MAJOR PROJECT

LOAN PREDICTION DATASET

*Project by **Meet Vijay Raychura***
Data Science

Task 1:

You are provided with a loan sanction dataset where you will have to identify whether the loan of a particular person is approved or not depending on the information the individual has provided.

The dataset consists of the following attributes: **Atrribute**	**Significance**
Loan_id	Unique loan Id
Gender	Male/female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant education (Graduate/ Under graduate)
Self-employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_history	Credit history meet guidelines

Property_area	Urban / Semi urban / Rural
Loan_status	Loan approved (Y/N)

Steps for you to follow:

- Import all the necessary libraries
- Import the dataset provided
- Understand the data
- Deal with the missing values if any
- Do some visualization if necessary
- Divide the dataset into training and test datasets
- Build the machine learning model which ever is suitable for the dataset
- Fit the model on the training dataset
- Test the model and find the accuracy of the model on the test and the training datasets
- Create a confusion matrix

At last, draw conclusions based on the dataset provided and document the same on the jupyter notebook
Most importantly, please comment the usefulness of each cell.

Task- 2

On the same dataset draw conclusions from the dataset and create a tableau dashboard for the same.

Task-1

```
#Import dataframe from csv file.

import pandas as pd

df = pd.read_csv("/content/loan.csv")
df
```

```
#DATA PRE-PROCESSING

#Remove unnecessary columns

df = df.drop(['Loan_ID'], axis='columns')
```

```
df
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows x 12 columns

```
#DATA PRE-PROCESSING
```

```
#Remove rows with null values

df.dropna(inplace=True)
```

```
df
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

480 rows × 12 columns

```
# Checking if null values still remain

df['Loan_Status'].isnull().sum()
```

```
0
```

```
df = df.drop(['Dependents'],axis='columns')
```

```
# Taking every column except target column Loan_Status as input for model.

inputs = df.drop(['Loan_Status'], axis='columns')
```

```
inputs
```

	Gender	Married	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
1	Male	Yes	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural
2	Male	Yes	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban
3	Male	Yes	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban
4	Male	No	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban
5	Male	Yes	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban
...	...	...	...	...	...	...	...	...	...	...
609	Female	No	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural
610	Male	Yes	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural
611	Male	Yes	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban
612	Male	Yes	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban
613	Female	No	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban

480 rows × 10 columns

```
#Label encode each value from categorical to numerical
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
inputs['Gender'] = le.fit_transform(inputs['Gender'])
inputs['Married'] = le.fit_transform(inputs['Married'])
inputs['Education'] = le.fit_transform(inputs['Education'])
inputs['Self_Employed'] = le.fit_transform(inputs['Self_Employed'])
inputs['Property_Area'] = le.fit_transform(inputs['Property_Area'])
```

	Gender	Married	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
1	1	1	0	0	4583	1508.0	128.0	360.0	1.0	0
2	1	1	0	1	3000	0.0	66.0	360.0	1.0	3
3	1	1	1	0	2583	2358.0	120.0	360.0	1.0	3
4	1	0	0	0	6000	0.0	141.0	360.0	1.0	3
5	1	1	0	1	5417	4196.0	267.0	360.0	1.0	3
...	...	...	...	...	...	...	...	...	...	...
609	0	0	0	0	2900	0.0	71.0	360.0	1.0	0
610	1	1	0	0	4106	0.0	40.0	180.0	1.0	0
611	1	1	0	0	8072	240.0	253.0	360.0	1.0	3
612	1	1	0	0	7583	0.0	187.0	360.0	1.0	3
613	0	0	0	1	4583	0.0	133.0	360.0	0.0	2

480 rows × 10 columns

```
# Loan_Status column in target variable. It is then label encoded.

target = df.drop(inputs, axis='columns')
target['Loan_Status'] = le.fit_transform(target['Loan_Status'])
```

```
target
```

	Loan_Status
1	0
2	1
3	1
4	1
5	1
...	...
609	1
610	1
611	1

	Loan_Status
612	1
613	0

480 rows × 1 columns

```
# Importing accuracy score libraries and setting test_size to 20%.

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

inputs_train, inputs_test, target_train, target_test =
train_test_split(inputs,target,test_size=0.2)
```

```
from sklearn import tree
model = tree.DecisionTreeClassifier()
```

```
# Fitting our training data to DecisionTreeClassifier.

model.fit(inputs_train, target_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
# Testing model accuracy score by putting against previously unseen
testing data.

model.score(inputs_test,target_test)
```

0.71875

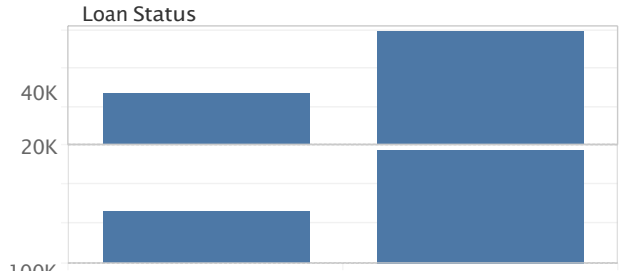
```
# Creating confusion matrix.

from sklearn.metrics import confusion_matrix
target_pred= model.predict(inputs_test)
cm = confusion_matrix(target_test, target_pred)
cm
```

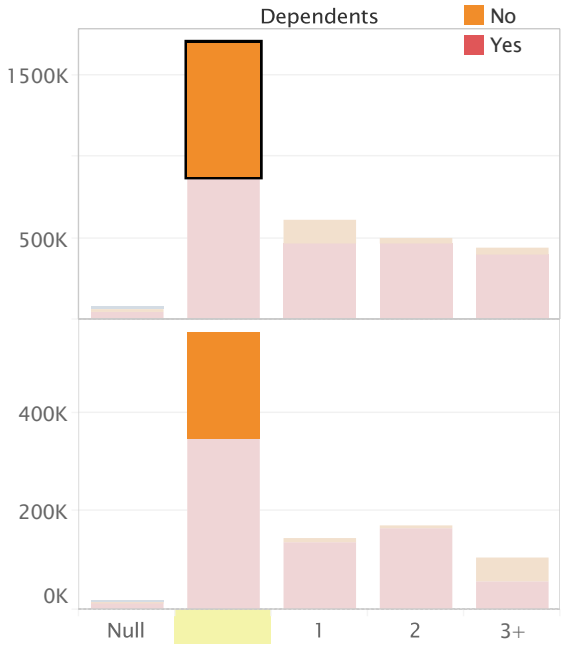
```
array([[20, 8], [19, 49]])
```

Task-2

Sheet 1



Sheet 2



Sheet-3

Property Area Rural Semiurban Urban

Education / Gender / Property Area