International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 09 | September - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Loan Distribution System Using Machine Learning

Karnati Sai Prashanth, Tsaliki Satya Ganesh Kumar, Garapati Venkata Krishna Rayalu, Svs Dhanush, Chandu Janakiram

INTRODUCTION

1. Loan-Prediction - This is the method a machine learning algorithm uses to determine whether or not a person will be approved for a loan.

2. The first and most important step is to comprehend the problem description. This would enable you to anticipate your challenges in advance. View the problem statement first.

3. Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan. Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customer segments, those are eligible for loanamount so that they can specifically target these customers.

4. It is a classification issue where we must determine whether or not a loan will be granted. We must forecast discrete values in a classification problem using the set of available independent variables. Two different classifications are possible:

5. Binary Classification: In this classification, we must choose between the two classes that are provided. For instance, identifying a person's gender as male or female, forecasting a win or loss, etc. Multiclass Classification: In this case, the data must be divided into at least three classes. For instance, you may categorise a film's genre as comedy, action, or romantic, or you could categorise fruits as oranges, apples, or pears.

6. Each retail bank encounters the very common challenge of loan projection at least once during its existence. At the conclusion of a retail bank, it can save a lot of man hours if done correctly.

Steps involved in machine learning

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 whichwe 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.**Choose a Model**

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

3. Train the Model

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

4. Evaluate the Model

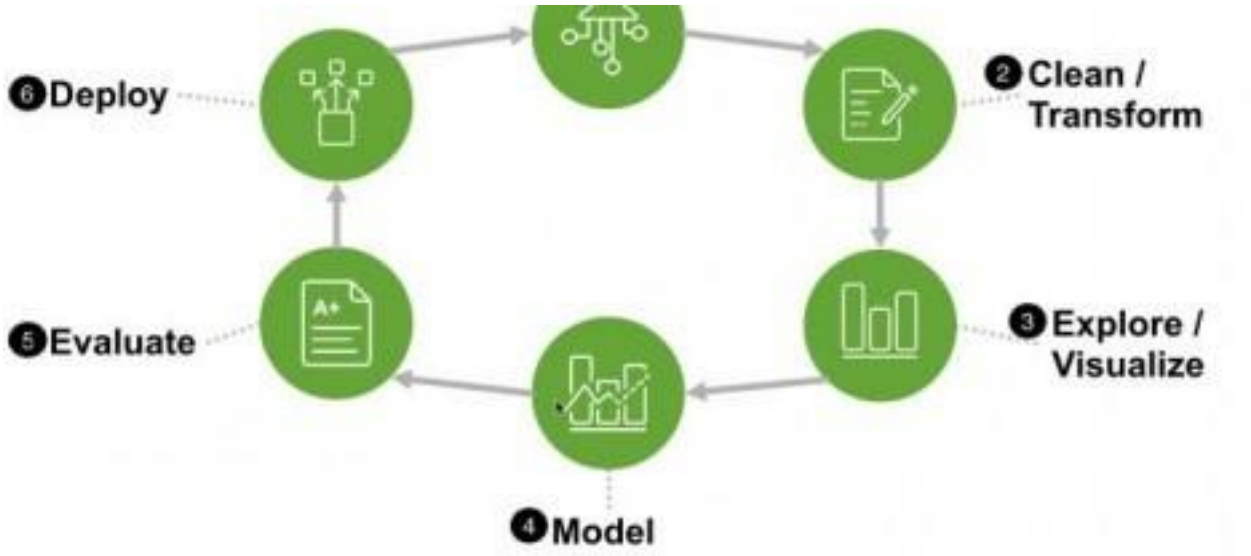
- Uses some metric or combination of metrics to "measure" objectiveperformance of model
- Test the model against previously unseen data
- This unseen data is meant to be somewhat representative of modelperformance in the real world, but still helps tune the model (as opposed to test data, which does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

5. Parameter Tuning

- This step refers to *hyper-parameter* tuning, which is an "art form" asopposed 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.

6. 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.



DATASETS

- Here we have two datasets. First is train_dataset.csv, test_dataset.csv.
- These are datasets of loan approval applications which are featuredwith annualincome, 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 loanstatus with the help of training dataset.
- These two datasets are already uploaded on google colab.

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 topay the loan.
- Credit_History – Whether the loan applicant has taken loan previouslyfrom same bank.
- Property_Area – This is about the area where the person stays(Rural/Urban).

LABELS

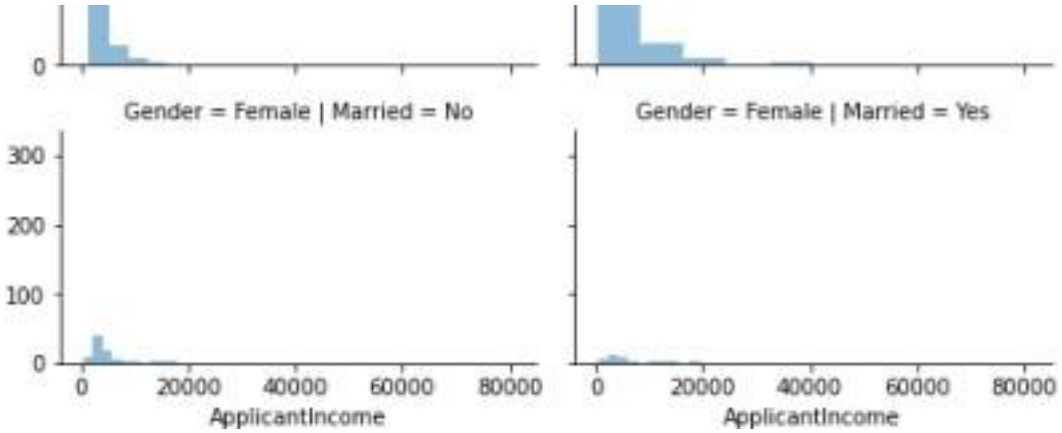
- LOAN_STATUS – Based on the mentioned features, the machine learning algorithm decides whether the person should be give loan ornot.

Visualizing data using google ColabCode and output

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

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

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

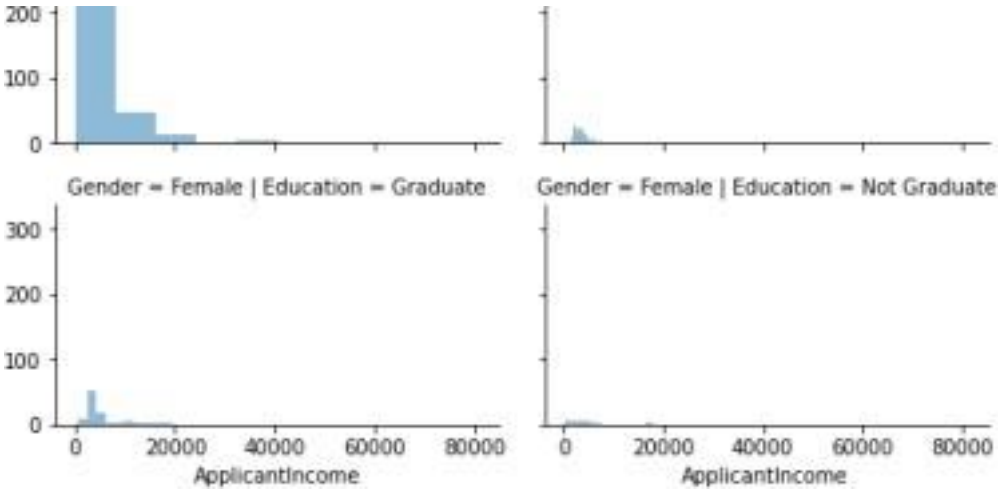
Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
LP001005	Male	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

```
# This code visualizes the people applying for loan who are categorized based on genderand 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()
```



