# Foundations of Data Science

**Assignment - 2**

## 2 - A

### 1. Correlation coefficients

We compute the correlation coefficient using Pearson correlation as a measure. For each feature, the correlation coefficient is computed with respect to the target attribute “Appliances”, and then stored in a list for easy access.

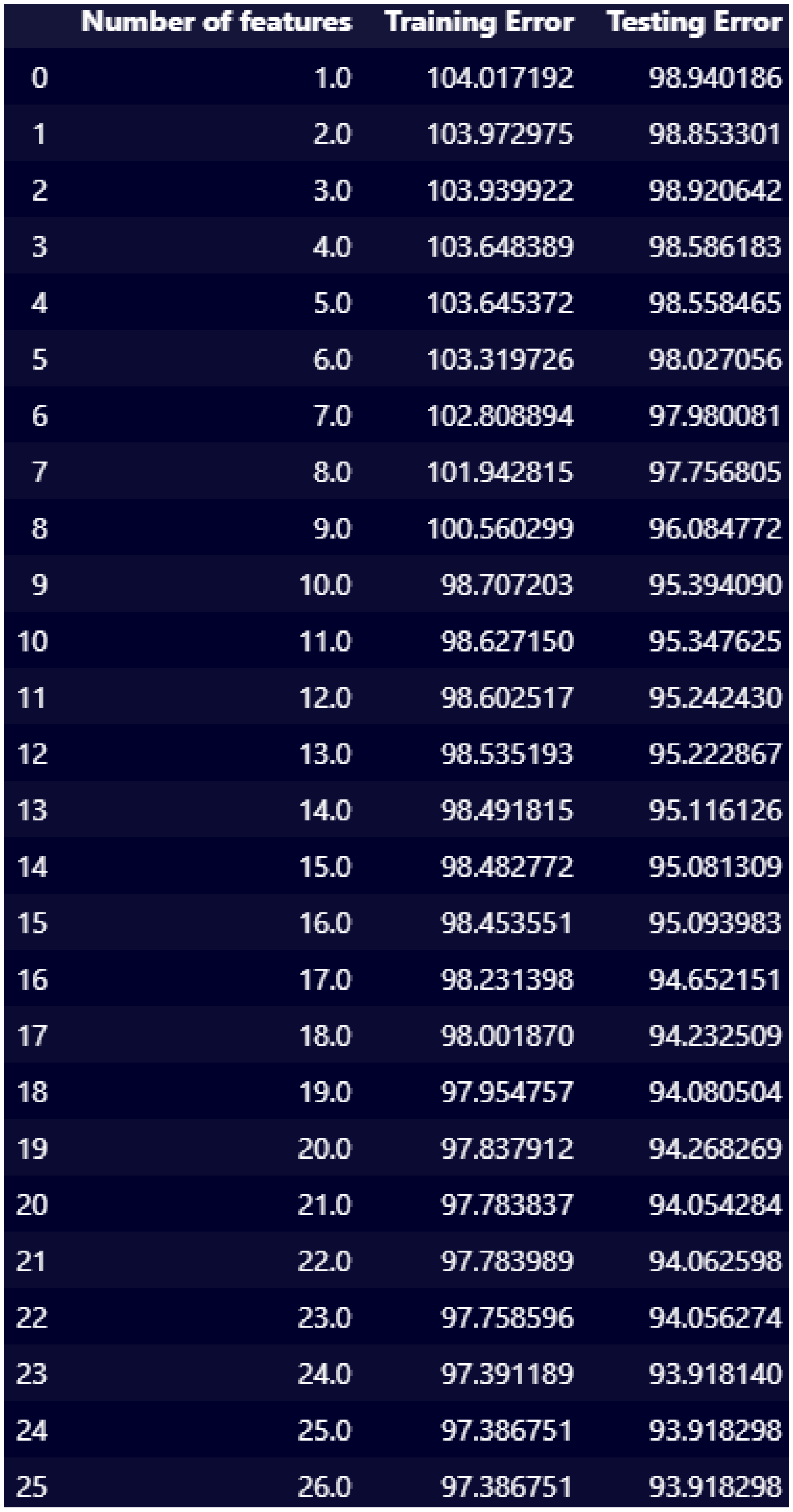
An important thing to note here is that even negative correlations with a high magnitude should be given proper consideration because they are strongly correlated, just in an inverse fashion. To overcome this issue, we computed the absolute valued coefficients, and then sorted the list, while keeping track of the feature that a particular coefficient represented.

