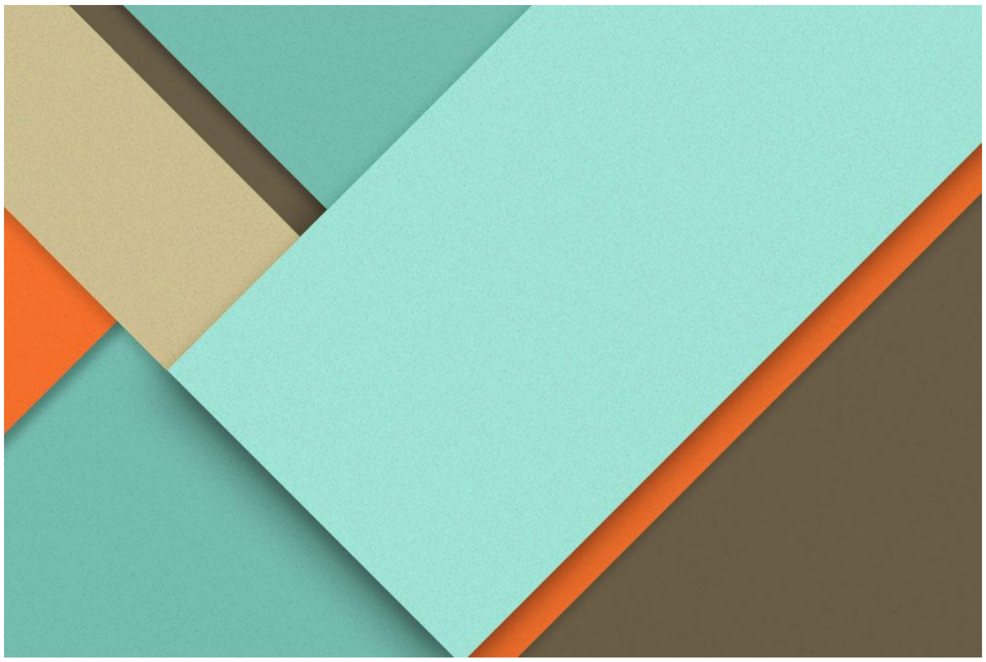
Comparing Feature Extraction Approaches



27.07.2022

Project members:

Tolulope Olusooto,

Noor-adien alshrah

Introduction

Feature selection is a vital tool used in our industry it improves the machine learning model performance and increases the prediction probability of getting the correct prediction.
A lot of data that is gathered is not always cleaned or ready to use some pre-processing procedures need to be conducted before building the machine learning model. we were introduced to some feature selection or extraction methods such as; Principle component analysis (PCA), forward search, backward elimination, and more.

The aim of this project is to use feature selection and principle component analysis (PCA) and apply it to large-scale data, choose one of these approaches (classification or regression), and observe how these models react when applying any of the two approaches. In this project, we choose to use the regression model and applied K-nearest neighbor (KNN) and linear regression to our data.
The data that we choose is the Power consumption of the Tetouan city Data Set. The data was taken from the UCI machine learning repository (http://archive.ics.uci.edu/ml/index.php).
This dataset is a record of the power consumption of three different distribution networks of Tetouan city which is located in northern Morocco. The attribute information consists of:

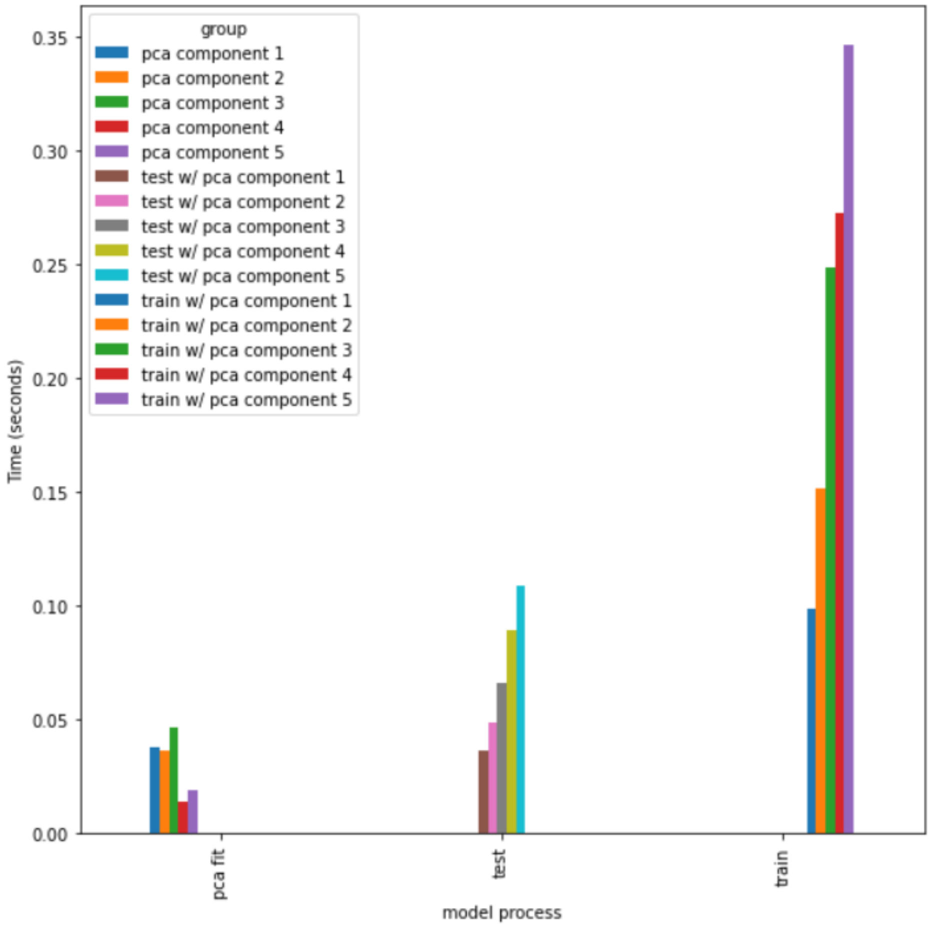
1. Temperature: Weather Temperature of Tetouan city.
2. Humidity: Weather Humidity of Tetouan city.
3. Wind Speed of Tetouan city.
4. general diffuse flows
5. diffuse flows
6. power consumption of zone 1 of Tetouan city.
7. power consumption of zone 2 of Tetouan city.
8. power consumption of zone 3 of Tetouan city.

