HW4_edgarsp2

October 14, 2019

1 STAT 542 / CS 598: Homework 4

Fall 2019, by Edgar Pino
Due: Monday, Oct 14 by 11:59 PM Pacific Time

[475]:
```
import pandas as pd
import numpy as np
import itertools
import operator
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
```

2 Question 1 [70 Points] Tuning Random Forests in Virtual Twins

[476]:
```
health_df = pd.read_csv('./data/sepsis.csv')
```

[477]:
```
health_df = health_df.drop(columns=['Unnamed: 0'])
```

[478]:
```
health_df.sample(10)
```

[478]:
```
        Health  THERAPY  PRAPACHE     AGE  BLGCS  ORGANNUM         BLIL6  \
165   0.356651        0        19  59.871     15         1     92.800000
271  -1.350463        0        26  56.098     15         1    406.600000
369   2.151098        0        21  53.656     15         2    406.600000
408   0.590776        0        37  49.823      6         2    301.800000
170  -0.251315        1        19  49.823     15         2    723.000000
47   -3.000959        0        20  33.174     11         1  28928.448742
270  -3.314716        0        26  65.891     14         2   1232.000000
260  -1.717820        1        25  33.174     15         2     37.100000
401  -0.023171        0        25  59.871     13         5   1232.000000
366  -2.139926        0        37  76.068     11         2     60.300000

     BLLPLAT  BLLBILI  BLLCREAT  TIMFIRST  BLADL  blSOFA  BEST
165    244.0      1.0       1.0     13.78    0.0     4.0     0
271    153.0      2.5       1.5     21.73    0.0     5.0     0
369    191.0      0.9       1.0     34.83    0.0     3.0     0
```

```
408    172.0   2.5  20.0  50.67  0.0  13.0  1
170     45.0   3.4   1.0  10.00  0.0  10.0  1
47     153.0   2.5   1.0  30.67  0.0   8.0  1
270    244.0   0.9   5.0  24.50  9.0   8.0  0
260     78.0   1.0   1.5  19.33  1.0  11.0  1
401     45.0   0.9   3.8  59.17  0.0  11.0  0
366    359.0   0.9   3.0  30.67  2.0   7.0  1
```

[479]:
```
health_active = health_df.loc[health_df['THERAPY'] == 1]
```

[480]:
```
health_control = health_df.loc[health_df['THERAPY'] == 0]
```

2.1 Randomly split the data into 75% for training and 25% for testing.

[481]:
```
y_active = health_active.Health
```

[482]:
```
x_active = health_active.drop(columns=['Health'])
```

[483]:
```
y_control = health_control.Health
```

[484]:
```
x_control = health_control.drop(columns=['Health'])
```

[485]:
```
X_active_train, X_active_test, y_active_train, y_active_test =␣
↪train_test_split(x_active, y_active, test_size=0.25)
```

[486]:
```
X_control_train, X_control_test, y_control_train, y_control_test =␣
↪train_test_split(x_control, y_control, test_size=0.25)
```

2.2 For the training data, fit the virtual twins model and then use the testing data to suggest the best treatment.

- You should not use the variable BEST when fitting the models
- Pick three different mtry values and three different nodesize, leave all other tuning parameters as default
- After predicting the best treatment in the testing data, compare it to the truth BEST

[487]:
```
max_features_vals = [6,8,10]
```

[488]:
```
min_samples_leaf = [5,10,20]
```

[489]:
```
params = []
```

[490]:
```
for r in itertools.product(max_features_vals, min_samples_leaf):
    params.append((r[0],r[1]))
```

[491]:
```
def fit_random_forest(x, y, max_features=5, min_samples_leaf=1):
    regresssor = RandomForestRegressor(max_features=max_features,␣
↪min_samples_leaf=min_samples_leaf, n_estimators=10)
    return regresssor.fit(x, y)
```

[492]:
```
def step(max_features=5, min_samples_leaf=1):
```

```
    X_active_train, X_active_test, y_active_train, y_active_test =␣
↪train_test_split(x_active, y_active, test_size=0.25)
    X_control_train, X_control_test, y_control_train, y_control_test =␣
↪train_test_split(x_control, y_control, test_size=0.25)

    active_regresssor = fit_random_forest(X_active_train.drop(columns=['BEST',␣
↪'THERAPY']), y_active_train, max_features, min_samples_leaf)
    control_regresssor = fit_random_forest(X_control_train.
↪drop(columns=['BEST', 'THERAPY']), y_control_train, max_features,␣
↪min_samples_leaf)

    x_test = pd.concat([X_active_test, X_control_test])

    y_active_pred = active_regresssor.predict(x_test.drop(columns=['BEST',␣
↪'THERAPY']))
    y_control_pred = control_regresssor.predict(x_test.drop(columns=['BEST',␣
↪'THERAPY']))

    y_pred = []
    for i in range(len(x_test)):
        if y_active_pred[i] > y_control_pred[i]:
            y_pred.append(1)
        else:
            y_pred.append(0)

    return accuracy_score(x_test.BEST, y_pred)
```

[493]:
```
def test_params(params):
    best_accuracy = 0
    best_params = {}

    for (features, lefs) in params:
        step_accuracy = step(max_features=features, min_samples_leaf=lefs)
        if step_accuracy > best_accuracy:
            best_params['max_features'] = features
            best_params['min_samples_leaf'] = lefs
            best_accuracy = step_accuracy

    return best_accuracy, best_params
```

[498]:
```
accuracies = []
top_param = {}
for i in range(100):
    accuracy, best_params = test_params(params)
    key = f"{best_params['max_features']}-{best_params['min_samples_leaf']}"
    if key in top_param:
        top_param[key]+= 1
```

```
    else:
        top_param[key] = 1

    accuracies.append(accuracy)
```

[499]:
```
accuracies = np.array(accuracies)
```

[500]:
```
(best_max_features, best_min_samples_leaf) = sorted(top_param.items(),␣
 ↪key=operator.itemgetter(1), reverse=True)[0][0].split('-')
```

[501]:
```
print(f"Got an average prediction accuracy of {accuracies.mean()}. The best␣
 ↪max_features is {best_max_features} and best min_samples_leaf is␣
 ↪{best_min_samples_leaf}")
```

```
Got an average prediction accuracy of 0.8639495798319327. The best max_features
is 10 and best min_samples_leaf is 20
```

3 Question 2 [30 Points] Second Step in Virtual Twins

[502]:
```
from sklearn import tree
```

3.0.1 Fit Virtual Twins model

[503]:
```
best_max_features = int(best_max_features)
```

[504]:
```
best_min_samples_leaf = int(best_min_samples_leaf)
```

[505]:
```
active_regresssor_2 = fit_random_forest(x_active.drop(columns=['BEST',␣
 ↪'THERAPY']), y_active, max_features=best_max_features,␣
 ↪min_samples_leaf=best_min_samples_leaf)
```

[506]:
```
control_regresssor_2 = fit_random_forest(x_control.drop(columns=['BEST',␣
 ↪'THERAPY']), y_control, max_features=best_max_features,␣
 ↪min_samples_leaf=best_min_samples_leaf)
```

[507]:
```
x_test_2 = pd.concat([x_active, x_control])
```

[508]:
```
y_active_pred = active_regresssor_2.predict(x_test_2.drop(columns=['BEST',␣
 ↪'THERAPY']))
y_control_pred = control_regresssor_2.predict(x_test_2.drop(columns=['BEST',␣
 ↪'THERAPY']))
```

[509]:
```
y_pred_twin = []
for i in range(len(x_test_2)):
    if y_active_pred[i] > y_control_pred[i]:
        y_pred_twin.append(1)
    else:
        y_pred_twin.append(0)
```

[510]:
```
twins_accuracy = accuracy_score(x_test_2.BEST, y_pred_twin)
```

[511]:
```
y_train = y_pred_twin
```

3.0.2 Fit CART model

[518]:
```
x_train = health_df.drop(columns=['BEST', 'THERAPY', 'Health'])
```

[534]:
```
clf = tree.DecisionTreeClassifier(max_features=best_max_features,␣
 ↪min_samples_leaf=best_min_samples_leaf)
```

[535]:
```
clf = clf.fit(x_train, y_train)
```

[536]:
```
y_pred_tree = clf.predict(x_train)
```

[537]:
```
tree_accuracy = accuracy_score(health_df.BEST, y_pred_tree)
```

3.0.3 Show accuracies

[539]:
```
print(f"Virtual Twins accuracy: {twins_accuracy}. CART model accuracy:␣
 ↪{tree_accuracy}")
```

```
Virtual Twins accuracy: 0.8404255319148937. CART model accuracy:
0.6191489361702127
```

[ ]:
```

```