2.DR-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')
```

[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]:
```
#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)
```

[9]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[ ]:
```

```

[10]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
param_grid = {'criterion':['mse','mae','friedman_mse'],
              'max_features': ['auto','sqrt','log2'],
              'splitter':['best','random']}
```

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

# fitting the model for grid search
grid.fit(X_train, y_train)
```

```
C:\Anaconda3\envs\ML\lib\site-packages\sklearn\model_selection\_split.py:1978:
FutureWarning: The default value of cv will change from 3 to 5 in version 0.22.
Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
```

```
Fitting 3 folds for each of 18 candidates, totalling 54 fits
```

```
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    4.5s
[Parallel(n_jobs=-1)]: Done  50 out of  54 | elapsed:    4.8s remaining:    0.3s
[Parallel(n_jobs=-1)]: Done  54 out of  54 | elapsed:    4.8s finished
C:\Anaconda3\envs\ML\lib\site-packages\sklearn\model_selection\_search.py:814:
DeprecationWarning: The default of the `iid` parameter will change from True to
False in version 0.22 and will be removed in 0.24. This will change numeric
results when test-set sizes are unequal.
  DeprecationWarning)
```

```
[10]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                   estimator=DecisionTreeRegressor(criterion='mse', max_depth=None,
                                                   max_features=None,
                                                   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,
                                                   presort=False, random_state=None,
                                                   splitter='best'),
                   iid='warn', n_jobs=-1,
                   param_grid={'criterion': ['mse', 'mae', 'friedman_mse'],
                               'max_features': ['auto', 'sqrt', 'log2'],
                               'splitter': ['best', 'random']},
                   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)
```

```
{'criterion': 'mae', 'max_features': 'auto', 'splitter': 'random'}
The R_score value for best parameter {'criterion': 'mae', 'max_features':
'auto', 'splitter': 'random'}: 0.7150762213240649
```

[13]:

```
table=pd.DataFrame.from_dict(re)
```

[14]:

```
table
```

[14]:

```
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        0.005985  1.409110e-03         0.002601        0.000353
1        0.004910  1.115174e-03         0.001689        0.000697
2        0.003957  4.484425e-05         0.001995        0.000000
3        0.004290  4.253465e-04         0.002125        0.000184
4        0.003149  2.514277e-04         0.001849        0.000643
5        0.002993  5.947204e-07         0.001995        0.000002
6        0.032343  1.138618e-03         0.001995        0.000001
7        0.020599  2.330010e-03         0.002158        0.000823
8        0.014959  8.157561e-04         0.001798        0.000639
9        0.011982  1.267615e-03         0.002923        0.001311
10       0.017405  2.710370e-03         0.001922        0.000101
11       0.016491  2.369476e-03         0.001588        0.000493
12       0.006316  9.395378e-04         0.002508        0.000408
13       0.004315  9.361913e-04         0.002187        0.000269
14       0.004615  1.638069e-03         0.002187        0.000266
15       0.003112  1.718468e-04         0.001781        0.000300
16       0.004992  2.157333e-03         0.002986        0.001414
17       0.003323  4.677745e-04         0.001542        0.000414

   param_criterion param_max_features param_splitter  \
0              mse               auto           best
1              mse               auto         random
2              mse               sqrt           best
3              mse               sqrt         random
4              mse               log2           best
5              mse               log2         random
6              mae               auto           best
7              mae               auto         random
8              mae               sqrt           best
9              mae               sqrt         random
10             mae               log2           best
```

```
11           mae              log2        random
12  friedman_mse              auto          best
13  friedman_mse              auto        random
14  friedman_mse              sqrt          best
15  friedman_mse              sqrt        random
16  friedman_mse              log2          best
17  friedman_mse              log2        random

                                               params  split0_test_score  \
0   {'criterion': 'mse', 'max_features': 'auto', '…           0.538684
1   {'criterion': 'mse', 'max_features': 'auto', '…           0.594686
2   {'criterion': 'mse', 'max_features': 'sqrt', '…           0.679781
3   {'criterion': 'mse', 'max_features': 'sqrt', '…           0.691520
4   {'criterion': 'mse', 'max_features': 'log2', '…           0.529707
5   {'criterion': 'mse', 'max_features': 'log2', '…           0.554397
6   {'criterion': 'mae', 'max_features': 'auto', '…           0.647916
7   {'criterion': 'mae', 'max_features': 'auto', '…           0.616340
8   {'criterion': 'mae', 'max_features': 'sqrt', '…           0.435070
9   {'criterion': 'mae', 'max_features': 'sqrt', '…           0.725520
10  {'criterion': 'mae', 'max_features': 'log2', '…           0.567610
11  {'criterion': 'mae', 'max_features': 'log2', '…           0.532352
12  {'criterion': 'friedman_mse', 'max_features': …           0.509355
13  {'criterion': 'friedman_mse', 'max_features': …           0.580917
14  {'criterion': 'friedman_mse', 'max_features': …           0.523019
15  {'criterion': 'friedman_mse', 'max_features': …           0.598454
16  {'criterion': 'friedman_mse', 'max_features': …           0.659411
17  {'criterion': 'friedman_mse', 'max_features': …           0.565314

    split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0            0.677963           0.694056         0.636791        0.069798
1            0.657129           0.675125         0.642260        0.034487
2            0.680816           0.641157         0.667265        0.018451
3            0.644275           0.591872         0.642610        0.040710
4            0.540047           0.683060         0.584210        0.069966
5            0.595953           0.555577         0.568626        0.019313
6            0.652077           0.688474         0.662806        0.018214
7            0.707618           0.695565         0.673111        0.040510
8            0.594457           0.685666         0.571578        0.103603
9            0.359665           0.504119         0.529987        0.150515
10           0.310512           0.495136         0.457876        0.108239
11           0.573472           0.664373         0.590001        0.055162
12           0.671252           0.691500         0.623907        0.081557
13           0.630293           0.631976         0.614358        0.023696
14           0.671363           0.676148         0.623397        0.071124
15           0.535226           0.656763         0.596816        0.049603
16           0.621462           0.720891         0.667246        0.040946
17           0.439232           0.515024         0.506589        0.051831
```

```
    rank_test_score
0                 7
1                 6
2                 2
3                 5
4                13
5                15
6                 4
7                 1
8                14
9                16
10               18
11               12
12                8
13               10
14                9
15               11
16                3
17               17
```

[15]:

```
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:"))
```

```
Age:4
BMI:56
Children:5
Sex Male 0 or 1:5
Smoker Yes 0 or 1:3
```

[16]:

```
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))
```

```
Future_Prediction=[49577.6624]
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: