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

LOGISTIC REGRESSION AND FINE TUNING IT

In [1]:

- 1 #Required libraries
- 2 **from** sklearn.datasets **import** make_classification
- 3 from sklearn.model selection import GridSearchCV
- 4 **from** sklearn.linear_model **import** LogisticRegression
- 5 from sklearn.model_selection import train_test_split
- 6 import pandas as pd

In [2]:

- 1 df=pd.read_csv("C:\\Users\\Atwongire Vianney\\Desktop\\AI_PRAC\\03_Cluster
- 2 df

Out[2]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyba
	0 2007	NaN	NaN	0	0	0	0	0	1
	1 2007	F	17.41	49	0	0	1	0	1
	2 2007	F	17.511	41	0	0	0	0	1
	3 2006	F	NaN	36	0	0	0	0	1
	4 2008	F	16.657	1	0	0	0	0	1
			•••				•••	•••	
1499	2 008	F	16.329	21	0	0	0	0	1
1499	2008	F	16.545	50	0	0	0	0	1
1499	2007	F	17.999	32	0	0	0	0	1
1499	2007	F	17.903	20	0	0	0	0	1
1499	9 2009	F	15.811	25	0	0	7	0	1

15000 rows × 40 columns

```
In [3]: 1 df.head()
```

Out[3]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyball	SI
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]: 1 df.tail()

Out[4]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyba
14995	2008	F	16.329	21	0	0	0	0	1
14996	2008	F	16.545	50	0	0	0	0	1
14997	2007	F	17.999	32	0	0	0	0	1
14998	2007	F	17.903	20	0	0	0	0	1
14999	2009	F	15.811	25	0	0	7	0	1

5 rows × 40 columns

In [5]: 1 x=df['NumberOffriends'].array.reshape(-1,1)
2 x

[0], [29], [0],

[21], [0],

[89], [37],

[89],

```
1 y=df['sex']
 In [6]:
           2 | y
 Out[6]: 0
                  0
         1
                  0
         2
                  3
         3
                  0
         4
                  0
         14995
                  0
         14996
                  0
         14997
                  0
         14998
                  0
         14999
                  а
         Name: sex, Length: 15000, dtype: int64
 In [7]:
           1 #splitting the data
           2 | x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
 In [8]:
           1 x_train.shape
Out[8]: (12000, 1)
 In [9]:
              model=LogisticRegression()
           2 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]:
           1 y_pred=model.predict(x_test)
           2 y_pred
Out[10]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
In [11]:
             score=model.score(x train,y train)
           2 score
Out[11]: 0.89125
```

FINE TUNING

```
In [12]:
           1 #Defining parameters
           2 param_grid={
           3
                  'penalty':['l1','l2'],
                  'C':[0.001,0.01,0.1,1,10,100],
           4
           5
                  'solver':['liblinear','saga']
           6 }
In [13]:
           1 #performing grid with cross validation
           2 grid search=GridSearchCV(estimator=model,param grid=param grid,cv=5,n jobs
           3 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
           1 | #best parameters and best score
In [14]:
           2 best_param=grid_search.best_params_
           3 grid search
Out[14]:
                    GridSearchCV
           ▶ estimator: LogisticRegression
                LogisticRegression
In [15]:
           1 best_score=grid_search.best_score_
           2 best_score
Out[15]: 0.891416666666666
In [16]:
           1 | #evaluating the performance
           2 | score=grid_search.score(x_test,y_test)
           3 score
Out[16]: 0.9046666666666666
In [17]:
           1 y pred=model.predict(x test)
           2 y_pred
Out[17]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

FOR LINEAR REGRESSION

- In [18]: | 1 | import pandas as pd
 - 2 from sklearn.linear_model import LinearRegression
 - 3 from sklearn.model_selection import train_test_split

Out[19]:

gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyba
2007	NaN	NaN	0	0	0	0	0	
2007	F	17.41	49	0	0	1	0	1
2007	F	17.511	41	0	0	0	0	1
2006	F	NaN	36	0	0	0	0	1
2008	F	16.657	1	0	0	0	0	1
2008	F	16.329	21	0	0	0	0	1
2008	F	16.545	50	0	0	0	0	1
2007	F	17.999	32	0	0	0	0	1
2007	F	17.903	20	0	0	0	0	1
2009	F	15.811	25	0	0	7	0	I
	2007 2007 2007 2006 2008 2008 2008 2007 2007	2007 NaN 2007 F 2007 F 2006 F 2008 F 2008 F 2008 F 2008 F 2007 F 2007 F	2007 NaN NaN 2007 F 17.41 2007 F 17.511 2006 F NaN 2008 F 16.657 2008 F 16.329 2008 F 16.545 2007 F 17.999 2007 F 17.903	2007 NaN NaN 0 2007 F 17.41 49 2007 F 17.511 41 2006 F NaN 36 2008 F 16.657 1 2008 F 16.329 21 2008 F 16.545 50 2007 F 17.999 32 2007 F 17.903 20	2007 NaN NaN 0 0 2007 F 17.41 49 0 2007 F 17.511 41 0 2006 F NaN 36 0 2008 F 16.657 1 0 2008 F 16.329 21 0 2008 F 16.545 50 0 2007 F 17.999 32 0 2007 F 17.903 20 0	2007 NaN NaN 0 0 0 2007 F 17.41 49 0 0 2007 F 17.511 41 0 0 2006 F NaN 36 0 0 2008 F 16.657 1 0 0 2008 F 16.329 21 0 0 2008 F 16.545 50 0 0 2007 F 17.999 32 0 0 2007 F 17.903 20 0 0	2007 NaN NaN 0 0 0 0 2007 F 17.41 49 0 0 1 2007 F 17.511 41 0 0 0 2006 F NaN 36 0 0 0 2008 F 16.657 1 0 0 0 2008 F 16.329 21 0 0 0 2008 F 16.545 50 0 0 0 2007 F 17.999 32 0 0 0 2007 F 17.903 20 0 0 0	2007 NaN NaN 0 0 0 0 0 2007 F 17.41 49 0 0 1 0 2007 F 17.511 41 0 0 0 0 2006 F NaN 36 0 0 0 0 2008 F 16.657 1 0 0 0 0 2008 F 16.329 21 0 0 0 0 2008 F 16.545 50 0 0 0 0 2007 F 17.999 32 0 0 0 0 2007 F 17.903 20 0 0 0 0

15000 rows × 40 columns

In [20]: 1 df.head()

Out[20]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyball	SI
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]: 1 df.tail()
```

