EDA Loan Status Prediction

Project Description:

In India, the number of people applying for loans has increased for various reasons in recent years. The bank employees are not able to analyse or predict whether the customer can pay back the amount or not (good customer or bad customer) for the given interest rate. The aim of this paper is to find the nature of the client applying for the personal loan. An exploratory data analysis technique is used to deal with this problem.

The result of the analysis shows that short term loans are preferred by the majority of the clients and the clients majorly apply loans for debt consolidation. The results are shown in graphs that help the bankers to understand the client's behavior.

Solution:

Here to make the process simple & easy . The bankers can get the data within a few minutes. Whether the person is eligible or Not for loan application. Customers first

apply for a home loan after that company validates the customer eligibility for the loan. However doing this manually takes a lot of time.

Hence it wants to automate the loan eligibility process (real time) based on customer information So the final thing is to identify the factors/ customer segments that are eligible for taking loan. How will the company benefit if we give the customer segments is the immediate question that arises? The solution is, banks would give loans to only those customers that are eligible so that they can be assured of getting the money back.

Steps involved in this machine learning project:

Following are the steps involved in creating a well-defined ML project:

- Understand and define the problem
- Analyse and prepare the data
- Apply the algorithms
- Reduce the errors
- Predict the result

Implementation:

1. Download the loan prediction data set from kaggle.

2. Import necessary python libraries. Import numpy, matplotlib, pandas and seaborn.

```
Import numpy as np

Import pandas as pd

Import matplotlib.pyplot as plt

Import seaborn as sns
```

We have renamed the libraries with aliases for simplicity.

3. Read test data set and training data set. The data set has to be divided into two parts. One is a training data set and another is a training data set. Preferably you should take 80% as a training set and 20% as a test data set.

```
df_train = pd.read_csv(“training.csv”)
```

4. Check for missing values. Drop the rows and columns that are not needed in this project.

5. You can also fix the values that are not present in the data set or have null values.

6. Now let us visualize the data. In x-axis, we will be plotting count and in the y-axis, we will plot gender.

Sns.countplot(y = ‘gender’, hue = ‘loan’, data = df_train).

Now continue the same process with married age, employment status, credit history and other data (fields) you have in the data set. Just change the y-axis accordingly while the x-axis will remain the same. Credit history includes data of the customers’ previous loan whether the person was a defaulter in his previous transactions.

Lets load the required libraries for our analysis

```
#Load libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import model_selection

from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

Splitting the Data Set:

As we have seen already, In Machine learning we have two kinds of datasets

- Training dataset - used to train our model
- Testing dataset - used to test if our model is making accurate predictions

Data:

Overview of the data

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

```
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
```

Evaluating the model and training the Model:

We are going to apply the below four algorithms to this problem and evaluate its effectiveness. And finally choose the best algorithm and train it.

- **Logistic Regression** : Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables

```
model = LogisticRegression()
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(accuracy_score(y_test, predictions))
```

- **Decision tree** : Decision tree is a type of supervised learning algorithm (having a predefined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables

```
model = DecisionTreeClassifier()
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(accuracy_score(y_test, predictions))
```

- **Random forest** : Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees

```
model = RandomForestClassifier(n_estimators=100)
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(accuracy_score(y_test, predictions))
```

Conclusion:

Predicting a person will be able to pay the debts manually is a tiring job and not always as accurate as we get when we use Machine Learning.