3.RF-Grid-Without Training and Testing

October 13, 2022

[1]:
```
#importing the Libraies
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[2]:
```
# Reading the Dataset
dataset = pd.read_csv('insurance_pre.csv')
```

[3]:
```
dataset
```

[3]:

	age	sex	bmi	children	smoker	charges
0	19	female	27.900	0	yes	16884.92400
1	18	male	33.770	1	no	1725.55230
2	28	male	33.000	3	no	4449.46200
3	33	male	22.705	0	no	21984.47061
4	32	male	28.880	0	no	3866.85520
…	…	…	…	…	…	…
1333	50	male	30.970	3	no	10600.54830
1334	18	female	31.920	0	no	2205.98080
1335	18	female	36.850	0	no	1629.83350
1336	21	female	25.800	0	no	2007.94500
1337	61	female	29.070	0	yes	29141.36030

[1338 rows x 6 columns]

[4]:
```
dataset
```

[4]:

	age	sex	bmi	children	smoker	charges
0	19	female	27.900	0	yes	16884.92400
1	18	male	33.770	1	no	1725.55230
2	28	male	33.000	3	no	4449.46200
3	33	male	22.705	0	no	21984.47061
4	32	male	28.880	0	no	3866.85520
…	…	…	…	…	…	…
1333	50	male	30.970	3	no	10600.54830
1334	18	female	31.920	0	no	2205.98080
1335	18	female	36.850	0	no	1629.83350

```
1336   21  female  25.800         0     no   2007.94500
1337   61  female  29.070         0    yes  29141.36030

[1338 rows x 6 columns]
```

[5]:
```
dataset=pd.get_dummies(dataset,drop_first=True)
```

[6]:
```
dataset
```

[6]:
```
      age     bmi  children      charges  sex_male  smoker_yes
0      19  27.900         0  16884.92400         0           1
1      18  33.770         1   1725.55230         1           0
2      28  33.000         3   4449.46200         1           0
3      33  22.705         0  21984.47061         1           0
4      32  28.880         0   3866.85520         1           0
…     …      …        …           …        …          …
1333   50  30.970         3  10600.54830         1           0
1334   18  31.920         0   2205.98080         0           0
1335   18  36.850         0   1629.83350         0           0
1336   21  25.800         0   2007.94500         0           0
1337   61  29.070         0  29141.36030         0           1

[1338 rows x 6 columns]
```

[7]:
```
indep=dataset[['age', 'bmi', 'children','sex_male', 'smoker_yes']]
dep=dataset['charges']
```

[8]:
```

```

[ ]:
```

```

[9]:
```
from sklearn.ensemble import RandomForestRegressor


```

[10]:
```
from sklearn.model_selection import GridSearchCV
#from sklearn.tree import DecisionTreeRegressor
param_grid = {'criterion':['mse','mae'],
              'max_features': ['auto','sqrt','log2'],
              'n_estimators':[10,100]}



grid = GridSearchCV(RandomForestRegressor(), param_grid, refit = True, verbose␣
 ↪= 3,n_jobs=-1)

# fitting the model for grid search
grid.fit(indep, dep)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits

C:\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:391: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use
`criterion='squared_error'` which is equivalent.
  FutureWarning,
```

[10]:
```
GridSearchCV(estimator=RandomForestRegressor(), n_jobs=-1,
             param_grid={'criterion': ['mse', 'mae'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [10, 100]},
             verbose=3)
```

[11]:
```
# print best parameter after tuning
#print(grid.best_params_)
re=grid.cv_results_

print("The R_score value for best parameter {}:".format(grid.best_params_))
```

```
The R_score value for best parameter {'criterion': 'mse', 'max_features':
'log2', 'n_estimators': 100}:
```

[12]:
```
table=pd.DataFrame.from_dict(re)
```

[13]:
```
table
```

[13]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	\
0	0.284063	0.062524	0.030955	0.005931	
1	2.156609	0.359442	0.090185	0.018605	
2	0.205565	0.036899	0.025583	0.007600	
3	1.545019	0.343842	0.097193	0.031486	
4	0.283845	0.201678	0.018975	0.001118	
5	1.475561	0.152391	0.079584	0.006714	
6	1.310922	0.170651	0.020398	0.001354	
7	9.778991	0.565477	0.070934	0.006059	
8	0.722489	0.066411	0.019980	0.001973	
9	5.684262	0.741965	0.077712	0.012003	
10	0.811606	0.066913	0.034295	0.009108	
11	5.470432	0.400919	0.068839	0.005975	

	param_criterion	param_max_features	param_n_estimators	\
0	mse	auto	10	
1	mse	auto	100	
2	mse	sqrt	10	
3	mse	sqrt	100	
4	mse	log2	10	

```
5              mse               log2              100
6              mae               auto               10
7              mae               auto              100
8              mae               sqrt               10
9              mae               sqrt              100
10             mae               log2               10
11             mae               log2              100

                                                 params  split0_test_score  \
0   {'criterion': 'mse', 'max_features': 'auto', '…           0.839061
1   {'criterion': 'mse', 'max_features': 'auto', '…           0.849500
2   {'criterion': 'mse', 'max_features': 'sqrt', '…           0.833721
3   {'criterion': 'mse', 'max_features': 'sqrt', '…           0.860212
4   {'criterion': 'mse', 'max_features': 'log2', '…           0.846953
5   {'criterion': 'mse', 'max_features': 'log2', '…           0.859825
6   {'criterion': 'mae', 'max_features': 'auto', '…           0.823591
7   {'criterion': 'mae', 'max_features': 'auto', '…           0.836650
8   {'criterion': 'mae', 'max_features': 'sqrt', '…           0.841550
9   {'criterion': 'mae', 'max_features': 'sqrt', '…           0.857936
10  {'criterion': 'mae', 'max_features': 'log2', '…           0.830176
11  {'criterion': 'mae', 'max_features': 'log2', '…           0.859750

    split1_test_score  split2_test_score  split3_test_score  \
0            0.765054           0.856413           0.818758
1            0.772910           0.861810           0.820069
2            0.763187           0.836738           0.806137
3            0.782689           0.859529           0.828218
4            0.741272           0.843339           0.805839
5            0.783732           0.856990           0.827978
6            0.732555           0.820898           0.820429
7            0.766024           0.853419           0.824384
8            0.755256           0.846631           0.821434
9            0.778566           0.851517           0.828466
10           0.768303           0.862483           0.802276
11           0.779176           0.855837           0.830336

    split4_test_score  mean_test_score  std_test_score  rank_test_score
0            0.826940         0.821245        0.030823                7
1            0.837232         0.828304        0.030945                5
2            0.827917         0.813540        0.027358               10
3            0.838493         0.833828        0.028373                2
4            0.819372         0.811355        0.038201               11
5            0.842367         0.834178        0.027671                1
6            0.820418         0.803578        0.035531               12
7            0.839276         0.823951        0.030398                6
8            0.825758         0.818126        0.032812                8
9            0.844862         0.832269        0.028588                4
```

10	0.825689	0.817785	0.031314	9
11	0.843122	0.833644	0.029126	3

[ ]:
```
age_input=float(input("Age:"))
bmi_input=float(input("BMI:"))
children_input=float(input("Children:"))
sex_male_input=int(input("Sex Male 0 or 1:"))
smoker_yes_input=int(input("Smoker Yes 0 or 1:"))
```

[ ]:
```
Future_Prediction=grid.
 ↪predict([[age_input,bmi_input,children_input,sex_male_input,smoker_yes_input]])#
 ↪change the paramter,play with it.
print("Future_Prediction={}".format(Future_Prediction))
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: