LOANS are the major requirement of the modern world. By this only, Banks get a major part of the total profit. It is beneficial for students to manage their education and living expenses, and for people to buy any kind of luxury like houses, cars, etc.

But when it comes to deciding whether the applicant's profile is relevant to be granted with loan or not. Banks have to look after many aspects.

So, here we will be using Machine Learning with Python to ease their work and predict whether the candidate's profile is relevant or not using key features like Marital Status, Education, Applicant Income, Credit History, etc.

Loan Approval Prediction using Machine Learning**

You can download the used data by visiting this link.**

https://github.com/makhan010385/DataSet/blob/main/LoanApprovalPrediction.csv

The dataset contains 13 features :

1 Loan A unique id

2 Gender Gender of the applicant Male/female

3 Married Marital Status of the applicant, values will be Yes/ No

4 Dependents It tells whether the applicant has any dependents or not.

5 Education It will tell us whether the applicant is Graduated or not.

6 Self_Employed This defines that the applicant is self-employed i.e. Yes/ No

7 ApplicantIncome Applicant income

8 CoapplicantIncome Co-applicant income

9 LoanAmount Loan amount (in thousands)

10 Loan_Amount_Term Terms of loan (in months)

11 Credit_History Credit history of individual's repayment of their debts

12 Property_Area Area of property i.e. Rural/Urban/Semi-urban

13 Loan_Status Status of Loan Approved or not i.e. Y- Yes, N-No

Importing Libraries and Dataset

Firstly we have to import libraries :

Pandas – To load the Dataframe

Matplotlib – To visualize the data features i.e. barplot

Seaborn – To see the correlation between features using heatmap

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ] data= pd.read_csv("/content/drive/My Drive/Colab Notebooks/Dataset/LoanApprovalPrediction.csv")
```

```
[ ] data.head(5)
```

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

Data Preprocessing and Visualization

Get the number of columns of object datatype.

```
[ ] obj = (data.dtypes == 'object')
    print("Categorical variables:",len(list(obj[obj].index)))
```

```
Categorical variables: 7
```

As Loan_ID is completely unique and not correlated with any of the other column, So we will drop it using .drop() function.

```
[ ] # Dropping Loan_ID column
    data.drop(['Loan_ID'],axis=1,inplace=True)
```

```
[ ] data
```

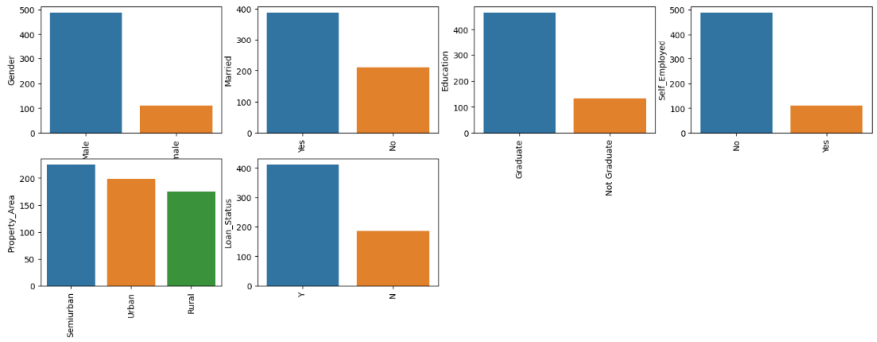
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
593	Female	No	0.0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y

594	Male	Yes	3.0	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
595	Male	Yes	1.0	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
596	Male	Yes	2.0	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
597	Female	No	0.0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

598 rows × 12 columns

Visualize all the unique values in columns using barplot. This will simply show which value is dominating as per our dataset.

```
obj = (data.dtypes == 'object')
object_cols = list(obj[obj].index)
plt.figure(figsize=(18,36))
index = 1

for col in object_cols:
  y = data[col].value_counts()
  plt.subplot(11,4,index)
  plt.xticks(rotation=90)
  sns.barplot(x=list(y.index), y=y)
  index +=1
```



As all the categorical values are binary so we can use Label Encoder for all such columns and the values will change into int datatype.

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how
# to understand word labels.
label_encoder = preprocessing.LabelEncoder()
obj = (data.dtypes == 'object')
for col in list(obj[obj].index):
  data[col] = label_encoder.fit_transform(data[col])
```

