Boosting

68_Adnan Shaikh

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
import pandas as pd
df=pd.read_csv('./apples_and_oranges.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Weight	Size	Class
0	69	4.39	orange
1	69	4.21	orange
2	65	4.09	orange
3	72	5.85	apple
4	67	4.70	orange

In [4]:

```
df.shape
```

Out[4]:

```
(40, 3)
```

In [5]:

```
x=df.drop("Class",axis="columns")
y=df.Class
```

In [6]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

In [7]:

```
x_test.shape
```

Out[7]:

```
(8, 2)
```

In [8]:

```
x_train.shape
```

Out[8]:

```
(32, 2)
```

In [9]:

```
from sklearn.ensemble import AdaBoostClassifier
```

In [10]:

```
ada=AdaBoostClassifier(n_estimators=100,base_estimator=None,learning_rate=1,random_state=1)
ada.fit(x_train,y_train)
```

Out[10]:

```
AdaBoostClassifier(learning_rate=1, n_estimators=100, random_state=1)
```

In [11]:

```
y_pred=ada.predict(x_test)
```

In [12]:

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[3 0]
 [0 5]]
```

In [13]:
```
accuracy=float(cm.diagonal().sum())/len(y_test)
```

In [14]:
```
print("Accuracy of Adaboost for Given Data set :",accuracy)
```

```
Accuracy of Adaboost for Given Data set : 1.0
```

In [15]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00         3
      orange       1.00      1.00      1.00         5

    accuracy                           1.00         8
   macro avg       1.00      1.00      1.00         8
weighted avg       1.00      1.00      1.00         8
```

Gradient Boosting

In [16]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [17]:
```
gb = GradientBoostingClassifier(n_estimators=100)

gb.fit(x_train,y_train)
```

Out[17]:
```
GradientBoostingClassifier()
```

In [18]:
```
y_pred = gb.predict(x_test)
```

In [19]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00         3
      orange       1.00      1.00      1.00         5

    accuracy                           1.00         8
   macro avg       1.00      1.00      1.00         8
weighted avg       1.00      1.00      1.00         8
```

Xtreme Gradient Boosting

In [20]:
```
from xgboost import XGBClassifier
```

In [21]:
```
xgb = XGBClassifier(n_estimators=200,reg_alpha=1)
```

In [22]:
```
xgb.fit(x_train,y_train)
```

```
[15:02:49] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:10
95: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logis
tic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore th
e old behavior.
```

Out[22]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=200, n_jobs=12, num_parallel_tree=1, random_state=0,
              reg_alpha=1, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [23]:
```
y_pred = xgb.predict(x_test)
```

```
In [24]: print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

       apple       0.60      1.00      0.75         3
      orange       1.00      0.60      0.75         5

    accuracy                           0.75         8
   macro avg       0.80      0.80      0.75         8
weighted avg       0.85      0.75      0.75         8
```