Graduate	No	5720	0	110	360	1	Urban
3	LP001022	Male	Yes	1	Graduate	No	3076	1500	126	360	1	Urban
4	LP001031	Male	Yes	2	Graduate	No	5000	1800	208	360	1	Urban
5	LP001035	Male	Yes	2	Graduate	No	2340	2546	100	360		Urban
6	LP001051	Male	No	3+	Not Gradu	No	3276	0	78	360	1	Rural

```
# Handling Missing Values

# Check How many Null Values in each columns
train.isnull().sum()

# Train Categorical Variables Missisng values
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['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)

# Train Numerical Variables Missing values
train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0],
inplace=True) train['LoanAmount'].fillna(train['LoanAmount'].median(),
inplace=True)

# Train Check if any Null Values Exits
train.isnull().sum()

# Test Check How many Null Values in each columns
test.isnull().sum()

# test Categorical Variables Missisng values
test['Gender'].fillna(test['Gender'].mode()[0], inplace=True)
test ['Married'].fillna(test['Married'].mode()[0],inplace=True)
test['Dependents'].fillna(test['Dependents'].mode()[0], inplace=True)
test['Self_Employed'].fillna(test['Self_Employed'].mode()[0], inplace=True)
test['Credit_History'].fillna(test['Credit_History'].mode()[0], inplace=True)

# test Numerical Variables Missing Values
test['Loan_Amount_Term'].fillna(test['Loan_Amount_Term'].mode()[0],
inplace=True) test['LoanAmount'].fillna(test['LoanAmount'].median(),
inplace=True)

 # test Check if any Null Values Exits
test.isnull().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
dtype: int64
```

```
# Outlier treatment
train['LoanAmount'] = np.log(train['LoanAmount'])
test['LoanAmount'] = np.log(test['LoanAmount'])

 # Separting the Variable into Independent and Dependent
X = train.iloc[:, 1:-1].values y = train.iloc[:, -1].values

 # Converting Categorical variables into dummy
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
labelencoder_
X = LabelEncoder()

# Gender
X[:,0] = labelencoder_X.fit_transform(X[:,0])
```

```
# Marraige
X[:,1] = labelencoder_X.fit_transform(X[:,1])
# Education
X[:,3] = labelencoder_X.fit_transform(X[:,3])


# Self Employed
X[:,4] = labelencoder_X.fit_transform(X[:,4])

# Property Area
X[:,-1] = labelencoder_X.fit_transform(X[:,-1])


# Splitting the dataset into the Training set and Test set
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,
random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Logistic Regression to our training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
# Predecting the results
y_pred = classifier.predict(X_test)

# Printing values of whether loan is accepted or rejected
y_pred[:50]
```

```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y'],
      dtype=object)
```

