ClassificationRegaression

In [24]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.preprocessing import OneHotEncoder,LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report
```

In [5]:
```
df=pd.read_csv("C:\\Users\\hp\\Downloads\\bank_loan.csv")
df.head()
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applicant
0	LP001002	Male	No	0	Graduate	No	
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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 [9]:
```
df.shape
```

Out[9]:
```
(614, 13)
```

In [51]:
```
df.columns
```

Out[51]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [11]:
```
df.fillna(df.median(numeric_only=True), inplace=True)
```

In [13]:
```
df.isnull().sum()
```

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

In [15]:
```
df.fillna(df['Gender'].mode()[0],inplace=True)
```

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
Loan_ID              0
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
```

In [19]:
```
for col in df.select_dtypes(include=[object]).columns:
    df[col]=LabelEncoder().fit_transform(df[col])
```

In [21]:
```
X=df.drop('Loan_Status',axis=1)
y=df[['Loan_Status']]
```

In [25]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.2,random_state
```

In [54]:
```
model2 = LogisticRegression(max_iter=1000)
model2.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning: lbfgs failed to converge after 1000 iteration(s) (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=1000).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[54]:

LogisticRegression

Parameters

penalty	'l2'
dual	False
tol	0.0001
C	1.0
fit_intercept	True
intercept_scaling	1
class_weight	None
random_state	None
solver	'lbfgs'
max_iter	1000
multi_class	'deprecated'
verbose	0
warm_start	False
n_jobs	None
l1_ratio	None

In [55]:
```
# 5. Evaluate performance
y_pred=model2.predict(X_test)
```

In [56]:
```
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

Out[56]: 0.7886178861788617

DecisionTreeClassifier

In [57]:
```
#
model3=DecisionTreeClassifier(max_depth=4, criterion='entropy',random_state=42
model3.fit(X_train,y_train)
```

Out[57]:

DecisionTreeClassifier

Parameters

criterion	'entropy'
splitter	'best'
max_depth	4
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0.0
max_features	None
random_state	42
max_leaf_nodes	None
min_impurity_decrease	0.0
class_weight	None
ccp_alpha	0.0
monotonic_cst	None

In [58]:
```
y_pred=model3.predict(X_test)
```

In [59]:
```
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

Out[59]: 0.7886178861788617

In [46]:
```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.95	0.42	0.58	43
1	0.76	0.99	0.86	80
accuracy			0.79	123
macro avg	0.85	0.70	0.72	123
weighted avg	0.83	0.79	0.76	123

Random Forest Classifier

In [60]:
```
model4=RandomForestClassifier(n_estimators=100,random_state=42)
model4.fit(X_train,y_train)
```

Out[60]:

RandomForestClassifier

Parameters

n_estimators	100
criterion	'gini'
max_depth	None
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0.0
max_features	'sqrt'
max_leaf_nodes	None
min_impurity_decrease	0.0
bootstrap	True
oob_score	False
n_jobs	None
random_state	42
verbose	0
warm_start	False
class_weight	None
ccp_alpha	0.0
max_samples	None
monotonic_cst	None

In [61]:
```
y_pred=model4.predict(X_test)
```

In [62]:
```
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

Out[62]: 0.7804878048780488

In [50]:
```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.95	0.42	0.58	43
1	0.76	0.99	0.86	80
accuracy			0.79	123
macro avg	0.85	0.70	0.72	123
weighted avg	0.83	0.79	0.76	123

In [63]:
```
# 7. Save model
joblib.dump(model2, "bank_loan_model.pkl")
print("✅ Model saved as bank_loan_model.pkl")
```

```
✅ Model saved as bank_loan_model.pkl
```