Bootstrap_Random_Forest

May 12, 2020

1 Application of Bootstrap samples in Random Forest

[1]:
```
import numpy as np
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error
```

[2]:
```
boston = load_boston()
x=boston.data #independent variables
y=boston.target #target variable
```

[3]:
```
x.shape
```

[3]:
```
(506, 13)
```

2 TASK 1

[4]:
```
import random
from sklearn.tree import DecisionTreeRegressor
```

[5]:
```
#function Task1
def Task1(x,y):
    num=len(x)
    sample_index=[]
    column_index=[]
    oob_index=[]

    for i in range(30):
        sample=np.random.choice(num,size=int(0.6*num),replace=False)
        sample_rep=np.random.choice(sample,size=num-int(0.6*num),replace=True)
        sample_index.append(np.concatenate((sample,sample_rep)))
        size=random.randint(3,12)
        column=np.random.choice(13,size=size,replace=False)
        column_index.append(column)
        oob_index.append(list(set(range(num))-set(sample)))

    regressor = DecisionTreeRegressor(random_state=0)
```

```
    y_pred=[]
    y_pred_oob=[]
    for row in range(num):
        datapoint=x[row]
        y_predsample=[]
        count=0
        y_pred_oob_sum=0
        for i in range(30):
            x_sampledata=x[np.ix_(sample_index[i],column_index[i])]
            y_sampledata=np.take(y,sample_index[i])
            regressor.fit(x_sampledata,y_sampledata)
            predicted=regressor.predict(np.take(datapoint,column_index[i]).
↪reshape(1,-1))
            y_predsample.extend(predicted)
            if row in oob_index[i]:
                count+=1
                y_pred_oob_sum+=predicted
        y_pred.append(np.mean(y_predsample))
        y_pred_oob.extend(y_pred_oob_sum/count)

    MSE=np.mean(np.square(y-y_pred))
    oobscore=np.mean(np.square(y-y_pred_oob))

    return MSE,oobscore
```

[6]:
```
MSE,oobscore=Task1(x,y)
print("MSE value is {}".format(MSE))
print("OOB Score is {}".format(oobscore))
```

```
MSE value is 2.3889127287039673
OOB Score is 13.625955623196374
```

3 TASK 2

[7]:
```
mse_=[]
oob_=[]
for k in range(35):
    MSE,oobscore=Task1(x,y)
    mse_.append(MSE)
    oob_.append(oobscore)
```

[8]:
```
# confidence interval of MSE
mse_sample=np.array(mse_)
sample_mean = mse_sample.mean()
sample_std =  mse_sample.std()
```

```
sample_size = len(mse_sample)

lower  = np.round(sample_mean - 2*(sample_std/np.sqrt(sample_size)), 3)
upper = np.round(sample_mean + 2*(sample_std/np.sqrt(sample_size)), 3)
print('95 percent confidence interval of MSE  %.1f and %.1f' % (lower, upper))
```

```
95 percent confidence interval of MSE  2.4 and 2.7
```

[9]:
```
# confidence interval of OOB Score
oob_sample=np.array(oob_)
sample_mean = oob_sample.mean()
sample_std =  oob_sample.std()
sample_size = len(oob_sample)

lower  = np.round(sample_mean - 2*(sample_std/np.sqrt(sample_size)), 3)
upper = np.round(sample_mean + 2*(sample_std/np.sqrt(sample_size)), 3)
print('95 percent confidence interval of OOB Score %.1f and %.1f' % (lower,␣
 ↪upper))
```

```
95 percent confidence interval of OOB Score 13.7 and 14.8
```

4 TASK 3

[10]:
```
xq=[0.18,20.0,5.00,0.0,0.421,5.60,72.2,7.95,7.0,30.0,19.1,372.13,18.60]
```

[11]:
```
num=len(x)
sample_index=[]
column_index=[]

for i in range(30):
    sample=np.random.choice(num,size=int(0.6*num),replace=False)
    sample_rep=np.random.choice(sample,size=num-int(0.6*num),replace=True)
    sample_index.append(np.concatenate((sample,sample_rep)))
    size=random.randint(3,12)
    column=np.random.choice(13,size=size,replace=False)
    column_index.append(column)

regressor = DecisionTreeRegressor(random_state=0)
y_predsample=[]
for i in range(30):
    x_sampledata=x[np.ix_(sample_index[i],column_index[i])]
    y_sampledata=np.take(y,sample_index[i])
    regressor.fit(x_sampledata,y_sampledata)
    predicted=regressor.predict(np.take(xq,column_index[i]).reshape(1,-1))
    y_predsample.extend(predicted)
y_pred=np.mean(y_predsample)
print("Predicted House Price for given query point xq is {} ".format(y_pred))
```

```
Predicted House Price for given query point xq is 20.788888888888888
```

[ ]: