3.RF-Grid

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')
```

[ ]:
```
dataset
```

[ ]:
```
dataset
```

[3]:
```
dataset=pd.get_dummies(dataset,drop_first=True)
```

[4]:
```
dataset
```

[4]:

	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]

[5]:
```
indep=dataset[['age', 'bmi', 'children','sex_male', 'smoker_yes']]
dep=dataset['charges']
```

[6]:
```
#split into training set and test
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(indep, dep, test_size = 1/
↪3, random_state = 0)
```

[7]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[8]:
```
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(X_train, y_train)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    4.9s
[Parallel(n_jobs=-1)]: Done  26 out of  36 | elapsed:    5.5s remaining:    2.1s
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed:    7.0s finished
```

[10]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
             estimator=RandomForestRegressor(bootstrap=True, criterion='mse',
                                             max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators='warn', n_jobs=None,
                                             oob_score=False, random_state=None,
                                             verbose=0, warm_start=False),
             iid='warn', n_jobs=-1,
```

```
                param_grid={'criterion': ['mse', 'mae'],
                            'max_features': ['auto', 'sqrt', 'log2'],
                            'n_estimators': [10, 100]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                scoring=None, verbose=3)
```

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

# print classification report
from sklearn.metrics import r2_score
r_score=r2_score(y_test,grid_predictions)

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

```
The R_score value for best parameter {'criterion': 'mae', 'max_features':
'sqrt', 'n_estimators': 100}: 0.87438313765281
```

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

[13]:
```
table
```

[13]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	\
0	0.064955	0.004217	0.004987	1.946680e-07	
1	0.504170	0.003438	0.025830	1.421202e-03	
2	0.049131	0.005116	0.005000	8.146892e-04	
3	0.443521	0.018677	0.022813	1.652083e-03	
4	0.048582	0.002549	0.005331	9.494264e-04	
5	0.437335	0.019040	0.022984	1.642447e-03	
6	0.221658	0.013660	0.004001	1.596079e-05	
7	1.746727	0.027610	0.024313	4.391143e-04	
8	0.124352	0.011595	0.003989	2.247832e-07	
9	1.127960	0.011441	0.022014	8.171223e-04	
10	0.120176	0.005492	0.004501	4.195174e-04	
11	1.134786	0.008669	0.022513	1.459310e-03	

	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', '…           0.772643
1   {'criterion': 'mse', 'max_features': 'auto', '…           0.790400
2   {'criterion': 'mse', 'max_features': 'sqrt', '…           0.776431
3   {'criterion': 'mse', 'max_features': 'sqrt', '…           0.806334
4   {'criterion': 'mse', 'max_features': 'log2', '…           0.785895
5   {'criterion': 'mse', 'max_features': 'log2', '…           0.805157
6   {'criterion': 'mae', 'max_features': 'auto', '…           0.736597
7   {'criterion': 'mae', 'max_features': 'auto', '…           0.791955
8   {'criterion': 'mae', 'max_features': 'sqrt', '…           0.787623
9   {'criterion': 'mae', 'max_features': 'sqrt', '…           0.808999
10  {'criterion': 'mae', 'max_features': 'log2', '…           0.765594
11  {'criterion': 'mae', 'max_features': 'log2', '…           0.800216

    split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0            0.765234           0.790219         0.776028        0.010473
1            0.795172           0.798572         0.794710        0.003353
2            0.808969           0.761584         0.782321        0.019779
3            0.803550           0.791685         0.800529        0.006352
4            0.786061           0.780230         0.784064        0.002710
5            0.800521           0.794355         0.800017        0.004425
6            0.762197           0.771544         0.756757        0.014780
7            0.789747           0.786850         0.789520        0.002091
8            0.787097           0.774438         0.783058        0.006094
9            0.802397           0.796112         0.802510        0.005263
10           0.767119           0.765989         0.766233        0.000646
11           0.807552           0.791503         0.799758        0.006556

    rank_test_score
0                10
1                 5
2                 9
3                 2
4                 7
5                 3
6                12
7                 6
8                 8
9                 1
10               11
```

11 4

[ ]:
```
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))
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: