FODS-A2 REPORT

Name	ID No.
1. SUPRIYAM AGARWAL	2020B4A7PS2341H
2. KARTIKAY DHALL	2020A7PS2087H
3. RAGHAV KRISHNA	2020B3PS1382H

2A- Building a regression model by finding max linear relationship.

After loading the dataset, we drop the "Appliances" column from the data frame and store it in the variable Y, while the remaining columns are stored in the variable X. We select the first 25 columns of the data frame and add a column of ones to the front, which is then stored in the X variable. The 26th column of the dataframe is then extracted and stored in the y variable as a NumPy array.

The theta variable is then initialized as an array of zeros to be used as the initial values for the model's parameters in the linear regression algorithm.

We implement a function pearson that calculates the Pearson correlation coefficient between two vectors. It then calculates the Pearson correlation coefficient between each column of a list of 26 vectors, var_list, and a label vector, Y. It then sorts the resulting list of correlation coefficients in descending order and prints the resulting list.

Pearson correlation coefficient with Appliances Column

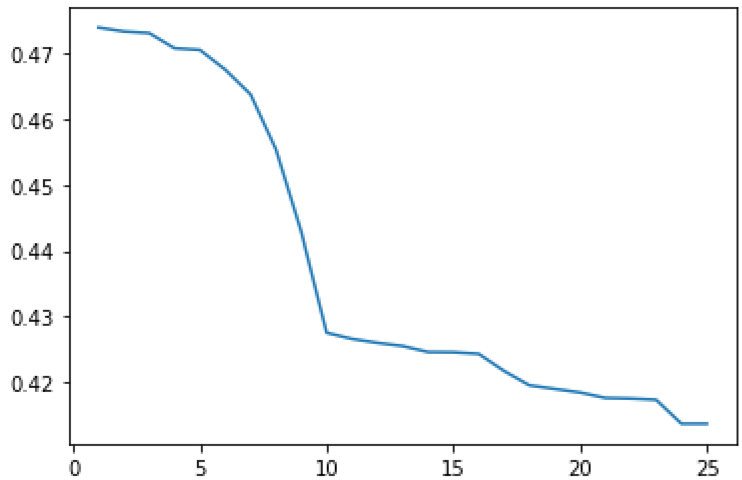
Appliances	1.000000
RH_out	0.157888
T6	0.115799
T2	0.109542
T_out	0.098908
RH_8	0.091040
Windspeed	0.090048
RH_6	0.083081
T3	0.080892
RH_1	0.076796

RH_2	0.066223
RH_7	0.059254
RH_9	0.051292
T1	0.047674
T4	0.035641
T8	0.032599
Press_mm_hg	0.029705
RH_3	0.026723
T7	0.020004
RH_4	0.013502
T5	0.012428
RH_5	0.011210
Tdewpoint	0.009925
Visibility	0.004848
T9	0.004509
rv1	0.000058
rv2	0.000058

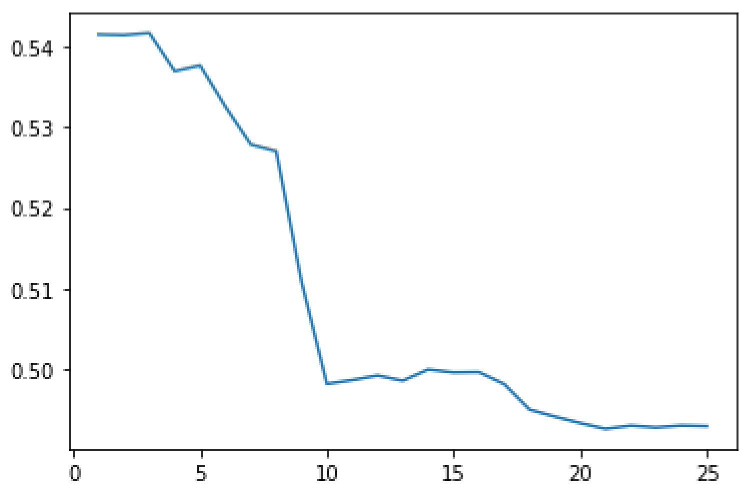
No. of features	Training Error	Testing Error
1	0.4921568413562548	0.5520359362626265
2	0.4920410454077345	0.5518863960110728
3	0.48769953551004896	0.5486912354772934
4	0.4870886520933431	0.5486692542982696
5	0.4865683920662211	0.5460424973543863
6	0.4850319236138159	0.54552161208197
7	0.47808223710328723	0.5383632317919431
8	0.47124404444404716	0.5340475380333828

9	0.46505767168266715	0.5349830658313164
10	0.45375709493471295	0.5181125758079219
11	0.45550548652278844	0.5173472279323555
12	0.44986000354321765	0.5137618055774885
13	0.4496963695253775	0.514309520750087
14	0.45022316657735784	0.5142857708087621
15	0.45163832886960603	0.5164220948027677
16	0.45141614049830486	0.5160875186345181
17	0.4477061681402078	0.5135101368001967
18	0.44044782006302163	0.5072498835297861
19	0.44030161716089833	0.5067868193522788
20	0.44496503293198186	0.5105151952012404
21	0.4433505470922482	0.5092754477315665
22	0.4399134847500075	0.5061823521695902
23	0.44916532481635624	0.513839820893271
24	0.44817223274792706	0.5129475163979652
25	0.4423838116098763	0.5107451635987253
26	0.4422423179729102	0.5105948374622102

No. of features vs. Training errors



No. of features vs. Testing errors



Principal Component Analysis

Principal component analysis (PCA) is a statistical technique that is commonly used for dimensionality reduction. It works by transforming the data into a new set of orthogonal axes, called principal components, which are ordered such that the first principal component has the largest possible variance, the second principal component has the second largest variance, and so on.

In the pca_function function, the PCA class is first instantiated with n_components set to the value of the num_components parameter. The fit method is then called on this object, which fits the PCA model to the data. The transform method is then called on the pca object to transform the data using the fitted model. The transformed data is returned by the function. We also define a list called pca_list and a function called pca_for_all that iterates over a range of values from 1 to 27, calling the pca_function for each value and appending the result to the pca_list list

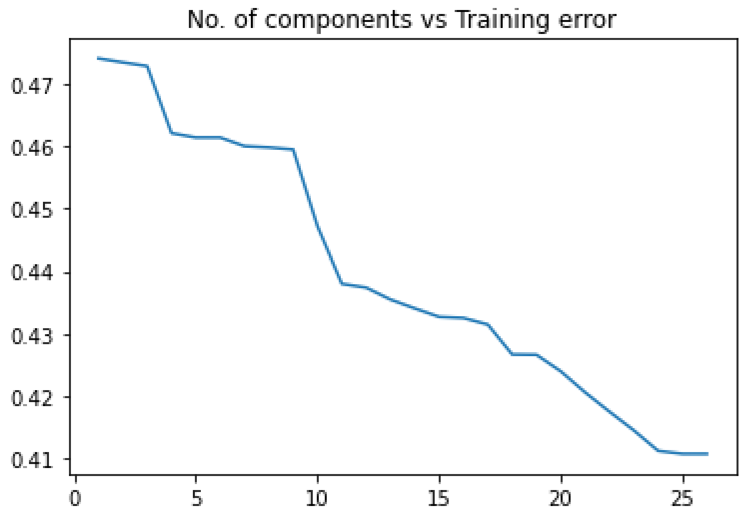
No.of principal components	Training error	Testing error
1	0.4740006626515382	0.5928531127592372
2	0.4733636927971104	0.5928686007895481
3	0.4727770217198106	0.5931331379380866
4	0.46203327484179635	0.5805071307456146
5	0.4613382919990127	0.5803379630335854
6	0.46133559595993534	0.5803710906780852
7	0.45995570140893616	0.5807448783287809
8	0.45975153464699214	0.5810017325981207
9	0.4594219406130878	0.580781606029845
10	0.44713554614229933	0.5636552156710399
11	0.43798722626210324	0.5500285726729904

