VIDYALANKAR INSTITUTE OF TECHNOLOGY

(Affiliated to University of Mumbai)

Wadala (E), Mumbai –400 037

A MINI-PROJECT REPORT

ON

"**Loan Prediction System**"

(Course – Business Intelligence Lab)

Submitted by

STUDENT NAME	ROLL NO
Shravani Kadam	**18101A0011**
Dhanesh Salgaonkar	18101A0013
Bhishmesh Chaudhari	**18101A0024**
Prajwal Gawande	18101A0032
Paresh Devlekar	18101A0047

Department of Information Technology VIT,
Wadala (E), Mumbai-400 037

2020-2021

VIDYALANKAR INSTITUTE OF TECHNOLOGY

Department of Information Technology

CERTIFICATE

Certified that the mini-project work entitled **"Loan Prediction System"** is a bonafide work carried out by

STUDENT NAME	ROLL NO
Shravani Kadam	**18101A0011**
Dhanesh Salgaonkar	18101A0013
Bhishmesh Chaudhari	**18101A0024**
Prajwal Gawande	18101A0032
Paresh Devlekar	18101A0047

The report has been approved as it satisfies the academic requirements in respect of mini-project work prescribed for the **course - Business Intelligence Lab**.

……

Prof. Vidya Chitre
Faculty In-Charge

……

Internal Examiner

……

External Examiner

TABLE OF CONTENT

1. Abstract

Finance raising and lending for real estate, consumer, mortgage and companies‘ loans is the central part of almost every bank‘s business model. Lending money to inappropriate customers forms the major source of credit risk. The major share of the bank‘s assets comes directly from the profit derived from the bank‘s loans. The banking companies‘ face, however dual challenge to distinguish the possible deliberate defaulters from the applicants . As the number of transactions in banking sector is rapidly growing and huge data volumes are available, the customers’ behavior can be easily analyzed and the risks around loan can be reduced.

The primary goal of the banking community is to safely invest their capital. Banks wanted to automate the loan eligibility process (real time) based on customer details such as Gender, Marital Status, Age, Occupation, Income, debts, and others provided in their online application form.

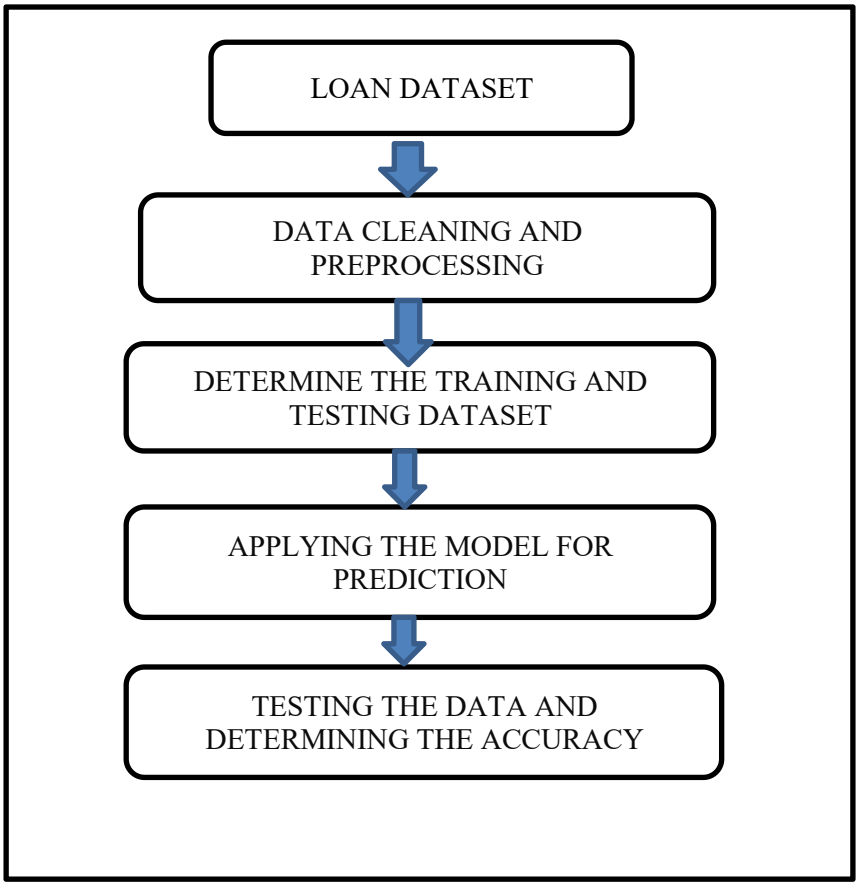
In the current scenario ,banks approve loans after a clear verification and authentication process, however, it remains uncertain whether the candidate selected is the worthy correct of all the applicants. Through this method, we can predict whether or not that particular applicant is secure and the machine learning technique automates the entire process of authentication.

This project seeks to ensure that the deserving customers can be quickly selected with ease which offers various benefits to the bank itself. This method will measure the weight automatically of each criterion that participates in the loan processing and process the same with regards to the associated weight of the new test data.

2. Objectives

- To obtain the loan prediction dataset including different parameters used in our project.
- To perform an efficient pre-processing task to make data suitable for analysis & Visualization .
- To predict whether the person who applied for the loan is eligible to get the loan or not .
- To assist the Banking Systems in automating the process of Loan Approval.