Here’s what the Pearson correlation measure looks like in a mathematical form:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{(x_i - \bar{x})^2 (y_i - \bar{y})^2}}$$

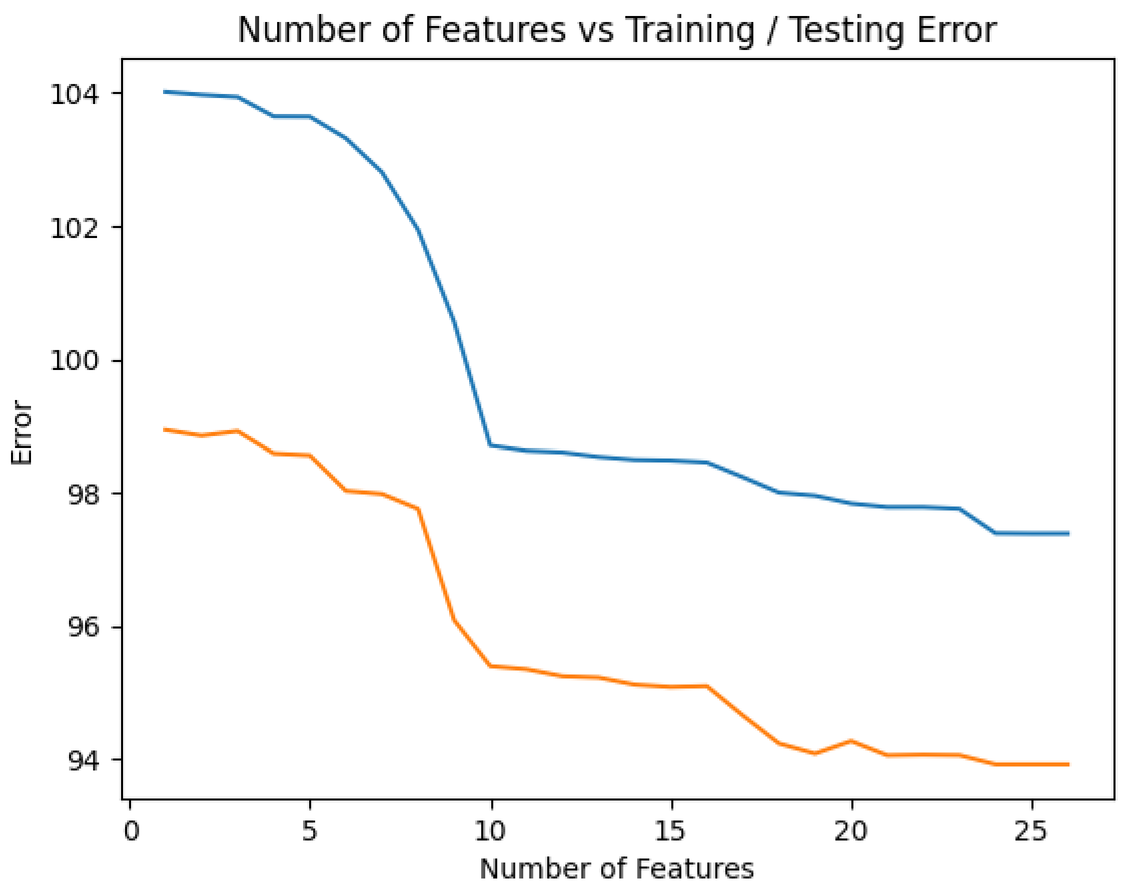
Now that we have a sorted list of coefficients with their respective features, we build models where we vary the number of features taken from 1 to 26. For a model with n features, we take the first n features in the sorted coefficient list. We train each model taking the number of iterations as 10000 and setting the learning rate as 0.00001. The reason we take such a low value of the learning rate is because even after normalization, the range of the numbers in the dataset is very high valued, so if we take a larger value of the learning rate (let’s suppose 0.001), during gradient descent, we will see an overshoot in the cost which will increase with the number of iterations.