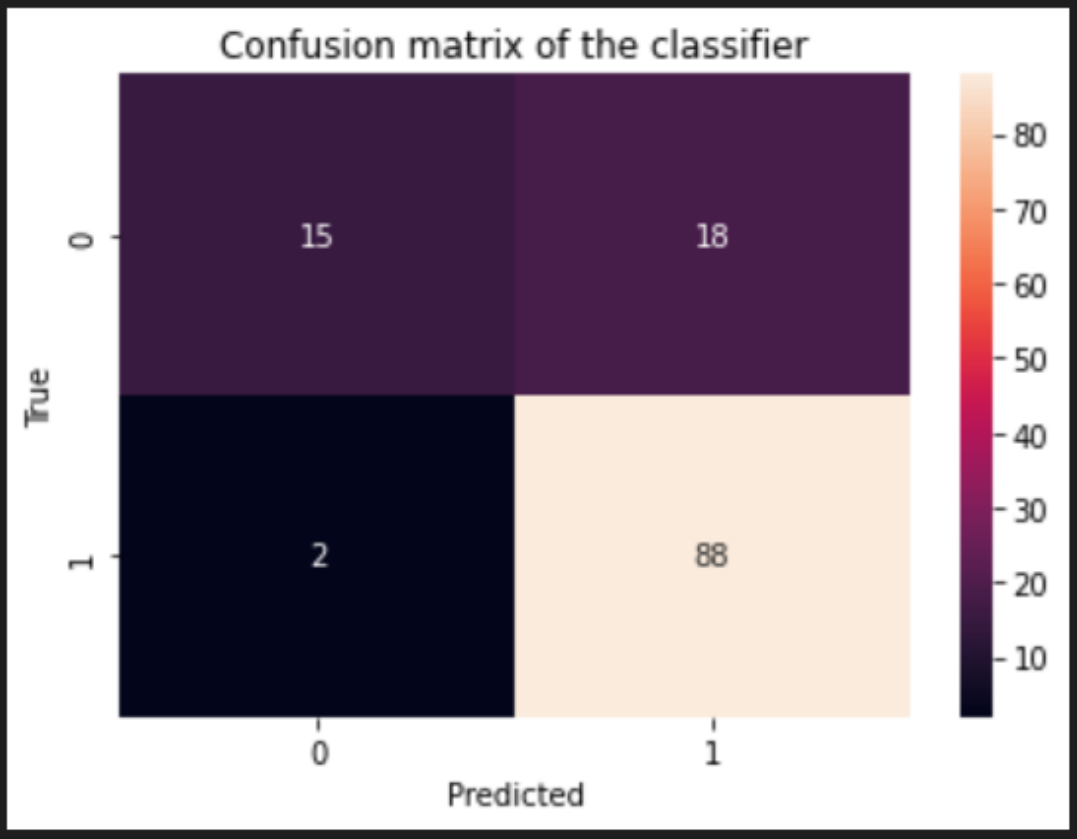
```
# import classification_report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           N       0.88      0.45      0.60        33
           Y       0.83      0.98      0.90        90

    accuracy                           0.84       123
   macro avg       0.86      0.72      0.75       123
weighted avg       0.84      0.84      0.82       123
```

```
# implementing the confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

# f, ax = plt.subplots(figsize=(9, 6))
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
```

plt.xlabel('Predicted')
plt.ylabel('True')

```
[[15 18]
 [ 2 88]]

Text(33.0, 0.5, 'True')
```



# Check Accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)

```
the accuracy of Logistic Regression is:

0.8373983739837398
```

```
# Applying k-Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv
= 10)

accuracies.mean()
# accuracies.std()
```

```
0.8024081632653062
```

2. Using Random Forest Classification

The code till feature scaling is same, there onwards code is slightly different

```
# Fitting Random Forest Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion="entropy", random_state=0)
classifier.fit(X_train,y_train)
```

```
# Printing values of whether loan is accepted or rejected
y_pred[:50]
```

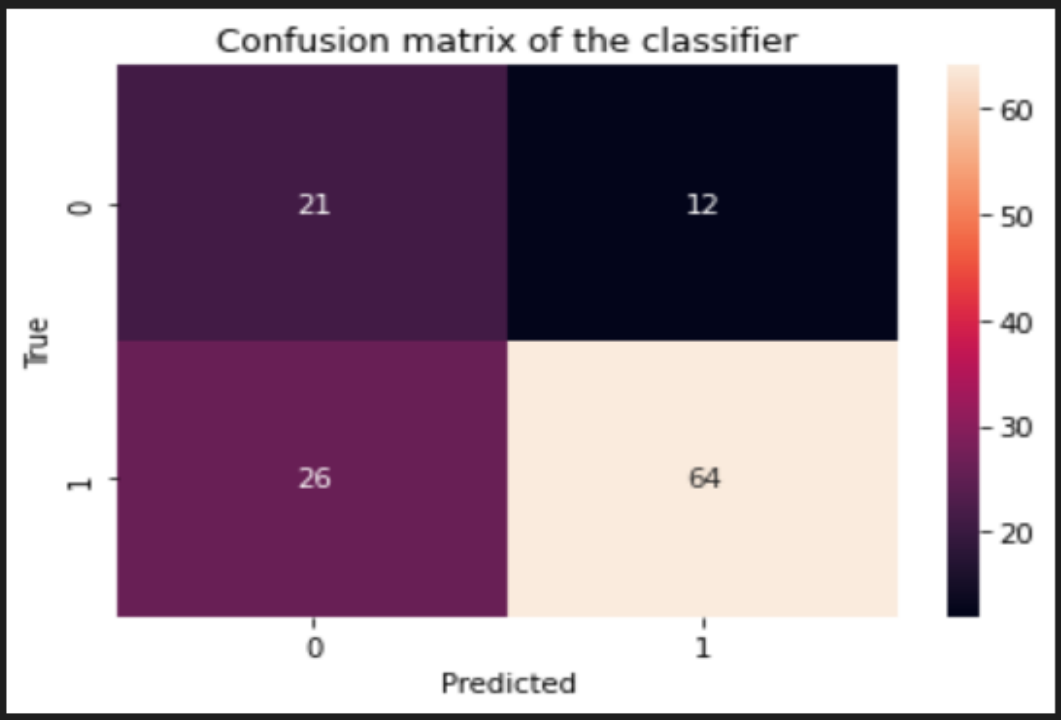
```
array(['N', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y',
       'N', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'N', 'N', 'N', 'Y', 'Y', 'Y',
       'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y'],
      dtype=object)
```

```
# import classification_report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
N	0.45	0.64	0.52	33
Y	0.84	0.71	0.77	90
accuracy			0.69	123
macro avg	0.64	0.67	0.65	123
weighted avg	0.74	0.69	0.71	123

