Vins_Random_Tree_Regressor_Airbnb

December 3, 2019

[43]:
```
import os
datapath = "finaldata.csv"

# Load system libraries
import sys
import datetime
import random
import numpy as np

# Load ML libraries
import pandas as pd

from matplotlib import pyplot
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor

from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.model_selection import cross_val_score

import warnings
warnings.filterwarnings("ignore")

from sklearn.tree import export_graphviz

# To plot pretty figures
%matplotlib inline
```

[44]:
```
def _train(alg, algName, X_train, Y_train, X_test, Y_test):
    print(datetime.datetime.now(), "Begin training: ", algName)
    alg.fit(X_train, Y_train)
    print(datetime.datetime.now(), "End training: ", algName)
```

[45]:
```
def _predict(alg, algName, X_train, Y_train, X_test, Y_test):
    print(datetime.datetime.now(), "Begin prediction: ", algName)
    predictions = alg.predict(X_test)
    print(datetime.datetime.now(), "End prediction: ", algName)
```

[46]:
```
# Load the data
NYC_data = pd.read_csv(datapath)
```

[47]:
```
# split data into train/test datasets
print("Splitting data into training and test sets")
array = NYC_data.values
X = array[1:,:6]
Y = array[1:,6:]

print("X Shape: " , X.shape)
print("Y Shape: " , Y.shape)

test_size = 0.20
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y,␣
 ↪test_size=test_size, random_state=seed)
```

```
Splitting data into training and test sets
X Shape:  (47188, 6)
Y Shape:  (47188, 1)
```

[48]:
```
#RANDOM FOREST MODEL
from sklearn.ensemble import RandomForestRegressor

#Initialize regressor
rnd_rgr = RandomForestRegressor(bootstrap="true", random_state=42,␣
 ↪n_estimators=100)

#train it
rnd_rgr.fit(X_train, Y_train.ravel())
```

[48]:
```
RandomForestRegressor(bootstrap='true', criterion='mse', 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=100, n_jobs=None,
           oob_score=False, random_state=42, verbose=0, warm_start=False)
```

[49]:
```
#EVALUATING RESULTS
print("Accuracy: " + str(round((1-rnd_rgr.score(X_test, Y_test))*100,2)) + "%")
```

```
Accuracy: 77.43%
```

[50]:
```
#How useful were the features?
print(rnd_rgr.feature_importances_)
```

```
[0.05128558 0.31901556 0.31714861 0.14146499 0.08813001 0.08295526]
```

[51]:
```
#MAKING A SINGLE PREDICTION
print(rnd_rgr.predict([[102, 40.72042, -73.98662, 1, 30, 200]]))
```

```
[104.23]
```

[52]:
```
from sklearn.linear_model import LogisticRegression

lor = LogisticRegression()
_train(lor, "Logistic Regression", X_train, Y_train, X_test, Y_test)
_predict(lor, "Logistic Regression", X_train, Y_train, X_test, Y_test)

lor.score(X_test,Y_test)
```

```
2019-12-03 19:23:02.314334 Begin training:  Logistic Regression
2019-12-03 19:25:35.647569 End training:  Logistic Regression
2019-12-03 19:25:35.647882 Begin prediction:  Logistic Regression
2019-12-03 19:25:35.728564 End prediction:  Logistic Regression
```

[52]:
```
0.06675143038779402
```

[37]:
```
#SUPPORT VECTOR REGRESSOR
from sklearn.svm import SVR

X = array[1:,:6]
Y = array[1:,6]

X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y,␣
 ↪test_size=test_size, random_state=seed)

svm = SVR(kernel='rbf', C=.1, gamma=0.1, epsilon=.1)

_train(svm, "Support Vector", X_train, Y_train, X_test, Y_test)
_predict(svm, "Support Vector", X_train, Y_train, X_test, Y_test)

print("Error score:")
svm.score(X_test,Y_test)
#NOTE: SVM does return a negative score, -0.038, this is normal and expected
```

```
2019-12-03 18:56:42.862210 Begin training:  Support Vector
2019-12-03 18:57:47.944007 End training:  Support Vector
2019-12-03 18:57:47.944327 Begin prediction:  Support Vector
2019-12-03 18:57:56.017192 End prediction:  Support Vector
Error score:
```

[37]:
```
-0.038257963935729666
```

[54]:
```
#NEW, BETTER REGRESSOR
X = array[1:,:6]
Y = array[1:,6:]

X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y,␣
 ↪test_size=test_size, random_state=seed)
#Initialize regressor
rnd_rgr = RandomForestRegressor(random_state=1 ,n_estimators=3)
#Train Random Forest Model
rnd_rgr.fit(X_train, Y_train)
#Evaluate results
print("Accuracy: " + str(round((1-rnd_rgr.score(X_test, Y_test))*100,2)) + "%")
```

```
Accuracy: 99.64%
```

[ ]:
```

```