2 Methodology



The machine learning techniques are tools for data mining in order to learn from the data and derive the prediction models.

The supervised learning tasks address the following questions: “Can we predict if a person will get loan from the bank or not? What factors make a person eligible for getting loan?”

The methodology of our project follows collecting the Loan Dataset , understanding the dataset and its attributes followed by data preprocessing and cleaning (replacing missing values , normalizing and scaling the dataset & removal of outliers).The next step followed is preparing the datset for modelling (splitting the dataset) .The Decision Tree algorithm is used for model building and prediction.The last step is to predict the test data by accepting input values and calculate the accuracy of the algorithm and thus the system.

Software Requirements : Juptyer Notebook OR Google Colab,RapidMiner

Decision Tree :

Decision Tree is a supervised learning algorithm used to solve classification and regression problems too. Here, DT uses tree representation to solve the prediction problem, i.e., external node and leaf node in a tree represents attribute and class labels respectively. The pseudo code for DT model is depicted in the following section: Step 1: Best attribute is chosen as the tree‘s root. Step 2: Training set is divided into subsets, such that, each subset comprises similar value for an attribute. Step 3: Step 1 and Step 2 are repeated for all subsets until all the leaf nodes are traversed in a tree.

3. Implementation

Loan Prediction System

The objective of this system is to predict either a person is eligible to apply for Loan or Not based on certain parameters or attributes like gender,marital status,employment,Income,credit history etc.The Prediction algorithm used for the system's implementation is Decision Tree Algorithm due to its maximum accuracy score.

Rows : 614

Columns : 13

Dataset : Kaggle

Jupyter Notebook : https://github.com/shravanikadam

```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline


# Reading the training dataset in a dataframe using Pandas
dataset = pd.read_csv("train.csv")
```

 # Understanding the column (Attributes) of the dataset

```
# First 10 Rows of training Dataset
dataset.head(10)


#No of Entries and Columns of training dataset
dataset.shape

# Concise Summary of data in the training dataset
dataset.info()
```

```
# Summary of numerical variables for training data set
dataset.describe()
```

1.For the non-numerical values (e.g. Property_Area, Credit_History etc.), we can look at frequency distribution to understand whether they make sense or not.

```
# Get the unique values and their frequency of variable Property_Area

dataset['Property_Area'].value_counts()

pd.crosstab(dataset['Property_Area'],dataset['Loan_Status'],margins=True)

# Loan approval rates in absolute numbers
loan_approval = dataset['Loan_Status'].value_counts()['Y']
print('Total no of Loans Approved :',loan_approval)

# Credit History and Loan Status
pd.crosstab(dataset['Credit_History'],dataset['Loan_Status'],margins=True)

def percentageConvert(ser):
    return ser/float(ser[-1])

 #Loan approval rate for customers having Credit_History (1)
#dataset['Y'] = pd.crosstab(dataset ["Credit_History"], dataset ["Loan_Status"],
margins=True).apply(percentageConvert, axis=1)
#loan_approval_with_Credit_1 = dataset['Y'][1]
#print(loan_approval_with_Credit_1*100)
```

- 79.58 % of the applicants whose loans were approved have Credit_History equals to 1.

```
pd.crosstab(dataset['Education'],dataset['Loan_Status'],margins=True)
```

Understanding Distribution of Numerical Variables

1.ApplicantIncome

2.LoanAmount

```
# Understanding the Outliers and treatment

# Box Plot for variable ApplicantIncome of training data set
dataset.boxplot(column='ApplicantIncome')
```

3.The above Box Plot confirms the presence of a lot of outliers/extreme values. This can be attributed to the income disparity in the society.

```
# Box Plot for variable CoapplicantIncome of training data set
dataset.boxplot(column='CoapplicantIncome')

# Histogram of variable ApplicantIncome
dataset['ApplicantIncome'].hist(bins=20)

# Histogram of variable CoapplicantIncome
dataset['CoapplicantIncome'].hist(bins=20)

# Box Plot for variable ApplicantIncome by variable Education of training data set
dataset.boxplot(column='ApplicantIncome', by= 'Education')
```

4.We can see that there is no substantial different between the mean income of graduate and non-graduates. But there are a higher number of graduates with very high incomes, which are appearing to be the outliers

```
# Box Plot for variable LoanAmount of training data set
dataset.boxplot(column='LoanAmount')
```

5. LoanAmount has missing as well as extreme values, while ApplicantIncome has a few extreme values

```
# Histogram of variable LoanAmount
dataset['LoanAmount'].hist(bins=20)
```

6. The extreme values are practically possible, i.e. some people might apply for high value loans due to specific needs. So instead of treating them as outliers, let’s try a log transformation to nullify their effect

```
# Perform log transformation of TLoanAmount to make it closer to normal
dataset['Loanamt_log']=np.log(dataset['LoanAmount'])
# Histogram of variable LoanAmount
dataset['Loanamt_log'].hist(bins=20)

# Add both ApplicantIncome and CoapplicantIncome to TotalIncome
dataset['TotalIncome']= dataset['ApplicantIncome'] + dataset['CoapplicantIncome']
# Perform log transformation of TotalIncome to make it closer to normal
dataset['TotalIncome_log']= np.log(dataset['TotalIncome'])
```

```
# Box Plot for variable TotalIncome of training data set
dataset.boxplot(column='TotalIncome_log')

# Looking at the distribtion of TotalIncome
dataset['TotalIncome_log'].hist(bins=20)
```

 # Data Preparation for Model Building

- sklearn requires all inputs to be numeric, we should convert all our categorical variables into numeric by encoding the categories. Before that we will fill all the missing values in the dataset.

```
#returns total no of missing values for each attribute
dataset.isnull().sum()

# Impute missing values for Gender
dataset['Gender'].fillna(dataset['Gender'].mode()[0],inplace=True)

# Impute missing values for Married
dataset['Married'].fillna(dataset['Married'].mode()[0],inplace=True)

# Impute missing values for Dependents
dataset['Dependents'].fillna(dataset['Dependents'].mode()[0],inplace=True)

# Impute missing values for Self_Employed
dataset['Self_Employed'].fillna(dataset['Self_Employed'].mode()[0],inplace=True)

# Impute missing values for LoanAmount
dataset.LoanAmount = dataset.LoanAmount.fillna(dataset.LoanAmount.mean())
dataset.Loanamt_log = dataset.Loanamt_log.fillna(dataset.Loanamt_log.mean())

# Impute missing values for Loan_Amount_Term
dataset['Loan_Amount_Term'].fillna(dataset['Loan_Amount_Term'].mode()[0],inplac
e=True)

# Impute missing values for Credit_History
dataset['Credit_History'].fillna(dataset['Credit_History'].mode()[0],inplace=True)

dataset.isnull().sum()

dataset.head()

dataset.info()
```

```
# Selecting Attribute Data for Model Training
x= dataset.iloc[:,np.r_[1:5,9:11,13:15]].values
y= dataset.iloc[:,12].values

#print x
x

#print y
y
```

 # Model Building

```
#Dividing the training dataset(80%) in train and test sets(20%)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)

print(x_train)

#To convert the categorical values into numerical values
from sklearn.preprocessing import LabelEncoder
labelencoder_x = LabelEncoder()

#for loop for conversion of first 5 attributes
for i in range(0,5):
    x_train[:,i]= labelencoder_x.fit_transform(x_train[:,i])

x_train[:,7]= labelencoder_x.fit_transform(x_train[:,7])

x_train


labelencoder_y = LabelEncoder()
y_train= labelencoder_y.fit_transform(y_train)

y_train


for i in range(0,5):
    x_test[:,i]= labelencoder_x.fit_transform(x_test[:,i])

x_train[:,7]= labelencoder_x.fit_transform(x_train[:,7])
```

```
labelencoder_y = LabelEncoder()
y_test= labelencoder_y.fit_transform(y_test)

x_test

y_test

#Scaling the training dataset
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
x_test=ss.fit_transform(x_test)
```

 # Decision Tree

```
#checking accuracy for Decision Tree Algorithm
from sklearn.tree import DecisionTreeClassifier
DTClassifier= DecisionTreeClassifier(criterion='entropy',max_depth=4)
#Training the model
DTClassifier.fit(x_train,y_train)

#predicting the values of x_test
y_pred= DTClassifier.predict(x_test)
y_pred

#calculating accuracy for decision tree algorithm
from sklearn import metrics
print('The accuracy of decision tree is : ',metrics.accuracy_score(y_pred,y_test))

# Prediction for Testing Dataset

LoanEligible = ['Congratulations your eligible for loan', 'Sorry! your not eligible for
loan']

attributes = ['Gender', 'Married', 'Dependents', 'Education',
      'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
      'Loan_Amount_Term', 'Credit_History', 'Property_Area']

dataset.replace({'Gender':{'Male':0,'Female':1}},inplace=True)
dataset.replace({'Married':{'Yes':1,'No':0}},inplace=True)
dataset.replace({'Dependents':{'+3':0,'3+':1}},inplace=True)
dataset.replace({'Education':{'Not Graduate':0,'Graduate':1}},inplace=True)
dataset.replace({'Self_Employed':{'No':0,'Yes':1}},inplace=True)
```

```
dataset.replace({'Property_Area':{'Urban':0,'Rural':1, 'Semiurban': 2}},inplace=True)
dataset.replace({'Loan_Status':{'Y':1,'N':0}},inplace=True)
dataset = dataset.fillna(0)

dataset.head(10)

X= dataset[attributes]
Y = dataset[["Loan_Status"]]
np.ravel(Y)

testdata= pd.read_csv("train.csv")

testdata.head()

testdata.info()

testdata.isnull().sum()

testdata['Gender'].fillna(testdata['Gender'].mode()[0],inplace=True)
testdata['Dependents'].fillna(testdata['Dependents'].mode()[0],inplace=True)
testdata['Self_Employed'].fillna(testdata['Self_Employed'].mode()[0],inplace=True)
testdata['Credit_History'].fillna(testdata['Credit_History'].mode()[0],inplace=True)
testdata['Loan_Amount_Term'].fillna(testdata['Loan_Amount_Term'].mode()[0],inpl
ace=True)


testdata.isnull().sum()

testdata.boxplot(column='LoanAmount')


testdata.boxplot(column='ApplicantIncome')

testdata.LoanAmount = dataset.LoanAmount.fillna(testdata.LoanAmount.mean())


testdata['Loanamt_log']=np.log(testdata['LoanAmount'])

testdata.isnull().sum()

testdata['TotalIncome']= testdata['ApplicantIncome'] + testdata['CoapplicantIncome']
testdata['TotalIncome_log']= np.log(testdata['TotalIncome'])
```

```
testdata.boxplot(column='TotalIncome')

testdata.head()

testdata.replace({'Gender':{'Male':0,'Female':1}},inplace=True)
testdata.replace({'Married':{'Yes':1,'No':0}},inplace=True)
testdata.replace({'Dependents':{'+3':0,'3+':1}},inplace=True)
testdata.replace({'Education':{'Not Graduate':0,'Graduate':1}},inplace=True)
testdata.replace({'Self_Employed':{'No':0,'Yes':1}},inplace=True)
testdata.replace({'Property_Area':{'Urban':0,'Rural':1, 'Semiurban': 2}},inplace=True)
testdata.replace({'Loan_Status':{'Y':1,'N':0}},inplace=True)
testdata = testdata.fillna(0)

X_test= testdata[attributes]
Y_test = testdata[["Loan_Status"]]
np.ravel(Y_test)

def DecisionTree(userinput):
    #import tree
    from sklearn import tree

    # empty model of the decision tree
    DTClassifier = tree.DecisionTreeClassifier()
    DTClassifier =  DTClassifier.fit(X,Y)

    # calculating accuracy----------------------
    from sklearn.metrics import accuracy_score
    ypred= DTClassifier.predict(X_test)
    # ----------------------------------------

    # Prediction
    inputtest = [userinput]
    predict = DTClassifier.predict(inputtest)
    predicted=predict[0]

    # check if prediction is done successfully
    h='no'
    for a in range(0,len(LoanEligible)):
        if(predicted == a):
            h='yes'
            break

    if (h=='yes'):
```

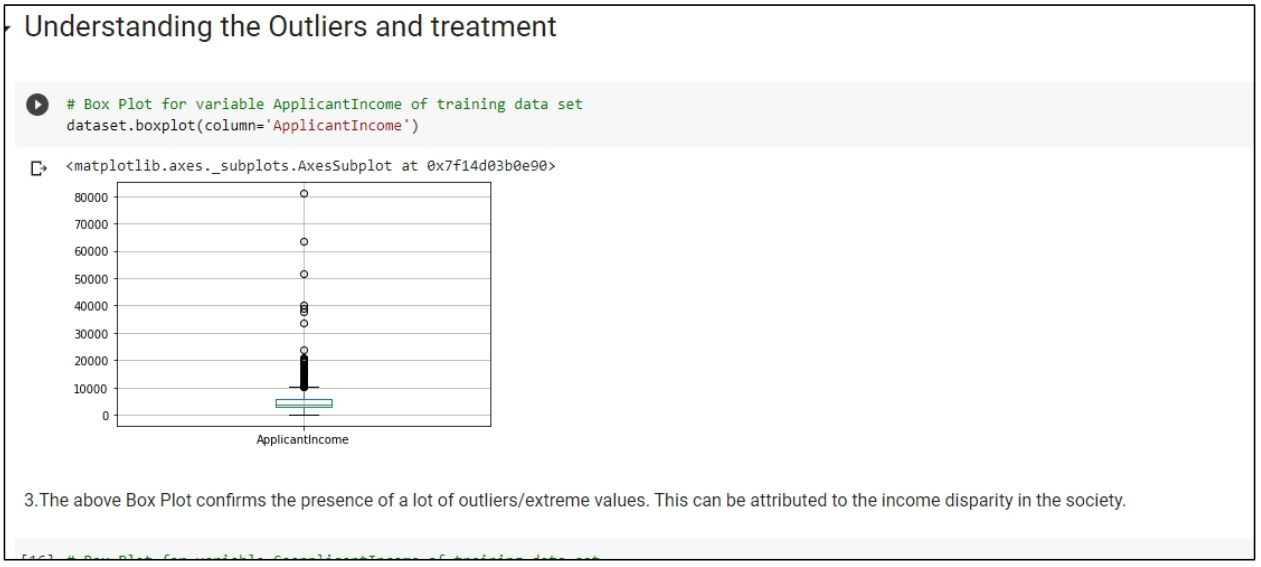
```
        return(LoanEligible[a], accuracy_score(Y_test, ypred,normalize=True)*100)
    else:
        return("Not Found", "Not Found")



Gender = input("Gender (0.0 for Male 1.0 for Female): ")
Married = input("Married (1.0 for Married 0.0 for Unmarried): ")
Dependents = input("Dependents (0 for Independent 1 for Dependent): ")
Education = input("Education (0 for Not graduated 1 for Graduated): ")
Self_Employed = input("Self_Employed (0.0 for No 1.0 for Yes): ")
ApplicantIncome =  input("Applicant Income (Eg.4583): ")
CoapplicantIncome = input("Coapplicant Income (Eg. 1516.0): ")
LoanAmount = input("Loan Amount (Eg. 120.0): ")
Loan_Amount_Term = input("Loan Amount Term (Eg. 360.0): ")
Credit_History = input("Credit History (Eg. 1.0 or 0.0): ")
Property_Area = input("Property Area  (0 for Urban, 1 for Rural, 2 for Semiurban): ")
userinput = [Gender, Married, Dependents, Education,
      Self_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount,
      Loan_Amount_Term, Credit_History, Property_Area]

print("==========================================================")
print("Using Decision Tree Algorithm:", DecisionTree(userinput))
```

4. Result and Screenshots

In this project, we have employed machine learning approaches to study the bank loan dataset.We have implemented the Decision tree Algorithm in order to perform analysis and visualize the dataset.The results have achieved the accuracy of around 83% .We have also determined the important factors that influence the loan eligibility criteria for an individual.At the end we have tested the model and evaluated its performance by taking input parameters in predicting the eligibility for loan approval .We have also implemented our system in RapidMiner tool for better visualization & exploratory analysis of the dataset.