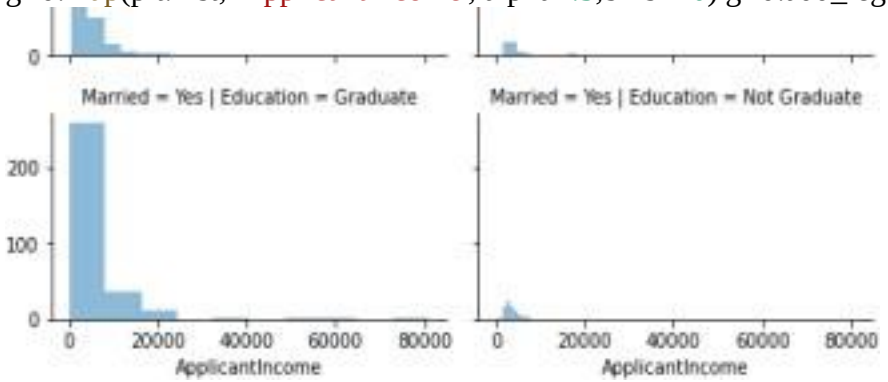
#

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()



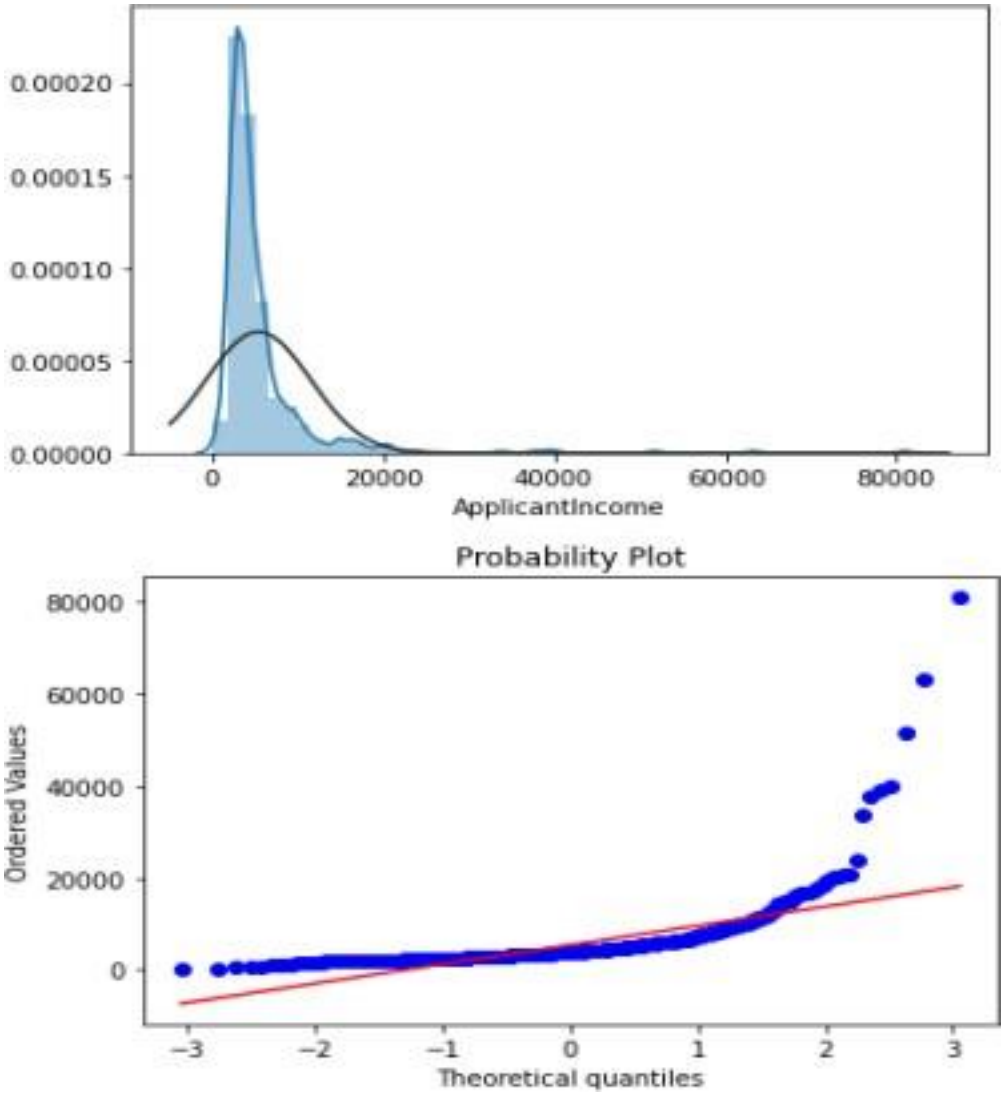
#

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()

