BANGIRANA DOUGLAS /00573

```
LOGISTIC REGRESSION
```

In [1]:
```python
#Required libraries
from sklearn.datasets import make_classification
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
```

In [2]:
```python
df=pd.read_csv("C:\\Users\\Atwongire Vianney\\Desktop\\AI_PRAC\\loan_data.(
df
```

Out[2]:

	Gender	Married	Dependents	Education	ApplicantIncome	CoapplicantIncome	LoanAmount
0	Male	Yes	1	Graduate	4583	1508.0	128
1	Male	Yes	0	Graduate	3000	0.0	66
2	Male	Yes	0	Not Graduate	2583	2358.0	120
3	Male	No	0	Graduate	6000	0.0	141
4	Male	Yes	0	Not Graduate	2333	1516.0	95
...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	5703	0.0	128
377	Male	Yes	0	Graduate	3232	1950.0	108
378	Female	No	0	Graduate	2900	0.0	71
379	Male	Yes	3+	Graduate	4106	0.0	40
380	Female	No	0	Graduate	4583	0.0	133

381 rows × 9 columns

In [3]:
```
df.head()
```

Out[3]:

	Gender	Married	Dependents	Education	ApplicantIncome	CoapplicantIncome	LoanAmount	L
0	Male	Yes	1	Graduate	4583	1508.0	128	
1	Male	Yes	0	Graduate	3000	0.0	66	
2	Male	Yes	0	Not Graduate	2583	2358.0	120	
3	Male	No	0	Graduate	6000	0.0	141	
4	Male	Yes	0	Not Graduate	2333	1516.0	95	

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

Out[4]:

	Gender	Married	Dependents	Education	ApplicantIncome	CoapplicantIncome	LoanAmount
376	Male	Yes	3+	Graduate	5703	0.0	128
377	Male	Yes	0	Graduate	3232	1950.0	108
378	Female	No	0	Graduate	2900	0.0	71
379	Male	Yes	3+	Graduate	4106	0.0	40
380	Female	No	0	Graduate	4583	0.0	133

In [5]:
```
#checking for missing values
print(df.isnull().sum())
```

```
Gender                5
Married               0
Dependents            8
Education             0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term     11
Loan_Status           0
dtype: int64
```

In [6]:

```
#Handling missing values appropriately
filled_df=df.fillna(0)
print(filled_df)

#Dropping rows with missing values
dropped_df=df.dropna()
print(dropped_df)

#saving the cleaned dataset to a new csv file
filled_df.to_csv('cleaned_file.csv', index=False)
dropped_df.to_csv('cleaned_file_drop.ped.csv', index=False)
```

```
    Gender Married Dependents     Education  ApplicantIncome  \
0     Male     Yes          1      Graduate             4583   
1     Male     Yes          0      Graduate             3000   
2     Male     Yes          0  Not Graduate             2583   
3     Male      No          0      Graduate             6000   
4     Male     Yes          0  Not Graduate             2333   
..     ...     ...        ...           ...              ...   
376   Male     Yes         3+      Graduate             5703   
377   Male     Yes          0      Graduate             3232   
378 Female      No          0      Graduate             2900   
379   Male     Yes         3+      Graduate             4106   
380 Female      No          0      Graduate             4583   

    CoapplicantIncome  LoanAmount  Loan_Amount_Term Loan_Status  
0              1508.0         128             360.0           N  
1                 0.0          66             360.0           Y  
2              2358.0         120             360.0           Y  
3                 0.0         141             360.0           Y  
4              1516.0          95             360.0           Y  
..                ...         ...               ...         ...  
376               0.0         128             360.0           Y  
377            1950.0         108             360.0           Y  
378               0.0          71             360.0           Y  
379               0.0          40             180.0           Y  
380               0.0         133             360.0           N  

[381 rows x 9 columns]
    Gender Married Dependents     Education  ApplicantIncome  \
0     Male     Yes          1      Graduate             4583   
1     Male     Yes          0      Graduate             3000   
2     Male     Yes          0  Not Graduate             2583   
3     Male      No          0      Graduate             6000   
4     Male     Yes          0  Not Graduate             2333   
..     ...     ...        ...           ...              ...   
376   Male     Yes         3+      Graduate             5703   
377   Male     Yes          0      Graduate             3232   
378 Female      No          0      Graduate             2900   
379   Male     Yes         3+      Graduate             4106   
380 Female      No          0      Graduate             4583   

    CoapplicantIncome  LoanAmount  Loan_Amount_Term Loan_Status  
0              1508.0         128             360.0           N  
1                 0.0          66             360.0           Y  
2              2358.0         120             360.0           Y  
3                 0.0         141             360.0           Y  
4              1516.0          95             360.0           Y  
..                ...         ...               ...         ...  
376               0.0         128             360.0           Y  
377            1950.0         108             360.0           Y  
378               0.0          71             360.0           Y  
379               0.0          40             180.0           Y  
380               0.0         133             360.0           N  

[358 rows x 9 columns]
```

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

Out[7]:

	Gender	Married	Dependents	Education	ApplicantIncome	CoapplicantIncome	LoanAmount
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
376	False	False	False	False	False	False	False
377	False	False	False	False	False	False	False
378	False	False	False	False	False	False	False
379	False	False	False	False	False	False	False
380	False	False	False	False	False	False	False

381 rows × 9 columns

In [8]:
```
df_cleaned = df.dropna()
df_cleaned
```

Out[8]:

	Gender	Married	Dependents	Education	ApplicantIncome	CoapplicantIncome	LoanAmount
0	Male	Yes	1	Graduate	4583	1508.0	128
1	Male	Yes	0	Graduate	3000	0.0	66
2	Male	Yes	0	Not Graduate	2583	2358.0	120
3	Male	No	0	Graduate	6000	0.0	141
4	Male	Yes	0	Not Graduate	2333	1516.0	95
...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	5703	0.0	128
377	Male	Yes	0	Graduate	3232	1950.0	108
378	Female	No	0	Graduate	2900	0.0	71
379	Male	Yes	3+	Graduate	4106	0.0	40
380	Female	No	0	Graduate	4583	0.0	133

358 rows × 9 columns

In [9]:
```
filled_df.to_csv('cleaned_file.csv', index=False)
```

In [10]:
```
x=df_cleaned['LoanAmount'].array.reshape(-1,1)
x
```

Out[10]:
```
<PandasArray>
[
[128],
[66],
[120],
[141],
[95],
[70],
[109],
[114],
[17],
[125],
[100],
[76],
[133],
[104],
[116],
[122],
[110],
```

In [11]:
```
y=df_cleaned['Loan_Amount_Term']
y
```

Out[11]:
```
0      360.0
1      360.0
2      360.0
3      360.0
4      360.0
       ...
376    360.0
377    360.0
378    360.0
379    180.0
380    360.0
Name: Loan_Amount_Term, Length: 358, dtype: float64
```

In [12]:
```
#splitting the data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
```

In [13]:
```
x_train.shape
```

Out[13]:
```
(286, 1)
```

In [14]:
```
model=LogisticRegression()
model.fit(x_train,y_train)
```

```
C:\Users\Atwongire Vianney\anaconda3\Lib\site-packages\sklearn\linear_model\_
logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[14]:
```
▾ LogisticRegression
LogisticRegression()
```

In [15]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[15]:
```
array([360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360.])
```

In [16]:
```
score=model.score(x_train,y_train)
score
```

Out[16]:
```
0.8251748251748252
```

Fine Tune

In [17]:
```
#Defining parameters
param_grid={
    'penalty':['l1','l2'],
    'C':[0.001,0.01,0.1,1,10,100],
    'solver':['liblinear','saga']
}
```

In [18]:
```
#performing grid with cross_validation
grid_search=GridSearchCV(estimator=model,param_grid=param_grid,cv=5,n_jobs
grid_search.fit(x_train,y_train)
```

```
C:\Users\Atwongire Vianney\anaconda3\Lib\site-packages\sklearn\model_selectio
n\_split.py:725: UserWarning: The least populated class in y has only 1 membe
rs, which is less than n_splits=5.
  warnings.warn(
```

Out[18]:
```
▸ GridSearchCV
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

In [19]:
```
#best parameters and best score
best_param=grid_search.best_params_
grid_search
```

Out[19]:
```
▸ GridSearchCV
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

In [20]:
```
best_score=grid_search.best_score_
best_score
```

Out[20]: 0.8251663641863279

In [21]:
```
#evaluating the performance
score=grid_search.score(x_test,y_test)
score
```

Out[21]: 0.9166666666666666

In [22]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[22]:
```
array([360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360., 360., 360., 360., 360., 360.,
       360., 360., 360., 360., 360., 360.])
```

LINEAR REGRESSION

In [23]:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [24]:

```
df.head()
```

Out[24]:

	Gender	Married	Dependents	Education	ApplicantIncome	CoapplicantIncome	LoanAmount
0	Male	Yes	1	Graduate	4583	1508.0	128
1	Male	Yes	0	Graduate	3000	0.0	66
2	Male	Yes	0	Not Graduate	2583	2358.0	120
3	Male	No	0	Graduate	6000	0.0	141
4	Male	Yes	0	Not Graduate	2333	1516.0	95

In [25]:

```
df.tail()
```

Out[25]:

	Gender	Married	Dependents	Education	ApplicantIncome	CoapplicantIncome	LoanAmount
376	Male	Yes	3+	Graduate	5703	0.0	128
377	Male	Yes	0	Graduate	3232	1950.0	108
378	Female	No	0	Graduate	2900	0.0	71
379	Male	Yes	3+	Graduate	4106	0.0	40
380	Female	No	0	Graduate	4583	0.0	133

In [26]:

```
df.isnull().sum()
```

Out[26]:

```
Gender                5
Married               0
Dependents            8
Education             0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term     11
Loan_Status           0
dtype: int64
```

In [27]:

```
df_cleaned=df.dropna()
df_cleaned
```

Out[27]:

	Gender	Married	Dependents	Education	ApplicantIncome	CoapplicantIncome	LoanAmount
0	Male	Yes	1	Graduate	4583	1508.0	128
1	Male	Yes	0	Graduate	3000	0.0	66
2	Male	Yes	0	Not Graduate	2583	2358.0	120
3	Male	No	0	Graduate	6000	0.0	141
4	Male	Yes	0	Not Graduate	2333	1516.0	95
...	...	...	...	...	...	...	...
376	Male	Yes	3+	Graduate	5703	0.0	128
377	Male	Yes	0	Graduate	3232	1950.0	108
378	Female	No	0	Graduate	2900	0.0	71
379	Male	Yes	3+	Graduate	4106	0.0	40
380	Female	No	0	Graduate	4583	0.0	133

358 rows × 9 columns

In [28]:

```
x=df_cleaned['LoanAmount'].array.reshape(-1,1)
x
```

Out[28]:

```
<PandasArray>
[
[128],
[66],
[120],
[141],
[95],
[70],
[109],
[114],
[17],
[125],
[100],
[76],
[133],
[104],
[116],
[122],
[110],
```

In [29]:
```
y=df_cleaned['Loan_Amount_Term']
y
```

Out[29]:
```
0      360.0
1      360.0
2      360.0
3      360.0
4      360.0
       ...
376    360.0
377    360.0
378    360.0
379    180.0
380    360.0
Name: Loan_Amount_Term, Length: 358, dtype: float64
```

In [30]:
```
#splitting the data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
```

In [31]:
```
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[31]:
```
▾ LinearRegression
LinearRegression()
```

In [32]:
```
score=model.score(x_train,y_train)
score
```

Out[32]:
```
0.013044972022888901
```

In [33]:
```
model.coef_
```

Out[33]:
```
array([0.28778211])
```

In [34]:
```
model.intercept_
```

Out[34]:
```
308.4210137519316
```

FINE TUNING

In [35]:
```
#Defining parameters
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
```

In [36]:
```python
param_grid={
    'copy_X':[True,False],
    'fit_intercept':[True,False],
    'n_jobs':[True,False],
    'positive':[True,False]

}
```

In [37]:
```python
#performing grid with cross_validation
grid_search=GridSearchCV(model,param_grid,cv=5,scoring='neg_mean_squared_e
grid_search.fit(x_train,y_train)
```

Out[37]:
```
▸ GridSearchCV
▸ estimator: LinearRegression
    ▸ LinearRegression
```

In [38]:
```python
#Best model
best_model=grid_search.best_estimator_
best_model
```

Out[38]:
```
▾ LinearRegression
LinearRegression(n_jobs=True, positive=True)
```

In [39]:
```python
best_score=model.score(x_test,y_test)
best_score
```

Out[39]:
```
-0.009395552322493339
```

In [40]:
```python
y_pred=best_model.predict(x_test)
y_pred
```

Out[40]:
```
array([345.83268771, 341.8037382 , 337.19922449, 348.13494456,
       346.12046981, 339.21369924, 343.53043085, 331.73136445,
       340.07704556, 337.19922449, 345.5449056 , 333.45805709,
       327.70241494, 342.37930242, 325.40015808, 346.12046981,
       321.65899069, 341.5159561 , 342.95486663, 339.50148134,
       348.71050878, 347.27159824, 323.67346544, 342.95486663,
       343.81821295, 342.95486663, 334.32140341, 345.25712349,
       321.9467728 , 347.84716246, 340.07704556, 344.39377717,
       325.9757223 , 348.13494456, 334.32140341, 349.86163721,
       349.86163721, 338.92591713, 340.65260977, 342.95486663,
       336.04809606, 342.95486663, 351.01276564, 325.11237598,
       347.27159824, 342.95486663, 338.35035291, 328.56576127,
       347.55938035, 345.25712349, 346.40825192, 340.94039188,
       345.83268771, 315.61556644, 332.59471077, 345.83268771,
       348.99829089, 342.95486663, 329.71688969, 348.13494456,
       346.12046981, 336.33587816, 337.19922449, 328.56576127,
       321.08342647, 337.19922449, 350.43720142, 327.70241494,
       340.94039188, 330.0046718 , 340.07704556, 345.83268771])
```

In [41]:
```
mse=mean_squared_error(y_test,y_pred)
mse
```

Out[41]: 2688.917596325753

In [ ]:
```

```