Bootstrap_Random_Forest_instructions

February 15, 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)
```

```
In [5]: 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

                    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(y_pred_sample)
                    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_sampl
                    #         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 (1/506)*np.sum(np.subtract(y_orig,self.y_pred) )

            def mean_square_error_oob(self,y_orig):
                return (1/506)*np.sum(np.subtract(y_orig,self.y_pred_oob) )
```

```
In [6]: 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 [7]: model.create_n_samples()
```

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

```
In [8]: model.train_model()
In [9]: model.mean_square_error(y)

Out[9]: -0.08279268706968855
```

1.1.2 Step 3 Calculating the OOB score :

```
In [10]: model.train_model_oob()

In [11]: model.mean_square_error_oob(y)

Out[11]: -41.21965559590377
```

1.2 Task: 2

```
In [12]: 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 [16:02<00:00, 27.60s/it]


In [13]: 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:  0.0005637626761705887 MSE Std:  0.03298793171491998
OOB Score Mean:  0.2963369381016723 OOB Score Std:  16.728015839311727


In [14]: print('Confidence Interval of MSE: [',np.array(mse).mean()-2*np.array(mse).std(),',',n
         print('Confidence Interval of OOB Score: [',np.array(oob_score).mean()-2*np.array(oob_

Confidence Interval of MSE: [ -0.06541210075366938 , 0.06653962610601055 ]
Confidence Interval of OOB Score: [ -33.15969474052178 , 33.75236861672513 ]
```

1.3 Task: 3

```
In [15]: 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[15]: array([20.12333333])
```