```
# implementing the confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

# f, ax = plt.subplots(figsize=(9, 6))
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
[[21 12]
 [26 64]]

Text(33.0, 0.5, 'True')
```



# Check Accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)

```
the accuracy of Random Forest is:

0.6910569105691057
```

# Applying k-Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv= 10)
accuracies.mean()
# accuracies.std()

```
0.7148163265306122
```

3. Using Decision Tree Classification Model

```
# Fitting Decision Tree Classification to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train,y_train)

# Predecting the results
y_pred = classifier.predict(X_test)

# Printing values of whether loan is accepted or rejected
y_pred[:50]
```

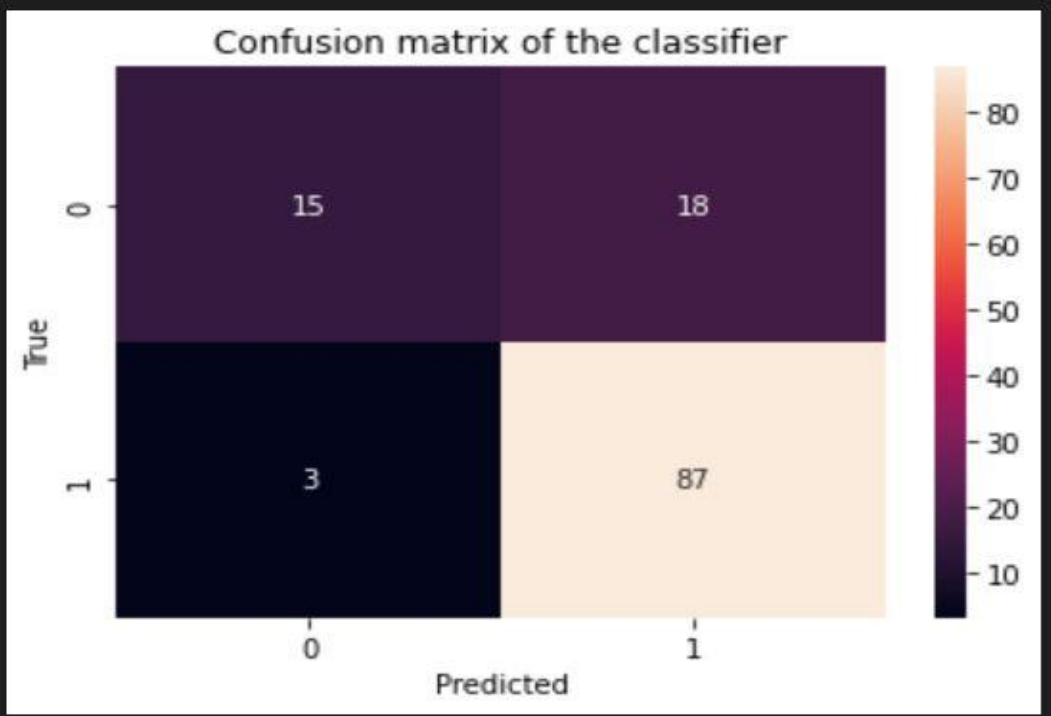
```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y'], dtype='<U1')
```

