```
Name : Amishi Anand
Batch: aimlmyb0524
```

Problem Statement

In the financial sector, accurately predicting whether a loan applicant is likely to repay their loan is crucial for minimizing financial risk and maximizing returns. Traditional manual methods of evaluating loan applicants are time-consuming and prone to human error. With the increasing availability of applicant data, there is a need for a robust machine learning solution that can classify loan applicants based on their likelihood of loan repayment.

The goal of this project is to build a machine learning model that can classify loan applicants into categories (approved or rejected) based on a variety of applicant data, such as income, credit history, and employment status. By using historical loan data, we will build a predictive model that automates the loan approval process, helping to minimize financial risks and streamline decision-making.

Content - The **loan_detection.csv** file contains a set of **41188** records under **60** attributes:

```
1.      Loan ID: A unique identifier for each loan application.
2.      Gender: The gender of the loan applicant.
3.      Marital Status: The marital status of the applicant.
4.      Dependents: The number of dependents.
5.      Education: The education level of the applicant.
6.      Self Employed: Whether the applicant is self-employed or not.
7.      Applicant Income: The income of the loan applicant.
8.      Coapplicant Income: The income of the coapplicant (if applicable).
9.      Loan Amount: The amount of the loan.
10.     Loan Amount Term: The term of the loan in months.
11.     Credit History: Whether the applicant has a good credit history.
12.     Property Area: The area type where the property is located (urban, semiurba
n, rural).
13.     Loan Status: Indicating whether the loan was approved or rejected.
```

AIM : Using machine learning techniques to predict loan payments.

Target value : `Loan_Status`

Importing Required Libraries

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

import xgboost
from xgboost import XGBClassifier

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

import warnings
warnings.filterwarnings('ignore')
```

Load Sample Dataset

In [2]:
```
df = pd.read_csv("/Users/amishi/Desktop/Internship Study Material/Datasets/loan_detection.cs
df.head()
```

Out[2]:

	age	campaign	pdays	previous	no_previous_contact	not_working	job_admin.	job_blue-collar	job_entrepreneur	job_housemai
0	56	1	999	0	1	0	0	0	0	
1	57	1	999	0	1	0	0	0	0	
2	37	1	999	0	1	0	0	0	0	
3	40	1	999	0	1	0	1	0	0	
4	56	1	999	0	1	0	0	0	0	

5 rows × 60 columns

Basic EDA

```
- Imbalance Data
- Missing Data
- Duplicate Data
- Outliers or Anomalies
- Data Visualization
- Feature Encoding
- Feature Selection
```

In [3]:
```
df.shape
```

Out[3]: (41188, 60)

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 60 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   age                            41188 non-null  int64
 1   campaign                       41188 non-null  int64
 2   pdays                          41188 non-null  int64
 3   previous                       41188 non-null  int64
 4   no_previous_contact            41188 non-null  int64
 5   not_working                    41188 non-null  int64
 6   job_admin.                     41188 non-null  int64
 7   job_blue-collar                41188 non-null  int64
 8   job_entrepreneur               41188 non-null  int64
 9   job_housemaid                  41188 non-null  int64
 10  job_management                 41188 non-null  int64
 11  job_retired                    41188 non-null  int64
 12  job_self-employed              41188 non-null  int64
 13  job_services                   41188 non-null  int64
 14  job_student                    41188 non-null  int64
 15  job_technician                 41188 non-null  int64
 16  job_unemployed                 41188 non-null  int64
 17  job_unknown                    41188 non-null  int64
 18  marital_divorced               41188 non-null  int64
 19  marital_married                41188 non-null  int64
 20  marital_single                 41188 non-null  int64
 21  marital_unknown                41188 non-null  int64
 22  education_basic.4y             41188 non-null  int64
 23  education_basic.6y             41188 non-null  int64
 24  education_basic.9y             41188 non-null  int64
 25  education_high.school          41188 non-null  int64
 26  education_illiterate           41188 non-null  int64
 27  education_professional.course  41188 non-null  int64
 28  education_university.degree    41188 non-null  int64
 29  education_unknown              41188 non-null  int64
 30  default_no                     41188 non-null  int64
 31  default_unknown                41188 non-null  int64
 32  default_yes                    41188 non-null  int64
 33  housing_no                     41188 non-null  int64
 34  housing_unknown                41188 non-null  int64
 35  housing_yes                    41188 non-null  int64
 36  loan_no                        41188 non-null  int64
 37  loan_unknown                   41188 non-null  int64
 38  loan_yes                       41188 non-null  int64
 39  contact_cellular               41188 non-null  int64
 40  contact_telephone              41188 non-null  int64
 41  month_apr                      41188 non-null  int64
 42  month_aug                      41188 non-null  int64
 43  month_dec                      41188 non-null  int64
 44  month_jul                      41188 non-null  int64
 45  month_jun                      41188 non-null  int64
 46  month_mar                      41188 non-null  int64
 47  month_may                      41188 non-null  int64
 48  month_nov                      41188 non-null  int64
 49  month_oct                      41188 non-null  int64
 50  month_sep                      41188 non-null  int64
 51  day_of_week_fri                41188 non-null  int64
 52  day_of_week_mon                41188 non-null  int64
 53  day_of_week_thu                41188 non-null  int64
 54  day_of_week_tue                41188 non-null  int64
 55  day_of_week_wed                41188 non-null  int64
 56  poutcome_failure               41188 non-null  int64
 57  poutcome_nonexistent           41188 non-null  int64
 58  poutcome_success               41188 non-null  int64
 59  Loan_Status_label              41188 non-null  int64
dtypes: int64(60)
memory usage: 18.9 MB
```

