sk_randomForest

April 15, 2019

1 Ransom Forest Algorithm

Random forest algorithm is an ensemble classification algorithm. Ensemble classifier means a group of classifiers. Instead of using only one classifier to predict the target, In ensemble, we use multiple classifiers to predict the target.

In case, of random forest, these ensemble classifiers are the randomly created decision trees. Each decision tree is a single classifier and the target prediction is based on the majority voting method.

The majority voting concept is same as the political votings. Each person votes per one political party out all the political parties participating in elections. In the same way, every classifier will votes to one target class out of all the target classes.

To declare the election results. The votes will calculate and the party which got the most number of votes treated as the election winner. In the same way, the target class which got the most number of votes considered as the final predicted target class.

```
In [1]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import accuracy_score, confusion_matrix
        from sklearn import datasets
```

```
In [2]: dataset = datasets.load_breast_cancer()
```

```
In [3]: cancer = pd.DataFrame(dataset.data, columns = dataset.feature_names)
        cancer.head()
```

Out[3]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	
1	0.07864	0.0869	0.07017	0.1812	
2	0.15990	0.1974	0.12790	0.2069	
3	0.28390	0.2414	0.10520	0.2597	

```
4          0.13280          0.1980              0.10430         0.1809

   mean fractal dimension           ...             worst radius  \
0                 0.07871           ...                    25.38
1                 0.05667           ...                    24.99
2                 0.05999           ...                    23.57
3                 0.09744           ...                    14.91
4                 0.05883           ...                    22.54

   worst texture  worst perimeter  worst area  worst smoothness  \
0          17.33           184.60      2019.0            0.1622
1          23.41           158.80      1956.0            0.1238
2          25.53           152.50      1709.0            0.1444
3          26.50            98.87       567.7            0.2098
4          16.67           152.20      1575.0            0.1374

   worst compactness  worst concavity  worst concave points  worst symmetry  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   worst fractal dimension
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678

[5 rows x 30 columns]
```

In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(cancer, dataset.target)
```

In [5]:
```
# model
rndmfrst = RandomForestClassifier(n_estimators=1000)
rndmfrst.fit(X_train, y_train)
```

Out[5]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=1000, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [6]:
```
# Prediction
y_pred = rndmfrst.predict(X_test)
```

```
        for i in range(0, 5):
            print("Actual outcome :: {} and predicted outcome :: {}".format(y_test[i], y_pred[i
```

```
Actual outcome :: 1 and predicted outcome :: 1
Actual outcome :: 0 and predicted outcome :: 0
Actual outcome :: 0 and predicted outcome :: 0
Actual outcome :: 0 and predicted outcome :: 0
Actual outcome :: 0 and predicted outcome :: 1
```

```
In [7]: # Train and Test Accuracy
        print ("Train Accuracy :: ", accuracy_score(y_train, rndmfrst.predict(X_train)))
        print ("Test Accuracy  :: ", accuracy_score(y_test, y_pred))
        print ("Confusion matrix :: \n", confusion_matrix(y_test, y_pred))
```

```
Train Accuracy ::  1.0
Test Accuracy  ::  0.958041958041958
Confusion matrix ::
 [[54  6]
 [ 0 83]]
```

1.1 References:

1. http://dataaspirant.com/2017/06/26/random-forest-classifier-python-scikit-learn/
2. https://en.wikipedia.org/wiki/Random_forest
3. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html