ATWONGERE VIANNEY 2023/U/MMU/BSSE/00574.

LOGISTIC REGRESSION AND FINE TUNING IT

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
```

In [2]:

```python
df=pd.read_csv("C:\\Users\\Atwongire Vianney\\Desktop\\AI_PRAC\\03_Cluster
df
```

Out[2]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyba
0	2007	NaN	NaN	0	0	0	0	0	
1	2007	F	17.41	49	0	0	1	0	
2	2007	F	17.511	41	0	0	0	0	
3	2006	F	NaN	36	0	0	0	0	
4	2008	F	16.657	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	.
14995	2008	F	16.329	21	0	0	0	0	
14996	2008	F	16.545	50	0	0	0	0	
14997	2007	F	17.999	32	0	0	0	0	
14998	2007	F	17.903	20	0	0	0	0	
14999	2009	F	15.811	25	0	0	7	0	

15000 rows × 40 columns

In [3]:

```python
df.head()
```

Out[3]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyball	s
0	2007	NaN	NaN	0	0	0	0	0	0	
1	2007	F	17.41	49	0	0	1	0	0	
2	2007	F	17.511	41	0	0	0	0	0	
3	2006	F	NaN	36	0	0	0	0	0	
4	2008	F	16.657	1	0	0	0	0	0	

5 rows × 40 columns

In [4]:

```python
df.tail()
```

Out[4]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyba
14995	2008	F	16.329	21	0	0	0	0	
14996	2008	F	16.545	50	0	0	0	0	
14997	2007	F	17.999	32	0	0	0	0	
14998	2007	F	17.903	20	0	0	0	0	
14999	2009	F	15.811	25	0	0	7	0	

5 rows × 40 columns

In [5]:

```python
x=df['NumberOffriends'].array.reshape(-1,1)
x
```

Out[5]:

```
<PandasArray>
[
[0],
[49],
[41],
[36],
[1],
[32],
[18],
[0],
[0],
[21],
[0],
[0],
[29],
[0],
[89],
[37],
[89],
```

In [6]:
```
y=df['sex']
y
```

Out[6]:
```
0        0
1        0
2        3
3        0
4        0
        ..
14995    0
14996    0
14997    0
14998    0
14999    0
Name: sex, Length: 15000, dtype: int64
```

In [7]:
```
#splitting the data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
```

In [8]:
```
x_train.shape
```

Out[8]:
```
(12000, 1)
```

In [9]:
```
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[9]:
```
▾ LogisticRegression

LogisticRegression()
```

In [10]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[10]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [11]:
```
score=model.score(x_train,y_train)
score
```

Out[11]:
```
0.89125
```

FINE TUNING

In [12]:
```
#Defining parameters
param_grid={
    'penalty':['l1','l2'],
    'C':[0.001,0.01,0.1,1,10,100],
    'solver':['liblinear','saga']
}
```

In [13]:
```
#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[13]:
```
▸            GridSearchCV
▸ estimator: LogisticRegression
        ▸ LogisticRegression
```

In [14]:
```
#best parameters and best score
best_param=grid_search.best_params_
grid_search
```

Out[14]:
```
▸            GridSearchCV
▸ estimator: LogisticRegression
        ▸ LogisticRegression
```

In [15]:
```
best_score=grid_search.best_score_
best_score
```

Out[15]:
```
0.8914166666666666
```

In [16]:
```
#evaluating the performance
score=grid_search.score(x_test,y_test)
score
```

Out[16]:
```
0.9046666666666666
```

In [17]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[17]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

FOR LINEAR REGRESSION

In [18]:

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

In [19]:

```python
df=pd.read_csv("C:\\Users\\Atwongire Vianney\\Desktop\\AI_PRAC\\03_Clusteri
df
```

Out[19]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyba
0	2007	NaN	NaN	0	0	0	0	0	
1	2007	F	17.41	49	0	0	1	0	
2	2007	F	17.511	41	0	0	0	0	
3	2006	F	NaN	36	0	0	0	0	
4	2008	F	16.657	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	.
14995	2008	F	16.329	21	0	0	0	0	
14996	2008	F	16.545	50	0	0	0	0	
14997	2007	F	17.999	32	0	0	0	0	
14998	2007	F	17.903	20	0	0	0	0	
14999	2009	F	15.811	25	0	0	7	0	

15000 rows × 40 columns

In [20]:

```python
df.head()
```

Out[20]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyball	s
0	2007	NaN	NaN	0	0	0	0	0	0	
1	2007	F	17.41	49	0	0	1	0	0	
2	2007	F	17.511	41	0	0	0	0	0	
3	2006	F	NaN	36	0	0	0	0	0	
4	2008	F	16.657	1	0	0	0	0	0	

5 rows × 40 columns

In [21]:

```
df.tail()
```

Out[21]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyba
14995	2008	F	16.329	21	0	0	0	0	
14996	2008	F	16.545	50	0	0	0	0	
14997	2007	F	17.999	32	0	0	0	0	
14998	2007	F	17.903	20	0	0	0	0	
14999	2009	F	15.811	25	0	0	7	0	

5 rows × 40 columns

In [22]:

```
x=df['NumberOffriends'].array.reshape(-1,1)
x
```

Out[22]:

```
<PandasArray>
[
[0],
[49],
[41],
[36],
[1],
[32],
[18],
[0],
[0],
[21],
[0],
[0],
[29],
[0],
[89],
[37],
[89],
[98]
```

In [23]:

```
y=df['sex']
y
```

Out[23]:

```
0        0
1        0
2        3
3        0
4        0
        ..
14995    0
14996    0
14997    0
14998    0
14999    0
Name: sex, Length: 15000, dtype: int64
```

In [24]:

```
#splitting the data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
```

In [25]:

```
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[25]:

```
▾ LinearRegression

LinearRegression()
```

In [26]:

```
score=model.score(x_train,y_train)
score
```

Out[26]:

```
9.581120157475809e-06
```

In [27]:

```
model.coef_
```

Out[27]:

```
array([0.00011896])
```

In [28]:

```
model.intercept_
```

Out[28]:

```
0.2200046732690924
```

FINE TUNING

In [29]:

```
#Defining parameters
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
```

In [30]:

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

}
```

In [31]:

```
#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[31]:

```
▸         GridSearchCV
▸ estimator: LinearRegression
      ▸ LinearRegression
```

In [32]:

```
#Best model
best_model=grid_search.best_estimator_
best_model
```

Out[32]:

```
▾ LinearRegression
LinearRegression(n_jobs=True, positive=True)
```

In [33]:

```
best_score=model.score(x_test,y_test)
best_score
```

Out[33]:

```
-0.0035208057122375624
```

In [34]:

```
y_pred=best_model.predict(x_test)
y_pred
```

Out[34]:

```
array([0.22143224, 0.2202426 , 0.22143224, ..., 0.22464428, 0.2202426 ,
       0.22285981])
```

In [35]:

```
mse=mean_squared_error(y_test,y_pred)
mse
```

Out[35]:

```
0.6336346329671283
```

In [ ]:

```

```