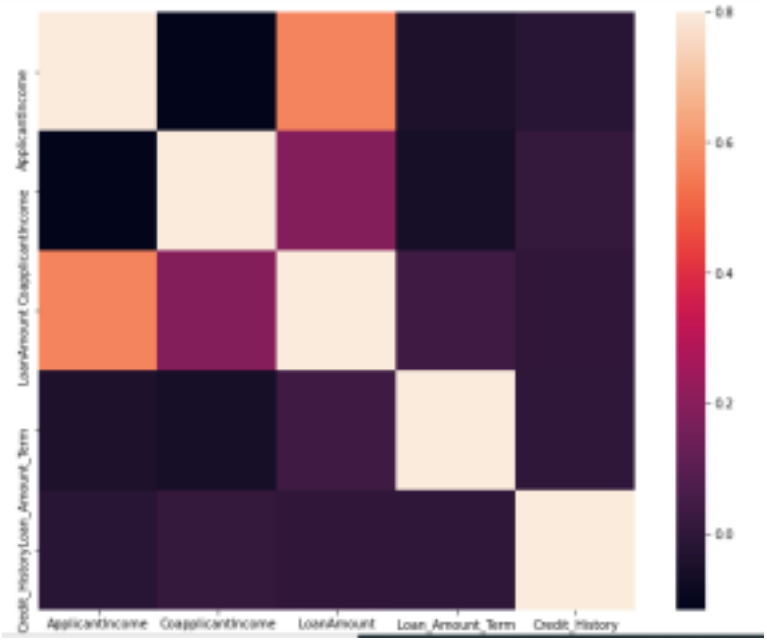


#histogram and normal probability plot sns.distplot(df_train['ApplicantIncome'], fit=norm); fig =

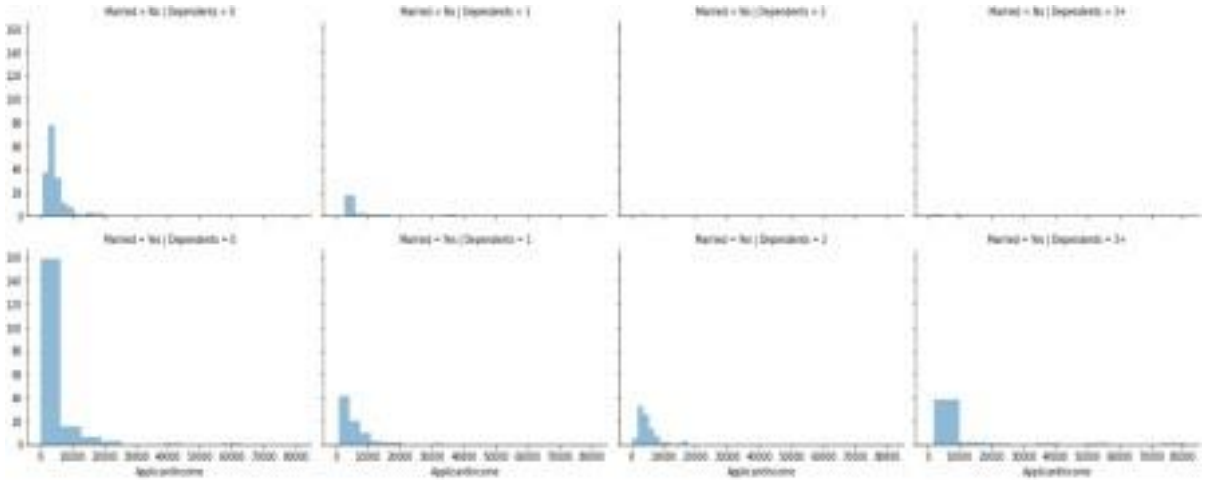
```
plt.figure()
res = stats.probplot(df_train['ApplicantIncome'], plot=plt)
```




