Ensemble Methods

RandomForest

Comparing the Decision Tree & Random Forest for MNIST data

```
from sklearn.datasets import load_digits
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
digits = load_digits()
X = digits.data
y = digits.target
trainX, testX, trainY, testY = train_test_split(X,y)
dt = DecisionTreeClassifier()
dt.fit(trainX,trainY)
dt.score(testX,testY)
rf = RandomForestClassifier()
rf.fit(trainX,trainY)
rf.score(testX,testY)
```

```
0.9777777777777777
```

Important Hyper-parameters

```
rf.feature_importances_
```

```
array([0.00000000e+00, 2.32963239e-03, 1.88528633e-02, 9.92553213e-03,
       8.97869357e-03, 2.02796036e-02, 6.66994345e-03, 9.19316774e-04,
       0.00000000e+00, 1.09732112e-02, 2.79608251e-02, 7.36534586e-03,
       1.52660488e-02, 2.73800861e-02, 5.46515165e-03, 6.98788222e-04,
       9.38989356e-05, 9.24805900e-03, 1.74748070e-02, 2.53573243e-02,
       3.03037526e-02, 4.76319396e-02, 7.60957370e-03, 3.44664321e-04,
       6.71365418e-05, 1.26498911e-02, 4.74825583e-02, 2.37436571e-02,
       3.60216512e-02, 2.38686480e-02, 3.20394087e-02, 6.86854412e-05,
       0.00000000e+00, 3.14180940e-02, 2.19818991e-02, 1.58647188e-02,
       3.47904652e-02, 2.41431298e-02, 2.73144385e-02, 0.00000000e+00,
       3.20285155e-05, 1.21782526e-02, 3.48600994e-02, 4.33129883e-02,
       2.09005271e-02, 2.00248309e-02, 2.10802784e-02, 2.03504875e-04,
       4.58013065e-05, 2.90810286e-03, 1.46544163e-02, 2.13935996e-02,
       1.61705101e-02, 2.47115218e-02, 2.30374624e-02, 2.35128653e-03,
       6.38731122e-05, 2.32151293e-03, 2.34900976e-02, 8.86341121e-03,
       2.67099176e-02, 3.01076430e-02, 1.37928249e-02, 4.20206517e-03])
```

AdaBoost

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

```
from sklearn.ensemble import AdaBoostClassifier
ab = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=8),n_estimators=60
ab.fit(trainX,trainY)
ab.score(testX,testY)
ab = AdaBoostClassifier(base_estimator=RandomForestClassifier(n_estimators=20),n_estimator
ab.fit(trainX,trainY)
ab.score(testX,testY)
```

```
0.9644444444444444
```

GradientBoostingTree

```
from sklearn.datasets import load_boston
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
house_data = load_boston()
X = house_data.data
y = house_data.target
from sklearn.ensemble import GradientBoostingRegressor
gbt = GradientBoostingRegressor()
gbt
```

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3,
                          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, n_estimators=100,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

```
from sklearn.model_selection import train_test_split
trainX, testX, trainY, testY = train_test_split(X,y)
gbt.fit(trainX,trainY)
test_score = np.zeros(100, dtype=np.float64)
for i, y_pred in enumerate(gbt.staged_predict(testX)):
 test_score[i] = gbt.loss_(testY, y_pred)
test_score
plt.plot(test_score)
plt.xlabel('Iterations')
plt.ylabel('Least squares Loss')
```

```
Text(0, 0.5, 'Least squares Loss')
```

VotingClassier

```
from sklearn.ensemble import VotingClassifier,RandomForestClassifier,AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
estimators = [
 ('rf',RandomForestClassifier(n_estimators=20)),
 ('svc',SVC(kernel='rbf', probability=True)),
 ('knc',KNeighborsClassifier()),
 ('abc',AdaBoostClassifier(base_estimator=DecisionTreeClassifier() ,n_estimators=20)),
 ('lr',LogisticRegression())
]
vc = VotingClassifier(estimators=estimators, voting='hard')
digits = load_digits()
X,y = digits.data, digits.target
trainX, testX, trainY, testY = train_test_split(X,y)
vc.fit(trainX,trainY)
vc.score(testX,testY)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
0.9755555555555555
```

```
for est,name in zip(vc.estimators_,vc.estimators):
 print (name[0], est.score(testX,testY))
 vc = VotingClassifier(estimators=estimators, voting='soft', weights=[2,.1,3,2,2])
 vc.fit(trainX,trainY)
 vc.score(testX,testY)
```

```
rf 0.9577777777777777
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
svc 0.98
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
knc 0.9822222222222222
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
abc 0.8222222222222222
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
lr 0.96
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```