1.SVR-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]

[ ]:
```
dataset
```

[4]:
```
dataset=pd.get_dummies(dataset,drop_first=True)
```

[5]:
```
dataset
```

[5]:

	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]
```

[6]:
```
indep=dataset[['age', 'bmi', 'children','sex_male', 'smoker_yes']]
dep=dataset['charges']
```

[7]:
```
#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)
```

[8]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[ ]:
```

```

[9]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVR
param_grid = {'kernel':['rbf','poly','sigmoid','linear'],
              'C':[10,100,1000,2000,3000],'gamma':['auto','scale']}



grid = GridSearchCV(SVR(), param_grid, refit = True, verbose = 3,n_jobs=-1)

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

```
Fitting 5 folds for each of 40 candidates, totalling 200 fits
```

[9]:
```
GridSearchCV(estimator=SVR(), n_jobs=-1,
             param_grid={'C': [10, 100, 1000, 2000, 3000],
                         'gamma': ['auto', 'scale'],
                         'kernel': ['rbf', 'poly', 'sigmoid', 'linear']},
             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 {'C': 3000, 'gamma': 'scale', 'kernel':
'poly'}:
```

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

[13]:
```
table
```

[13]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	\
0	0.225589	0.046237	0.108398	0.019392	10	
1	0.191590	0.008035	0.036402	0.005314	10	
2	0.277604	0.028574	0.053994	0.010414	10	
3	0.194797	0.017412	0.034620	0.004876	10	
4	0.285293	0.064482	0.127277	0.015214	10	
5	0.245986	0.057148	0.038208	0.007833	10	
6	0.347123	0.084148	0.044281	0.007096	10	
7	0.223376	0.051269	0.032241	0.003642	10	
8	0.254402	0.044610	0.140991	0.021690	100	
9	0.271605	0.072528	0.033775	0.005381	100	
10	0.298807	0.029107	0.043606	0.004195	100	
11	0.265804	0.048751	0.034005	0.005580	100	
12	0.278847	0.043272	0.111162	0.012941	100	
13	0.220319	0.020449	0.030294	0.002021	100	
14	0.276371	0.043382	0.045598	0.004599	100	
15	0.246357	0.020494	0.029416	0.003064	100	
16	0.322198	0.026580	0.117421	0.013363	1000	
17	0.318823	0.060011	0.030774	0.002317	1000	
18	0.342817	0.026006	0.041988	0.005437	1000	
19	0.335825	0.062581	0.027385	0.001353	1000	
20	0.347422	0.049640	0.110973	0.011608	1000	
21	0.327104	0.048083	0.036786	0.007283	1000	
22	0.328180	0.027982	0.043382	0.002231	1000	
23	0.323196	0.026447	0.031988	0.006815	1000	
24	0.390991	0.031962	0.116941	0.006559	2000	
25	0.666942	0.293389	0.035566	0.006084	2000	
26	0.378941	0.050494	0.046677	0.010079	2000	
27	0.445278	0.038421	0.036954	0.008009	2000	
28	0.399297	0.056169	0.118997	0.009284	2000	
29	0.768545	0.663585	0.030829	0.004483	2000	
30	0.359056	0.104075	0.041220	0.011444	2000	
31	0.339239	0.034891	0.033002	0.006061	2000	
32	0.372433	0.041572	0.108621	0.009229	3000	
33	1.244789	0.896527	0.031993	0.004470	3000	
34	0.369150	0.120150	0.039554	0.006811	3000	

35	0.525386	0.110224	0.028983	0.003986	3000
36	0.424375	0.048907	0.102556	0.018475	3000
37	0.702857	0.217121	0.030980	0.004711	3000
38	0.402735	0.088892	0.065773	0.032096	3000
39	0.426197	0.090825	0.029754	0.003584	3000

	param_gamma	param_kernel	\
0	auto	rbf	
1	auto	poly	
2	auto	sigmoid	
3	auto	linear	
4	scale	rbf	
5	scale	poly	
6	scale	sigmoid	
7	scale	linear	
8	auto	rbf	
9	auto	poly	
10	auto	sigmoid	
11	auto	linear	
12	scale	rbf	
13	scale	poly	
14	scale	sigmoid	
15	scale	linear	
16	auto	rbf	
17	auto	poly	
18	auto	sigmoid	
19	auto	linear	
20	scale	rbf	
21	scale	poly	
22	scale	sigmoid	
23	scale	linear	
24	auto	rbf	
25	auto	poly	
26	auto	sigmoid	
27	auto	linear	
28	scale	rbf	
29	scale	poly	
30	scale	sigmoid	
31	scale	linear	
32	auto	rbf	
33	auto	poly	
34	auto	sigmoid	
35	auto	linear	
36	scale	rbf	
37	scale	poly	
38	scale	sigmoid	
39	scale	linear	

