In [1]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\dell\appdata\local\programs\pyt
hon\python311\lib\site-packages (2.0.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\dell\appdata\lo
cal\programs\python\python311\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\appdata\local\progra
ms\python\python311\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\dell\appdata\local\prog
rams\python\python311\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\dell\appdata\local\progr
ams\python\python311\lib\site-packages (from pandas) (1.24.2)
Requirement already satisfied: six>=1.5 in c:\users\dell\appdata\local\programs\p
ython\python311\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:
```
import pandas as pd
```

In [3]:
```
data = pd.read_csv('C:/Users/DELL/Downloads/Data/Loan Status Prediction/Loan_Sta
```

Display top 5 rows of dataset

In [4]:
```
data.head(5)
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
0	LP001002	Male	No	0	Graduate	No	584
1	LP001003	Male	Yes	1	Graduate	No	458
2	LP001005	Male	Yes	0	Graduate	Yes	300
3	LP001006	Male	Yes	0	Not Graduate	No	258
4	LP001008	Male	No	0	Graduate	No	600

Display Last 5 rows of dataset

In [5]:
```
data.tail()
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
609	LP002978	Female	No	0	Graduate	No	2
610	LP002979	Male	Yes	3+	Graduate	No	4
611	LP002983	Male	Yes	1	Graduate	No	8
612	LP002984	Male	Yes	2	Graduate	No	7
613	LP002990	Female	No	0	Graduate	Yes	4

Shape of Dataset

In [6]:
```
data.shape
```

Out[6]: (614, 13)

Get information about Dataset

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
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)
memory usage: 62.5+ KB
```

Null Values in DataSet

In [8]:
```
data.isnull()
```

Out[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncon
0	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	
609	False	False	False	False	False	False	Fal
610	False	False	False	False	False	False	Fal
611	False	False	False	False	False	False	Fal
612	False	False	False	False	False	False	Fal
613	False	False	False	False	False	False	Fal

614 rows × 13 columns

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

In [10]:
```
data.isnull().sum()*100/len(data)
```

Out[10]:
```
Loan_ID             0.000000
Gender              2.117264
Married             0.488599
Dependents          2.442997
Education           0.000000
Self_Employed       5.211726
ApplicantIncome     0.000000
CoapplicantIncome   0.000000
LoanAmount          3.583062
Loan_Amount_Term    2.280130
Credit_History      8.143322
Property_Area       0.000000
Loan_Status         0.000000
dtype: float64
```

Missing Values

In [11]:
```
columns=['Gender','Dependents','LoanAmount','Loan_Amount_Term']
```

In [12]:
```
data=data.dropna(subset=columns)
```

In [13]:
```
data.isnull().sum()*100/len(data)
```

Out[13]:
```
Loan_ID              0.000000
Gender               0.000000
Married              0.000000
Dependents           0.000000
Education            0.000000
Self_Employed        5.424955
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           0.000000
Loan_Amount_Term     0.000000
Credit_History       8.679928
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

In [14]:
```
data['Self_Employed'].mode()[0]
```

Out[14]:
```
'No'
```

In [15]:
```
data['Self_Employed']=data['Self_Employed'].fillna(data['Self_Employed'].mode()[
```

In [16]:
```
data.isnull().sum()*100/len(data)
```

Out[16]:
```
Loan_ID              0.000000
Gender               0.000000
Married              0.000000
Dependents           0.000000
Education            0.000000
Self_Employed        0.000000
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           0.000000
Loan_Amount_Term     0.000000
Credit_History       8.679928
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

In [17]:
```
data['Credit_History'].mode()[0]
```

Out[17]:
```
1.0
```

In [18]:
```
data['Credit_History']=data['Credit_History'].fillna(data['Credit_History'].mode
```

In [19]:
```
data.isnull().sum()*100/len(data)
```

Out[19]:
```
Loan_ID              0.0
Gender               0.0
Married              0.0
Dependents           0.0
Education            0.0
Self_Employed        0.0
ApplicantIncome      0.0
CoapplicantIncome    0.0
LoanAmount           0.0
Loan_Amount_Term     0.0
Credit_History       0.0
Property_Area        0.0
Loan_Status          0.0
dtype: float64
```

Categorical Columns

In [20]:
```
data.sample(5)
```

Out[20]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
351	LP002140	Male	No	0	Graduate	No	8
17	LP001036	Female	No	0	Graduate	No	3
546	LP002768	Male	No	0	Not Graduate	No	3
527	LP002706	Male	Yes	1	Not Graduate	No	5
290	LP001936	Male	Yes	0	Graduate	No	3

In [21]:
```
data['Dependents']=data['Dependents'].replace(to_replace='3+',value='4')
```

In [22]:
```
data['Dependents'].unique()
```

Out[22]:
```
array(['1', '0', '2', '4'], dtype=object)
```

In [23]:
```
data['Gender'].unique()
```

Out[23]:
```
array(['Male', 'Female'], dtype=object)
```

In [24]:
```
data['Gender']=data['Gender'].map({'Male':1,'Female':0}).astype('int')
```

In [25]:
```
data['Gender'].unique()
```

Out[25]:
```
array([1, 0])
```

In [26]:
```
data['Married'].unique()
```

Out[26]:
```
array(['Yes', 'No'], dtype=object)
```

In [27]:
```
data['Married']=data['Married'].map({'Yes':1,'No':0}).astype('int')
```

In [28]:
```
data['Married'].unique()
```

Out[28]:
```
array([1, 0])
```

In [29]:
```
data['Education'].unique()
```

Out[29]:
```
array(['Graduate', 'Not Graduate'], dtype=object)
```

In [30]:
```
data['Education']=data['Education'].map({'Graduate':1,'Not Graduate':0}).astype(
```

In [31]:
```
data['Education'].unique()
```

Out[31]:
```
array([1, 0])
```

In [32]:
```
data['Self_Employed'].unique()
```

Out[32]:
```
array(['No', 'Yes'], dtype=object)
```

In [33]:
```
data['Self_Employed']=data['Self_Employed'].map({'Yes':1,'No':0}).astype('int')
```

In [34]:
```
data['Self_Employed'].unique()
```

Out[34]:
```
array([0, 1])
```

In [35]:
```
data['Property_Area'].unique()
```

Out[35]:
```
array(['Rural', 'Urban', 'Semiurban'], dtype=object)
```

In [36]:
```
data['Property_Area']=data['Property_Area'].map({'Urban':1,'Rural':0,'Semiurban'
```

In [37]:
```
data['Property_Area'].unique()
```

Out[37]:
```
array([0, 1, 2])
```

In [38]:
```
data.head()
```

Out[38]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
1	LP001003	1	1	1	1	0	458
2	LP001005	1	1	0	1	1	300
3	LP001006	1	1	0	0	0	258
4	LP001008	1	0	0	1	0	600
5	LP001011	1	1	2	1	1	541

In [39]:
```
data.drop('Loan_ID', axis=1, inplace=True)
```

In [40]:
```
data.head()
```

Out[40]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
1	1	1	1	1	0	4583	
2	1	1	0	1	1	3000	
3	1	1	0	0	0	2583	
4	1	0	0	1	0	6000	
5	1	1	2	1	1	5417	

In [41]:
```
data['Loan_Status'].unique()
```

Out[41]:
```
array(['N', 'Y'], dtype=object)
```

In [42]:
```
data['Loan_Status']=data['Loan_Status'].map({'Y':1,'N':0}).astype('int')
```

In [43]:
```
data.head()
```

Out[43]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
1	1	1	1	1	0	4583	
2	1	1	0	1	1	3000	
3	1	1	0	0	0	2583	
4	1	0	0	1	0	6000	
5	1	1	2	1	1	5417	

Store Feature Matrix in X and Response(Target) in Vector Y

In [44]:
```
x=data.drop('Loan_Status',axis=1)
```

In [45]:
```
y=data['Loan_Status']
```

In [46]:
```
y
```

Out[46]:
```
1      0
2      1
3      1
4      1
5      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 553, dtype: int32
```

Feature Scaling

In [47]:
```
data.head()
```

Out[47]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
1	1	1	1	1	0	4583	
2	1	1	0	1	1	3000	
3	1	1	0	0	0	2583	
4	1	0	0	1	0	6000	
5	1	1	2	1	1	5417	

In [48]:
```
cols=['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term']
```

In [49]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\dell\appdata\local\progra
ms\python\python311\lib\site-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\dell\appdata\local\progr
ams\python\python311\lib\site-packages (from scikit-learn) (1.24.2)
Requirement already satisfied: scipy>=1.3.2 in c:\users\dell\appdata\local\progra
ms\python\python311\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\dell\appdata\local\progr
ams\python\python311\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\dell\appdata\loca
l\programs\python\python311\lib\site-packages (from scikit-learn) (3.1.0)
[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [50]:
```
from sklearn.preprocessing import StandardScaler
st = StandardScaler()
x[cols]=st.fit_transform(x[cols])
```

In [51]:
```
x
```

Out[51]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapp
1	1	1	1	1	0	-0.128694	
2	1	1	0	1	1	-0.394296	
3	1	1	0	0	0	-0.464262	
4	1	0	0	1	0	0.109057	
5	1	1	2	1	1	0.011239	
...	...	...	...	...	...	...	
609	0	0	0	1	0	-0.411075	
610	1	1	4	1	0	-0.208727	
611	1	1	1	1	0	0.456706	
612	1	1	2	1	0	0.374659	
613	0	0	0	1	1	-0.128694	

553 rows × 11 columns

Splitting The Dataset Into The Training Set And Test Set & Applying K-Fold Cross Validation

In [52]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
import numpy as np
```

