raj_choudhary_16bce1384_bagging_random_forest

September 18, 2018

0.1 Raj Choudhary

0.2 16BCE1384

0.3 Bagging –> Random Forest

0.3.1 Importing libraries

```
In [1]: import pandas as pd
```

0.3.2 Importing the dataset

```
In [2]: df_wine = pd.read_csv('wine.data',
                              header=None)

        df_wine.columns = ['Class label', 'Alcohol', 'Malic acid', 'Ash',
                           'Alcalinity of ash', 'Magnesium', 'Total phenols',
                           'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
                           'Color intensity', 'Hue', 'OD280/OD315 of diluted wines',
                           'Proline']

In [3]: df_wine = df_wine[df_wine['Class label'] != 1]

        y = df_wine['Class label'].values
        X = df_wine[['Alcohol', 'OD280/OD315 of diluted wines']].values
```

0.3.3 Preprocessing the dataset

```
In [4]: from sklearn.preprocessing import LabelEncoder
        from sklearn.model_selection import train_test_split

In [5]: le = LabelEncoder()
        y = le.fit_transform(y)

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

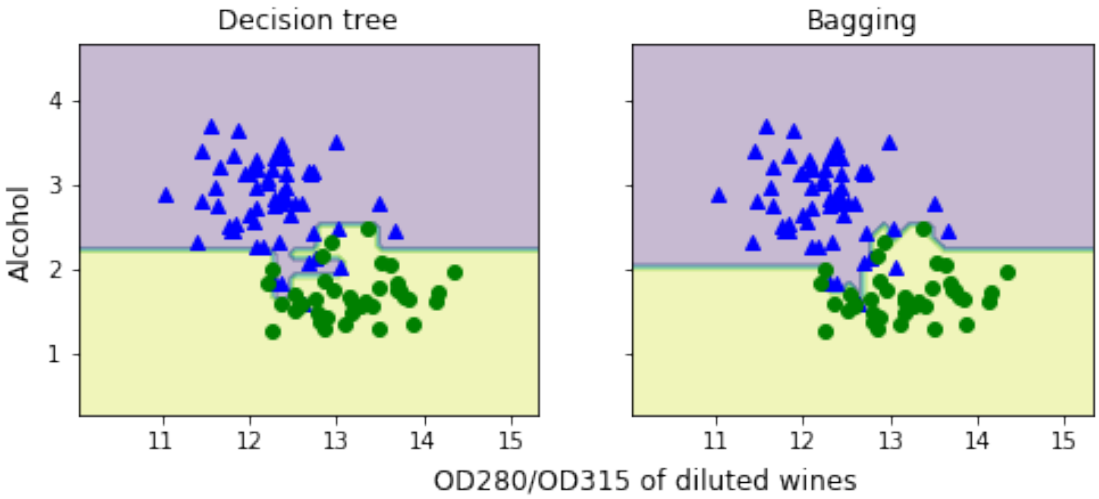
0.3.4 Comparing the result of classification between Decision Tree Classifier and Bagging Classifier

```
In [6]: from sklearn.ensemble import BaggingClassifier
        from sklearn.tree import DecisionTreeClassifier

In [7]: tree = DecisionTreeClassifier(criterion = 'entropy',
                                      max_depth = None,
                                      random_state = 1)
        bag = BaggingClassifier(base_estimator = tree,
                                n_estimators = 500,
                                max_samples = 1.0,
                                max_features = 1.0,
                                bootstrap = True,
                                bootstrap_features = False,
                                random_state = 1,
                                n_jobs = -1)

In [8]: from sklearn.metrics import accuracy_score

In [9]: tree = tree.fit(X_train, y_train)
        y_train_pred = tree.predict(X_train)
        y_test_pred = tree.predict(X_test)

        tree_train = accuracy_score(y_train, y_train_pred)
        tree_test = accuracy_score(y_test, y_test_pred)
        print('Decision tree train/test accuracies %.3f/%.3f'
              % (tree_train, tree_test))

Decision tree train/test accuracies 1.000/0.833


In [10]: bag = bag.fit(X_train, y_train)
         y_train_pred = bag.predict(X_train)
         y_test_pred = bag.predict(X_test)

         bag_train = accuracy_score(y_train, y_train_pred)
         bag_test = accuracy_score(y_test, y_test_pred)
         print('Bagging train/test accuracies %.3f/%.3f'
               % (bag_train, bag_test))

Bagging train/test accuracies 1.000/0.917
```

From the above train/test accuracies score, we conclude that Bagging Classifier even though it is also overfitting the training set data as the Decision Tree Classifier but it is better able to generalize the result for test set. As it has a test set accuracy score of 91.7% as compared to 83.3% of the Decision Tree Classifier

0.3.5 Plotting the decision boundary of Decision Tree Classifier and Bagging Classifier

In [11]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x_min = X_train[:, 0].min() - 1
x_max = X_train[:, 0].max() + 1
y_min = X_train[:, 1].min() - 1
y_max = X_train[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

f, axarr = plt.subplots(nrows=1, ncols=2,
                        sharex='col',
                        sharey='row',
                        figsize=(8, 3))


for idx, clf, tt in zip([0, 1],
                        [tree, bag],
                        ['Decision tree', 'Bagging']):
    clf.fit(X_train, y_train)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    axarr[idx].contourf(xx, yy, Z, alpha=0.3)
    axarr[idx].scatter(X_train[y_train == 0, 0],
                       X_train[y_train == 0, 1],
                       c='blue', marker='^')

    axarr[idx].scatter(X_train[y_train == 1, 0],
                       X_train[y_train == 1, 1],
                       c='green', marker='o')

    axarr[idx].set_title(tt)

axarr[0].set_ylabel('Alcohol', fontsize=12)
plt.text(10.2, -0.5,
         s='OD280/OD315 of diluted wines',
         ha='center', va='center', fontsize=12)

plt.show()
```



0.3.6 Comparing the classification result of Decision Tree Classifier and Random Forest Classifier

```
In [12]: from sklearn.ensemble import RandomForestClassifier
```

```
In [13]: tree = tree.fit(X_train, y_train)
         y_train_pred = tree.predict(X_train)
         y_test_pred = tree.predict(X_test)

         tree_train = accuracy_score(y_train, y_train_pred)
         tree_test = accuracy_score(y_test, y_test_pred)
         print('Decision tree train/test accuracies %.3f/%.3f'
               % (tree_train, tree_test))
```

```
Decision tree train/test accuracies 1.000/0.833
```

```
In [14]: forest = RandomForestClassifier(criterion = 'entropy', n_estimators = 1000, random_sta
         forest = forest.fit(X_train, y_train)
```

```
In [15]: y_train_pred = forest.predict(X_train)
         y_test_pred = forest.predict(X_test)

         forest_train = accuracy_score(y_train, y_train_pred)
         forest_test = accuracy_score(y_test, y_test_pred)
         print('Random Forest train/test accuracies %.3f/%.3f'
               % (forest_train, forest_test))
```

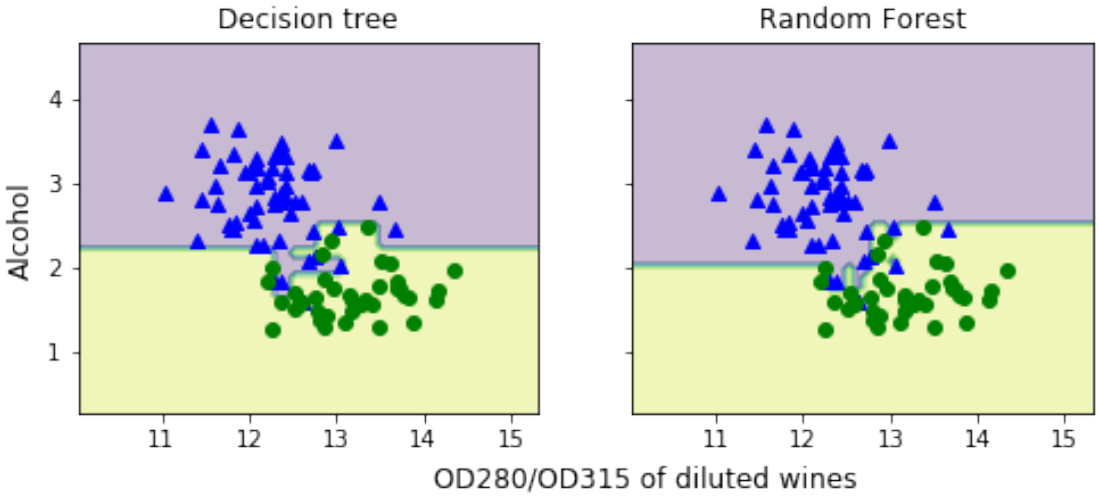
```
Random Forest train/test accuracies 1.000/0.917
```

From the above train/test accuracies score, we conclude that Random Forest Classifier even though it is also overfitting the training set data as the Decision Tree Classifier but it is better able to generalize the result for test set. As it has a test set accuracy score of 91.7% as compared to 83.3% of the Decision Tree Classifier

0.3.7 Plotting the decision boundary of Decision Tree Classifier and Random Forest Classifier

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x_min = X_train[:, 0].min() - 1
x_max = X_train[:, 0].max() + 1
y_min = X_train[:, 1].min() - 1
y_max = X_train[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

f, axarr = plt.subplots(nrows=1, ncols=2,
                        sharex='col',
                        sharey='row',
                        figsize=(8, 3))


for idx, clf, tt in zip([0, 1],
                        [tree, forest],
                        ['Decision tree', 'Random Forest']):
    clf.fit(X_train, y_train)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    axarr[idx].contourf(xx, yy, Z, alpha=0.3)
    axarr[idx].scatter(X_train[y_train == 0, 0],
                       X_train[y_train == 0, 1],
                       c='blue', marker='^')

    axarr[idx].scatter(X_train[y_train == 1, 0],
                       X_train[y_train == 1, 1],
                       c='green', marker='o')

    axarr[idx].set_title(tt)

axarr[0].set_ylabel('Alcohol', fontsize=12)
plt.text(10.2, -0.5,
         s='OD280/OD315 of diluted wines',
         ha='center', va='center', fontsize=12)
```

```
plt.show()
```



From the results, we can conclude that Bagging Classifier and Random Forest Classifier are better able to classify the result as compared to simple Decision Tree Classifier. We can also improve the performance of the two classifier by further tuning the respective hyperparameter via GridSearchCV.