We used the data in our model, applying both PCA and backward selection approaches.

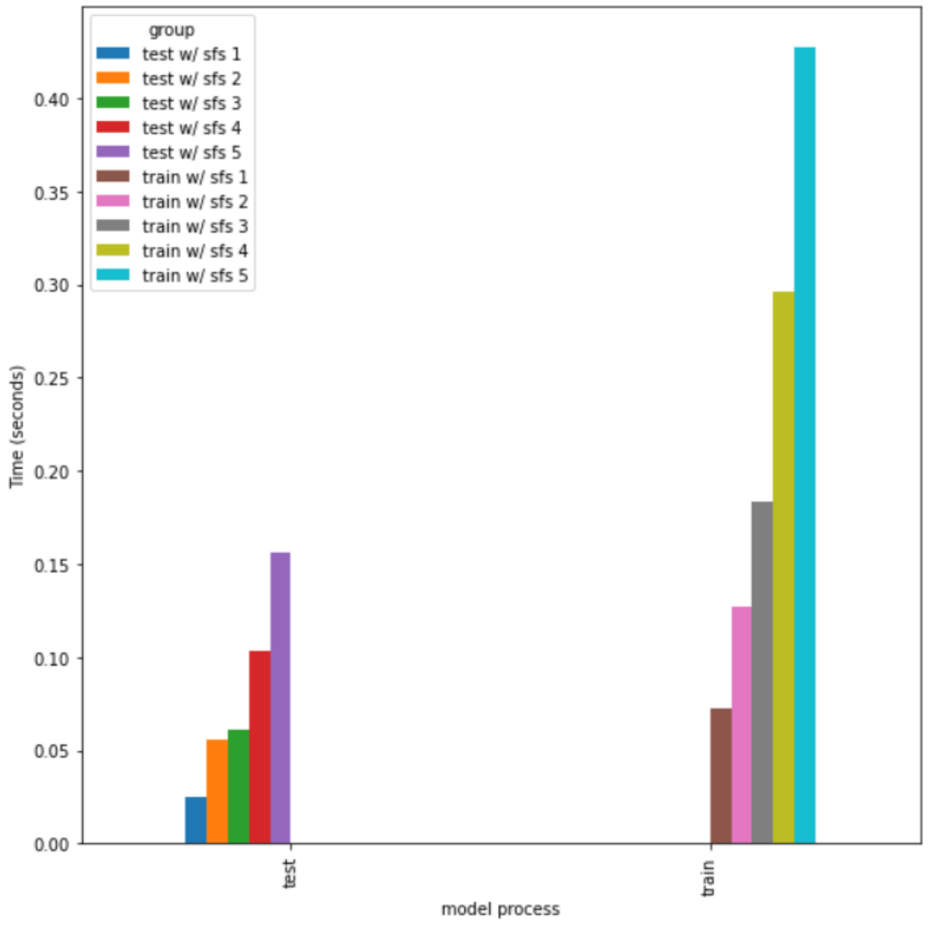
Results

➔ **Computational Times for both training and testing:**

1. For the training/testing using k-nearest neighbor the computational time:

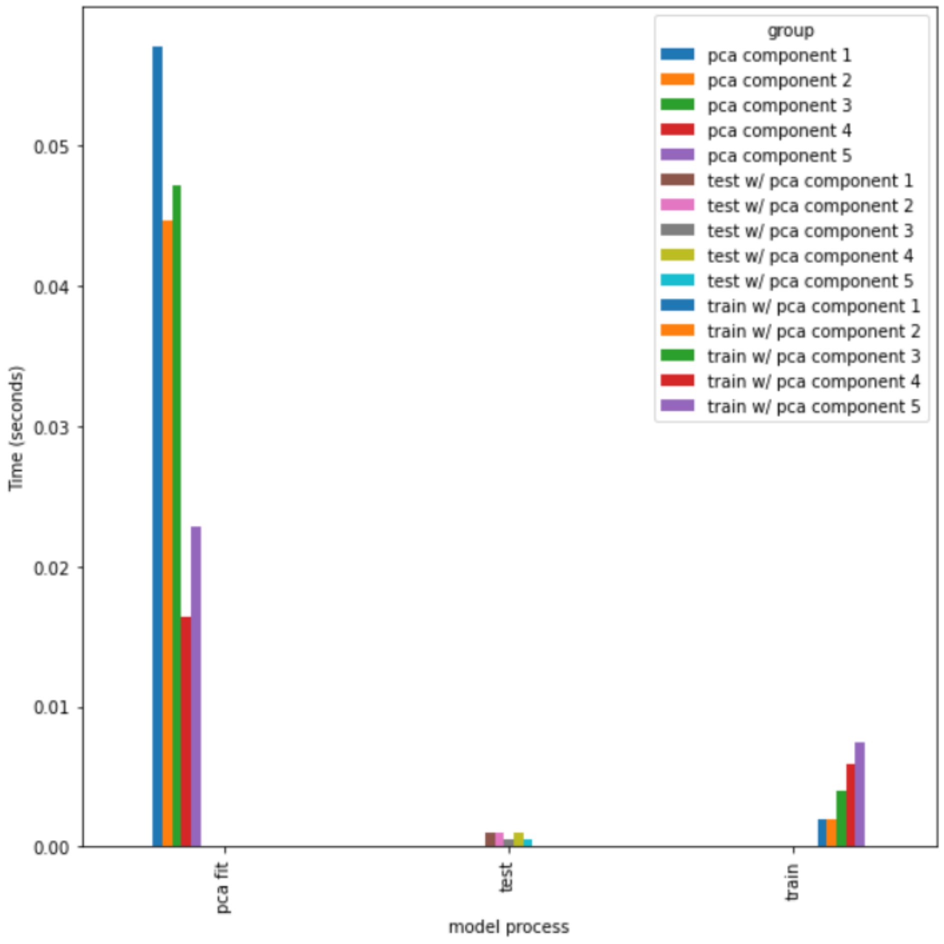


PCA

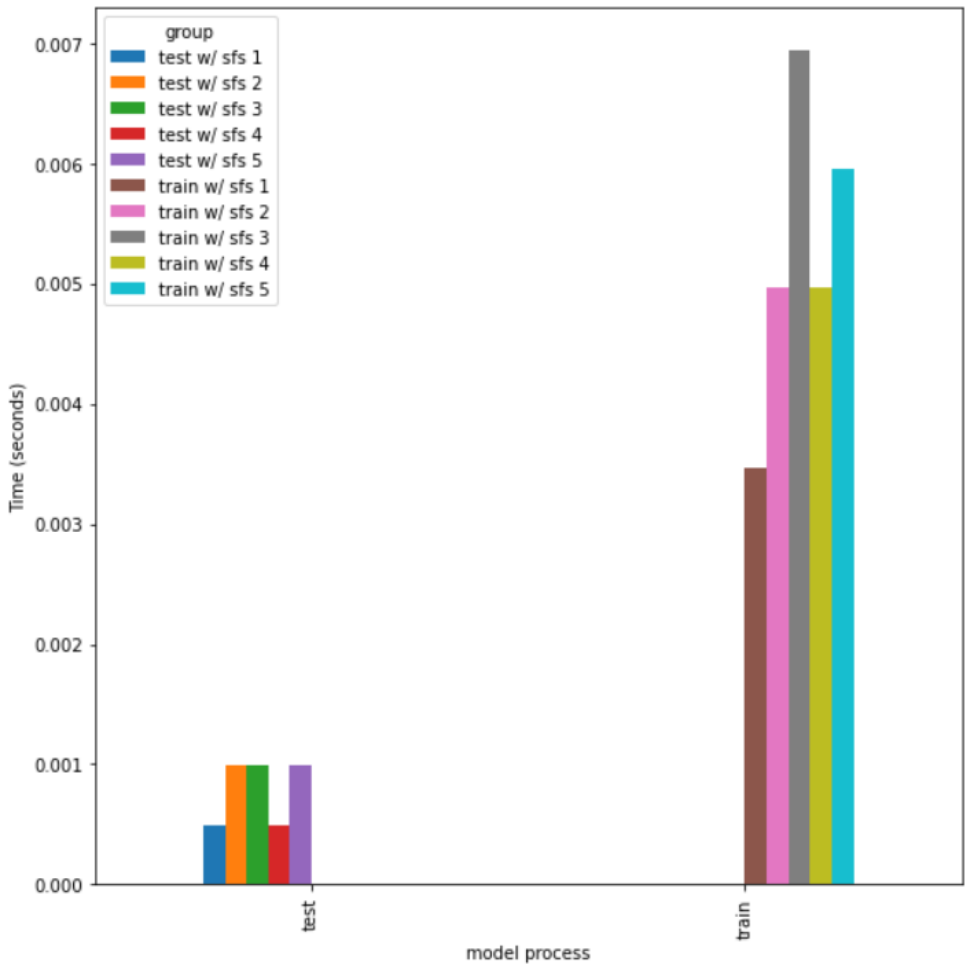


Feature selection

2. **For the training/testing using linear regression the computational time:**


group
pca component 1
pca component 2
pca component 3
pca component 4
pca component 5
test w/ pca component 1
test w/ pca component 2
test w/ pca component 3
test w/ pca component 4
test w/ pca component 5
train w/ pca component 1
train w/ pca component 2
train w/ pca component 3
train w/ pca component 4
train w/ pca component 5
Time (seconds)
0.00
0.01
0.02
0.03
0.04
0.05
pca fit
test
train
model process

PCA