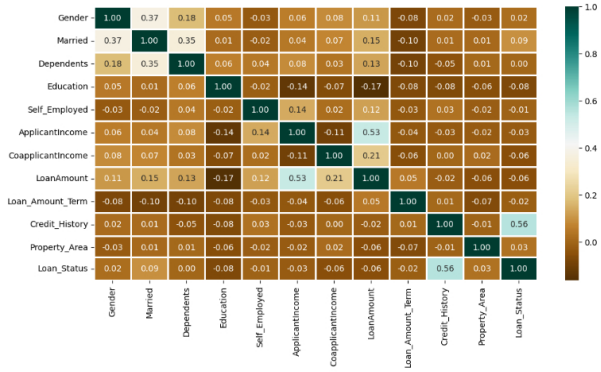
Again check the object datatype columns. Let's find out if there is still any left.

```
# To find the number of columns with
# datatype==object
obj = (data.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

```
Categorical variables: 0
```

```
plt.figure(figsize=(12,6))

sns.heatmap(data.corr(),cmap='BrBG',fmt='.2f',
            linewidths=2,annot=True)
```

```
<Axes: >
```



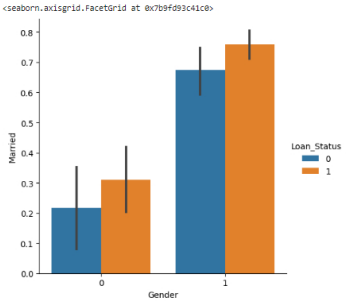
The above heatmap is showing the correlation between Loan Amount and ApplicantIncome. It also shows that Credit_History has a high impact on Loan_Status.

Now we will use Catplot to visualize the plot for the Gender, and Marital Status of the applicant.

```
sns.catplot(x="Gender", y="Married",
            hue="Loan_Status",
            kind="bar",
```

```
                data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7b9fd93c41c0>
```



Now we will find out if there is any missing values in the dataset using below code.

```
[ ] for col in data.columns:
      data[col] = data[col].fillna(data[col].mean())

    data.isna().sum()
```

```
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
```

As there is no missing value then we must proceed to model training.

Splitting Dataset

```
[ ]
    from sklearn.model_selection import train_test_split

    X = data.drop(['Loan_Status'],axis=1)
    Y = data['Loan_Status']
    X.shape,Y.shape

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                        test_size=0.4,
                                                        random_state=1)
    X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

```
((358, 11), (240, 11), (358,), (240,))
```

```
[ ] print(Y)
```

```
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: int64
```

Model Training and Evaluation

As this is a classification problem so we will be using these models :

KNeighborsClassifiers

RandomForestClassifiers

Support Vector Classifiers (SVC)

Logistics Regression

```
[ ] from sklearn.neighbors import KNeighborsClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.svm import SVC
    from sklearn.linear_model import LogisticRegression

    from sklearn import metrics

    knn = KNeighborsClassifier(n_neighbors=3)
    rfc = RandomForestClassifier(n_estimators = 7,
                criterion = 'entropy',
                random_state =7)
    svc = SVC()
    lc = LogisticRegression()

    # making predictions on the training set
    for clf in (rfc, knn, svc,lc):
      clf.fit(X_train, Y_train)
      Y_pred = clf.predict(X_train)
      print("Accuracy score of ",
        clf.__class__.__name__,
        "=",100*metrics.accuracy_score(Y_train,
                        Y_pred))
```

```
Accuracy score of  RandomForestClassifier = 98.04469273743017
Accuracy score of  KNeighborsClassifier = 78.49162011173185
Accuracy score of  SVC = 68.71508379888269
Accuracy score of  LogisticRegression = 80.44692737430168
```

```
[ ] X_train
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
168	1	1	3.0	1	0	3522	0.0	81.0	180.000000	1.0	0
370	1	0	0.0	0	0	3069	0.0	71.0	480.000000	1.0	2
519	1	0	2.0	0	0	3588	0.0	110.0	360.000000	0.0	0

121	1	1	2.0	0	0	2957	0.0	81.0	360.000000	1.0	1
274	1	0	1.0	1	1	4053	2426.0	158.0	360.000000	0.0	2
...	...	...	...	...	...	...	...	...	...	...	...
129	1	0	0.0	0	0	2718	0.0	70.0	360.000000	1.0	1
144	1	1	1.0	0	0	1538	1425.0	30.0	360.000000	1.0	2
72	1	1	3.0	1	0	4755	0.0	95.0	341.917808	0.0	1
235	1	1	1.0	1	0	2510	1983.0	140.0	180.000000	1.0	2
37	1	0	0.0	0	0	4166	7210.0	184.0	360.000000	1.0	2

358 rows × 11 columns

Gender ,Merried,Education, Self Employeed,Property Area ,Loan status

```
[ ] Y_pred= clf.predict([[1,1,3.0,1,0,3522,0.0,81.0,180.000000,1.0,0]])
    print(Y_pred )
```

```
[1]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

```
[ ] print(X)
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
0         1        0         0.0          0              0             5849
1         1        1         1.0          0              0             4583
2         1        1         0.0          0              1             3000
3         1        1         0.0          1              0             2583
4         1        0         0.0          0              0             6000
..      ...      ...         ...        ...            ...              ...
593       0        0         0.0          0              0             2900
594       1        1         3.0          0              0             4106
595       1        1         1.0          0              0             8072
596       1        1         2.0          0              0             7583
597       0        0         0.0          0              1             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                  0.0  144.968804             360.0             1.0
1               1508.0  128.000000             360.0             1.0
2                  0.0   66.000000             360.0             1.0
3               2358.0  120.000000             360.0             1.0
4                  0.0  141.000000             360.0             1.0
..                 ...         ...               ...             ...
593                0.0   71.000000             360.0             1.0
594                0.0   40.000000             180.0             1.0
595              240.0  253.000000             360.0             1.0
596                0.0  187.000000             360.0             1.0
597                0.0  133.000000             360.0             0.0

     Property_Area
0                2
1                0
2                2
3                2
4                2
..             ...
593              0
594              0
595              2
596              2
597              1

[598 rows x 11 columns]
```

Double-click (or enter) to edit

Prediction on the test set:

```
[ ] # making predictions on the testing set
    for clf in (rfc, knn, svc,lc):
        clf.fit(X_train, Y_train)
        Y_pred = clf.predict(X_test)
        print("Accuracy score of ",
              clf.__class__.__name__,"=",
              100*metrics.accuracy_score(Y_test,
                                         Y_pred))
```

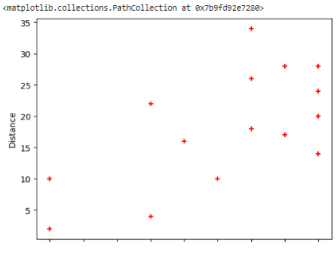
```
Accuracy score of  RandomForestClassifier = 82.5
Accuracy score of  KNeighborsClassifier = 63.74999999999999
Accuracy score of  SVC = 69.16666666666667
Accuracy score of  LogisticRegression = 80.83333333333333
```

```
[ ] # Import pandas library
    import pandas as pd

    # initialize list elements

    data={'speed': [4,4,7,7,8,9,10,10,10,11,11,12,12,12,12],
          'distance': [2,10,4,22,16,10,18,26,34,17,28,14,20,24,28]}
    # Create the pandas DataFrame with column name is provided explicitly
    df = pd.DataFrame(data, columns=['speed','distance'])

    # print dataframe.
    df
```

	speed	distance
0	4	2
1	4	10
2	7	4
3	7	22
4	8	16
5	9	10
6	10	18
7	10	26
8	10	34
9	11	17
10	11	28
11	12	14
12	12	20
13	12	24
14	12	28

```
[ ] %matplotlib inline
    plt.xlabel('Speed')
    plt.ylabel('Distance')
    plt.scatter(df.speed,df.distance,color='red',marker='+')
```

```
<matplotlib.collections.PathCollection at 0x7b9fd92e7280>
```



4 5 6 7 8 9 10 11 12
Speed

```
new_df = df.drop('speed',axis='columns')
new_df
```

	distance
0	2
1	10
2	4
3	22
4	16
5	10
6	18
7	26
8	34
9	17
10	28
11	14
12	20
13	24
14	28

```
speed = df.speed
speed
```

```
0     4
1     4
2     7
3     7
4     8
5     9
6    10
7    10
8    10
9    11
10   11
11   12
12   12
13   12
14   12
Name: speed, dtype: int64
```

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
```

```
# Create linear regression object
reg = linear_model.LinearRegression()
reg.fit(new_df,speed)
```

```
LinearRegression()
```

```
reg.predict([[10]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
array([7.69692848])
```