The result of our experiment with the data is as follows:



| | Number of features | Training Error | Testing Error |
|---|---|---|---|
| 0 | 1.0 | 104.017192 | 98.940186 |
| 1 | 2.0 | 103.972975 | 98.853301 |
| 2 | 3.0 | 103.939922 | 98.920642 |
| 3 | 4.0 | 103.648389 | 98.586183 |
| 4 | 5.0 | 103.645372 | 98.558465 |
| 5 | 6.0 | 103.319726 | 98.027056 |
| 6 | 7.0 | 102.808894 | 97.980081 |
| 7 | 8.0 | 101.942815 | 97.756805 |
| 8 | 9.0 | 100.560299 | 96.084772 |
| 9 | 10.0 | 98.707203 | 95.394090 |
| 10 | 11.0 | 98.627150 | 95.347625 |
| 11 | 12.0 | 98.602517 | 95.242430 |
| 12 | 13.0 | 98.535193 | 95.222867 |
| 13 | 14.0 | 98.491815 | 95.116126 |
| 14 | 15.0 | 98.482772 | 95.081309 |
| 15 | 16.0 | 98.453551 | 95.093983 |
| 16 | 17.0 | 98.231398 | 94.652151 |
| 17 | 18.0 | 98.001870 | 94.232509 |
| 18 | 19.0 | 97.954757 | 94.080504 |
| 19 | 20.0 | 97.837912 | 94.268269 |
| 20 | 21.0 | 97.783837 | 94.054284 |
| 21 | 22.0 | 97.783989 | 94.062598 |
| 22 | 23.0 | 97.758596 | 94.056274 |
| 23 | 24.0 | 97.391189 | 93.918140 |
| 24 | 25.0 | 97.386751 | 93.918298 |
| 25 | 26.0 | 97.386751 | 93.918298 |

Here's a graphical representation of the same for easier understanding:



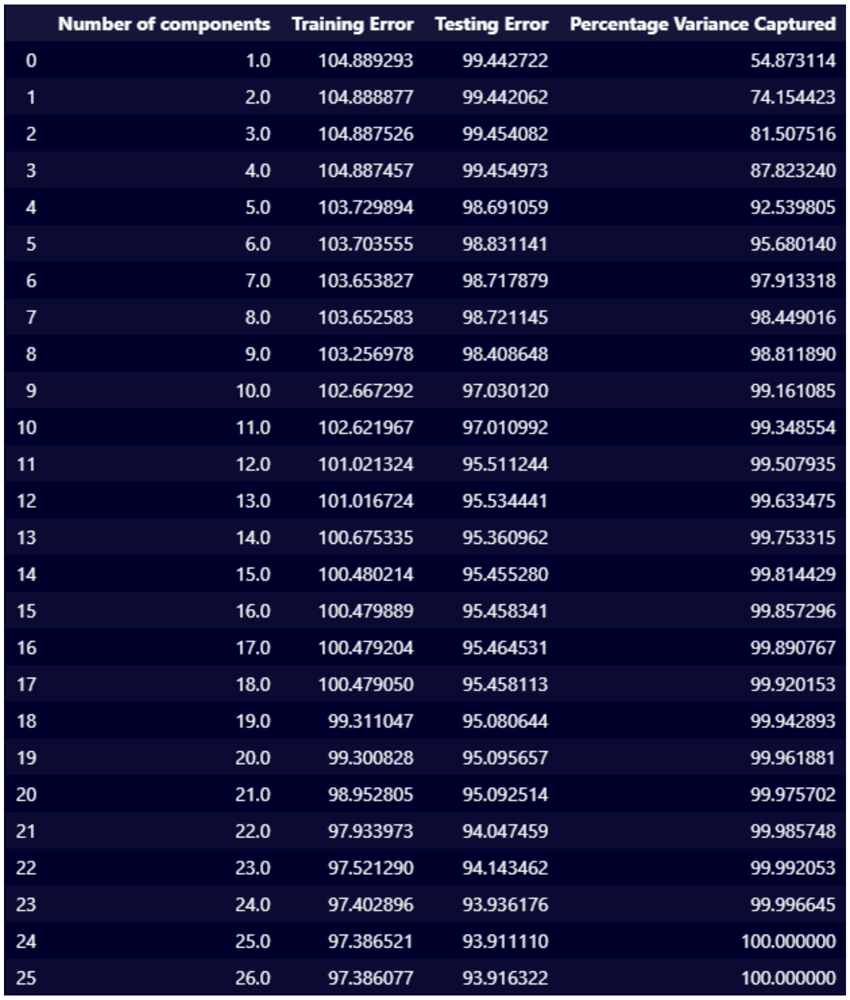
The blue line represents the training error, and the orange line represents the testing error.