In [5]:
```
df.describe()
```

Out[5]:

	age	campaign	pdays	previous	no_previous_contact	not_working	job_admin.	job_blue colla
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.00000
mean	40.02406	2.567593	962.475454	0.172963	0.963217	0.087623	0.253035	0.22467
std	10.42125	2.770014	186.910907	0.494901	0.188230	0.282749	0.434756	0.41737
min	17.00000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	32.00000	1.000000	999.000000	0.000000	1.000000	0.000000	0.000000	0.00000
50%	38.00000	2.000000	999.000000	0.000000	1.000000	0.000000	0.000000	0.00000
75%	47.00000	3.000000	999.000000	0.000000	1.000000	0.000000	1.000000	0.00000
max	98.00000	56.000000	999.000000	7.000000	1.000000	1.000000	1.000000	1.00000

8 rows × 60 columns

Type *Markdown* and LaTeX: α^2

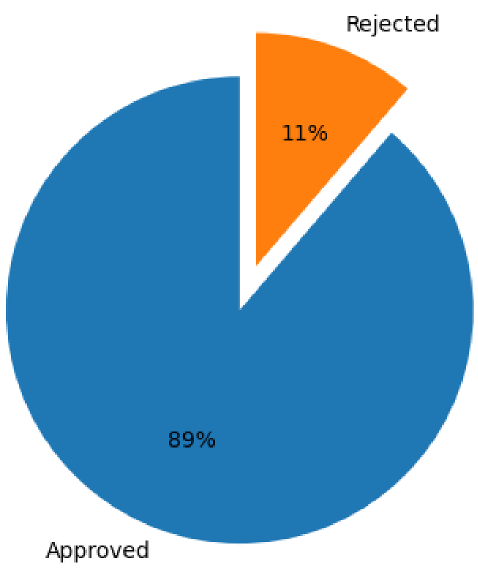
1) Imbalance Data

In [6]:
```
df['Loan_Status_label'].value_counts()
```

Out[6]:
```
Loan_Status_label
0    36548
1     4640
Name: count, dtype: int64
```

In [7]:
```
plt.pie(df['Loan_Status_label'].value_counts(), autopct="%1.0f%%", startangle=90,
        labels = ['Approved','Rejected'], explode=(0, 0.2))
plt.show
```

Out[7]: <function matplotlib.pyplot.show(close=None, block=None)>



2) Missing Data

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
age                              0
campaign                         0
pdays                            0
previous                         0
no_previous_contact              0
not_working                      0
job_admin.                       0
job_blue-collar                  0
job_entrepreneur                 0
job_housemaid                    0
job_management                   0
job_retired                      0
job_self-employed                0
job_services                     0
job_student                      0
job_technician                   0
job_unemployed                   0
job_unknown                      0
marital_divorced                 0
marital_married                  0
marital_single                   0
marital_unknown                  0
education_basic.4y               0
education_basic.6y               0
education_basic.9y               0
education_high.school            0
education_illiterate             0
education_professional.course    0
education_university.degree      0
education_unknown                0
default_no                       0
default_unknown                  0
default_yes                      0
housing_no                       0
housing_unknown                  0
housing_yes                      0
loan_no                          0
loan_unknown                     0
loan_yes                         0
contact_cellular                 0
contact_telephone                0
month_apr                        0
month_aug                        0
month_dec                        0
month_jul                        0
month_jun                        0
month_mar                        0
month_may                        0
month_nov                        0
month_oct                        0
month_sep                        0
day_of_week_fri                  0
day_of_week_mon                  0
day_of_week_thu                  0
day_of_week_tue                  0
day_of_week_wed                  0
poutcome_failure                 0
poutcome_nonexistent             0
poutcome_success                 0
Loan_Status_label                0
dtype: int64
```

3) Duplicate Data

In [9]:
```
df[df.duplicated()]
```

Out[9]:

	age	campaign	pdays	previous	no_previous_contact	not_working	job_admin.	job_blue-collar	job_entrepreneur	job_hous
10	41	1	999	0	1	0	0	1	0	
11	25	1	999	0	1	0	0	0	0	
16	35	1	999	0	1	0	0	1	0	
31	59	1	999	0	1	0	0	0	0	
104	52	1	999	0	1	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	
40928	21	1	999	0	1	1	0	0	0	
41131	58	1	999	0	1	0	0	0	0	
41167	32	3	999	0	1	0	1	0	0	
41172	31	1	999	0	1	0	1	0	0	
41181	37	1	999	0	1	0	1	0	0	

2417 rows × 60 columns

In [10]:
```
df.drop_duplicates(keep='first', inplace=True)
```

In [11]:
```
df.duplicated().sum()
```

Out[11]: 0

Type *Markdown* and LaTeX: α^2

4) Outliers or Anomalies

In [12]:
```
# Using IQR
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

lower_quartile = (Q1 - 1.5*IQR)
upper_quartile = (Q3 + 1.5*IQR)
```

In [13]:
```
data=df[~((df<lower_quartile) | (df>upper_quartile)) .any(axis=1)]
data.head()
```

Out[13]:

age	campaign	pdays	previous	no_previous_contact	not_working	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid

0 rows × 60 columns

Type *Markdown* and LaTeX: α^2

5) Feature Encoding

In [14]:
```
df.columns
```

Out[14]:
```
Index(['age', 'campaign', 'pdays', 'previous', 'no_previous_contact',
       'not_working', 'job_admin.', 'job_blue-collar', 'job_entrepreneur',
       'job_housemaid', 'job_management', 'job_retired', 'job_self-employed',
       'job_services', 'job_student', 'job_technician', 'job_unemployed',
       'job_unknown', 'marital_divorced', 'marital_married', 'marital_single',
       'marital_unknown', 'education_basic.4y', 'education_basic.6y',
       'education_basic.9y', 'education_high.school', 'education_illiterate',
       'education_professional.course', 'education_university.degree',
       'education_unknown', 'default_no', 'default_unknown', 'default_yes',
       'housing_no', 'housing_unknown', 'housing_yes', 'loan_no',
       'loan_unknown', 'loan_yes', 'contact_cellular', 'contact_telephone',
       'month_apr', 'month_aug', 'month_dec', 'month_jul', 'month_jun',
       'month_mar', 'month_may', 'month_nov', 'month_oct', 'month_sep',
       'day_of_week_fri', 'day_of_week_mon', 'day_of_week_thu',
       'day_of_week_tue', 'day_of_week_wed', 'poutcome_failure',
       'poutcome_nonexistent', 'poutcome_success', 'Loan_Status_label'],
      dtype='object')
```

In [15]:
```
df.drop(columns=['marital_divorced', 'marital_married', 'marital_single',
       'marital_unknown','contact_cellular','contact_telephone','month_apr', 'month_aug', 'm
       'month_mar', 'month_may', 'month_nov', 'month_oct', 'month_sep',
       'day_of_week_fri', 'day_of_week_mon', 'day_of_week_thu',
       'day_of_week_tue', 'day_of_week_wed'], axis=1, inplace=True)
```

Type *Markdown* and LaTeX: α^2

6) Feature Selection

In [16]:
```
df.corr()['Loan_Status_label']
```

Out[16]:
```
age                            0.026865
campaign                      -0.074539
pdays                         -0.324611
previous                       0.228665
no_previous_contact           -0.324572
not_working                    0.118294
job_admin.                     0.035250
job_blue-collar               -0.075625
job_entrepreneur              -0.019306
job_housemaid                 -0.007496
job_management                -0.001627
job_retired                    0.091108
job_self-employed             -0.006224
job_services                  -0.033042
job_student                    0.092536
job_technician                -0.003678
job_unemployed                 0.012016
job_unknown                   -0.002082
education_basic.4y            -0.011243
education_basic.6y            -0.024966
education_basic.9y            -0.047056
education_high.school         -0.008653
education_illiterate           0.006922
education_professional.course  0.002225
education_university.degree    0.054144
education_unknown              0.019507
default_no                     0.103282
default_unknown               -0.103227
default_yes                   -0.003224
housing_no                    -0.011295
housing_unknown               -0.004555
housing_yes                    0.012686
loan_no                        0.011501
loan_unknown                  -0.004555
loan_yes                      -0.010239
poutcome_failure               0.028345
poutcome_nonexistent          -0.191993
poutcome_success               0.316035
Loan_Status_label              1.000000
Name: Loan_Status_label, dtype: float64
```

Type *Markdown* and LaTeX: α^2

Model Building

```
- Separate your Independent and Dependent data
- Split your data into Training and Test set
- Model Selection
- Model Training
- Model Prediction
- Model Evaluation
- Hyperparameter Tuning
```

1) Separate your Independent and Dependent data

In [17]:
```
X = df.iloc[:,:-1]
X
```

Out[17]:

	age	campaign	pdays	previous	no_previous_contact	not_working	job_admin.	job_blue-collar	job_entrepreneur	job_hous
0	56	1	999	0	1	0	0	0	0	
1	57	1	999	0	1	0	0	0	0	
2	37	1	999	0	1	0	0	0	0	
3	40	1	999	0	1	0	1	0	0	
4	56	1	999	0	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
41183	73	1	999	0	1	1	0	0	0	
41184	46	1	999	0	1	0	0	1	0	
41185	56	2	999	0	1	1	0	0	0	
41186	44	1	999	0	1	0	0	0	0	
41187	74	3	999	1	1	1	0	0	0	

38771 rows × 38 columns

In [18]:
```
y = df['Loan_Status_label']
y
```

Out[18]:
```
0        0
1        0
2        0
3        0
4        0
        ..
41183    1
41184    0
41185    0
41186    1
41187    0
Name: Loan_Status_label, Length: 38771, dtype: int64
```

Type *Markdown* and LaTeX: α^2

2) Split your data into Training and Test set

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Type *Markdown* and LaTeX: α^2

3) Model Selection

In [20]:
```
# Using Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Out[20]:
```
▾ LogisticRegression
LogisticRegression()
```

In [21]:
```
print(f'Training Accuracy : {lr.score(X_train, y_train)}')
print(f'Test Accuracy : {lr.score(X_test, y_test)}')
```

```
Training Accuracy : 0.8916688160949188
Test Accuracy : 0.8985170857511283
```

In [22]:
```
# Using Decision tree
dt = DecisionTreeClassifier(max_depth=4)
dt. fit(X_train, y_train)
```

Out[22]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4)
```

In [23]:
```
print(f'Training Accuracy : {dt.score(X_train, y_train)}')
print(f'Test Accuracy : {dt.score(X_test, y_test)}')
```

```
Training Accuracy : 0.8919589889089502
Test Accuracy : 0.898388136686009
```

In [24]:
```
# Using XGBoost
xgb = XGBClassifier()
xgb.fit(X_train,y_train)
```

Out[24]:
```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
```

In [25]:
```
print(f'Training Accuracy : {xgb.score(X_train, y_train)}')
print(f'Test Accuracy : {xgb.score(X_test, y_test)}')
```

```
Training Accuracy : 0.9045331441836472
Test Accuracy : 0.895164410058027
```

Type *Markdown* and LaTeX: α^2

4) Model Prediction

In [26]:
```
y_pred_tr=dt.predict(X_train)
y_pred_ts=dt.predict(X_test)
```

In [27]:
```
y_train[:3]
```

Out[27]:
```
34767    0
32132    0
9868     0
Name: Loan_Status_label, dtype: int64
```

In [28]:
```
y_pred_tr[:3]
```

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

In [29]:
```
y_test[:3]
```

Out[29]:
```
37815    0
5279     0
40810    0
Name: Loan_Status_label, dtype: int64
```

In [30]:
```
y_pred_ts[:3]
```

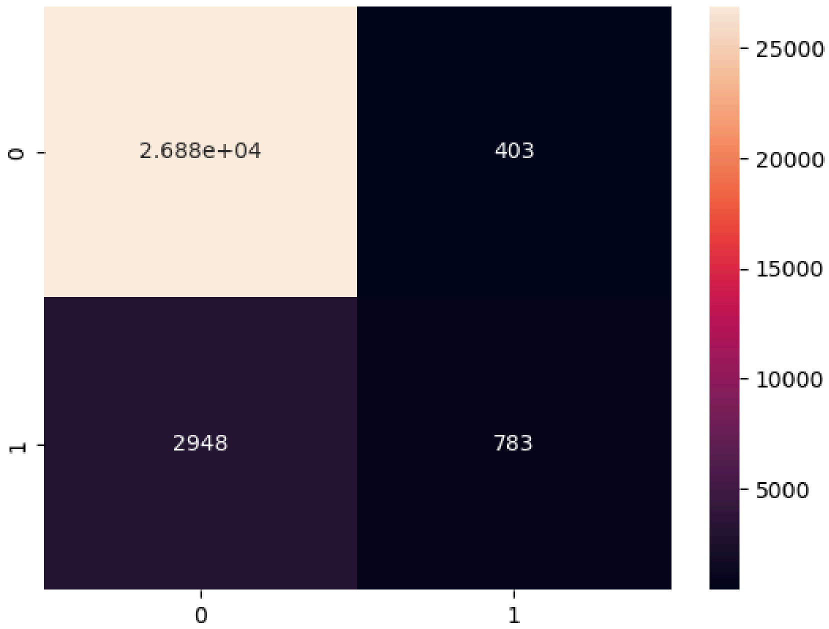
Out[30]:
```
array([0, 0, 0])
```

Type *Markdown* and LaTeX: α^2

5) Model Evaluation

In [31]:
```
# training data
sns.heatmap(confusion_matrix(y_train, y_pred_tr), annot=True, fmt='.4g')
```

Out[31]: <Axes: >



In [32]:
```
accuracy_score(y_train, y_pred_tr)
```

Out[32]: 0.8919589889089502

In [33]:
```
print(classification_report(y_train, y_pred_tr))
```

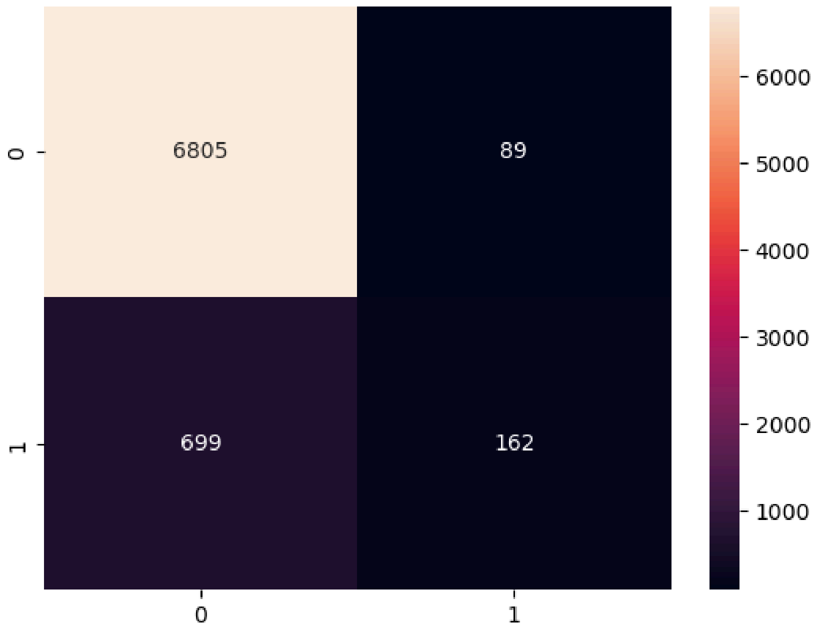
```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94     27285
           1       0.66      0.21      0.32      3731

    accuracy                           0.89     31016
   macro avg       0.78      0.60      0.63     31016
weighted avg       0.87      0.89      0.87     31016
```

In [34]:
```
# testing data
sns.heatmap(confusion_matrix(y_test, y_pred_ts), annot=True, fmt='.4g')
```

Out[34]: <Axes: >



In [35]:
```
accuracy_score(y_test, y_pred_ts)
```

Out[35]: 0.898388136686009

Type *Markdown* and LaTeX: α^2

Hyperparameter Tuning

In [36]:
```
parameters = {
    'n_estimators' : [100, 200],
    'learning_rate': [0.1, 0.01, 1.0, 0.05],
    'max_depth': [3,4,5],
    'gamma' : [0.2, 0.3],
    'reg_alpha' : [0.1,1,0.2],
    'reg_lambda' : [0.1, 1]
}


parameters
```

Out[36]:
```
{'n_estimators': [100, 200],
 'learning_rate': [0.1, 0.01, 1.0, 0.05],
 'max_depth': [3, 4, 5],
 'gamma': [0.2, 0.3],
 'reg_alpha': [0.1, 1, 0.2],
 'reg_lambda': [0.1, 1]}
```

In [37]:
```
#Grid Search CV
```

In [38]:
```
grid_search = GridSearchCV(estimator=xgb, param_grid=parameters, scoring='accuracy', cv=5, v
grid_search.fit(X_train, y_train)
```

```
_lambda=0.1;, score=0.887 total time=   0.3s
[CV 1/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=1, reg
_lambda=1;, score=0.893 total time=   0.3s
[CV 2/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=1, reg
_lambda=1;, score=0.896 total time=   0.4s
[CV 3/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=1, reg
_lambda=1;, score=0.890 total time=   0.3s
[CV 4/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=1, reg
_lambda=1;, score=0.892 total time=   0.3s
[CV 5/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=1, reg
_lambda=1;, score=0.887 total time=   0.3s
[CV 1/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=0.2, r
eg_lambda=0.1;, score=0.893 total time=   0.4s
[CV 2/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=0.2, r
eg_lambda=0.1;, score=0.896 total time=   0.4s
[CV 3/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=0.2, r
eg_lambda=0.1;, score=0.891 total time=   0.4s
[CV 4/5] END gamma=0.2, learning_rate=0.05, max_depth=4, n_estimators=200, reg_alpha=0.2, r
eg_lambda=0.1;, score=0.891 total time=   0.4s
```

In [39]:
```
print(f'Best Selected Hyperparameters: \n\n{grid_search.best_params_}\n')
print(f'Best Estimator: \n\n{grid_search.best_estimator_}')
```

```
Best Selected Hyperparameters:

{'gamma': 0.2, 'learning_rate': 0.05, 'max_depth': 3, 'n_estimators': 200, 'reg_alpha': 0.
2, 'reg_lambda': 0.1}

Best Estimator:

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0.2, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.05, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=3, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=200, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

In [40]:
```
print(f'Training Accuracy : {grid_search.score(X_train, y_train)}')
print(f'Test Accuracy : {grid_search.score(X_test, y_test)}')
```

```
Training Accuracy : 0.893345370131545
Test Accuracy : 0.8989039329464862
```

Thank you!!