Feature selection

In general we found that the training time took longer regardless of the dimension reduction technique or model used. However, the difference in speed is minimal as no one process in any of the experiment trails was above half a second long. The more features were added the longer the process took. On average PCA was faster than backward feature selection. Linear regression was the faster process regardless of the dimension reduction technique, and this is likely due to the simplicity of the linear model.

➔ **Expremint conducted**

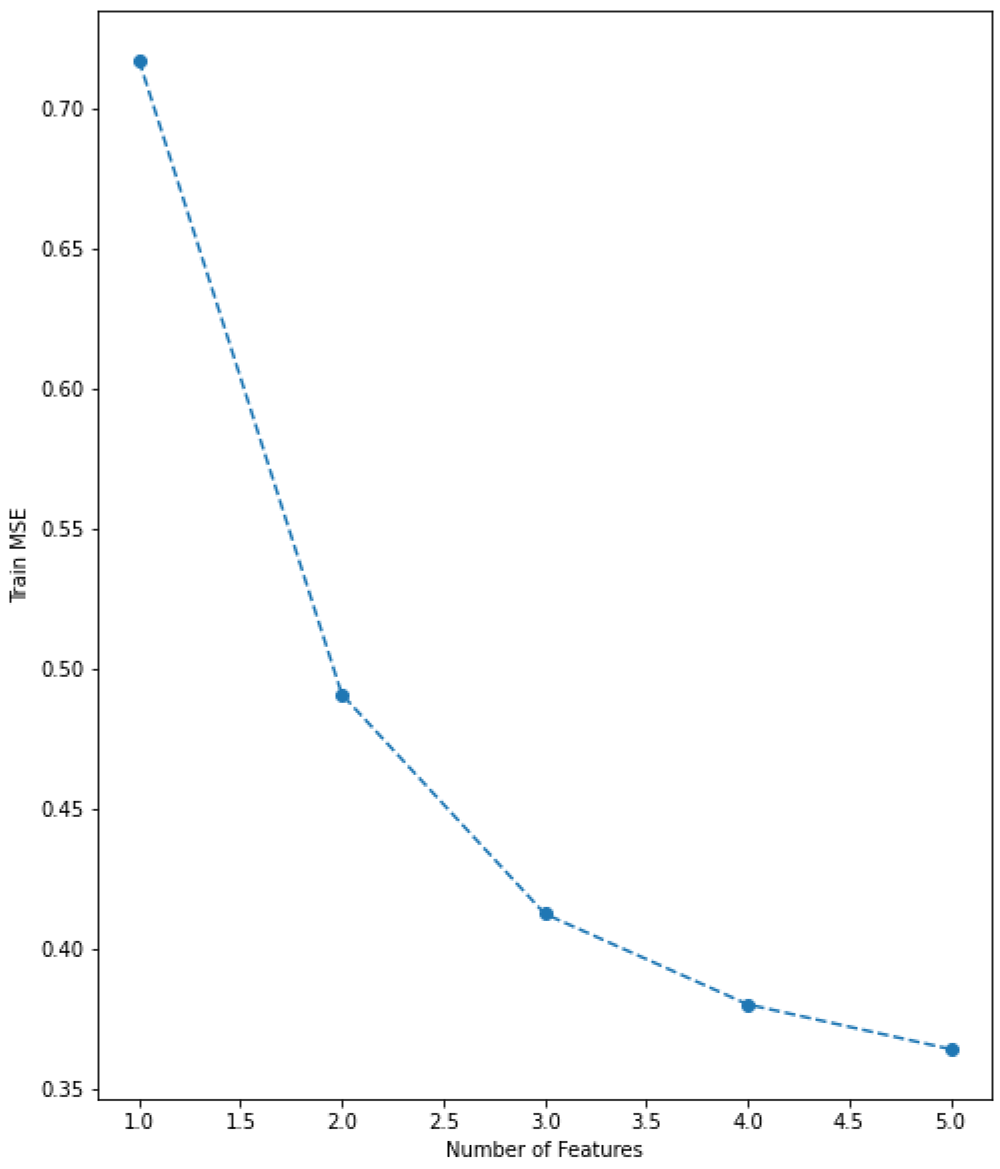
Firstly the data needed to be cleaned before the start of working with one of the approaches, and the three zones of the power consumption were summed up and dropped the date and time. then a standard scaler was performed, then split the data 30-70 for training and testing.

