EnsembleLearning

November 18, 2021

SL Lab with Python 6: Ensemble Learning

Statistical Learning (Sejong University)

Date: 2021.11.17 (By: S. M. Riazul Islam)

1. Random Forest

```
In [58]: import numpy as np
         import pandas as pd

         from sklearn import datasets
         from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestClassifier
         from sklearn import metrics

In [59]: #Load iris dataset
         iris = datasets.load_iris()

         iris.keys()

Out[59]: dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])

In [60]: print(iris['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 [61]: # print the names of the four features
         print(iris.feature_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
In [62]: # print the label species(setosa, versicolor,virginica)
         print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

```
In [63]: # Create the pandas data frame
         irisdata = pd.DataFrame(iris['data'],columns=iris['feature_names'])
```

```
In [64]: irisdata.head()
```

```
Out[64]:    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
```

```
In [65]: irisdata["species"]=iris.target
         irisdata.head()
```

```
Out[65]:    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

            species
         0        0
         1        0
         2        0
         3        0
         4        0
```

```
In [66]: irisdata.sample(5)
```

```
Out[66]:      sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
         57                 4.9               2.4                3.3               1.0
         117                7.7               3.8                6.7               2.2
         1                  4.9               3.0                1.4               0.2
         12                 4.8               3.0                1.4               0.1
         141                6.9               3.1                5.1               2.3

              species
         57         1
         117        2
         1          0
         12         0
         141        2
```

```
In [67]: X=irisdata.iloc[:,:-1]
         # Predictors

         y=irisdata.iloc[:,-1]
         # Response
```

```
In [72]: # Split dataset into training set and test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

         #Create a RF Classifier
         rfclf=RandomForestClassifier(n_estimators=100)

         #Train the model
         rfclf.fit(X_train,y_train)

         #Predict
         y_pred=rfclf.predict(X_test)
```

```
In [69]: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9555555555555556
```

2. AdaBoost

```
In [73]: from sklearn.ensemble import AdaBoostClassifier

         # Split dataset into training set and test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

         # Create adaboost classifer object
         adaclf = AdaBoostClassifier(n_estimators=50,
                                     learning_rate=1)
         # Train the model
         adaclf.fit(X_train, y_train)
```

```
        #Predict the response
        y_pred = adaclf.predict(X_test)
```

```
In [74]: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9333333333333333
```

3. Gradient Boosting

```
In [75]: from sklearn.ensemble import GradientBoostingRegressor

         from sklearn.metrics import mean_squared_error
         from sklearn.metrics import mean_absolute_error

         from sklearn.datasets import load_boston

         boston = load_boston()
```

```
In [76]: housedata = pd.DataFrame(boston['data'],columns=boston['feature_names'])
         housedata["Price"]=boston.target

         housedata.head()
```

Out[76]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	

	PTRATIO	B	LSTAT	Price
0	15.3	396.90	4.98	24.0
1	17.8	396.90	9.14	21.6
2	17.8	392.83	4.03	34.7
3	18.7	394.63	2.94	33.4
4	18.7	396.90	5.33	36.2

```
In [77]: X=housedata.iloc[:,:-1]
         # Predictors

         y=housedata.iloc[:,-1]
         # Response
```

```
In [78]: # Split dataset into training set and test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
In [82]: # Create the model
         gboostreg = GradientBoostingRegressor(max_depth=2, n_estimators=5, learning_rate=1.0)

         # Train the modek
         gboostreg.fit(X_train, y_train)

         # Predict
         y_pred = gboostreg.predict(X_test)

In [83]: print("Average Eror:",mean_absolute_error(y_test, y_pred))

Average Eror: 3.0613357422697836


In [84]: print("Average Eror:",mean_squared_error(y_test, y_pred))

Average Eror: 24.219994889335933
```