LOAN PREDICTION(PROJECT 1)

TUSAR KUMAR SAMAL

LOAN PREDICTION

The main income earning assets for a bank are loan, but bank are not providing everyone to taking loan from the bank. There are so many criteria to predict the loan like gender, age, income source, credit history and others.

INTRODUCTION

A person wants to automate the loan eligibility process based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, LoanAmount, Credit History and others. To automate this process, they have given a problem to identify the customers'segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a partial data set.

DATA UNDERTSTANDING

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Sklearn

STEPS ARE USING TO FINDOUT THE LOAN PREDICTION

- Importing Data with read_csv()
- Train test
- Univariate Analysis
- Visualising Variables
- Logistic Regression
- Random Forest

1.Importing Data with read_csv() :

The first step to any data science project is to import your data. Often, you'll work with data in Comma Separated Value (CSV) files and run into problems at the very start of your workflow.

```
train = pd.read_csv("E:\\pin2\\New folder\\PYdata\\train.csv")
test = pd.read_csv("E:\\pin2\\New folder\\PYdata\\test.csv")
```

```
train_original = train.copy()
test_original = test.copy()
```

```
train.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000

2.Univariate Analysis:

Univariate analysis is the simplest form of analyzing data. "Uni" means "one", so in other words your data has only one variable. It doesn't deal with causes or relationships (unlike regression) and it'smajor purpose is to describe; it takes data, summarizes that data and finds patterns in the data.

```
In [7]: # Univariate Analysis
        print(train.shape)
        print(test.shape)

        (614, 13)
        (367, 12)

In [9]: #proportion will be
        train['Loan_Status'].value_counts()

Out[9]: Y    422
        N    192
        Name: Loan_Status, dtype: int64
```

3.Visualizing variable:

There are 3 types of visualizing the data

i. Categorical

ii. ordinal

iii. Numerical

Logistic regression:

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

In [40]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.3)
```

In [41]:
```
                    #LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state = 0)
```

In [42]:
```
model.fit(xtrain,ytrain)
```

```
C:\Users\tusar_000\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be chan
ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\tusar_000\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[42]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=0, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [43]:
```
model.score(xtest,ytest)
```

Out[43]:
```
0.8402777777777778
```

In [44]:
```
model.score(xtrain,ytrain)
```

Out[44]:
```
0.7976190476190477
```

Random forest: A random forest is a data construct applied to machine learning that develops large numbers of random decision trees analyzing sets of variables. This type of algorithm helps to enhance the ways that technologies analyze complex data.

```
from sklearn.ensemble import RandomForestClassifier
model_random = RandomForestClassifier(n_estimators = 60, max_depth =1,random_state = 0,max_features=7)
```

In [46]:
```
model_random.fit(xtrain,ytrain)
```

```
C:\Users\tusar_000\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: DataConversionWarning: A column-vec
n a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  """Entry point for launching an IPython kernel.
```

Out[46]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=1, max_features=7, 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, n_estimators=60, n_jobs=None,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [47]:
```
model_random.score(xtrain,ytrain)
```

Out[47]:
```
0.7946428571428571
```

In [48]:
```
model_random.score(xtest,ytest)
```

THANK YOU