12	0.4373899567612342	0.5492125495578889
13	0.4354799743595863	0.5437996438292892
14	0.43406141392990016	0.5398656218196638
15	0.4327104131442044	0.5344334233806202
16	0.4325274203958965	0.5335014286107879
17	0.4314795125939306	0.5324061877326598
18	0.42668967631169086	0.527489140393435
19	0.4266409084714607	0.5275038375475165
20	0.4239835715563329	0.5212375201881939
21	0.42061121578837074	0.5154002293358066
22	0.41752473339288076	0.5128099208458522
23	0.4145341734161583	0.5113376316864465
24	0.41125511258169406	0.5041796275737003
25	0.41077474915064266	0.50399979960485
26	0.41075533214126825	0.5040519703718098

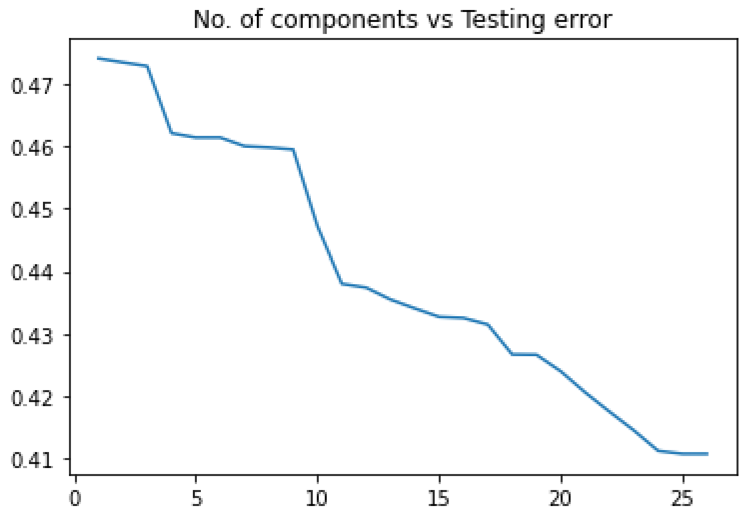
Percentage variance captured by each feature sets-

1. 0.35896089
2. 0.6307988
3. 0.70818005
4. 0.77699652
5. 0.8183059
6. 0.85539066
7. 0.89131133
8. 0.91280068
9. 0.93306365
10. 0.94783511
11. 0.9582695
12. 0.96503675
13. 0.97062846
14. 0.97593166
15. 0.98057846
16. 0.9849769
17. 0.98864056
18. 0.99145579
19. 0.99400824
20. 0.99571697
21. 0.99735433
22. 0.99845068
23. 0.99930645
24. 0.99986255
25. 1
26. 1

No.of principal components vs Training error



No.of principal components vs Testing error

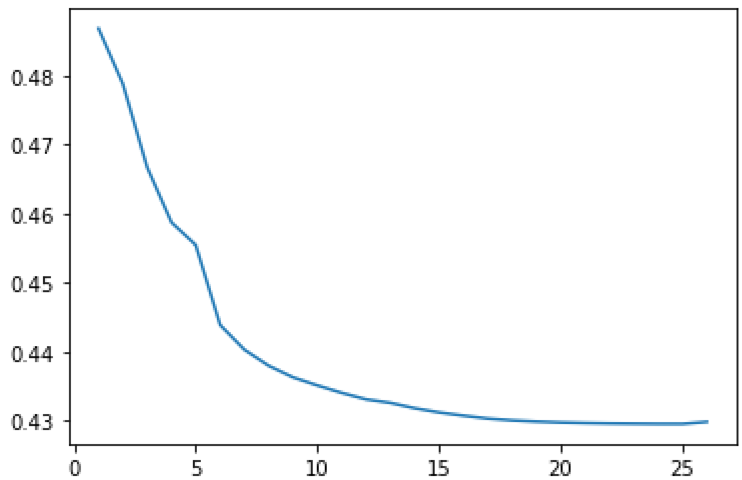


2B- Greedy Forward Feature Selection

We perform greedy forward feature selection by first calculating training error for individual features. Out of these, we select the feature with min. error and make subsets including this particular feature. We repeat the process until we get all the features in our subset. To obtain the best model, we take the one with the least error.