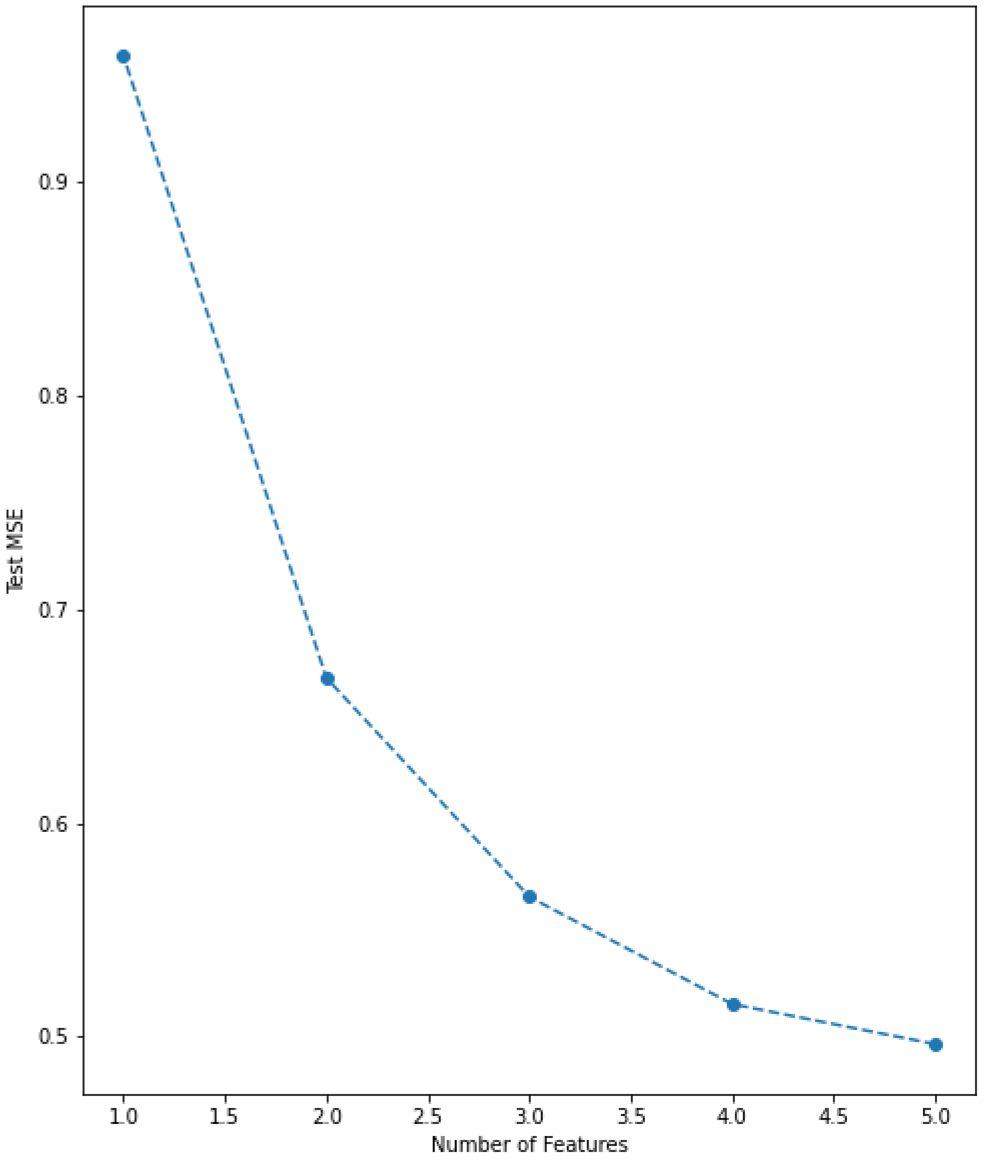
- Experiment 1 with k-nearest Neighbour (KNN)

Before the PCA approach could be applied KNN, the number of neighbours was determined by finding the number neighbours that minimized the MSE. The optimal neighbour count was 7 with the MSE value of 0.496. Below are the results of PCA approach on KNN:

- **PCA using K-nearest neighbor.**

```
training R^2 with 5 features is 0.6353333310168895 (pca w/ knn).
test R^2 with 5 features is 0.504207491508797 (pca w/ knn).
training R^2 with 4 features is 0.619342530302668 (pca w/ knn).
test R^2 with 4 features is 0.4855304557150968 (pca w/ knn).
training R^2 with 3 features is 0.5869586736174339 (pca w/ knn).
test R^2 with 3 features is 0.43548472089315693 (pca w/ knn).
training R^2 with 2 features is 0.5085579209874372 (pca w/ knn).
test R^2 with 2 features is 0.33330098399737695 (pca w/ knn).
training R^2 with 1 features is 0.2819193391337109 (pca w/ knn).
test R^2 with 1 features is 0.04286684001651753 (pca w/ knn).
```


Train MSE
Number of Features
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

