logistics_regression

February 10, 2019

```
In [5]: from sklearn.datasets import load_iris
        import pandas as pd

        iris_data = load_iris()
        print(iris_data.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 ...
```

In [7]:
```
X = iris_data.data
X_df = pd.DataFrame(X, columns=iris_data.feature_names)
X_df
```

Out[7]:
```
     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
5                  5.4               3.9                1.7               0.4
6                  4.6               3.4                1.4               0.3
7                  5.0               3.4                1.5               0.2
8                  4.4               2.9                1.4               0.2
```

9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3.0	1.4	0.1
13	4.3	3.0	1.1	0.1
14	5.8	4.0	1.2	0.2
15	5.7	4.4	1.5	0.4
16	5.4	3.9	1.3	0.4
17	5.1	3.5	1.4	0.3
18	5.7	3.8	1.7	0.3
19	5.1	3.8	1.5	0.3
20	5.4	3.4	1.7	0.2
21	5.1	3.7	1.5	0.4
22	4.6	3.6	1.0	0.2
23	5.1	3.3	1.7	0.5
24	4.8	3.4	1.9	0.2
25	5.0	3.0	1.6	0.2
26	5.0	3.4	1.6	0.4
27	5.2	3.5	1.5	0.2
28	5.2	3.4	1.4	0.2
29	4.7	3.2	1.6	0.2
..	...	...	...	...
120	6.9	3.2	5.7	2.3
121	5.6	2.8	4.9	2.0
122	7.7	2.8	6.7	2.0
123	6.3	2.7	4.9	1.8
124	6.7	3.3	5.7	2.1
125	7.2	3.2	6.0	1.8
126	6.2	2.8	4.8	1.8
127	6.1	3.0	4.9	1.8
128	6.4	2.8	5.6	2.1
129	7.2	3.0	5.8	1.6
130	7.4	2.8	6.1	1.9
131	7.9	3.8	6.4	2.0
132	6.4	2.8	5.6	2.2
133	6.3	2.8	5.1	1.5
134	6.1	2.6	5.6	1.4
135	7.7	3.0	6.1	2.3
136	6.3	3.4	5.6	2.4
137	6.4	3.1	5.5	1.8
138	6.0	3.0	4.8	1.8
139	6.9	3.1	5.4	2.1
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
145	6.7	3.0	5.2	2.3

```
        146                6.3               2.5                5.0               1.9
        147                6.5               3.0                5.2               2.0
        148                6.2               3.4                5.4               2.3
        149                5.9               3.0                5.1               1.8

        [150 rows x 4 columns]

In [10]: X_df['flower_type']=iris_data.target #append the target column, or Y
         X_df.groupby('flower_type').mean()

Out[10]:              sepal length (cm)  sepal width (cm)  petal length (cm)  \
         flower_type
         0                        5.006             3.428              1.462
         1                        5.936             2.770              4.260
         2                        6.588             2.974              5.552

                      petal width (cm)
         flower_type
         0                       0.246
         1                       1.326
         2                       2.026

In [12]: X_df.groupby('flower_type').max()

Out[12]:              sepal length (cm)  sepal width (cm)  petal length (cm)  \
         flower_type
         0                          5.8               4.4                1.9
         1                          7.0               3.4                5.1
         2                          7.9               3.8                6.9

                      petal width (cm)
         flower_type
         0                         0.6
         1                         1.8
         2                         2.5

In [13]: X_df.groupby('flower_type').min()

Out[13]:              sepal length (cm)  sepal width (cm)  petal length (cm)  \
         flower_type
         0                          4.3               2.3                1.0
         1                          4.9               2.0                3.0
         2                          4.9               2.2                4.5

                      petal width (cm)
         flower_type
         0                         0.1
         1                         1.0
         2                         1.4
```

```
In [23]: from sklearn.linear_model import LogisticRegression
         model = LogisticRegression(multi_class='ovr', solver='liblinear')
```

```
In [24]: from sklearn.model_selection import train_test_split
         y = iris_data.target
         X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33,random_state=
```

```
In [25]: y_test.shape
```

```
Out[25]: (50,)
```

```
In [26]: model.fit(X_train,y_train)
```

```
Out[26]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='ovr',
                    n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
                    tol=0.0001, verbose=0, warm_start=False)
```

```
In [28]: model.predict(X_train[0].reshape(1,-1))
```

```
Out[28]: array([1])
```

```
In [29]: y_train[0]
```

```
Out[29]: 1
```

```
In [30]: model.predict(X_test)
```

```
Out[30]: array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2, 0,
                2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1,
                2, 2, 1, 2])
```

```
In [31]: y_test
```

```
Out[31]: array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2, 0,
                2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1,
                2, 2, 1, 2])
```

```
In [33]: model.predict(X_test) == y_test
```

```
Out[33]: array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
                 True,  True,  True,  True,  True,  True,  True,  True,  True,
                 True,  True,  True,  True,  True,  True,  True,  True,  True,
                 True,  True,  True,  True,  True,  True,  True,  True,  True,
                 True,  True,  True,  True,  True,  True,  True,  True,  True,
                 True,  True,  True,  True,  True], dtype=bool)
```

```
In [34]: y_pred = model.predict(X_test)
         model.score(X_test,y_test)
```

```
Out[34]: 1.0
```

```
In [35]: from sklearn.metrics import classification_report
         print(classification_report(y_pred=y_pred, y_true=y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        15
           2       1.00      1.00      1.00        16

   micro avg       1.00      1.00      1.00        50
   macro avg       1.00      1.00      1.00        50
weighted avg       1.00      1.00      1.00        50
```