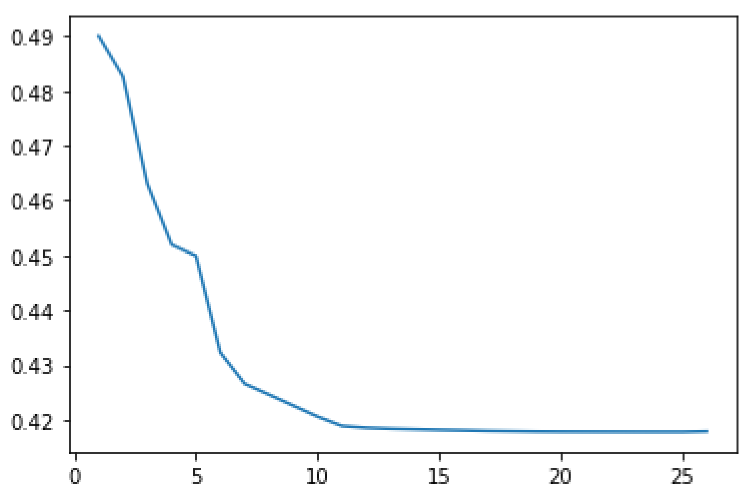
Subset of Features -	Training Error for subset
20	0.48680015461964066
20, 1	0.47876425151156193
20, 1, 13	0.4666603806862409
20, 1, 13, 3	0.4586681323141413
20, 1, 13, 3, 4	0.4553998929958801
20, 1, 13, 3, 4, 16	0.4438715738285372
20, 1, 13, 3, 4, 16, 2	0.4402796080303145
20, 1, 13, 3, 4, 16, 2, 15	0.43795452191317347
20, 1, 13, 3, 4, 16, 2, 15, 14	0.4362773385293425
20, 1, 13, 3, 4, 16, 2, 15, 14, 10	0.4351074668752276
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18	0.4340268728461876
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9	0.4330884731463834
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11	0.4325762267681773
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12	0.4317988224114434
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22	0.43119176059793324
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17	0.4307254531547224
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7	0.43030312708673796
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0	0.43004183354412656

20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23	0.4298658430262162
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21	0.429739142819956
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8	0.42965805991529105
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19	0.4295988940580948
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24	0.42956388814876895
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24, 6	0.429534988010786
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24, 6, 5	0.42953139922111233
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24, 6, 5, 25	0.4298261516231515



Subset of Features -	Testing Error for subset
20	0.49005330524602564
20, 1	0.4826694827488317
20, 1, 13	0.4630711516307173
20, 1, 13, 3	0.45197750995775404
20, 1, 13, 3, 4	0.4499250405572726
20, 1, 13, 3, 4, 16	0.43230224306078935
20, 1, 13, 3, 4, 16, 2	0.4266002682364598
20, 1, 13, 3, 4, 16, 2, 15	0.42461635463702363

20, 1, 13, 3, 4, 16, 2, 15, 14	0.42260255536680724
20, 1, 13, 3, 4, 16, 2, 15, 14, 10	0.42059716882519904
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18	0.41889638921527844
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9	0.41856128439184503
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11	0.41839935949532386
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12	0.41828840411846746
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22	0.4181813888202191
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17	0.41810222368517597
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7	0.4180026227052914
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0	0.4179453605172386
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23	0.4178856425727393
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21	0.41785704182919275
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8	0.4178441838713094
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19	0.41783569998587566
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24	0.41782799861676
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24, 6	0.41782135718172236
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24, 6, 5	0.4178211520695694
20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24, 6, 5, 25	0.417917510103053