**Inference**: One thing to note from the table is that the training and testing errors for 25 features and 26 features are identical. The reason for that is that the features 'rv1' and 'rv2' are identical. So the way that we have written the code ignores the repeated columns while building the dataset out of the most correlated features, and the errors that we see for 25 and 26 features are coming from the same model. In conclusion, we see that taking 24 features out of the entire feature set is optimal to build a regression model for the given dataset.

## 2. Principal Component Analysis

Principal Component Analysis is not a machine learning algorithm, but rather a tool that helps us to reduce the dimensions for a given feature set. In most cases, it is not possible to work with all the features from the dataset, because it becomes computationally heavy. In those kinds of situations, PCA jumps in to reduce the number of features we build our model with, by transforming our dataset into a vector space wherein we can control the number of dimensions we want to work with. We do so, by computing the eigenvalues of the dataset (only the input features), and take the eigenvectors corresponding to the first n eigenvalues in a sorted list. Now we have an n dimensional vector space wherein we can project the data points onto these n dimensions to reduce the number of features to n.

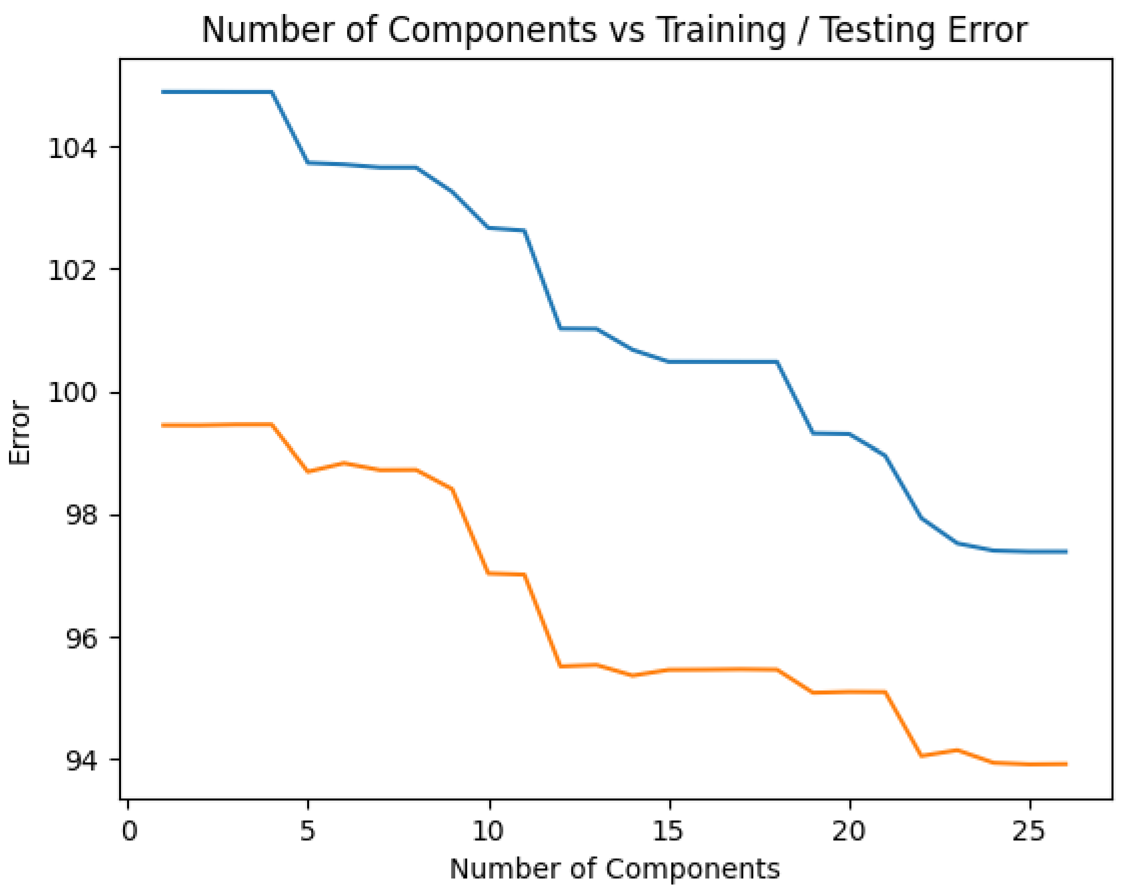
Here are the results of our experiment with the dataset:



| | Number of components | Training Error | Testing Error | Percentage Variance Captured |
|---|---|---|---|---|
| 0 | 1.0 | 104.889293 | 99.442722 | 54.873114 |
| 1 | 2.0 | 104.888877 | 99.442062 | 74.154423 |
| 2 | 3.0 | 104.887526 | 99.454082 | 81.507516 |
| 3 | 4.0 | 104.887457 | 99.454973 | 87.823240 |
| 4 | 5.0 | 103.729894 | 98.691059 | 92.539805 |
| 5 | 6.0 | 103.703555 | 98.831141 | 95.680140 |
| 6 | 7.0 | 103.653827 | 98.717879 | 97.913318 |
| 7 | 8.0 | 103.652583 | 98.721145 | 98.449016 |
| 8 | 9.0 | 103.256978 | 98.408648 | 98.811890 |
| 9 | 10.0 | 102.667292 | 97.030120 | 99.161085 |
| 10 | 11.0 | 102.621967 | 97.010992 | 99.348554 |
| 11 | 12.0 | 101.021324 | 95.511244 | 99.507935 |
| 12 | 13.0 | 101.016724 | 95.534441 | 99.633475 |
| 13 | 14.0 | 100.675335 | 95.360962 | 99.753315 |
| 14 | 15.0 | 100.480214 | 95.455280 | 99.814429 |
| 15 | 16.0 | 100.479889 | 95.458341 | 99.857296 |
| 16 | 17.0 | 100.479204 | 95.464531 | 99.890767 |
| 17 | 18.0 | 100.479050 | 95.458113 | 99.920153 |
| 18 | 19.0 | 99.311047 | 95.080644 | 99.942893 |
| 19 | 20.0 | 99.300828 | 95.095657 | 99.961881 |
| 20 | 21.0 | 98.952805 | 95.092514 | 99.975702 |
| 21 | 22.0 | 97.933973 | 94.047459 | 99.985748 |
| 22 | 23.0 | 97.521290 | 94.143462 | 99.992053 |
| 23 | 24.0 | 97.402896 | 93.936176 | 99.996645 |
| 24 | 25.0 | 97.386521 | 93.911110 | 100.000000 |
| 25 | 26.0 | 97.386077 | 93.916322 | 100.000000 |

We made the models by reducing the number of features to components with the number varying from 1 to 26. In addition to the training error and the testing error, we calculated the variance captured by each model. The way we computed the value of the variance was by taking the sum of the top n eigenvalues and dividing it by the sum of all eigenvalues, for a model where we consider n dimensions. The percentage of variance captured by the models vary from 54.87% to 100 %. 100 % variance being captured is intuitive because we are considering all the original features to build the model.

Here's a graphical representation of the errors for easier understanding:



The blue line represents the training error, and the orange line represents the testing error.

**Inference:** It turns out that the model where we have 25 features being taken into account is the one which has the lowest training and testing error.

**2 - B**

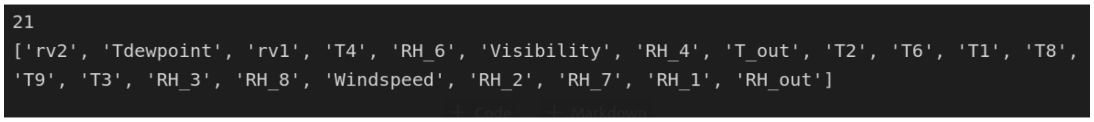
1. **Forward feature selection**

   In forward feature selection, we start with an empty set of features and greedily pick the subsequent features. We are using recursion to simulate the process of picking the features. At every iteration we go through the list of features that have not been added so far, among those we pick the feature which gives us the best model using Test RMS as the metric. If during the process adding any feature does not give us a better model than the previous iteration or the previous set of features, we stop the process and return our final set of selected features.

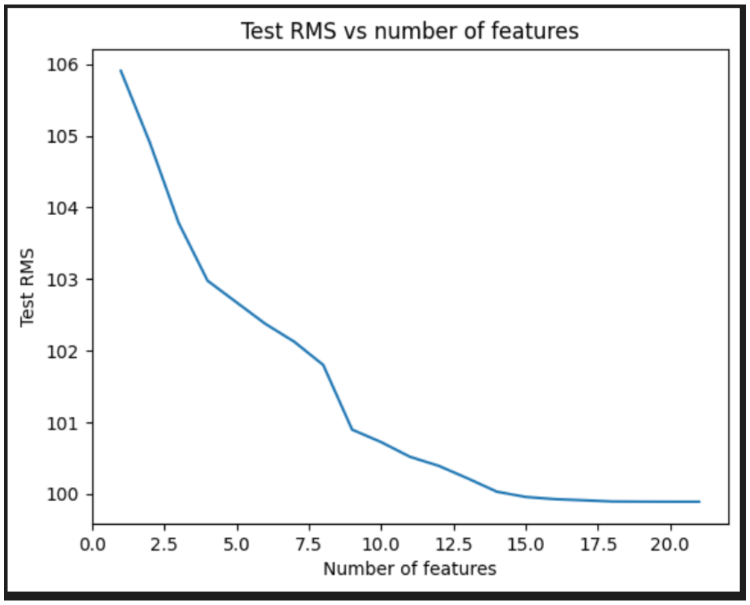
   **Inference:** Using the greedy forward feature selection algorithm, we got the set of following 21 features as the best. Adding any new feature to the model after this point did not give us a better model as compared to the previous iteration, hence we stop at this point.
   RMS : 93.64999685757

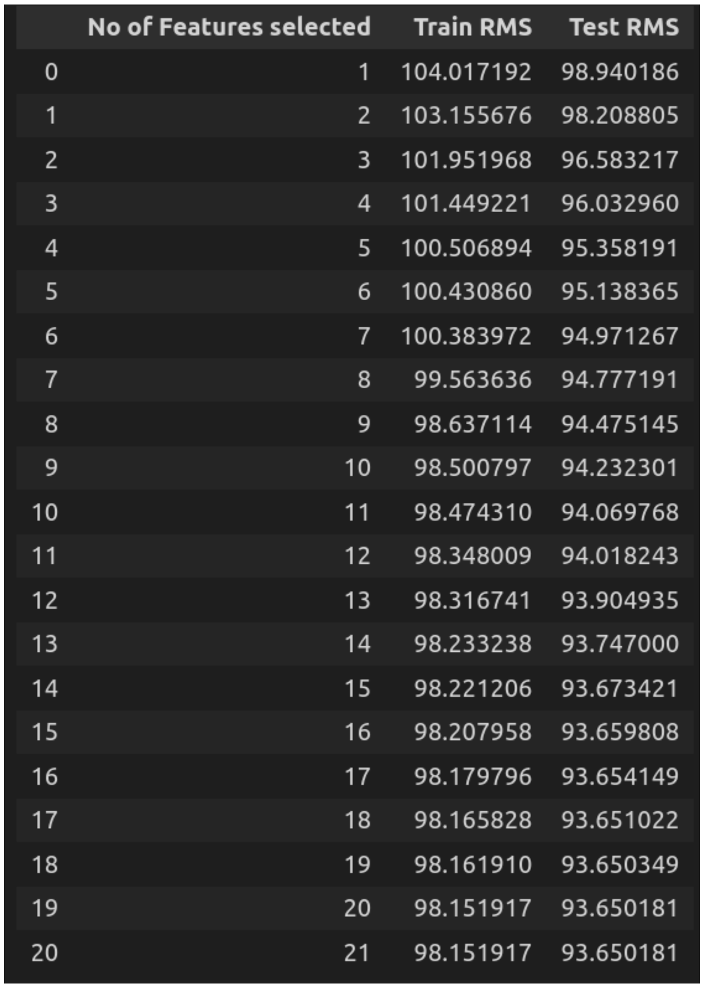
   Here are the results of our experiments:

   **<u>Final List of Selected Features :</u>**



```
21
['rv2', 'Tdewpoint', 'rv1', 'T4', 'RH_6', 'Visibility', 'RH_4', 'T_out', 'T2', 'T6', 'T1', 'T8',
'T9', 'T3', 'RH_3', 'RH_8', 'Windspeed', 'RH_2', 'RH_7', 'RH_1', 'RH_out']
```





