Bootstrap_Random_Forest_instructions

February 24, 2020

1 Application of Bootstrap samples in Random Forest

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

Load the boston house dataset

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

1.0.1 Task: 1

Step 1 Creating samples: Randomly create 30 samples from the whole boston data points.

Creating each sample: Consider any random 303(60% of 506) data points from whole data set and then replicate any 203 points from the sampled points

Ex: For better understanding of this procedure lets check this examples, assume we have 10 data points [1,2,3,4,5,6,7,8,9,10], first we take 6 data points randomly consider we have selected [4, 5, 7, 8, 9, 3] now we will replciate 4 points from [4, 5, 7, 8, 9, 3], consder they are [5, 8, 3,7] so our final sample will be [4, 5, 7, 8, 9, 3, 5, 8, 3,7]

we create 30 samples like this

Note that as a part of the Bagging when you are taking the random samples make sure each of the sample will have different set of columns

Ex: assume we have 10 columns for the first sample we will select [3, 4, 5, 9, 1, 2] and for the second sample [7, 9, 1, 4, 5, 6, 2] and so on...

Make sure each sample will have atleast 3 feautres/columns/attributes

Step 2 Building High Variance Models on each of the sample and finding train MSE value: Build a DecisionTreeRegressor on each of the sample.

Build a regression trees on each of 30 samples.

computed the predicted values of each data point(506 data points) in your corpus.

predicted house price of i^{th} data point $y^i_{pred} = \frac{1}{30}\sum_{k=1}^{30}$(predicted value of x^i with k^{th} model).

Now calculate the $MSE = \frac{1}{506}\sum_{i=1}^{506}(y^i - y^i_{pred})^2$.

Step 3 Calculating the OOB score :

Computed the predicted values of each data point(506 data points) in your corpus.

Predicted house price of i^{th} data point $y^i_{pred} = \frac{1}{k}\sum_{\text{k= model which was buit on samples not included } x^i}$(predicted value of x^i

Now calculate the $OOBScore = \frac{1}{506}\sum_{i=1}^{506}(y^i - y^i_{pred})^2$.

1.0.2 Task: 2

1.0.3 Task: 3

1.1 Task: 1

In [3]:
```
from sklearn.tree import DecisionTreeRegressor
import pandas as pd
import random
from tqdm import tqdm
```

In [4]:
```
# Converting x values to dataframe
data = pd.DataFrame(data=x[:,:], index= range(len(x)), columns=boston.feature_names)
data.head()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	

	PTRATIO	B	LSTAT
0	15.3	396.90	4.98
1	17.8	396.90	9.14
2	17.8	392.83	4.03
3	18.7	394.63	2.94
4	18.7	396.90	5.33

In [6]:
```
class Bootstrap_random_forest:
    def __init__(self,x,y,n = 30):
        self.n = n
        self.x = x
        self.y = y
        self.X_n_sample = {}
        self.Y_n_sample = {}
        self.column_sample_index = {}
        self.row_sample_index = {}
        self.y_pred = []
        self.y_pred_oob = []

    def create_n_samples(self):
        data_size_60 = (int)(0.6*self.x.shape[0])
        data_size_40 = self.x.shape[0] - data_size_60

        for i in range(self.n):
#       column sampling
            idx_col = random.sample(range(data.shape[1]),random.randrange(3, data.shape
            self.column_sample_index[i] = idx_col
```

```
#       row sampling
            idx = random.sample(range(self.x.shape[0]),data_size_60)
            idx2 = random.sample(idx,data_size_40)
            idx_row = idx + idx2
            self.row_sample_index[i] = idx_row
#             print('For n: ',i,' Col value: ',len(idx_col),' Row value: ',len(idx_row
            sample_x = self.x.iloc[idx_row,idx_col].values
            sample_y = self.y[idx_row]

            self.X_n_sample[i] = sample_x
            self.Y_n_sample[i] = sample_y

    def train_model(self):
        y_pred_total = np.zeros(506)
        regressor = DecisionTreeRegressor(random_state=0)
        for i in range(self.n):
            regressor.fit(self.X_n_sample[i],self.Y_n_sample[i])
            y_pred_sample = regressor.predict(self.x.iloc[:,self.column_sample_index[i]
#             print(self.x.iloc[:,self.column_sample_index[i]].shape[1])
            y_pred_total = np.add(y_pred_sample,y_pred_total)
        self.y_pred = (1/30)*y_pred_total

    def train_model_oob(self):
        for i in range(self.x.shape[0]):
            y_pred_sample = 0
            k = 0
            regressor = DecisionTreeRegressor(random_state=0)
            for j in range(self.n):
                if i  not in self.row_sample_index[j]:
                    k+=1
                    regressor.fit(self.X_n_sample[j],self.Y_n_sample[j])
#                     print(self.column_sample_data[j].shape)
                    y_pred_sample += regressor.predict(self.x.iloc[:,self.column_sample
            #         print(y_pred_sample)
            self.y_pred_oob.append((1/k)*y_pred_sample)

    def predict_sample(self,data):
        y_pred_total = np.zeros(len(data))
        regressor = DecisionTreeRegressor(random_state=0)
        for i in range(self.n):
            regressor.fit(self.X_n_sample[i],self.Y_n_sample[i])
            y_pred_sample = regressor.predict(data[:,self.column_sample_index[i]])
        #     print(y_pred_sample)
            y_pred_total = np.add(y_pred_sample,y_pred_total)
        y_pred = (1/30)*y_pred_total
        return y_pred

    def mean_square_error(self,y_orig):
```

```
            return np.mean(np.subtract(y_orig,self.y_pred))

        def mean_square_error_oob(self,y_orig):
            return np.mean(np.subtract(y_orig,self.y_pred_oob))
```

```
In [7]: model = Bootstrap_random_forest(data,y,30)
```

1.1.1 Step 1: Creating samples: Randomly create 30 samples from the whole boston data points.ű

```
In [8]: model.create_n_samples()
```

Step 2 Building High Variance Models on each of the sample and finding train MSE value'

```
In [9]: model.train_model()
```

```
In [10]: model.mean_square_error(y)
```

```
Out[10]: 0.05959952196464046
```

1.1.2 Step 3 Calculating the OOB score :

```
In [11]: model.train_model_oob()
```

```
In [12]: model.mean_square_error_oob(y)
```

```
Out[12]: 0.05989914777846328
```

1.2 Task: 2

```
In [13]: mse = []
         oob_score = []
         for i in tqdm(range(35)):
             model = Bootstrap_random_forest(data,y,30)
             model.create_n_samples()
             model.train_model()
             mse.append(model.mean_square_error(y))
             model.train_model_oob()
             oob_score.append(model.mean_square_error_oob(y))
```

```
100%|| 35/35 [16:55<00:00, 28.26s/it]
```

```
In [14]: print('MSE Mean: ',np.array(mse).mean(),'MSE Std: ',np.array(mse).std())
         print('OOB Score Mean: ',np.array(oob_score).mean(),'OOB Score Std: ',np.array(oob_sc
```

```
MSE Mean:  -0.011462985496974637 MSE Std:  0.030518836528231154
OOB Score Mean:  -0.011380826166990552 OOB Score Std:  0.030442073883171238
```

```
In [15]: print('Confidence Interval of MSE: [',np.array(mse).mean()-2*np.array(mse).std(),',',
         print('Confidence Interval of OOB Score: [',np.array(oob_score).mean()-2*np.array(oob
```

```
Confidence Interval of MSE: [ -0.07250065855343694 , 0.04957468755948767 ]
Confidence Interval of OOB Score: [ -0.07226497393333303 , 0.049503321599351925 ]
```

1.3 Task: 3

```
In [16]: xq = np.array([[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]])
         model.predict_sample(xq)
```

```
Out[16]: array([21.23611111])
```