Logistic Regression Implementation with IRIS dataset

February 29, 2024

1 Logistic Regression Implementation

[1]:
```python
from sklearn.datasets import load_iris
```

[2]:
```python
dataset=load_iris()
```

[4]:
```python
print(dataset.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================
```

```
    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more …
```

[8]:
```
dataset.keys()
```

[8]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

[5]:
```
import pandas as pd
import numpy as np
```

[12]:
```
df=pd.DataFrame(dataset.data,columns=dataset.feature_names)
```

[13]:
```
df.head()
```

```
[13]:    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
      0                5.1               3.5                1.4               0.2
      1                4.9               3.0                1.4               0.2
      2                4.7               3.2                1.3               0.2
      3                4.6               3.1                1.5               0.2
      4                5.0               3.6                1.4               0.2
```

```
[16]: df['target']=dataset.target
```

```
[19]: df.head()
```

```
[19]:    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
      0                5.1               3.5                1.4               0.2
      1                4.9               3.0                1.4               0.2
      2                4.7               3.2                1.3               0.2
      3                4.6               3.1                1.5               0.2
      4                5.0               3.6                1.4               0.2

         target
      0       0
      1       0
      2       0
      3       0
      4       0
```

```
[20]: dataset.target
```

```
[20]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
[23]: ## binary classification

      df_copy=df[df['target']!=2]
```

```
[26]: df_copy.head()
```

```
[26]:    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
      0                5.1               3.5                1.4               0.2
      1                4.9               3.0                1.4               0.2
      2                4.7               3.2                1.3               0.2
      3                4.6               3.1                1.5               0.2
      4                5.0               3.6                1.4               0.2
```

```
   target
0       0
1       0
2       0
3       0
4       0
```

1.1 independent and dependent features

[51]:
```python
x=df_copy.iloc[:,:-1]
y=df_copy.iloc[:,-1]
```

[52]:
```python
from sklearn.linear_model import LogisticRegression
```

1.2 train test split

[53]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪33,random_state=42)
```

[54]:
```python
classifier=LogisticRegression()
```

[55]:
```python
classifier.fit(x_train,y_train)
```

[55]:
```
LogisticRegression()
```

1.3 prediction

[56]:
```python
y_pred=classifier.predict(x_test)
```

[57]:
```python
y_pred
```

[57]:
```
array([1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1])
```

[60]:
```python
y_test
```

[60]:
```
83    1
53    1
70    1
45    0
44    0
39    0
22    0
80    1
```

```
10    0
0     0
18    0
30    0
73    1
33    0
90    1
4     0
76    1
77    1
12    0
31    0
55    1
88    1
26    0
42    0
69    1
15    0
40    0
96    1
9     0
72    1
11    0
47    0
85    1
Name: target, dtype: int64
```

1.4 confusion matrix, accuracy score, classification report

[58]:
```
from sklearn.metrics import␣
↪confusion_matrix,accuracy_score,classification_report
```

[61]:
```
print(confusion_matrix(y_pred,y_test))
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

```
[[19  0]
 [ 0 14]]
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        14

    accuracy                           1.00        33
   macro avg       1.00      1.00      1.00        33
weighted avg       1.00      1.00      1.00        33
```

[62]:
```
classifier.predict_proba(x_test)
```

[62]:
```
array([[0.00144451, 0.99855549],
       [0.01745661, 0.98254339],
       [0.00359664, 0.99640336],
       [0.96427472, 0.03572528],
       [0.93476204, 0.06523796],
       [0.96670676, 0.03329324],
       [0.99197567, 0.00802433],
       [0.03414716, 0.96585284],
       [0.97022597, 0.02977403],
       [0.97501069, 0.02498931],
       [0.94876941, 0.05123059],
       [0.95432911, 0.04567089],
       [0.00502877, 0.99497123],
       [0.98309763, 0.01690237],
       [0.01045624, 0.98954376],
       [0.97788605, 0.02211395],
       [0.00251305, 0.99748695],
       [0.00154296, 0.99845704],
       [0.96995374, 0.03004626],
       [0.95518866, 0.04481134],
       [0.00835685, 0.99164315],
       [0.02378303, 0.97621697],
       [0.95343728, 0.04656272],
       [0.98084327, 0.01915673],
       [0.02881772, 0.97118228],
       [0.97696692, 0.02303308],
       [0.97878312, 0.02121688],
       [0.01706457, 0.98293543],
       [0.96440543, 0.03559457],
       [0.0018048 , 0.9981952 ],
       [0.96391062, 0.03608938],
       [0.97430387, 0.02569613],
       [0.00915764, 0.99084236]])
```

1.5 hyperparameter tuning grid search CV

[63]:
```
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
```

[69]:
```
parameter={'penalty':('l1','l2','elasticnet',None),'C':[1,10,20]}
```

[70]:
```
clf=GridSearchCV(classifier,param_grid=parameter,cv=5)
```

[71]:
```
## splitting of train data to calidation data

clf.fit(x_train,y_train)
```

[71]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'C': [1, 10, 20],
                         'penalty': ('l1', 'l2', 'elasticnet', None)})
```

[72]:
```
clf.best_params_
```

[72]:
```
{'C': 1, 'penalty': 'l2'}
```

[74]:
```
classifier=LogisticRegression(C=1,penalty='l2')
```

[75]:
```
classifier.fit(x_train,y_train)
```

[75]:
```
LogisticRegression(C=1)
```

[76]:
```
y_pred=classifier.predict(x_test)
```

[77]:
```
print(confusion_matrix(y_pred,y_test))
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

```
[[19  0]
 [ 0 14]]
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        14

    accuracy                           1.00        33
   macro avg       1.00      1.00      1.00        33
weighted avg       1.00      1.00      1.00        33
```

1.6 randomized search cv

[78]:
```
from sklearn.model_selection import RandomizedSearchCV
```

[79]:
```
random_clf=RandomizedSearchCV(LogisticRegression(),param_distributions=parameter,cv=5)
```

[80]:
```
random_clf.fit(x_train,y_train)
```

[80]:
```
RandomizedSearchCV(cv=5, estimator=LogisticRegression(),
                   param_distributions={'C': [1, 10, 20],
```

```
                                          'penalty': ('l1', 'l2', 'elasticnet',
                                                      None)})
```

[81]:
```python
random_clf.best_params_
```

[81]: {'penalty': None, 'C': 20}

[87]:
```python
random_clf=LogisticRegression(C=20,penalty=None)
```

[89]:
```python
random_clf.fit(x_train,y_train)
```

[89]: LogisticRegression(C=20, penalty=None)

[90]:
```python
y_pred=random_clf.predict(x_test)
```

[91]:
```python
print(confusion_matrix(y_pred,y_test))
print(accuracy_score(y_pred,y_test))
print(classification_report(y_pred,y_test))
```

```
[[19  0]
 [ 0 14]]
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        14

    accuracy                           1.00        33
   macro avg       1.00      1.00      1.00        33
weighted avg       1.00      1.00      1.00        33
```

[ ]:
```python

```