Out[21]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyba
14995	2008	F	16.329	21	0	0	0	0	1
14996	2008	F	16.545	50	0	0	0	0	1
14997	2007	F	17.999	32	0	0	0	0	1
14998	2007	F	17.903	20	0	0	0	0	1
14999	2009	F	15.811	25	0	0	7	0	1

5 rows × 40 columns

14995

14996

14997

14998

14999

0

0

0

0

```
1 x=df['NumberOffriends'].array.reshape(-1,1)
In [22]:
            2 x
Out[22]: <PandasArray>
          [0],
          [49],
          [41],
          [36],
          [1],
          [32],
          [18],
          [0],
          [0],
          [21],
          [0],
          [0],
          [29],
          [0],
          [89],
          [37],
          [89],
In [23]:
           1 y=df['sex']
Out[23]: 0
                   0
          1
                   0
          2
                   3
          3
                   0
          4
                   0
```

localhost:8888/notebooks/ATWONGERE VIANNEY 00574 FINE TUNING LINEAR AND LOGISTIC REGRESSION.ipynb#

Name: sex, Length: 15000, dtype: int64

```
In [24]:
           1 #splitting the data
           2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
In [25]:
             model=LinearRegression()
           2 model.fit(x_train,y_train)
Out[25]:
          LinearRegression
          LinearRegression()
In [26]:
              score=model.score(x train,y train)
           2
              score
Out[26]: 9.581120157475809e-06
In [27]:
           1 model.coef
Out[27]: array([0.00011896])
In [28]:
           1 model.intercept
Out[28]: 0.2200046732690924
         FINE TUNING
In [29]:
           1 #Defining parameters
           2 from sklearn.model_selection import GridSearchCV
           3 from sklearn.metrics import mean squared error
In [30]:
              param_grid={
           2
                  'copy_X':[True,False],
                  'fit_intercept':[True,False],
           3
           4
                  'n_jobs':[True,False],
           5
                  'positive':[True,False]
           6
           7
In [31]:
           1 | #performing grid with cross_validation
           2 grid_search=GridSearchCV(model,param_grid,cv=5,scoring='neg_mean_squared_e
           3 grid_search.fit(x_train,y_train)
Out[31]:
                    GridSearchCV
           ▶ estimator: LinearRegression
                 LinearRegression
```

```
In [32]:
           1 #Best model
           2 best_model=grid_search.best_estimator_
           3 best_model
Out[32]:
                        LinearRegression
          LinearRegression(n_jobs=True, positive=True)
In [33]:
           1 best_score=model.score(x_test,y_test)
           2 best_score
Out[33]: -0.0035208057122375624
           1 y_pred=best_model.predict(x_test)
In [34]:
           2 y_pred
Out[34]: array([0.22143224, 0.2202426 , 0.22143224, ..., 0.22464428, 0.2202426 ,
                0.22285981])
In [35]:
           1 mse=mean_squared_error(y_test,y_pred)
Out[35]: 0.6336346329671283
 In [ ]:
```