```
# import classification_report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           N       0.83      0.45      0.59        33
           Y       0.83      0.97      0.89        90

    accuracy                           0.83       123
   macro avg       0.83      0.71      0.74       123
weighted avg       0.83      0.83      0.81       123
```

```
# implementing the confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

# f, ax = plt.subplots(figsize=(9, 6))
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion matrix of the classifier')
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
[[15 18]
 [ 3 87]]

Text(33.0, 0.5, 'True')
```



```
# Check Accuracy
from sklearn.metrics import accuracy_score
print("the accuracy of Decision Tree clasifier is:")
accuracy_score(y_test,y_pred)
```

```
the accuracy of Decision Tree clasifier is:

0.8292682926829268
```

```
# Applying k-Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv
= 10)

accuracies.mean()
# accuracies.std()
```

```
0.7922448979591836
```

5. Output Screens

5.1 Training Dataset

# take a look at the top 5 rows of the train set, notice the column "Loan_Status"
train.head()

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0

LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
NaN	360.0	1.0	Urban	Y
128.0	360.0	1.0	Rural	N
66.0	360.0	1.0	Urban	Y
120.0	360.0	1.0	Urban	Y
141.0	360.0	1.0	Urban	Y

5.2Test Dataset

# take a look at the top 5 rows of the test set, notice the absence of "Loan_Status" coloumn that we will predict
test.head()

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	LP001015	Male	Yes	0	Graduate	No	5720	0
1	LP001022	Male	Yes	1	Graduate	No	3076	1500
2	LP001031	Male	Yes	2	Graduate	No	5000	1800
3	LP001035	Male	Yes	2	Graduate	No	2340	2546
4	LP001051	Male	No	3	Not Graduate	No	3276	0

LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
110.0	360.0	1.0	Urban
126.0	360.0	1.0	Urban
208.0	360.0	1.0	Urban
100.0	360.0	NaN	Urban
78.0	360.0	1.0	Rural

5.3 Output Using LOGISTIC REGRESSION Model

```
# Printing values of whether loan is accepted or rejected
y_pred[:50]
```

✓ 0.5s

```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y'],
      dtype=object)
```

```
# Check Accuracy
from sklearn.metrics import accuracy_score
print("the accuracy of Logistic Regression is:")
accuracy_score(y_test,y_pred)
```

```
the accuracy of Logistic Regression is:

0.8373983739837398
```

5.4 Output using RANDOM FOREST classification Model

```
# Printing values of whether loan is accepted or rejected
y_pred[:50]
```

✓ 0.5s

```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y'],
      dtype=object)
```

```
# Check Accuracy
from sklearn.metrics import accuracy_score
print("the accuracy of Logistic Regression is:")
accuracy_score(y_test,y_pred)
```

```
the accuracy of Logistic Regression is:

0.8373983739837398
```

5.5 Output using DECISION TREE Classifier

```
# Printing values of whether loan is accepted or rejected
y_pred[:50]
```

✓ 0.4s

```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y'], dtype='<U1')
```

```
# Check Accuracy
from sklearn.metrics import accuracy_score
print("the accuracy of Decision Tree clasifier is:")
accuracy_score(y_test,y_pred)
```