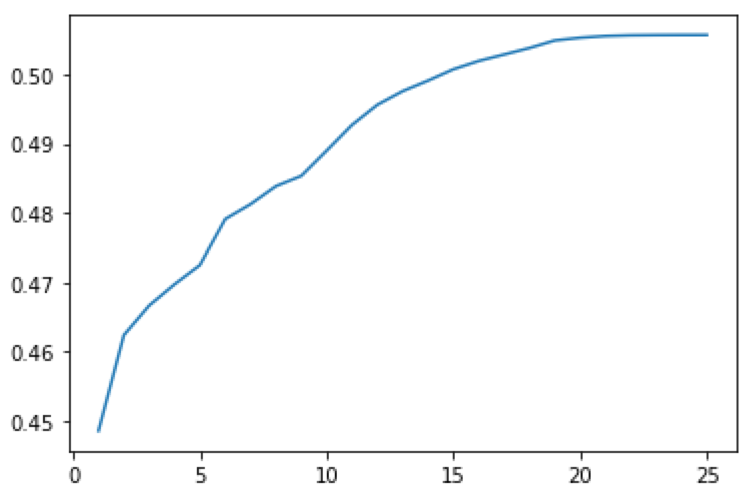
Greedy Backward Feature Selection

In the greedy backward feature selection method, we start off with all the features as our subset and keep on removing a feature to obtain the one with the least error. Once we get that subset, we build successive subsets from it and continue similarly to get the subset with least error until we get exactly one feature.

Subset of Features - Training Error for subset

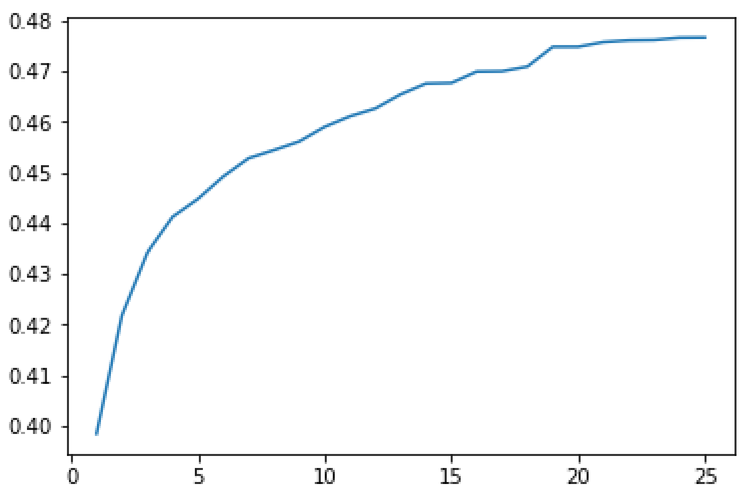
0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25	0.44851398853687796
0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.4623684564209405
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.46665595087069733
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.4697199191634084
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.47250331259379336
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.47910378302079176
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 18, 19, 20, 21, 22, 23, 24	0.481258496589659
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 18, 19, 20, 22, 23, 24	0.48386610483282466
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 18, 19, 22, 23, 24	0.4853495325742465
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 18, 19, 22, 24	0.4890012371199762
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 18, 19, 22, 24	0.49272158742872757
0, 2, 3, 6, 7, 8, 9, 10, 11, 12, 18, 19, 22, 24	0.4956400958241931
0, 3, 6, 7, 8, 9, 10, 11, 12, 18, 19, 22, 24	0.4976132611385842
3, 6, 7, 8, 10, 11, 12, 18, 19, 22, 24	0.49913804230272113

3, 6, 7, 8, 10, 11, 18, 19, 22, 24	0.5007819365564705
6, 7, 8, 10, 11, 12, 18, 19, 24	0.5019854528932531
6, 7, 8, 10, 11, 12, 19, 24	0.5029189047038953
6, 7, 8, 11, 12, 19, 24	0.5038578041951691
6, 7, 8, 12, 19, 24	0.5049538788701197
6, 7, 8, 12, 24	0.505348824850478
7, 8, 12, 24	0.5055987012166471
8, 12, 24	0.5057011104805499
8, 24	0.5057317747971535
24	0.5057448157960615



Subset of Features -	Testing Error for subset
0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25	0.3983779565976855
0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.42176418143494565
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.4341430827221132
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.44120223281223014
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.4447553138467746
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17, 18, 19, 20, 21, 22, 23, 24	0.44919366196079535
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 18, 19, 20, 21, 22, 23, 24	0.4528144579418528
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 18, 19, 20, 22, 23, 24	0.45442235443402995
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 18, 19, 22, 23, 24	0.45613270477078544
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 16, 18, 19, 22, 24	0.45902983675903497
0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 18, 19, 22, 24	0.4611154902666174
0, 2, 3, 6, 7, 8, 9, 10, 11, 12, 18, 19, 22, 24	0.46265714118149304
0, 3, 6, 7, 8, 9, 10, 11, 12, 18, 19, 22, 24	0.4654630463611771
3, 6, 7, 8, 10, 11, 12, 18, 19, 22, 24	0.46759782989293025
3, 6, 7, 8, 10, 11, 18, 19, 22, 24	0.46770620810032826
6, 7, 8, 10, 11, 12, 18, 19, 24	0.469962064688233
6, 7, 8, 10, 11, 12, 19, 24	0.47002684482090146
6, 7, 8, 11, 12, 19, 24	0.47094830016163264

6, 7, 8, 12, 19, 24	0.47486178200585666
6, 7, 8, 12, 24	0.4748675448504311
7, 8, 12, 24	0.4758058423209765
8, 12, 24	0.4761420922397325
8, 24	0.4762144373442774
24	0.47666924140204225



2C - Comparative Analysis

Regression model with all 26 features:

We have built the linear regression model using gradient descent for the dataset as:

$$y = w0 + w1x1 + w2x2 + \ldots..+w26x26$$

The optimal weights are:

[-0.00911838 -0.00031681 0.06172803 0.0347887 -0.02735899 0.03583962
0.03290145 -0.00933532 0.01735667 -0.02580388 0.01460424 0.03562633
-0.0116559 -0.01784221 -0.03917831 0.00010665 -0.05316706 -0.0304389
-0.02926148 0.02223491 -0.01333347 -0.05631209 0.04709571 0.01000656
-0.0102225   0.00322865 0.00322865]

The training error is: 0.4478255580731643
The testing error is: 0.5616366429583213

The best model obtained from **the Pearson correlation method** is with 22 features taken according to maximum correlation with target attribute.

The optimal weights for this model are:
[0.17404906 -0.03901422 0.24564854 -0.42448231 -0.18586083 -0.18338707
0.02804944 0.04998061 0.33036543 0.60634486 -0.65005187 -0.05028934
-0.06723513 0.09597149 0.01938376 0.09275283 0.01580222 0.12341389
-0.09818501 0.08839969 -0.08193843 0.0415883]

The training error for this model is:
0.4399134847500075

The testing error for this model is:
0.5061823521695902

Using **PCA** we found out that considering 22 principal components will be best on the basis of training error.

The training error is 0.41752473339288076 and the testing error for this model is 0.5128099208458522.

Using **Forward feature selection**, [20, 1, 13, 3, 4, 16, 2, 15, 14, 10, 18, 9, 11, 12, 22, 17, 7, 0, 23, 21, 8, 19, 24, 6, 5] is the best subset of features as the testing error is minimum for this model.
The testing error is 0.4178211520695694 and the training error is 0.42953139922111233.

Using **Backward Feature selection,** [0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25] is the best subset of features.

The testing error for this model is 0.3983779565976855 and training error is 0.44851398853687796.

Finally, the best model that we found is using the backward feature selection as it gives minimum testing error