CAP 4631C. Spring 2023

Assignment 5

Introduction

- This assignment is about Random Forest Regression.
- This assignment uses the Hitters data from assignment 4. Therefore, one more time, the outcome variable to **predict is the log of Salary**, where Salary is the player's salary in the 1987 season. Notice that the column called "Salary" in this dataset actually records the log of the Salary.
- Use the train_test_split() method and use 80% of the players to form the training dataset.
- Use all the predictors when constructing the forests with the exception of "League", "Division", and "New League". (this bullet point was added on Thursday 02-23-23)

Question 1: Apply Random Forest using approach 1 (the theory-based approach) and version 2 (where you need to select the best values for both the number of features and the number of trees)

1 a) Report the mean squared error of the chosen forest evaluated on the test dataset you created earlier (i.e., the 20% players that you left out earlier to serve as a test dataset).

Note: Do not forget that this question asks you to apply the second version of approach 1 that we practiced in class (i.e., you need to tune both the number of features and the number of trees)

In [21]:
```
#libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
from abess import LinearRegression
```

In [22]:
```
#load csv
hitters_df= pd.read_csv('Hitters_ML_HW4.csv')
```

In [23]:
```
#quick eda
hitters_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Hits       263 non-null    int64  
 1   HmRun      263 non-null    int64  
 2   Runs       263 non-null    int64  
 3   RBI        263 non-null    int64  
 4   Years      263 non-null    int64  
 5   CAtBat     263 non-null    int64  
 6   CHits      263 non-null    int64  
 7   CHmRun     263 non-null    int64  
 8   CRuns      263 non-null    int64  
 9   CRBI       263 non-null    int64  
 10  League     263 non-null    object 
 11  Division   263 non-null    object 
 12  NewLeague  263 non-null    object 
 13  Salary     263 non-null    float64
dtypes: float64(1), int64(10), object(3)
memory usage: 28.9+ KB
```

In [24]:
```
#quick eda cont.
hitters_df.head()
```

Out[24]:

	Hits	HmRun	Runs	RBI	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	League	Division	Nev
0	81	7	24	38	14	3449	835	69	321	414	N	W	
1	130	18	66	72	3	1624	457	63	224	266	A	W	
2	141	20	65	78	11	5628	1575	225	828	838	N	E	
3	87	10	39	42	2	396	101	12	48	46	N	E	
4	169	4	74	51	11	4408	1133	19	501	336	A	W	

In [25]:
```
#train/test split creation
x_train, x_test, y_train, y_test= train_test_split(
    hitters_df.iloc[:,:-4],hitters_df['Salary'], test_size=0.2, random_state=1)
```

Approach 1

(Theory Based)

In [26]:
```
#consider 50 to 500 trees in steps of 10
number_of_trees=np.arange(50,501,10)
```

In [27]:
```
#print feature count and square root of feature count
print(x_train.shape[1])
print(np.sqrt(x_train.shape[1]))
```

```
10
3.1622776601683795
```

In [28]:
```
#shape of trees with X features
number_of_features=np.arange(3,6)
mse_scores_rf_oob_matrix= np.empty((number_of_features.size, number_of_trees.size))
mse_scores_rf_oob_matrix.shape
```

Out[28]:
```
(3, 46)
```

In [29]:
```
#finding best MSE for combination of trees and features
r=0
for i in number_of_features:
    c=0
    for j in number_of_trees:
        rf_loop= RandomForestRegressor(n_estimators = j, oob_score= True, max_featu
        rf_loop.fit(x_train, y_train)
        mse_scores_rf_oob_matrix[r,c]= mean_squared_error(y_train, rf_loop.oob_pred
        c=c+1
    r= r+1
```

In [30]:
```
#print number of features
number_of_features[np.where(mse_scores_rf_oob_matrix == np.min(mse_scores_rf_oob_ma
```

Out[30]:
```
array([3])
```

In [31]:
```
#print number of trees
number_of_trees[np.where(mse_scores_rf_oob_matrix == np.min(mse_scores_rf_oob_matri
```

Out[31]:
```
array([460])
```

In [32]:
```
#Create random forest with number of features and trees
rf= RandomForestRegressor(n_estimators= 460, max_features=3, random_state=1)
```

In [33]:
```
#fit training set to random forest
rf.fit(x_train, y_train)
```

Out[33]:
```
RandomForestRegressor(max_features=3, n_estimators=460, random_state=1)
```

In [34]:
```
#MSE
mean_squared_error( y_test,rf.predict (x_test))
```

Out[34]:
```
0.15953913430856442
```

In [35]:
```
#RMSE
mean_squared_error( y_test,rf.predict (x_test), squared=False)
```

Out[35]:
```
0.39942350244892255
```

Here we have our MSE and RMSE (my personal favorite) using the theory based approach for non-random forests. They are .159 and .399 respectively. From a purely objective approach these seem to be very good values. I'd personnaly like to compare the RMSE to the average of the log of salary.

Question 2: Apply Random Forest using approach 2 (the practice-based approach)

2 a) Report the mean squared error of the chosen forest evaluated on the test dataset you created earlier (i.e., the 20% players that you left out earlier to serve as a test dataset).

In [36]:
```
#create parameters for gridsearch
param_grid_rf = {  'n_estimators': np.arange(50,501,20), 'max_features': np.arange(
```

In [37]:
```
#create grid with params
gridSearch_rf = GridSearchCV(RandomForestRegressor(), param_grid_rf, cv=5,scoring='
```

In [38]:
```
#fit training data to grid
gridSearch_rf.fit(x_train, y_train)
```

Out[38]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'max_features': array([3, 4, 5]),
                         'n_estimators': array([ 50,  70,  90, 110, 130, 150, 170,
190, 210, 230, 250, 270, 290,
       310, 330, 350, 370, 390, 410, 430, 450, 470, 490])},
             scoring='neg_mean_squared_error')
```

In [39]:
```
#find best params (random = 0)
print('Parameters: ', gridSearch_rf.best_params_)
```

```
Parameters:  {'max_features': 3, 'n_estimators': 270}
```

In [40]:
```
#define forest with best params
rf2= RandomForestRegressor(n_estimators= 270 , max_features=3, random_state=1)
```

In [41]:
```
#train forest
rf2.fit(x_train, y_train)
```

Out[41]:
```
RandomForestRegressor(max_features=3, n_estimators=50, random_state=1)
```

In [42]:
```
#MSE
mean_squared_error( y_test,rf2.predict (x_test))
```

Out[42]:
```
0.18296560494884415
```

In [43]:
```
#RMSE
mean_squared_error( y_test,rf2.predict (x_test), squared=False)
```

Out[43]:
```
0.4277447895051957
```

Here we have our MSE and RMSE using the practice based approach for non-random forests. They are .183 and .427 respectively. We can see that using the theory based approach is lending better results (MSE = .159).

2 b) Report the three most important predictors that form the trees part of this forest. Show how you selected the three most important predictors.

In [53]:
```
#feature importance with sorting
feature_imp = pd.Series(rf2.feature_importances_, hitters_df.iloc[:,:-4].columns [r
feature_imp.sort_values(ascending=False)
```

Out[53]:
```
CAtBat     0.213392
CRuns      0.199416
CHits      0.182947
CRBI       0.107660
CHmRun     0.066294
RBI        0.058874
Hits       0.057284
Years      0.046167
HmRun      0.034094
Runs       0.033872
dtype: float64
```

Our three most impotant features are CAtBat, CRuns, and CHits for predicting log of salary using our second non-random forest (theory based).

Question 3: Select the best Forest from those you got in questions 1 and 2. **Justify your selection**. Then, compare the chosen forest to the best single tree that you got in assignment 4. State which one is best (the best forest or the best single tree?). **Justify your answer**.

Note: You do not have to repeat the steps you did in assignment 4 to get the best single tree. Just mention what your best single tree was from assignment 4 and why. Then, compare it to the best forest.

Note: Do not forget that this question asks you to justify on two occasions.

Comparing our MSE we clearly see that our first random forest ourperformed our second. Our first MSE was .159 and our second MSE was .183. This clearly indicates that of the two possible models we should use the theory based approach for non-random forests.

In the previous assignment our best performing model was a post pruned tree (MSE equaling .179). When comparing our first non-random forest to our previous model from assignment 4, we find that our non-random forest utlizing a theory based approach performed much better than any of our existing models (MSE equaling .159 for our first non-random forest). That means that for our specific dataset, prediciting the log of salary is best done by using a non-random forest that utlilizes a theory based approach.