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

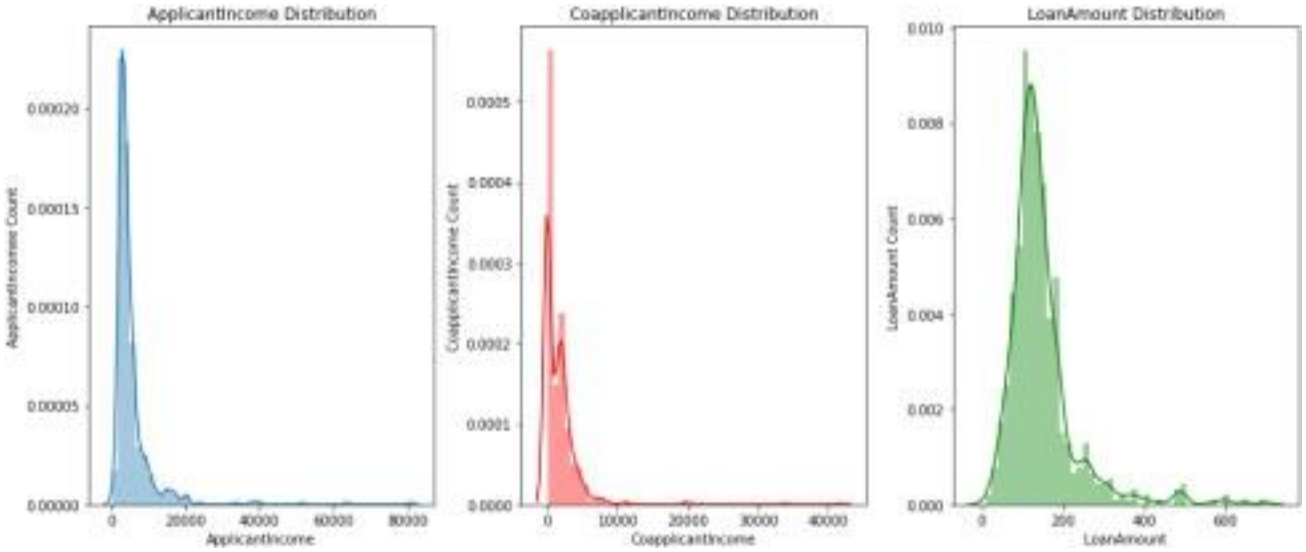


# This graph depicts the combination of applicant income, married people anddependent people in a family
grid = sns.FacetGrid(df_train, row='Married', col='Dependents', size=3.2, aspect=1.6) grid.map(plt.hist, 'ApplicantIncome', alpha=.5, bins=10)
grid.add_legend()

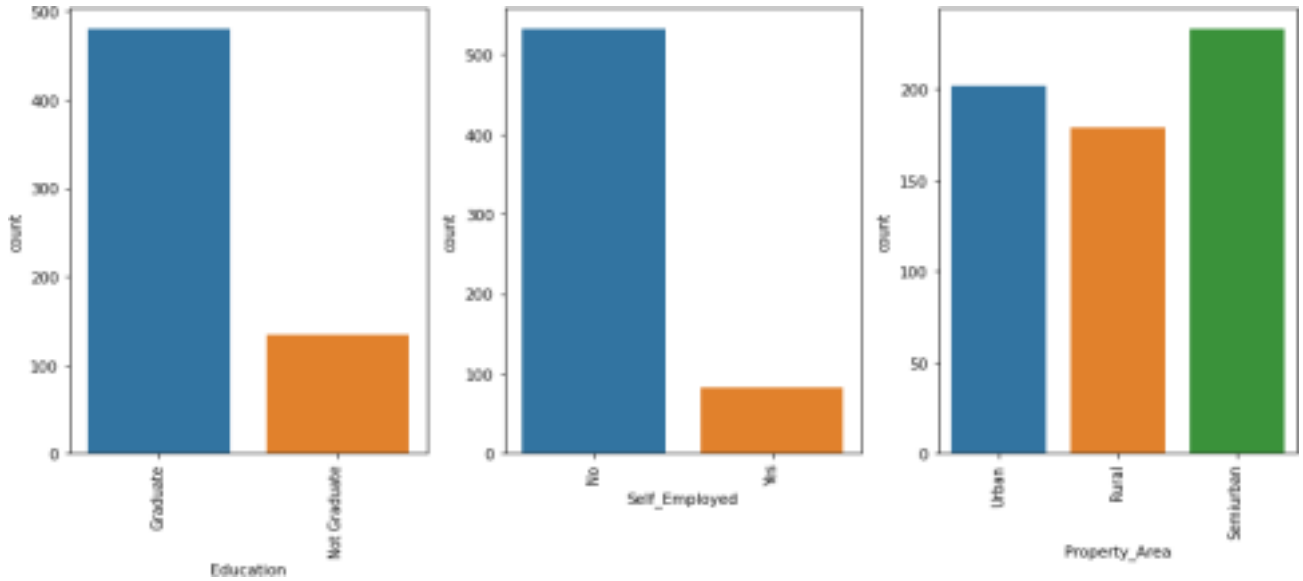


The graph which differentiates the applicant income distribution, Coapplicant incomedistribution, loan amount distribution
flg, axes = plt.subplots(nrows = 1, ncols = 3, figsize = (14,6)) sns.distplot(df_train['ApplicantIncome'], ax = axes[0]).set_title('ApplicantIncome Distribution')
axes[0].set_ylabel('ApplicantIncomee Count')

```
sns.distplot(df_train['CoapplicantIncome'], color = "r", ax = axes[1]).set_title('Coapplican tIncome 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()
```




```
# 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()
```



Explanation of the Main Code using GoogleColab 1. Logistic Regression model

```
# Importing required Librariesimport pandas as pd
import numpy as np # For mathematical calculationsimport 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	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

```
# take a look at the top 5 rows of the test set, notice the absense of "Loan_Status" that wewill predict
test.head()
```

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

```
# Handling Missing Values

# Check How many Null Values in each columnstrain.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 Exitstrain.isnull().sum()

# Test Check How many Null Values in each columnstest.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 Exitstest.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].valuesy = train.iloc[:, -1].values

# Converting Categorical variables into dummy
from sklearn.preprocessing import LabelEncoder,OneHotEncoderlabelencoder_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 setfrom 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 StandardScalersc = 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(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

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

# Printing values of whether loan is accepted or rejectedy_pred[:100]
```

```
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', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y'], dtype=object)
```

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

# implementing the confusion matrix
from sklearn.metrics import confusion_matrixcm = 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')
```

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

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 rejectedy_pred[:100]


#confusion matrix

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

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 Classifiaction 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 orrejected y_pred[:100]


#confusion matrix

# Check Accuracy
from sklearn.metrics import accuracy_scoreaccuracy_score(y_test,y_pred) 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_tr ain, cv = 10)

accuracies.mean() # accuracies.std()

0.7922448979591836
```

Loan prediction models comparison

Loan Prediction	Accuracy	Accuracy using K-fold Cross Validation
Using Logistic Regression	**0.8373983739837398**	**0.8024081632653062**
Using Random Forest Classification	**0.6910569105691057**	**0.7148163265306122**
Using Decision Tree Classification	**0.8292682926829268**	**0.7922448979591836**

SUMMARY

This machine learning paper's goal is to build a model that can approve or refuse loans. There are currently three models that we may train and test to determine whether or not additional candidates will receive loans. The accuracy of the first model, which uses a logistic regression model, is 0.8373, and it is 0.8024 when accuracy is measured using k-fold cross validation. Using k-fold cross validation, the second model provides accuracy values of 0.6910 and 0.7148. Accuracy values for the third model are 0.8292 and 0.7922. The accuracy of logistic regression is the best of all the models. A dataset was used to train this logistic regression model, and a different dataset was used to test it.