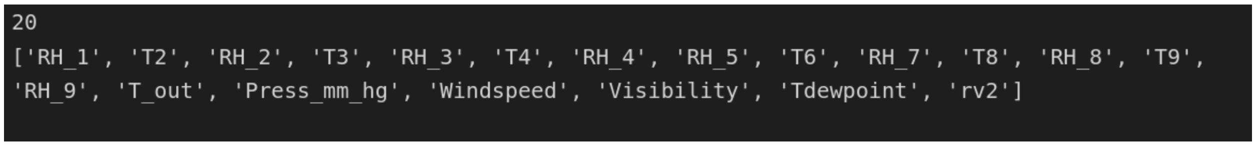
| | No of Features selected | Train RMS | Test RMS |
|---|---|---|---|
| 0 | 1 | 104.017192 | 98.940186 |
| 1 | 2 | 103.155676 | 98.208805 |
| 2 | 3 | 101.951968 | 96.583217 |
| 3 | 4 | 101.449221 | 96.032960 |
| 4 | 5 | 100.506894 | 95.358191 |
| 5 | 6 | 100.430860 | 95.138365 |
| 6 | 7 | 100.383972 | 94.971267 |
| 7 | 8 | 99.563636 | 94.777191 |
| 8 | 9 | 98.637114 | 94.475145 |
| 9 | 10 | 98.500797 | 94.232301 |
| 10 | 11 | 98.474310 | 94.069768 |
| 11 | 12 | 98.348009 | 94.018243 |
| 12 | 13 | 98.316741 | 93.904935 |
| 13 | 14 | 98.233238 | 93.747000 |
| 14 | 15 | 98.221206 | 93.673421 |
| 15 | 16 | 98.207958 | 93.659808 |
| 16 | 17 | 98.179796 | 93.654149 |
| 17 | 18 | 98.165828 | 93.651022 |
| 18 | 19 | 98.161910 | 93.650349 |
| 19 | 20 | 98.151917 | 93.650181 |
| 20 | 21 | 98.151917 | 93.650181 |

## 2. Backward feature selection

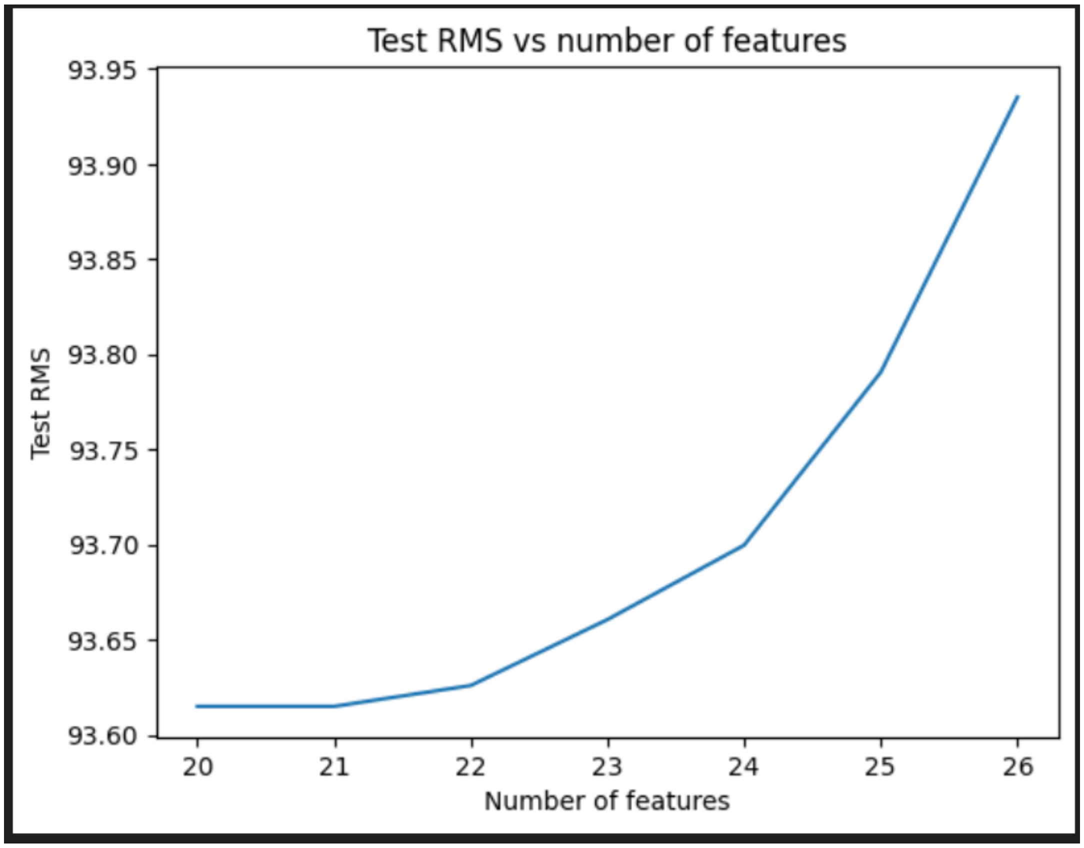
In backward feature selection we start off with all the features given and then the features with least relevance are removed one by one. We are using recursion to simulate the process of elimination. During the first iteration we have all the features and then for each feature present we build a regression model leaving that feature out, the feature whose removal yields the best model is chosen to be eliminated. We stop the process when elimination of no process yields a better model as compared to the current model or only one feature remains.

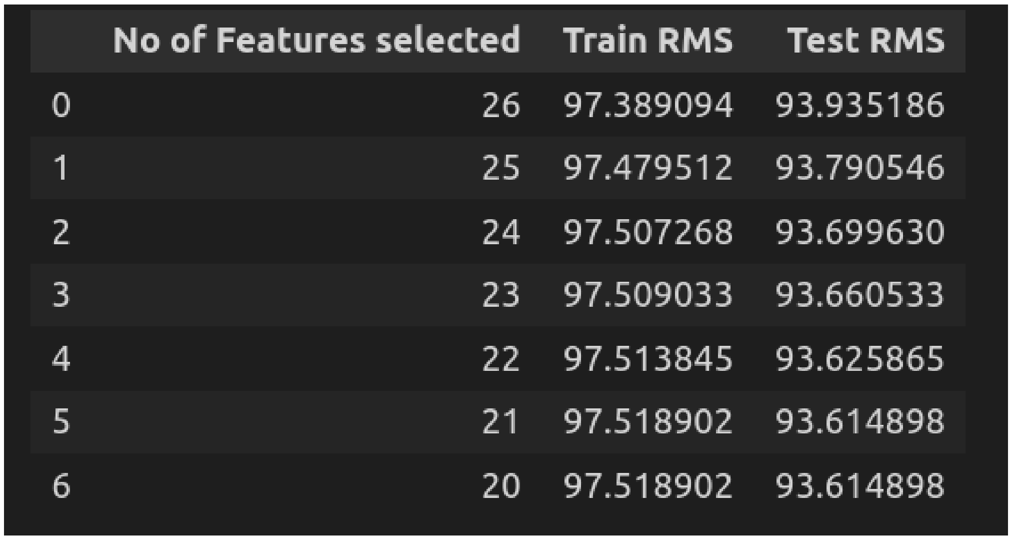
**Inference:** Using the greedy backward feature selection algorithm, we got the set of following 20 features as the best. Removing any feature after this point did not give us a better model as compared to the previous iteration, hence we stop at this point.
RMS : 93.91829821626636
Here are the results of our experiments:

**Final List of Selected Features :**



```
20
['RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T4', 'RH_4', 'RH_5', 'T6', 'RH_7', 'T8', 'RH_8', 'T9',
'RH_9', 'T_out', 'Press_mm_hg', 'Windspeed', 'Visibility', 'Tdewpoint', 'rv2']
```





| | No of Features selected | Train RMS | Test RMS |
|---|---|---|---|
| 0 | 26 | 97.389094 | 93.935186 |
| 1 | 25 | 97.479512 | 93.790546 |
| 2 | 24 | 97.507268 | 93.699630 |
| 3 | 23 | 97.509033 | 93.660533 |
| 4 | 22 | 97.513845 | 93.625865 |
| 5 | 21 | 97.518902 | 93.614898 |
| 6 | 20 | 97.518902 | 93.614898 |

**2 - C**

1. **Comparative Analysis of models and feature selection techniques**

    - The method of Correlation coefficients shows us that taking 24 features is optimal and the testing RMS for the model constructed using 24 features is 93.91

    - The method of Principal Component Analysis shows that using a model which requires 25 components is optimal with a testing RMS of 93.91

    - The method Forward Feature Selection shows that using a model which uses 21 of the original features to train itself returns a testing RMS of around 93.64.

    - The method Backward Feature Selection shows that using a model which uses 20 of the original features to train itself returns a testing RMS of around 93.61.

    Clearly, the Backward Feature Selection yields the best results out of the ones that have been computed as a part of this assignment.

## Group Members :

1. Arkishman Ghosh - 2020A7PS2077H
2. Ashwin Arun - 2020A7PS1291H
3. Ashwin Naveen Pugalia - 2020A7PS1080H