In [53]:
```
model_df={}
def model_val(model,x,y):
    x_train,x_test,y_train,y_test=train_test_split(x,y,
                                                   test_size=0.20,
                                                   random_state=42)
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    print(f"{model} accuracy is {accuracy_score(y_test,y_pred)}")

    score = cross_val_score(model,x,y,cv=5)
    print(f"{model} Avg cross val score is {np.mean(score)}")
    model_df[model]=round(np.mean(score)*100,2)
```

In [54]:
```
model_df
```

Out[54]: {}

Logistic Regression

In [55]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model_val(model,x,y)
```

```
LogisticRegression() accuracy is 0.8018018018018018
LogisticRegression() Avg cross val score is 0.8047829647829647
```

SVC

In [56]:
```
from sklearn import svm
model = svm.SVC()
model_val(model,x,y)
```

```
SVC() accuracy is 0.7927927927927928
SVC() Avg cross val score is 0.7938902538902539
```

Decision Tree Classifier

In [57]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model_val(model,x,y)
```

```
DecisionTreeClassifier() accuracy is 0.7297297297297297
DecisionTreeClassifier() Avg cross val score is 0.7306306306306307
```

Random Forest Classifier

In [58]:
```
from sklearn.ensemble import RandomForestClassifier
model =RandomForestClassifier()
model_val(model,x,y)
```

```
RandomForestClassifier() accuracy is 0.7567567567567568
RandomForestClassifier() Avg cross val score is 0.7848648648648648
```

Gradient Boosting Classifier

In [59]:
```
from sklearn.ensemble import GradientBoostingClassifier
model =GradientBoostingClassifier()
model_val(model,x,y)
```

```
GradientBoostingClassifier() accuracy is 0.7927927927927928
GradientBoostingClassifier() Avg cross val score is 0.7685503685503685
```

Hyperparameter Tuning

In [60]:
```
from sklearn.model_selection import RandomizedSearchCV
```

Logistic Regression

In [61]:
```
log_reg_grid={"C":np.logspace(-4,4,20),
              "solver":['liblinear']}
```

In [62]:
```
rs_log_reg=RandomizedSearchCV(LogisticRegression(),
                   param_distributions=log_reg_grid,
                  n_iter=20,cv=5,verbose=True)
```

In [63]:
```
rs_log_reg.fit(x,y)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

Out[63]:

▸ RandomizedSearchCV
▸ estimator: LogisticRegression
▸ LogisticRegression

In [64]:
```
rs_log_reg.best_score_
```

Out[64]:
```
0.8047829647829647
```

In [65]:
```
rs_log_reg.best_params_
```

Out[65]:
```
{'solver': 'liblinear', 'C': 0.23357214690901212}
```

SVC

In [66]:
```
svc_grid = {'C':[0.25,0.50,0.75,1],"kernel":["linear"]}
```

In [67]:
```
rs_svc=RandomizedSearchCV(svm.SVC(),
                  param_distributions=svc_grid,
                   cv=5,
                   n_iter=20,
                  verbose=True)
```

In [68]:
```
rs_svc.fit(x,y)
```

```
C:\Users\DELL\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_search.py:305: UserWarning: The total space of parameters 4 is smaller than n_iter=20. Running 4 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
Fitting 5 folds for each of 4 candidates, totalling 20 fits
```

Out[68]:

▸ RandomizedSearchCV
▸ estimator: SVC
▸ SVC

In [69]:
```
rs_svc.best_score_
```

Out[69]:
```
0.8066011466011467
```

In [70]:
```
rs_svc.best_params_
```

Out[70]:
```
{'kernel': 'linear', 'C': 0.25}
```

Random Forest Classifier

In [71]:
```
from sklearn.ensemble import RandomForestClassifier

# Create a RandomForestClassifier with max_features set to 'sqrt'
clf = RandomForestClassifier(max_features='sqrt')
```

In [72]:
```
from sklearn.ensemble import RandomForestClassifier

# Create a RandomForestClassifier without specifying max_features
clf = RandomForestClassifier()
```

In [73]:
```
RandomForestClassifier()
```

Out[73]:
```
RandomForestClassifier()
```

In [74]:
```
rf_grid={'n_estimators':np.arange(10,1000,10),
   'max_features':['auto','sqrt'],
  'max_depth':[None,3,5,10,20,30],
  'min_samples_split':[2,5,20,50,100],
  'min_samples_leaf':[1,2,5,10]
  }
```

In [75]:
```
rs_rf=RandomizedSearchCV(RandomForestClassifier(),
                  param_distributions=rf_grid,
                   cv=5,
                   n_iter=20,
                  verbose=True)
```

In [89]:
```
from sklearn.ensemble import RandomForestClassifier

# Create a RandomForestClassifier object
rs_rf = RandomForestClassifier()

# Fit the model
rs_rf.fit(x, y)
```

Out[89]:
```
RandomForestClassifier()
```

In [77]:
```
rs_rf.best_score_
```

Out[77]:
```
0.8066175266175266
```

In [78]:
```
rs_rf.best_params_
```

Out[78]:
```
{'n_estimators': 290,
 'min_samples_split': 5,
 'min_samples_leaf': 5,
 'max_features': 'auto',
 'max_depth': 30}
```

Save The Model

In [79]:
```
X = data.drop('Loan_Status',axis=1)
y = data['Loan_Status']
```

In [80]:
```
rf = RandomForestClassifier(n_estimators=270,
 min_samples_split=5,
 min_samples_leaf=5,
 max_features='sqrt',
 max_depth=5)
```

In [81]:
```
rf.fit(x,y)
```

Out[81]:
```
RandomForestClassifier(max_depth=5, min_samples_leaf=5, min_samples_split=5,
                       n_estimators=270)
```

In [82]:
```
import joblib
```

In [83]:
```
joblib.dump(rf,'loan_status_predict')
```

Out[83]:
```
['loan_status_predict']
```

In [84]:
```
model = joblib.load('loan_status_predict')
```

In [85]:
```
import pandas as pd
df = pd.DataFrame({
    'Gender':1,
    'Married':1,
    'Dependents':2,
    'Education':0,
    'Self_Employed':0,
    'ApplicantIncome':2889,
    'CoapplicantIncome':0.0,
    'LoanAmount':45,
    'Loan_Amount_Term':180,
    'Credit_History':0,
    'Property_Area':1
},index=[0])
```

In [86]:
```
df
```

Out[86]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
0	1	1	2	0	0	2889	

In [87]:
```
result = model.predict(df)
```

In [88]:
```
if result==1:
    print("Loan Approved")
else:
    print("Loan Not Approved")
```

```
Loan Not Approved
```