2.Hyper-parameter Tuning - Exercise

October 28, 2021

Hyperparameter Tuning for Multiple Regression - Exercise

Car Price Prediction We will find

optimal number of features - using recursive feature elimination (RFE)

k-value in cross validation - using grid search (GS)

[10]:
```
# Import necessary package
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings # supress warnings
warnings.filterwarnings('ignore')
```

[11]:
```
# reading the dataset
cars = pd.read_csv("E:\\MY LECTURES\\DATA SCIENCE\\3.
 ↪Programs\\dataset\\CarPrice_Assignment.csv")
cars.head()
```

[11]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	\
0	1	3	alfa-romero giulia	gas	std	two	
1	2	3	alfa-romero stelvio	gas	std	two	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	
3	4	2	audi 100 ls	gas	std	four	
4	5	2	audi 100ls	gas	std	four	

	carbody	drivewheel	enginelocation	wheelbase	…	enginesize	\
0	convertible	rwd	front	88.6	…	130	
1	convertible	rwd	front	88.6	…	130	
2	hatchback	rwd	front	94.5	…	152	
3	sedan	fwd	front	99.8	…	109	
4	sedan	4wd	front	99.4	…	136	

	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	\
0	mpfi	3.47	2.68	9.0	111	5000	21	
1	mpfi	3.47	2.68	9.0	111	5000	21	

```
2        mpfi       2.68    3.47              9.0        154      5000       19
3        mpfi       3.19    3.40             10.0        102      5500       24
4        mpfi       3.19    3.40              8.0        115      5500       18

   highwaympg    price
0          27  13495.0
1          27  16500.0
2          26  16500.0
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

[12]:
```
# All data preparation steps in this cell
import re
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import scale

# select specific features and split into x and y
x = cars[['symboling', 'fueltype', 'aspiration', 'doornumber',
       'carbody', 'drivewheel', 'enginelocation', 'wheelbase', 'carlength',
       'carwidth', 'carheight', 'curbweight', 'enginetype', 'cylindernumber',
       'enginesize', 'fuelsystem', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg']]
y = cars['price']

# creating dummy variables for categorical variables
cars_categorical = x.select_dtypes(include=['object'])
cars_categorical.head()

# convert into dummies
cars_dummies = pd.get_dummies(cars_categorical, drop_first=True)
cars_dummies.head()

# drop categorical variables
x = x.drop(list(cars_categorical.columns), axis=1)

# concat dummy variables with X
x = pd.concat([x, cars_dummies], axis=1)

# rescale the features
cols = x.columns
x = pd.DataFrame(scale(x))
x.columns = cols

# split into train and test
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.7,␣
↪test_size = 0.3, random_state=40)
```

[13]:
```
# number of features
len(x_train.columns)
```

[13]: 43

[14]:
```
# creating a KFold object with 5 splits
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV

folds = KFold(n_splits = 5, shuffle = True, random_state = 100)

# specify range of hyperparameters
hyper_params = [{'n_features_to_select': list(range(2, 40))}]

# specify model
lm = LinearRegression()
lm.fit(x_train, y_train)
rfe = RFE(lm)

# set up GridSearchCV()
model_cv = GridSearchCV(estimator = rfe, param_grid = hyper_params, scoring=␣
↪'r2', cv = folds, verbose = 1, return_train_score=True)

# fit the model
model_cv.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 38 candidates, totalling 190 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 190 out of 190 | elapsed:    5.4s finished
```

[14]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=100, shuffle=True),
             estimator=RFE(estimator=LinearRegression()),
             param_grid=[{'n_features_to_select': [2, 3, 4, 5, 6, 7, 8, 9, 10,
                                                   11, 12, 13, 14, 15, 16, 17,
                                                   18, 19, 20, 21, 22, 23, 24,
                                                   25, 26, 27, 28, 29, 30, 31,
…]}],
             return_train_score=True, scoring='r2', verbose=1)
```

[15]:
```
# cv results
cv_results = pd.DataFrame(model_cv.cv_results_)
cv_results
```

[15]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	\
0	0.038884	0.002353	0.002400	0.001625	
1	0.037378	0.001355	0.002599	0.000490	
2	0.035098	0.003161	0.003000	0.002449	
3	0.036077	0.003595	0.002000	0.001897	
4	0.036179	0.004212	0.002000	0.000633	
5	0.032271	0.001795	0.000800	0.000979	
6	0.032176	0.001985	0.001409	0.001962	
7	0.032614	0.003171	0.003413	0.001977	
8	0.030372	0.003487	0.003298	0.002749	
9	0.031442	0.001785	0.002504	0.001607	
10	0.030684	0.001223	0.000811	0.000993	
11	0.031766	0.001993	0.001105	0.001435	
12	0.030466	0.000827	0.000000	0.000000	
13	0.026909	0.001928	0.004827	0.001495	
14	0.029412	0.000708	0.001016	0.000850	
15	0.027022	0.002844	0.003304	0.001395	
16	0.028031	0.002469	0.002999	0.002449	
17	0.025008	0.000019	0.004000	0.002000	
18	0.025614	0.002980	0.002997	0.002651	
19	0.026192	0.001565	0.001110	0.001439	
20	0.021607	0.001079	0.003988	0.001994	
21	0.023255	0.002730	0.000403	0.000807	
22	0.021467	0.002654	0.002110	0.001911	
23	0.020986	0.001141	0.002598	0.002238	
24	0.022036	0.002451	0.003515	0.001344	
25	0.018758	0.001164	0.000506	0.001012	
26	0.018624	0.001977	0.002015	0.001911	
27	0.019246	0.000993	0.000812	0.000994	
28	0.017088	0.002182	0.003024	0.000822	
29	0.016295	0.002067	0.001114	0.001443	
30	0.016827	0.001108	0.002925	0.000484	
31	0.013617	0.002231	0.000996	0.001992	
32	0.014075	0.002786	0.002105	0.001905	
33	0.010954	0.001125	0.003208	0.002082	
34	0.010040	0.000013	0.002904	0.002053	
35	0.008646	0.002556	0.003101	0.002320	
36	0.008342	0.001823	0.002006	0.001974	
37	0.005643	0.001639	0.001708	0.001829	

	param_n_features_to_select	params	\
0	2	{'n_features_to_select': 2}	
1	3	{'n_features_to_select': 3}	

```
2                          4   {'n_features_to_select': 4}
3                          5   {'n_features_to_select': 5}
4                          6   {'n_features_to_select': 6}
5                          7   {'n_features_to_select': 7}
6                          8   {'n_features_to_select': 8}
7                          9   {'n_features_to_select': 9}
8                         10  {'n_features_to_select': 10}
9                         11  {'n_features_to_select': 11}
10                        12  {'n_features_to_select': 12}
11                        13  {'n_features_to_select': 13}
12                        14  {'n_features_to_select': 14}
13                        15  {'n_features_to_select': 15}
14                        16  {'n_features_to_select': 16}
15                        17  {'n_features_to_select': 17}
16                        18  {'n_features_to_select': 18}
17                        19  {'n_features_to_select': 19}
18                        20  {'n_features_to_select': 20}
19                        21  {'n_features_to_select': 21}
20                        22  {'n_features_to_select': 22}
21                        23  {'n_features_to_select': 23}
22                        24  {'n_features_to_select': 24}
23                        25  {'n_features_to_select': 25}
24                        26  {'n_features_to_select': 26}
25                        27  {'n_features_to_select': 27}
26                        28  {'n_features_to_select': 28}
27                        29  {'n_features_to_select': 29}
28                        30  {'n_features_to_select': 30}
29                        31  {'n_features_to_select': 31}
30                        32  {'n_features_to_select': 32}
31                        33  {'n_features_to_select': 33}
32                        34  {'n_features_to_select': 34}
33                        35  {'n_features_to_select': 35}
34                        36  {'n_features_to_select': 36}
35                        37  {'n_features_to_select': 37}
36                        38  {'n_features_to_select': 38}
37                        39  {'n_features_to_select': 39}

    split0_test_score  split1_test_score  split2_test_score  \
0            0.881340           0.673284           0.882871
1            0.892657           0.720574           0.835474
2            0.884742           0.744381           0.819150
3            0.869541           0.758867           0.853019
4            0.857359           0.759931           0.856447
5            0.848291           0.774516           0.844838
6            0.856797           0.772164           0.865752
7            0.845066           0.775609           0.867770
8            0.864097           0.778355           0.886196
```

```
9            0.864097           0.782456           0.836851
10           0.863401           0.786999           0.832292
11           0.861306           0.787046           0.849250
12           0.859188           0.782215           0.835798
13           0.888793           0.782215           0.856645
14           0.894933           0.781954           0.863213
15           0.895685           0.782008           0.882355
16           0.891110           0.775399           0.891473
17           0.890039           0.797709           0.891554
18           0.892676           0.789162           0.881252
19           0.889544           0.806964           0.881252
20           0.895019           0.791903           0.882540
21           0.893430           0.810184           0.885101
22           0.898803           0.814658           0.878401
23           0.905526           0.819384           0.891492
24           0.905159           0.818307           0.895491
25           0.899597           0.824489           0.891750
26           0.905798           0.813068           0.891493
27           0.908821           0.811574           0.887597
28           0.909721           0.811793           0.891984
29           0.911868           0.812738           0.889322
30           0.909482           0.813952           0.878364
31           0.907590           0.813877           0.891504
32           0.906685           0.814061           0.894935
33           0.906685           0.812555           0.895884
34           0.901623           0.811802           0.899548
35           0.904845           0.811802           0.900348
36           0.903629           0.813608           0.896504
37           0.901190           0.813871           0.896299

    split3_test_score  …  mean_test_score  std_test_score  rank_test_score  \
0            0.643727  …         0.759411        0.102770               27
1            0.675851  …         0.713387        0.156172               37
2            0.695744  …         0.712963        0.159642               38
3            0.711072  …         0.724029        0.159396               36
4            0.692448  …         0.726327        0.144522               35
5            0.645008  …         0.741160        0.104395               33
6            0.685292  …         0.751166        0.109374               29
7            0.685736  …         0.744943        0.116062               32
8            0.589406  …         0.729735        0.144385               34
9            0.633530  …         0.746908        0.102626               31
10           0.666386  …         0.751141        0.098531               30
11           0.679817  …         0.756717        0.098995               28
12           0.678216  …         0.767331        0.075764               26
13           0.678216  …         0.777421        0.086944               25
14           0.678216  …         0.784731        0.084806               24
15           0.666802  …         0.801543        0.082894               22
```

16	0.679983	…	0.803545	0.080031	21
17	0.658587	…	0.810030	0.085017	20
18	0.612406	…	0.800890	0.101325	23
19	0.660354	…	0.812433	0.082432	18
20	0.658878	…	0.810410	0.084667	19
21	0.652735	…	0.813024	0.086573	17
22	0.656825	…	0.816623	0.085354	16
23	0.649162	…	0.818195	0.091241	15
24	0.646119	…	0.819917	0.093165	10
25	0.645521	…	0.820209	0.092015	9
26	0.656254	…	0.821593	0.089193	5
27	0.660428	…	0.821466	0.087567	6
28	0.654976	…	0.821949	0.090515	3
29	0.663242	…	0.823689	0.087471	1
30	0.655194	…	0.819433	0.088336	11
31	0.651789	…	0.820400	0.091009	8
32	0.637433	…	0.818726	0.096859	14
33	0.640553	…	0.819304	0.095876	12
34	0.642025	…	0.819287	0.095068	13
35	0.643934	…	0.822838	0.095694	2
36	0.641766	…	0.821721	0.095641	4
37	0.641875	…	0.821223	0.095135	7

	split0_train_score	split1_train_score	split2_train_score	\
0	0.762090	0.809346	0.780621	
1	0.808676	0.868796	0.833966	
2	0.820593	0.890003	0.843983	
3	0.850299	0.903691	0.857307	
4	0.863832	0.906496	0.865698	
5	0.878932	0.912412	0.872259	
6	0.879152	0.916877	0.877898	
7	0.886022	0.917305	0.882350	
8	0.893817	0.917886	0.909495	
9	0.893817	0.926743	0.913194	
10	0.906552	0.927146	0.918228	
11	0.907780	0.927182	0.919668	
12	0.908590	0.937358	0.920457	
13	0.918726	0.937358	0.925112	
14	0.921396	0.940652	0.929727	
15	0.923773	0.941901	0.932459	
16	0.928168	0.942566	0.933475	
17	0.929927	0.947371	0.933566	
18	0.930159	0.949396	0.936528	
19	0.931624	0.953374	0.936528	
20	0.933098	0.955488	0.936538	
21	0.934073	0.956518	0.939325	
22	0.935051	0.956845	0.939727	

23	0.935376	0.957620	0.940248
24	0.935394	0.957966	0.941517
25	0.935662	0.959086	0.941756
26	0.936865	0.959541	0.942051
27	0.938053	0.959680	0.942981
28	0.938327	0.959698	0.943355
29	0.939258	0.960361	0.943431
30	0.940028	0.961422	0.944076
31	0.940044	0.961430	0.944076
32	0.940802	0.961488	0.944349
33	0.940802	0.961522	0.944495
34	0.941094	0.961559	0.944734
35	0.941756	0.961559	0.944734
36	0.941767	0.961583	0.945020
37	0.941819	0.961585	0.945021

	split3_train_score	split4_train_score	mean_train_score	std_train_score
0	0.798515	0.798908	0.789896	0.016691
1	0.850420	0.845840	0.841540	0.019888
2	0.853084	0.860087	0.853550	0.022581
3	0.862113	0.862891	0.867260	0.018759
4	0.895723	0.866674	0.879685	0.017845
5	0.905489	0.885241	0.890867	0.015481
6	0.908631	0.894029	0.895317	0.015547
7	0.909643	0.899599	0.898984	0.013374
8	0.929191	0.905300	0.911138	0.011902
9	0.935141	0.913816	0.916542	0.014036
10	0.937654	0.914816	0.920879	0.010674
11	0.940952	0.914844	0.922085	0.011357
12	0.941103	0.925372	0.926576	0.011745
13	0.941103	0.925372	0.929534	0.008352
14	0.941103	0.927699	0.932116	0.007665
15	0.943753	0.933120	0.935001	0.007215
16	0.945003	0.933799	0.936602	0.006243
17	0.946269	0.937527	0.938932	0.006884
18	0.947154	0.937902	0.940228	0.007106
19	0.948491	0.939028	0.941809	0.007972
20	0.949093	0.939330	0.942709	0.008318
21	0.950972	0.939502	0.944078	0.008317
22	0.952959	0.939969	0.944910	0.008434
23	0.953265	0.941115	0.945525	0.008444
24	0.953502	0.942953	0.946266	0.008258
25	0.955048	0.943298	0.946970	0.008724
26	0.955530	0.943420	0.947481	0.008590
27	0.956403	0.943628	0.948149	0.008368
28	0.956450	0.944904	0.948547	0.008142
29	0.956748	0.944906	0.948941	0.008146

30	0.957132	0.944953	0.949522	0.008248
31	0.957206	0.945059	0.949563	0.008249
32	0.957235	0.945085	0.949792	0.008060
33	0.957254	0.945093	0.949833	0.008053
34	0.957265	0.945110	0.949953	0.007967
35	0.957480	0.945137	0.950133	0.007859
36	0.957495	0.945148	0.950203	0.007826
37	0.957495	0.945149	0.950214	0.007816

[38 rows x 21 columns]

[16]:
```
# plotting cv results
plt.figure(figsize=(16,6))

plt.plot(cv_results["param_n_features_to_select"],␣
 ↪cv_results["mean_test_score"])
plt.plot(cv_results["param_n_features_to_select"],␣
 ↪cv_results["mean_train_score"])
plt.xlabel('number of features')
plt.ylabel('r-squared')
plt.title("Optimal Number of Features")
plt.legend(['test score', 'train score'], loc='upper left')
```

[16]: <matplotlib.legend.Legend at 0x22fc1d0b4c0>