Output :

1. Analysis of Data



Loan_Status	N	Y	All
Property_Area			
Rural	69	110	179
Semiurban	54	179	233
Urban	69	133	202
All	192	422	614

```
[11] # Loan approval rates in absolute numbers
     loan_approval = dataset['Loan_Status'].value_counts()['Y']
     print('Total no of Loans Approved :',loan_approval)
```

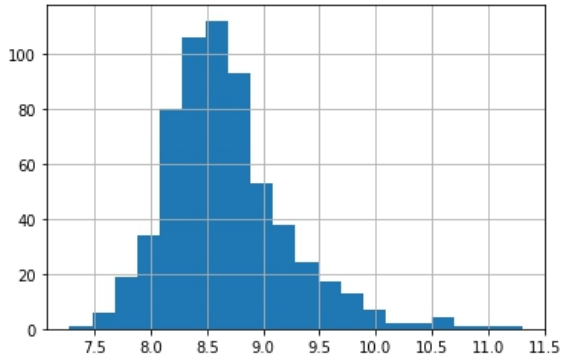
```
Total no of Loans Approved : 422
```

```
[12] # Credit History and Loan Status
     pd.crosstab(dataset['Credit_History'],dataset['Loan_Status'],margins=True)
```

Loan_Status	N	Y	All
Credit_History			
0.0	82	7	89
1.0	97	378	475
All	179	385	564

```
[25] # Looking at the distribtion of TotalIncome
     dataset['TotalIncome_log'].hist(bins=20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f14cfa42550>
```



2. Model Building : Decision Tree

Decision Tree

```
[54] #checking accuracy for Decision Tree Algorithm
     from sklearn.tree import DecisionTreeClassifier
     DTClassifier= DecisionTreeClassifier(criterion='entropy',max_depth=4)
     #Training the model
     DTClassifier.fit(x_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=4, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
[55] #predicting the values of x_test
     y_pred= DTClassifier.predict(x_test)
     y_pred
```

```
array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1])
```

```
[56] #calculating accuracy for decision tree algorithm
     from sklearn import metrics
     print('The accuracy of decision tree is : ',metrics.accuracy_score(y_pred,y_test))
```

```
The accuracy of decision tree is :  0.8292682926829268
```

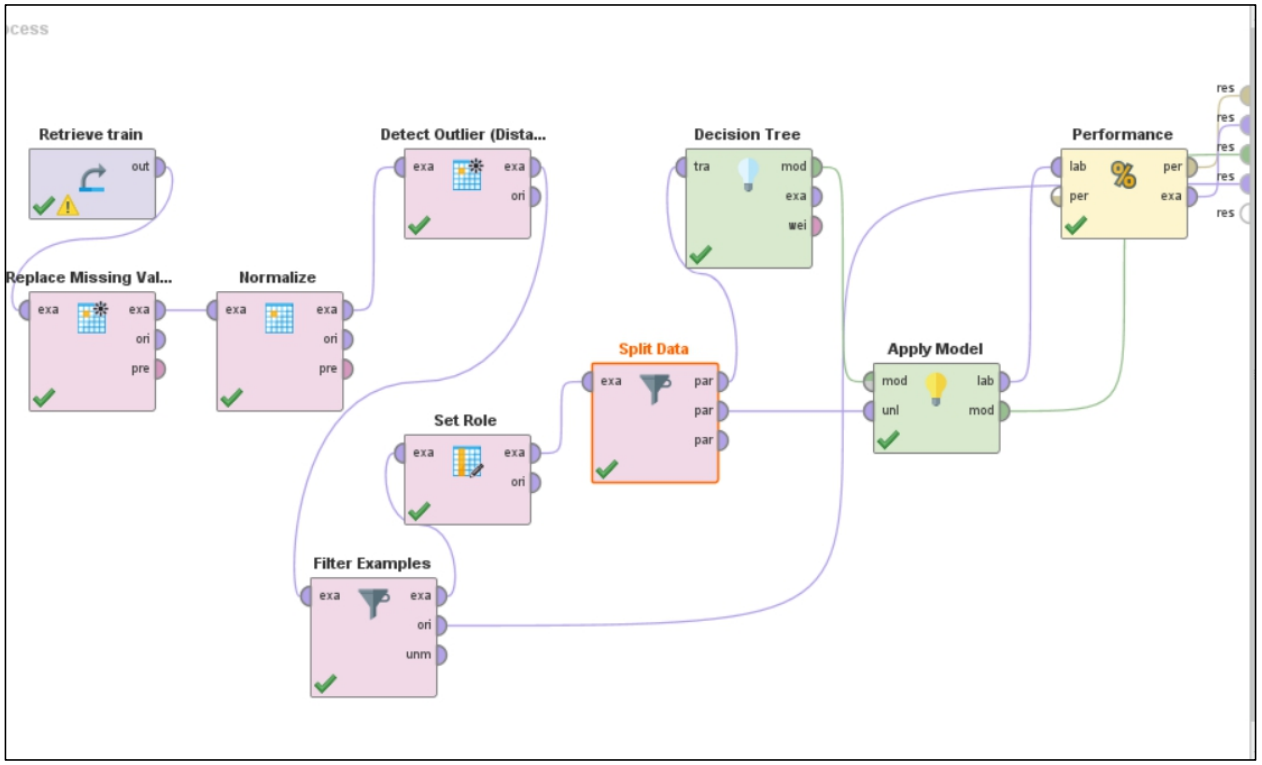
3. Predicting Loan Eligibility

```
Gender = input("Gender (0.0 for Male 1.0 for Female): ")
Married = input("Married (1.0 for Married 0.0 for Unmarried): ")
Dependents = input("Dependents (0 for Independent 1 for Dependent): ")
Education = input("Education (0 for Not graduated 1 for Graduated): ")
Self_Employed = input("Self_Employed (0.0 for No 1.0 for Yes): ")
ApplicantIncome =  input("Applicant Income (Eg.4583): ")
CoapplicantIncome = input("Coapplicant Income (Eg. 1516.0): ")
LoanAmount = input("Loan Amount in Rs. (Eg. 120.0):1
")
Loan_Amount_Term = input("Loan Amount Term in days (Eg. 360.0): ")
Credit_History = input("Credit History (Eg. 1.0 or 0.0): ")
Property_Area = input("Property Area  (0 for Urban, 1 for Rural, 2 for Semiurban): ")
userinput = [Gender, Married, Dependents, Education,
       Self_Employed, ApplicantIncome, CoapplicantIncome, LoanAmount,
       Loan_Amount_Term, Credit_History, Property_Area]

print("=====================================================")
print("Using Decision Tree Algorithm:", DecisionTree(userinput))
```

```
Gender (0.0 for Male 1.0 for Female): 0
Married (1.0 for Married 0.0 for Unmarried): 0
Dependents (0 for Independent 1 for Dependent): 0
Education (0 for Not graduated 1 for Graduated): 1
Self_Employed (0.0 for No 1.0 for Yes): 1
Applicant Income (Eg.4583): 5000
Coapplicant Income (Eg. 1516.0): 2000
Loan Amount in Rs. (Eg. 120.0): 700
Loan Amount Term in days (Eg. 360.0): 365
Credit History (Eg. 1.0 or 0.0): 1
Property Area  (0 for Urban, 1 for Rural, 2 for Semiurban): 0
=====================================================
Using Decision Tree Algorithm: ('Sorry! your not eligible for loan', 99.8371335504886)
```

```
Gender (0.0 for Male 1.0 for Female): 1
Married (1.0 for Married 0.0 for Unmarried): 0
Dependents (0 for Independent 1 for Dependent): 0
Education (0 for Not graduated 1 for Graduated): 1
Self_Employed (0.0 for No 1.0 for Yes): 1
Applicant Income (Eg.4583): 10000
Coapplicant Income (Eg. 1516.0): 10000
Loan Amount (Eg. 120.0): 25000
Loan Amount Term (Eg. 360.0): 360
Credit History (Eg. 1.0 or 0.0): 1
Property Area  (0 for Urban, 1 for Rural, 2 for Semiurban): 0
=====================================================
Using Decision Tree Algorithm: ('Congratulations your eligible for loan', 99.8371335504886)
```

4. RapidMiner Implementation

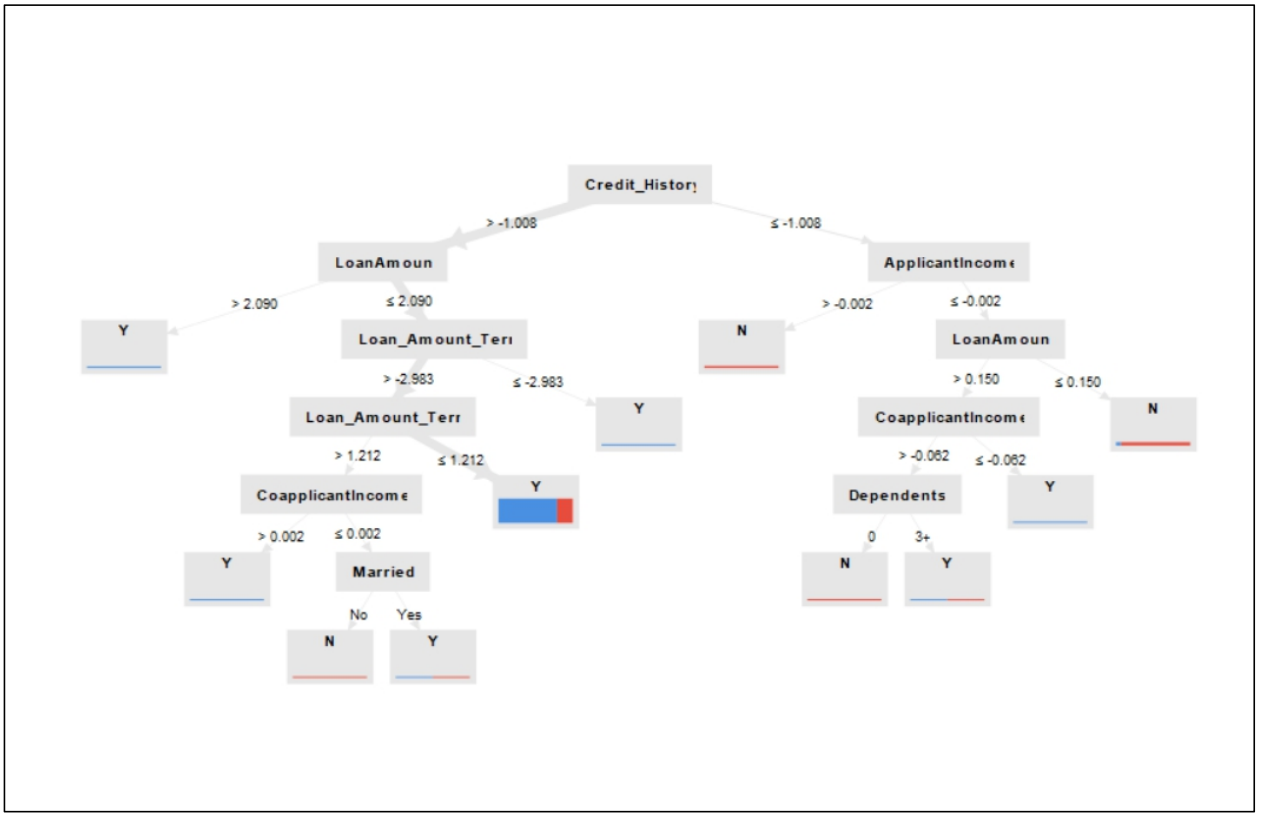


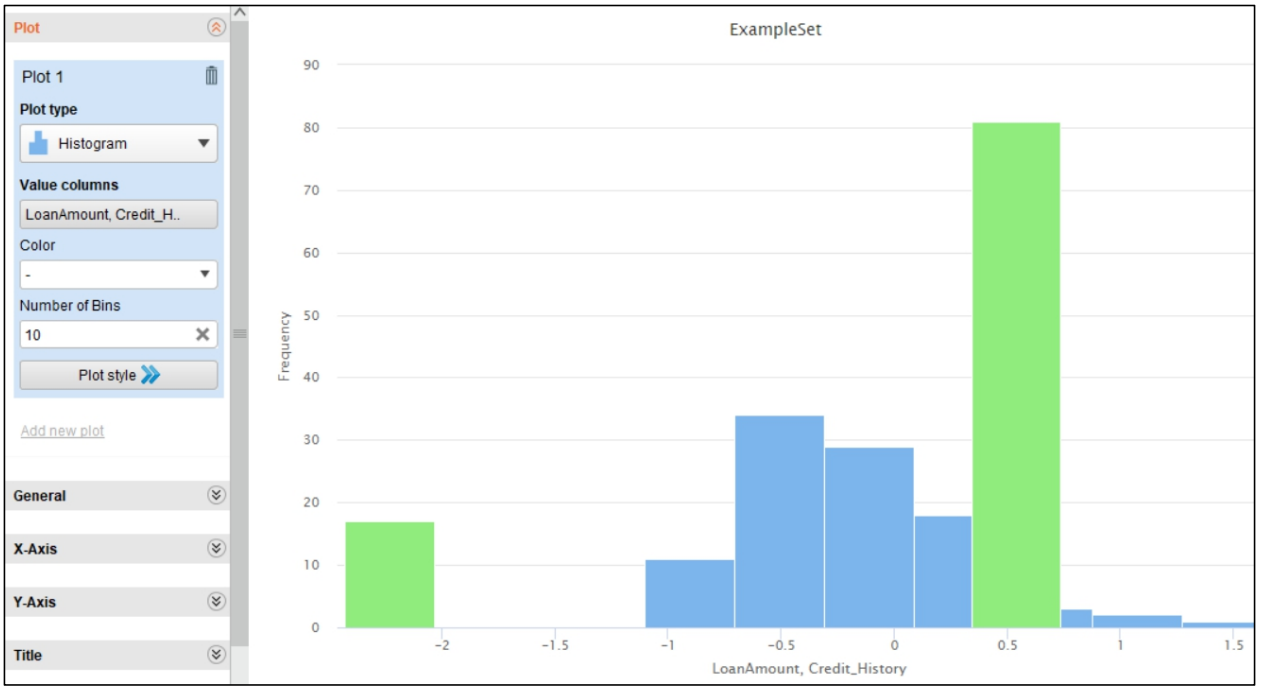
PerformanceVector

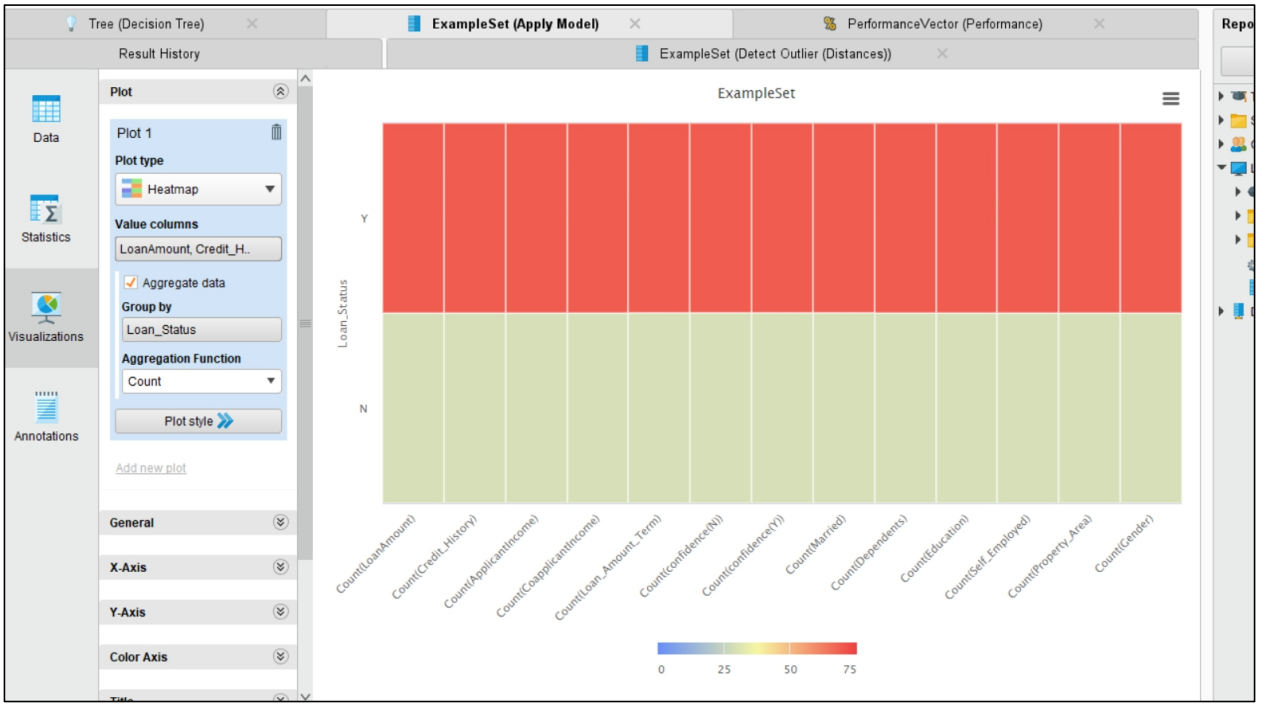
```
PerformanceVector:
accuracy: 86.73%
ConfusionMatrix:
True:   Y       N
Y:      68      12
N:      1       17
precision: 94.44% (positive class: N)
ConfusionMatrix:
True:   Y       N
Y:      68      12
N:      1       17
recall: 58.62% (positive class: N)
ConfusionMatrix:
True:   Y       N
Y:      68      12
N:      1       17
AUC (optimistic): 0.994 (positive class: N)
AUC: 0.787 (positive class: N)
AUC (pessimistic): 0.580 (positive class: N)
```

accuracy: 86.73%

	true Y	true N	class precision
pred. Y	68	12	85.00%
pred. N	1	17	94.44%
class recall	98.55%	58.62%	




Plot
Plot 1
Plot type
Histogram
Value columns
LoanAmount, Credit_H..
Color
-
Number of Bins
10
Plot style
Add new plot
General
X-Axis
Y-Axis
Title
ExampleSet
Frequency
LoanAmount, Credit_History


Tree (Decision Tree)
ExampleSet (Apply Model)
PerformanceVector (Performance)
Result History
ExampleSet (Detect Outlier (Distances))
Data
Statistics
Visualizations
Annotations
Plot
Plot 1
Plot type
Heatmap
Value columns
LoanAmount, Credit_H..
Aggregate data
Group by
Loan_Status
Aggregation Function
Count
Plot style
Add new plot
General
X-Axis
Y-Axis
Color Axis
ExampleSet
Loan_Status
Y
N
Count(LoanAmount)
Count(Credit_History)
Count(ApplicantIncome)
Count(CoapplicantIncome)
Count(Loan_Amount_Term)
Count(confidence(N))
Count(confidence(Y))
Count(Married)
Count(Dependents)
Count(Education)
Count(Self_Employed)
Count(Property_Area)
Count(Gender)
0
25
50
75

5. Conclusion

From the data analysis we can conclude :

- The chances of getting a loan will be higher for:

    - Applicants having a credit history (we observed this in exploration.)
    - Applicants with higher applicant and co-applicant incomes
    - Applicants with higher education level
    - Properties in urban areas with high growth perspectives

In this project we employed machine learning approaches particularly Decision tree algorithm to study the Loan dataset. The algorithm performed relatively well with a prediction accuracy ranging between 83% and 87%. Thus, this algorithm is very suitable for bank data analytics and prediction of Loan Eligibility in particular. More room for improvement can be found in the dataset. As demonstrated before, the precision of the algorithms increases when the size of dataset is increased. Hence, more data will surely make the model more accurate in detecting frauds and reduce the number of false positives. However, this requires official support from the banks themselves.

Thus by properly analyzing positive qualities and constraints, it can be concluded with confidence that the Decision Tree Algorithm is extremely efficient and gives a better result when compared to other models. It works correctly and fulfills all requirements of bankers and can be connected to many other systems. Machine learning helps to understand the factors which affect the specific outcomes most.