In [1]:

```
1 import pandas as pd
```

In [2]:

```
1 df = pd.read_csv('LoanApprovalPrediction.csv')
```

In [3]:

```
1 df.head()
```

Out[3]:

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

In [4]:

```
1 df.shape
```

Out[4]:

```
(598, 13)
```

In [5]:

```
1 df.info
```

Out[5]:

```
<bound method DataFrame.info of       Loan_ID  Gender Married  Dependents     Education Self_Employed  \
0    LP001002    Male      No         0.0      Graduate            No
1    LP001003    Male     Yes         1.0      Graduate            No
2    LP001005    Male     Yes         0.0      Graduate           Yes
3    LP001006    Male     Yes         0.0  Not Graduate            No
4    LP001008    Male      No         0.0      Graduate            No
..        ...     ...     ...         ...           ...           ...
593  LP002978  Female      No         0.0      Graduate            No
594  LP002979    Male     Yes         3.0      Graduate            No
595  LP002983    Male     Yes         1.0      Graduate            No
596  LP002984    Male     Yes         2.0      Graduate            No
597  LP002990  Female      No         0.0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5849                0.0         NaN             360.0
1               4583             1508.0       128.0             360.0
2               3000                0.0        66.0             360.0
3               2583             2358.0       120.0             360.0
4               6000                0.0       141.0             360.0
..               ...                ...         ...               ...
593             2900                0.0        71.0             360.0
594             4106                0.0        40.0             180.0
595             8072              240.0       253.0             360.0
596             7583                0.0       187.0             360.0
597             4583                0.0       133.0             360.0

     Credit_History Property_Area Loan_Status
0               1.0         Urban           Y
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
..              ...           ...         ...
593             1.0         Rural           Y
594             1.0         Rural           Y
595             1.0         Urban           Y
596             1.0         Urban           Y
597             0.0     Semiurban           N

[598 rows x 13 columns]>
```

In [6]:

```
df.describe()
```

Out[6]:

	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	586.000000	598.000000	598.000000	577.000000	584.000000	549.000000
mean	0.755973	5292.252508	1631.499866	144.968804	341.917808	0.843352
std	1.007751	5807.265364	2953.315785	82.704182	65.205994	0.363800
min	0.000000	150.000000	0.000000	9.000000	12.000000	0.000000
25%	0.000000	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	0.000000	3806.000000	1211.500000	127.000000	360.000000	1.000000
75%	1.750000	5746.000000	2324.000000	167.000000	360.000000	1.000000
max	3.000000	81000.000000	41667.000000	650.000000	480.000000	1.000000

In [7]:

```
df.isna().sum()
```

Out[7]:

```
Loan_ID               0
Gender                0
Married               0
Dependents           12
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           21
Loan_Amount_Term     14
Credit_History       49
Property_Area         0
Loan_Status           0
dtype: int64
```

In [8]:

```
#check the uniqueness of Loan Id column
df.Loan_ID.nunique()
```

Out[8]:

```
598
```

In [9]:

```
#drop the Loan_ID column as it is not required (no dupllicates)
df.drop(['Loan_ID'], axis = 1, inplace = True)
```

In [10]:

```
df.head()
```

Out[10]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	F
0	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	
1	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
3	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
4	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	

In [11]:

```
#Total missing value cells
df.isna().sum().sum()
```

Out[11]:

```
96
```

In [12]:

```
df.dtypes
```

Out[12]:

```
Gender               object
Married              object
Dependents          float64
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

In [13]:

```
#convert Gender column from object to int datatype
df.Gender = df.Gender.map({'Male' : 0, 'Female' : 1})
```

In [14]:

```
df.head()
```

Out[14]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	P
0	0	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	
1	0	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	0	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
3	0	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
4	0	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	

In [20]:

```
#Convert all categorical(object) columns using LabelEncoder
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
obj = (df.dtypes == 'object')
print(list(obj[obj].index))#List of categorical objects
for col in list(obj[obj].index):
    df[col] = label_encoder.fit_transform(df[col])
```

```
['Married', 'Education', 'Self_Employed', 'Property_Area', 'Loan_Status']
```

In [21]:

```
df.head()
```

Out[21]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	P
0	0	0	0.0	0	0	5849	0.0	NaN	360.0	1.0	
1	0	1	1.0	0	0	4583	1508.0	128.0	360.0	1.0	
2	0	1	0.0	0	1	3000	0.0	66.0	360.0	1.0	
3	0	1	0.0	1	0	2583	2358.0	120.0	360.0	1.0	
4	0	0	0.0	0	0	6000	0.0	141.0	360.0	1.0	

In [22]:

```
df.dtypes
```

Out[22]:

```
Gender                int64
Married               int32
Dependents          float64
Education             int32
Self_Employed         int32
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area         int32
Loan_Status           int32
dtype: object
```

In [23]:

```
df.columns
```

Out[23]:

```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [24]:

```
#fill in the missing rows with mean
for i in df.columns:
    df[i] = df[i].fillna(df[i].mean())
```

In [25]:

```
df.isna().sum()
```

Out[25]:

```
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
Loan_Status         0
dtype: int64
```

Training the model

In [26]:

```

X= df.drop(['Loan_Status'],axis=1)
y = df['Loan_Status']
```

In [27]:

```
X
```

Out[27]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0	0	0.0	0	0	5849	0.0	144.968804	360.0	1.0
1	0	1	1.0	0	0	4583	1508.0	128.000000	360.0	1.0
2	0	1	0.0	0	1	3000	0.0	66.000000	360.0	1.0
3	0	1	0.0	1	0	2583	2358.0	120.000000	360.0	1.0
4	0	0	0.0	0	0	6000	0.0	141.000000	360.0	1.0
...	...	...	...	...	...	...	...	...	...	...
593	1	0	0.0	0	0	2900	0.0	71.000000	360.0	1.0
594	0	1	3.0	0	0	4106	0.0	40.000000	180.0	1.0
595	0	1	1.0	0	0	8072	240.0	253.000000	360.0	1.0
596	0	1	2.0	0	0	7583	0.0	187.000000	360.0	1.0
597	1	0	0.0	0	1	4583	0.0	133.000000	360.0	0.0

598 rows × 11 columns

In [28]:

```
y
```

Out[28]:

```
0      1
1      0
2      1
3      1
4      1
      ..
593    1
594    1
595    1
596    1
597    0
Name: Loan_Status, Length: 598, dtype: int32
```

In [48]:

```
#Split the data into train and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =  train_test_split(X, y ,test_size=0.3, random_state=7)
```

Model Selection

In [49]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import RidgeClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [50]:

```
models = []
models.append(('Logistic Regression', LogisticRegression()))
models.append(('Ridge Classifier', RidgeClassifier()))
models.append(('Decision Tree Classifier', DecisionTreeClassifier()))
models.append(('K-Neighbors Classifier', KNeighborsClassifier()))
models.append(('Random Forest Classifier', RandomForestClassifier()))
```

In [51]:

```
def model_selection(model):
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    return accuracy_score(y_test,y_pred)*100
```

In [52]:

```
for name,model in models:
    #print(name,model)
    print(f'{name} : {model_selection(model)}')
```

```
Logistic Regression : 81.66666666666667
Ridge Classifier : 82.22222222222221
Decision Tree Classifier : 75.55555555555556
K-Neighbors Classifier : 66.11111111111111
Random Forest Classifier : 80.55555555555556
```

Ridge Classifier Performs better

In [ ]:

```

```