```
the accuracy of Decision Tree clasifier is:
0.8292682926829268
```

7. CONCLUSION & FUTURE ENHANCEMENT

Today's fast-growing IT industry needs to discover new technology and update the old technology that helps us to reduce human intervention and increase the efficiency of the work. This model is used for the banking system or anyone who wants to apply for a loan, it also helps the organizations in making right decision to approve or reject the loan request of the customers. It will be very helpful in bank management. Time is also very precious for everyone through this not only the bank but also the waiting time of the applicant will also reduce.

In future, this model can be used to compare various other machine learning algorithm generated prediction models and the model which will give higher accuracy will be chosen as the prediction model. This project can be extended to higher level in future. The system is trained on old training dataset in future software can be made such that new testing data should also take part in training data after some fix time. Predictive model for loans that uses machine learning algorithms, where the results from each graph of the paper can be taken as individual criteria for the machine learning algorithm.

8. REFERENCES

- https://ieeexplore.ieee.org/
- https://github.com/
- https://medium.com/topic/machine-learning
- https://www.kaggle.com/learn/machine-learning
- https://www.geeksforgeeks.org/machine-learning
- https://www.wikipedia.org/

APPENDIX-A :

MACHINE LEARNING

Machine learning is the scientific field dealing with the ways in which machines learn from experience. For many scientists, the term “machine learning” is identical to the term “artificial intelligence”, given that the possibility of learning is the main characteristic of an entity called intelligent in the broadest sense of the word. The purpose of machine learning is the construction of computer systems that can adapt and learn from their. A more detailed and formal definition of machine learning is A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. With the rise of Machine Learning approaches we have the ability to find a solution to this issue, we have developed a system using data mining which has the ability to predict whether the message is spam or not. This research has focused on developing a system based on three classification methods namely, Support Vector Machine, Logistic regression and Artificial Neural Network algorithms.

Supervised learning

In supervised learning, the system must “learn” inductively a function called target function, which is an expression of a model describing the data. The objective function is used to predict the value of a variable, called dependent variable or output variable, from a set of variables, called independent variables or input variables or characteristics or features. The set of possible input values of the function, i.e. its domain, are called instances. Each case is described by a set of characteristics (attributes or features). A subset of all cases, for which the output variable value is known, is called training data or examples. In order to infer the best target function, the learning system, given a training set, takes into

consideration alternative functions, called hypothesis and denoted by h. In supervised learning, there are two kinds of learning tasks: classification and regression. Classification models try to predict distinct classes, such as e.g. blood groups, while regression models predict numerical values. Some of the most common techniques are Decision Trees (DT), Rule Learning, and Instance Based Learning (IBL), such as k-Nearest Neighbours (k-NN), Genetic Algorithms (GA), Artificial Neural Networks (ANN), and Support Vector Machines (SVM).

Unsupervised learning

In unsupervised learning, the system tries to discover the hidden structure of data or associations between variables. In that case, training data consists of instances without any corresponding labels. Association Rule Mining appeared much later than machine learning and is subject to greater influence from the research area of databases. Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. IV. Reinforcement Learning The term Reinforcement Learning is a general term given to a family of techniques, in which the system attempts to learn through direct interaction with the environment so as to maximize some notion of cumulative reward. It is important to mention that the system has no prior knowledge about the behaviour of the environment and the only way to find out is through trial and failure (trial and error). Reinforcement learning is mainly applied to autonomous systems, due to its independence in relation to its environment.

Data Preprocessing

Data Preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format, techniques used for Data pre-processing are

Data Cleaning

Data cleaning is the process of detecting, rectifying, or removing inaccurate and corrupted information from the dataset or database. In addition, it recognizes inaccurate or unfinished parts of data, filling the missing ones, and removing the noisy data.

Data Integration

Data integration involves combining data residing in different sources and providing users with a unified view of these all data. Data integration may involve inconsistent data and therefor needs data cleaning.

Data Transformation.

Data transformation is the process of converting data from one from to another from. Data transformation is necessary to ensure that data from one application or database is understandable to other applications and databases.

Data Reduction

Data reduction involves in reducing the number of attributes, attribute values, number of tuples.

Logistic regression

Logistic regression is a regression model where the dependent variable is categorical, namely binary dependent variable-that is, where it can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed using logistic regression. Many other medical scales used to assess severity of a patient have been developed using logistic regression. The technique can also be used in

engineering, especially for predicting the probability of failure of a given process, system or product. It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription. In economics it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application is about to predict the likelihood of a homeowner defaulting on a mortgage. Conditional random fields, an extension of logistic regression to sequential data, are used in natural language processing.

Random forest

Are an ensemble learning method for classification and regression and other task that operates by constructing a multitude of decision tree at training time and outputting the class that is the mode of the classes or mean prediction of individual trees. The first algorithm for random decision forests was created by Tin Kam Ho using random subspace method. Ho established that to gain the accuracy it should over train where it can randomly restrict sensitive selected features of the given data.

Decision tree

Decision tree are a type of supervised machine learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decision or the final outcome and the decision nodes are where the data is split. The leaf node is labeled by attribute and each attribute is assigned by a target value. The highest information gain of all attribute id calculated first. It is a method commonly used for data mining.