```
                                              params  split0_test_score  \
0         {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}          -0.004176
1        {'C': 10, 'gamma': 'auto', 'kernel': 'poly'}           0.047420
2     {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}           0.044787
3      {'C': 10, 'gamma': 'auto', 'kernel': 'linear'}           0.387624
4        {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}          -0.003956
5       {'C': 10, 'gamma': 'scale', 'kernel': 'poly'}           0.043648
6    {'C': 10, 'gamma': 'scale', 'kernel': 'sigmoid'}           0.043946
7     {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}           0.387624
8        {'C': 100, 'gamma': 'auto', 'kernel': 'rbf'}           0.303414
9       {'C': 100, 'gamma': 'auto', 'kernel': 'poly'}           0.542212
10   {'C': 100, 'gamma': 'auto', 'kernel': 'sigmoid'}           0.492088
11    {'C': 100, 'gamma': 'auto', 'kernel': 'linear'}           0.596232
12      {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}           0.304939
13     {'C': 100, 'gamma': 'scale', 'kernel': 'poly'}           0.532310
14  {'C': 100, 'gamma': 'scale', 'kernel': 'sigmoid'}           0.491855
15   {'C': 100, 'gamma': 'scale', 'kernel': 'linear'}           0.596232
16      {'C': 1000, 'gamma': 'auto', 'kernel': 'rbf'}           0.731430
17     {'C': 1000, 'gamma': 'auto', 'kernel': 'poly'}           0.799185
18  {'C': 1000, 'gamma': 'auto', 'kernel': 'sigmoid'}           0.232428
19   {'C': 1000, 'gamma': 'auto', 'kernel': 'linear'}           0.686126
20     {'C': 1000, 'gamma': 'scale', 'kernel': 'rbf'}           0.732079
21    {'C': 1000, 'gamma': 'scale', 'kernel': 'poly'}           0.798212
22  {'C': 1000, 'gamma': 'scale', 'kernel': 'sigmo…           0.248266
23  {'C': 1000, 'gamma': 'scale', 'kernel': 'linear'}           0.686126
24      {'C': 2000, 'gamma': 'auto', 'kernel': 'rbf'}           0.788746
25     {'C': 2000, 'gamma': 'auto', 'kernel': 'poly'}           0.804603
26  {'C': 2000, 'gamma': 'auto', 'kernel': 'sigmoid'}          -0.656854
27   {'C': 2000, 'gamma': 'auto', 'kernel': 'linear'}           0.669958
28     {'C': 2000, 'gamma': 'scale', 'kernel': 'rbf'}           0.789144
29    {'C': 2000, 'gamma': 'scale', 'kernel': 'poly'}           0.804708
30  {'C': 2000, 'gamma': 'scale', 'kernel': 'sigmo…          -0.370980
31  {'C': 2000, 'gamma': 'scale', 'kernel': 'linear'}           0.669958
32      {'C': 3000, 'gamma': 'auto', 'kernel': 'rbf'}           0.795104
33     {'C': 3000, 'gamma': 'auto', 'kernel': 'poly'}           0.804915
34  {'C': 3000, 'gamma': 'auto', 'kernel': 'sigmoid'}          -1.795299
35   {'C': 3000, 'gamma': 'auto', 'kernel': 'linear'}           0.669819
36     {'C': 3000, 'gamma': 'scale', 'kernel': 'rbf'}           0.795353
37    {'C': 3000, 'gamma': 'scale', 'kernel': 'poly'}           0.805169
38  {'C': 3000, 'gamma': 'scale', 'kernel': 'sigmo…          -1.651895
39  {'C': 3000, 'gamma': 'scale', 'kernel': 'linear'}           0.669819

    split1_test_score  split2_test_score  split3_test_score  \
0            0.022594          -0.118956          -0.082926
1            0.077536          -0.060527          -0.009476
2            0.081689          -0.072355          -0.027541
```

3	0.461268	0.288301	0.340540
4	0.022453	-0.119035	-0.082925
5	0.079780	-0.059229	-0.009498
6	0.082230	-0.072132	-0.027546
7	0.461268	0.288301	0.340540
8	0.319385	0.155546	0.208414
9	0.566743	0.471172	0.537557
10	0.545107	0.438714	0.468576
11	0.635776	0.566816	0.588799
12	0.318480	0.155033	0.208422
13	0.571466	0.474948	0.537506
14	0.544604	0.439299	0.468556
15	0.635776	0.566816	0.588799
16	0.716349	0.686476	0.710172
17	0.786769	0.821307	0.810704
18	0.306815	0.236688	0.291707
19	0.598398	0.586104	0.573860
20	0.715992	0.686057	0.710177
21	0.787019	0.823118	0.810712
22	0.377692	0.315321	0.291779
23	0.598398	0.586104	0.573860
24	0.764261	0.818616	0.794131
25	0.784782	0.847674	0.807280
26	-0.432695	-0.346330	-0.484052
27	0.592446	0.586953	0.572862
28	0.764022	0.818319	0.794133
29	0.784713	0.847781	0.807295
30	-0.421011	-0.350248	-0.483750
31	0.592446	0.586953	0.572862
32	0.772917	0.846181	0.811601
33	0.783304	0.852190	0.810160
34	-1.629662	-1.461919	-1.712710
35	0.588893	0.587076	0.572849
36	0.772801	0.846034	0.811601
37	0.783163	0.852345	0.810160
38	-2.010670	-1.468842	-1.712109
39	0.588893	0.587076	0.572849

	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	-0.103473	-0.057387	0.056205	35
1	-0.050823	0.000826	0.054025	32
2	-0.051470	-0.004978	0.058648	34
3	0.297825	0.355112	0.063693	25
4	-0.103510	-0.057395	0.056230	36
5	-0.050317	0.000877	0.053658	31
6	-0.051337	-0.004968	0.058595	33
7	0.297825	0.355112	0.063693	25

8	0.161756	0.229703	0.069348	29
9	0.413719	0.506281	0.056081	22
10	0.425516	0.474000	0.042447	23
11	0.537415	0.585008	0.032600	19
12	0.161488	0.229672	0.069604	30
13	0.415264	0.506299	0.055077	21
14	0.424226	0.473708	0.042461	24
15	0.537415	0.585008	0.032600	19
16	0.613293	0.691544	0.041718	11
17	0.719528	0.787499	0.035885	8
18	0.346868	0.282901	0.043415	28
19	0.556109	0.600119	0.045217	13
20	0.613122	0.691485	0.041876	12
21	0.719979	0.787808	0.036003	7
22	0.346175	0.315846	0.044478	27
23	0.556109	0.600119	0.045217	13
24	0.709676	0.775086	0.036986	9
25	0.745448	0.797957	0.033268	4
26	-0.076409	-0.399268	0.190629	38
27	0.556109	0.595666	0.039218	15
28	0.709586	0.775041	0.036992	10
29	0.745732	0.798046	0.033220	3
30	-0.087414	-0.342681	0.135687	37
31	0.556109	0.595666	0.039218	15
32	0.725697	0.790300	0.040188	5
33	0.748175	0.799749	0.034104	2
34	-0.929359	-1.505790	0.308601	39
35	0.557014	0.595130	0.039068	17
36	0.725615	0.790281	0.040189	6
37	0.748201	0.799808	0.034165	1
38	-0.961821	-1.561068	0.346643	40
39	0.557014	0.595130	0.039068	17

[14]:
```
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:32
BMI:43
Children:2
Sex Male 0 or 1:0
Smoker Yes 0 or 1:1
```

[15]:
```
```

```
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=[3316415.72004342]
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: