Chapter 10: Demo BoostRegressor

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter10_Boosting/
```

In [3]:
```
# Load libraries
from sklearn.ensemble import AdaBoostRegressor
import pandas as pd
```

In [4]:
```
# Load data
iris = pd.read_excel("Iris.xls")
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepallength    150 non-null float64
sepalwidth     150 non-null float64
petallength    150 non-null float64
petalwidth     150 non-null float64
iris           150 non-null object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```
# 5 first samples
X = iris[["sepallength","sepalwidth","petalwidth"]]
X.head(3)
```

Out[5]:

	sepallength	sepalwidth	petalwidth
0	5.1	3.5	0.2
1	4.9	3.0	0.2
2	4.7	3.2	0.2

In [6]:
```
# 5 first result
y = iris[["petallength"]]
y.head(3)
```

Out[6]:

	petallength
0	1.4
1	1.4
2	1.3

AdaBoostRegressor

In [7]:
```
# Create adaboost-decision tree classifer object
# n_estimators: It controls the number of weak learners.
# learning_rate:Controls the contribution of weak learners in the final combinat
# There is a trade-off between learning_rate and n_estimators.
# base_estimators: It helps to specify different ML algorithm.
from sklearn.tree import DecisionTreeRegressor
ml = DecisionTreeRegressor(max_depth=4)
clf = AdaBoostRegressor(n_estimators=50,
                        base_estimator=ml,
                        learning_rate=1)
```

In [8]:
```
# Train model
clf.fit(X, y)
```

```
c:\program files\python36\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[8]:
```
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(criterion='mse',
                                                       max_depth=4,
                                                       max_features=None,
                                                       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,
                                                       presort=False,
                                                       random_state=None,
                                                       splitter='best'),
                  learning_rate=1, loss='linear', n_estimators=50,
                  random_state=None)
```

In [9]:
```
X_test = [[6.4, 3.2, 1.5], [5.9, 3. , 1.8]]
y_pred = clf.predict(X_test)
y_pred
```

Out[9]:
```
array([4.68235294, 4.95      ])
```

In [10]:
```
# với DecisionTreeRegressor mà không có AdaBoost
clf_1 = ml.fit(X,y)
```

In [11]:
```
y_pred = clf_1.predict(X_test)
y_pred
```

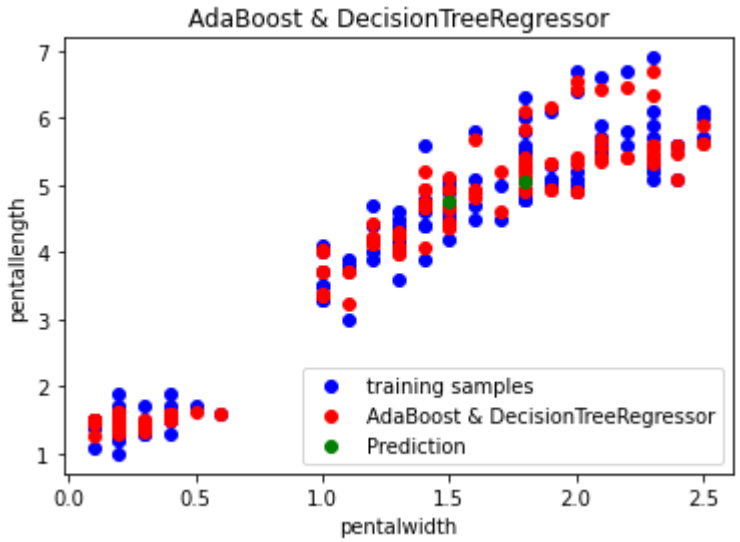
Out[11]:
```
array([4.74375   , 5.05217391])
```

In [12]:
```
X_test = pd.DataFrame(X_test)
X_test_width = X_test[2]
```

In [13]:
```
import matplotlib.pyplot as plt
```

In [14]:
```
# Plot the results
y_1 = clf.predict(X) # AdaBoost & DecisionTreeRegressor

plt.figure()
plt.scatter(X["petalwidth"], y, c="blue", label="training samples")
plt.scatter(X["petalwidth"], y_1, c="red",
            label="AdaBoost & DecisionTreeRegressor")
plt.scatter(X_test_width, y_pred, c="green", label="Prediction")
plt.xlabel("pentalwidth")
plt.ylabel("pentallength")
plt.title("AdaBoost & DecisionTreeRegressor")
plt.legend()
plt.show()
```



XGBRegressor

In [15]:
```
import xgboost as xgb
from sklearn.metrics import mean_squared_error
```

In [16]:
```
xgb_model = xgb.XGBRegressor(random_state=42)
xgb_model.fit(X, y)
```

```
[15:20:19] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objecti
ve/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarede
rror.
```

Out[16]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:linear', random_state=42,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=1, verbosity=1)
```

In [17]:
```
y_pred = xgb_model.predict(X)
print("Model r^2 score:", xgb_model.score(X,y))
```

```
Model r^2 score: 0.9917113219896987
```

In [18]:
```
mse=mean_squared_error(y, y_pred)
print(mse)
```

```
0.025632114175041725
```

In [19]:
```
# predict new sample
X_new = pd.DataFrame({'sepallength': [6.4,5.9],
                      'sepalwidth':[3.2,3],
                      'petalwidth':[1.5,1.8]
})
xgb_model.predict(X_new)
```

Out[19]:
```
array([4.634132, 4.925619], dtype=float32)
```

In [ ]:
```

```