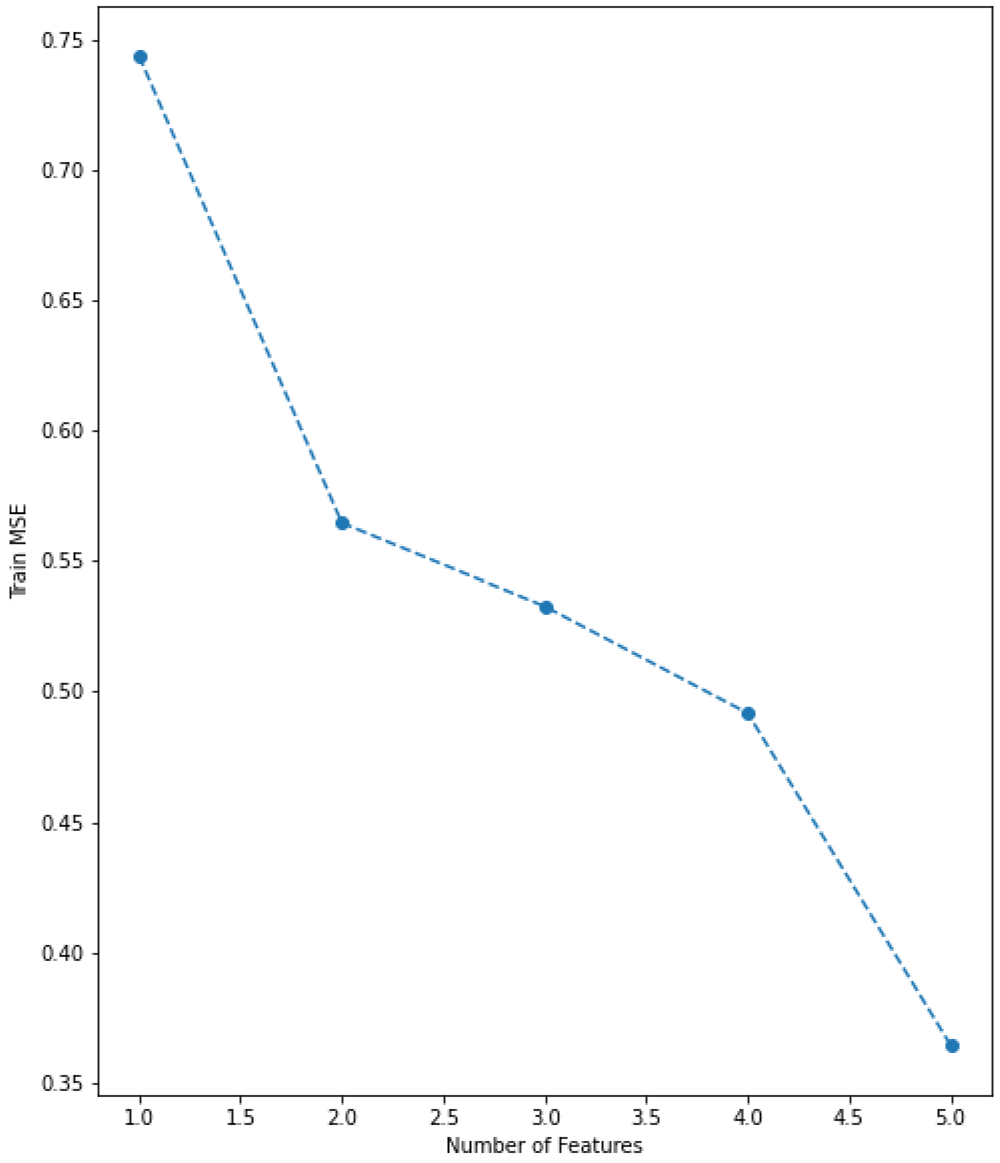


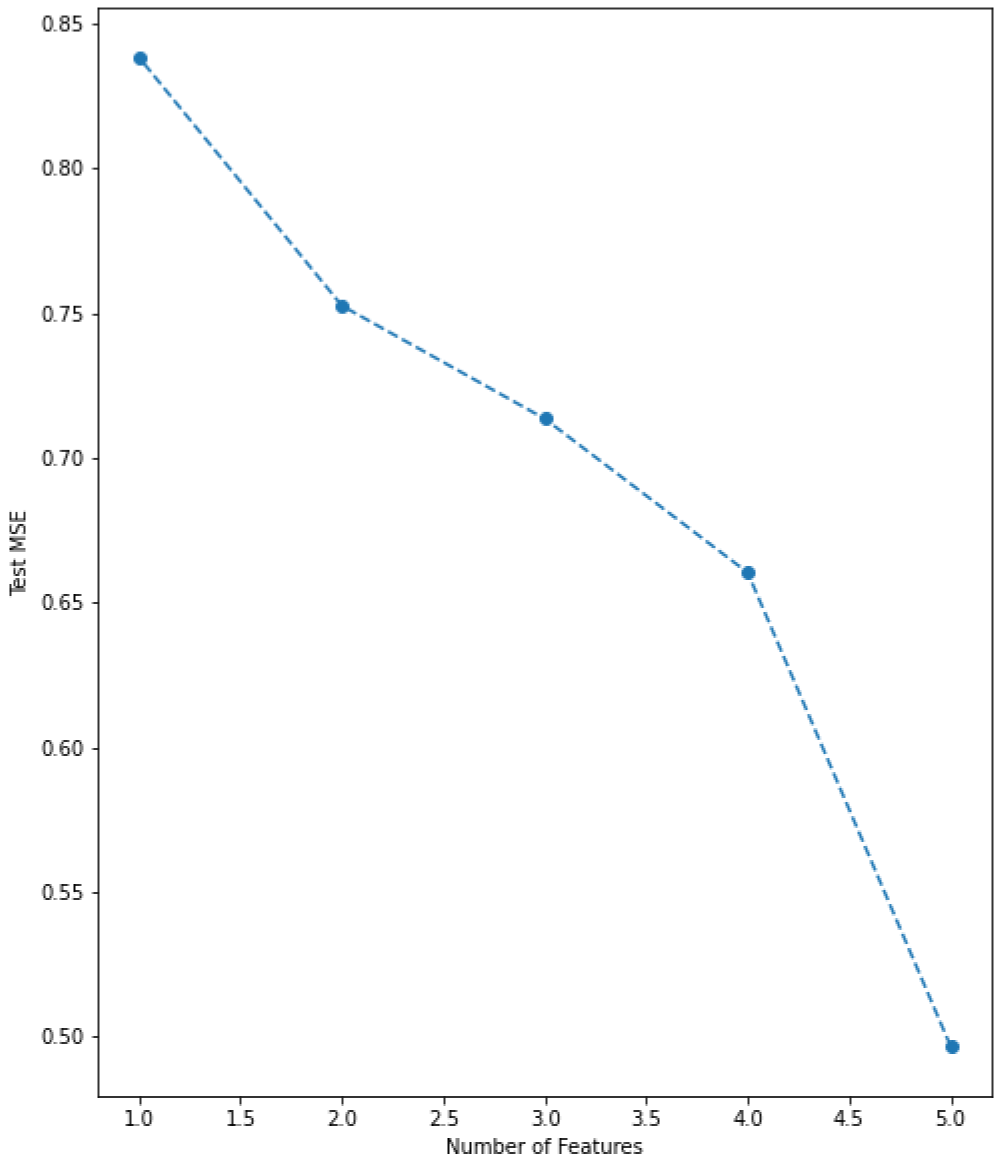
The R^2 values of the KNN model for both train and test decreased as features were removed from the data, intuitively this makes sense as less variance is being explained by the data. The MSE for both train and test were fairly similar with test having a slightly higher MSE value per number of features present in the data. After the 4th feature the reduction in MSE from 4 to 5 features was minimal. Meaning prediction accuracy of the model gains very little when we add more features after the 4th; a potential optimal dimension maybe 4.

- **Feature selection using K-nearest neighbor**

```
training R^2 with 5 features is 0.6353333310168895 (sfs w/ knn).
```

```
test R^2 with 5 features is 0.504207491508797 (sfs w/ knn).
training R^2 with 4 features is 0.507897016350235 (sfs w/ knn).
test R^2 with 4 features is 0.34073242761450295 (sfs w/ knn).
training R^2 with 3 features is 0.46693704025605076 (sfs w/ knn).
test R^2 with 3 features is 0.28767273424842443 (sfs w/ knn).
training R^2 with 2 features is 0.43505208562882514 (sfs w/ knn).
test R^2 with 2 features is 0.24867429058485568 (sfs w/ knn).
training R^2 with 1 features is 0.2557529519304005 (sfs w/ knn).
test R^2 with 1 features is 0.16302986572069667 (sfs w/ knn).
```



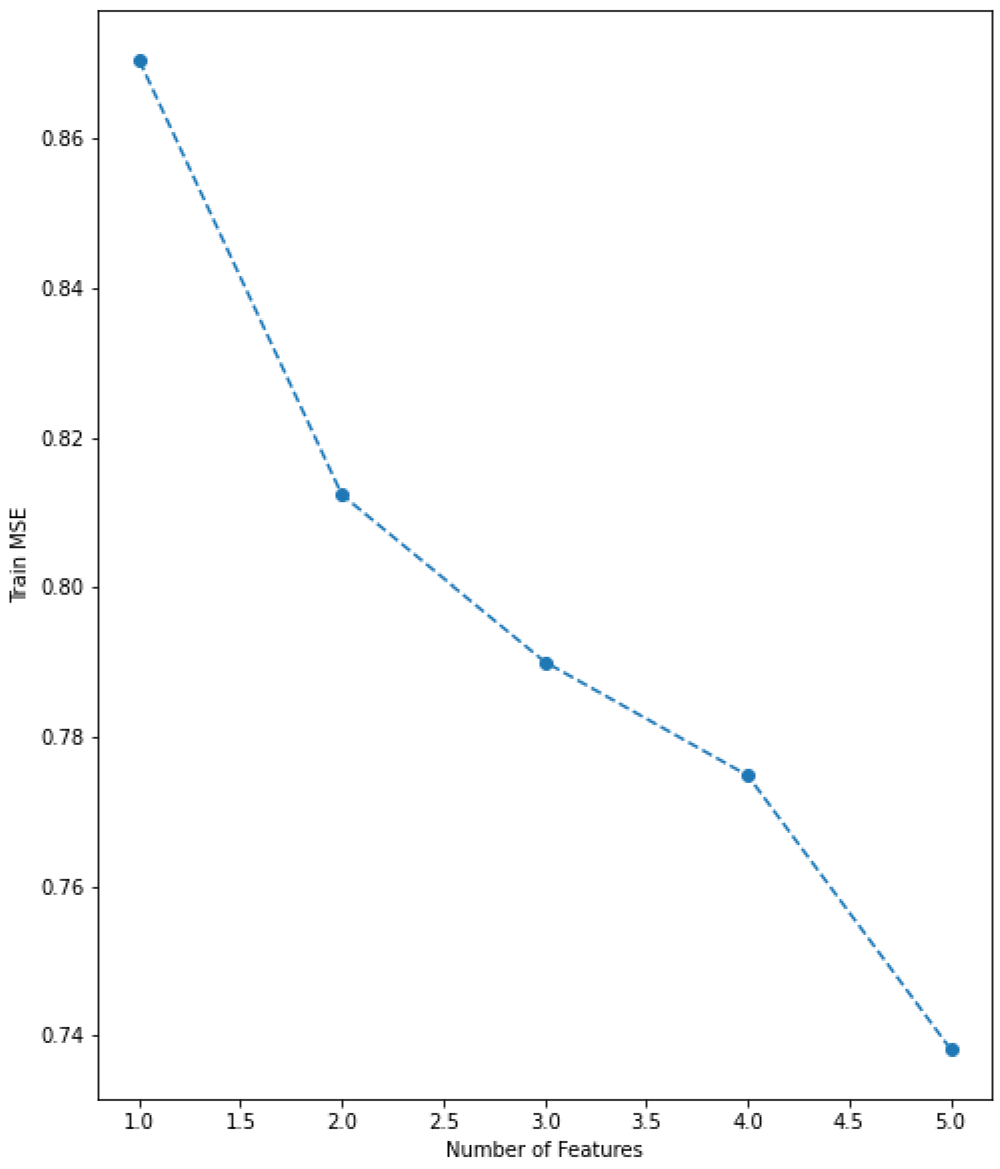


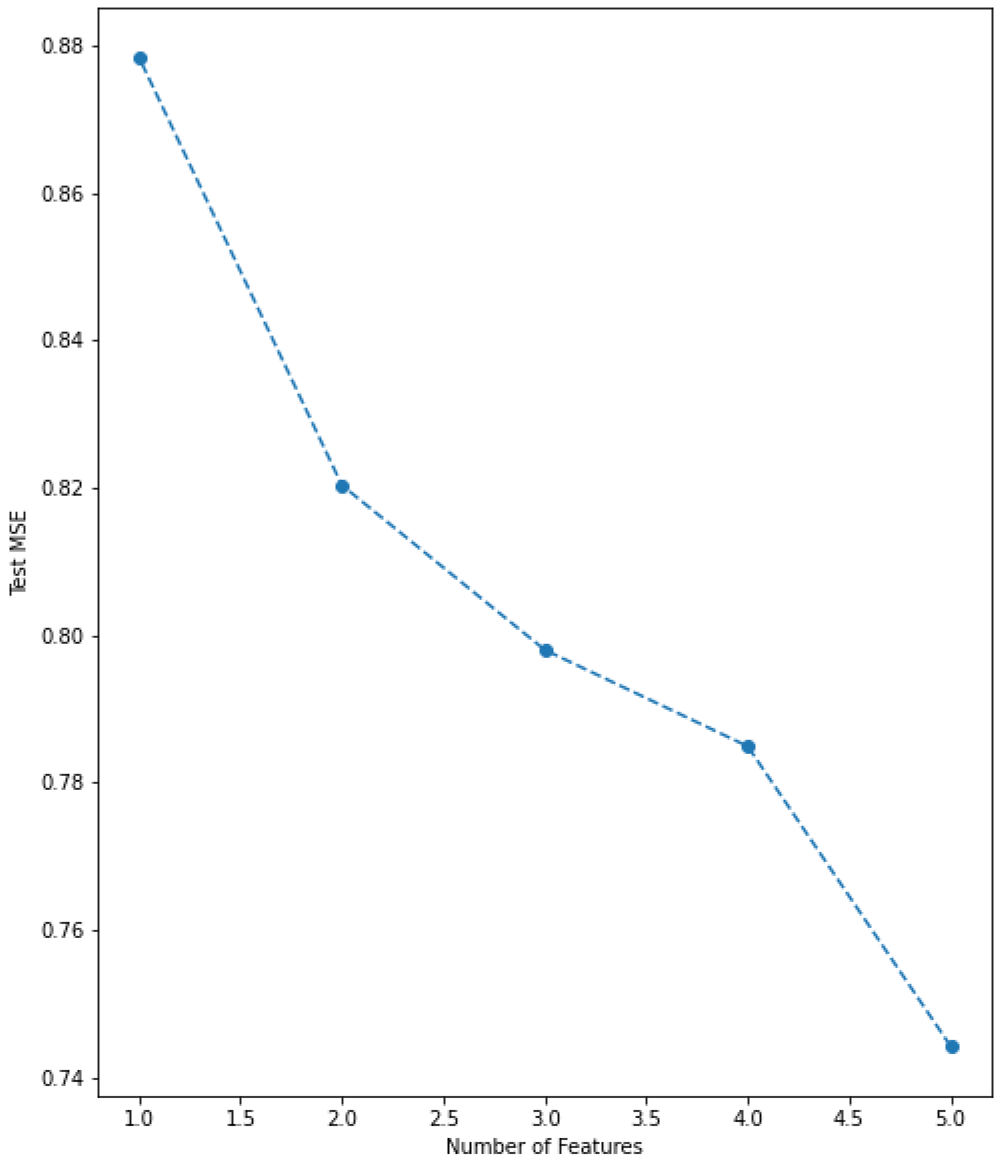
The results of the backward feature selection showed a more drastic difference in MSE value when features were removed. The R^2 values decreased as features were moved, like that of the PCA approach. However, on average, the PCA approach outperforms the backward selection approach.

- **PCA using linear regression.**

```
training R^2 with 5 features is 0.2612894141195038 (pca w/ lnreg).
test R^2 with 5 features is 0.2570583199401476 (pca w/ lnreg).
training R^2 with 4 features is 0.2246263135158647 (pca w/ lnreg).
test R^2 with 4 features is 0.21641565383846217 (pca w/ lnreg).
```

```
training R^2 with 3 features is 0.2094101943998583 (pca w/ lnreg).
test R^2 with 3 features is 0.20334988161387924 (pca w/ lnreg).
training R^2 with 2 features is 0.18709489662070367 (pca w/ lnreg).
test R^2 with 2 features is 0.1810223965771749 (pca w/ lnreg).
training R^2 with 1 features is 0.1288360772262901 (pca w/ lnreg).
test R^2 with 1 features is 0.12303431224566275 (pca w/ lnreg).
```

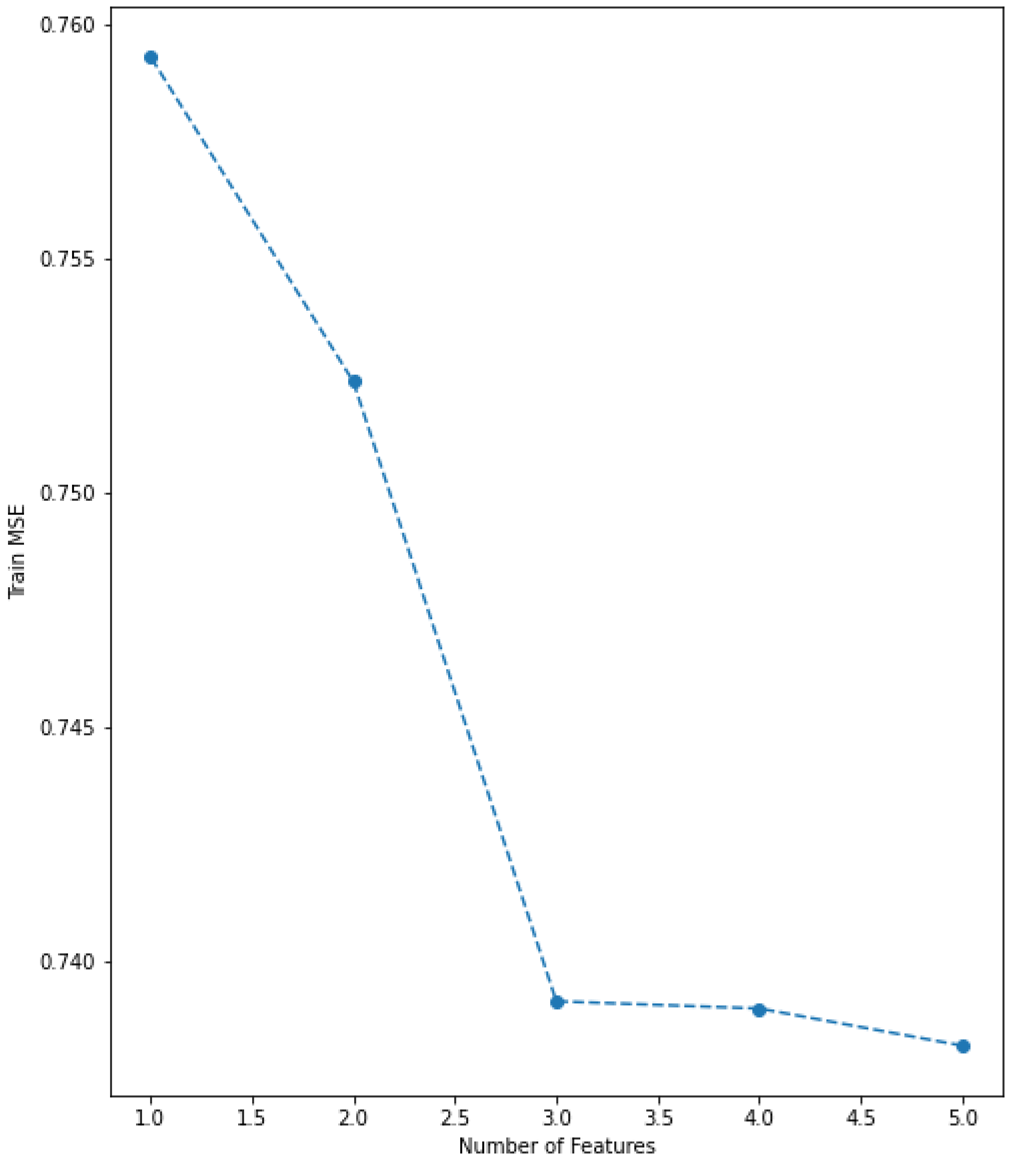


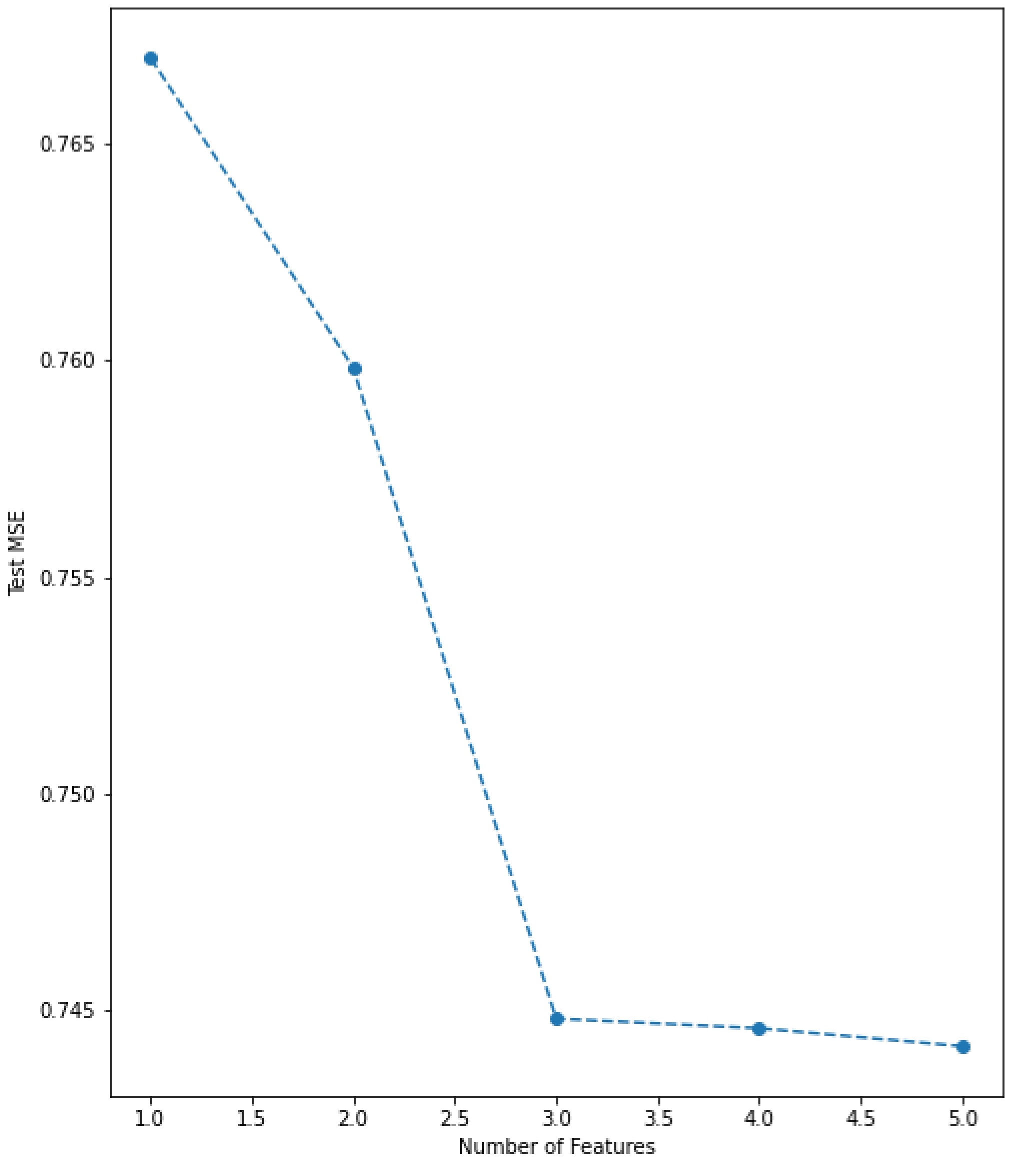


- **Feature selection using linear regression.**

```
training R^2 with 5 features is 0.2612894141195038 (sfs w/ lnreg).
test R^2 with 5 features is 0.2570583199401475 (sfs w/ lnreg).
training R^2 with 4 features is 0.26049493793488576 (sfs w/ lnreg).
test R^2 with 4 features is 0.2566445761129966 (sfs w/ lnreg).
training R^2 with 3 features is 0.2603399987195437 (sfs w/ lnreg).
test R^2 with 3 features is 0.25642312147781865 (sfs w/ lnreg).
training R^2 with 2 features is 0.247087035739448 (sfs w/ lnreg).
test R^2 with 2 features is 0.2414334851344443 (sfs w/ lnreg).
training R^2 with 1 features is 0.24012560011976924 (sfs w/ lnreg).
```

```
test R^2 with 1 features is 0.23429899078973127 (sfs w/ lnreg).
```





Both PCA and backward selection approaches were applied to a linear model and we found that on average the backward selection outperformed the PCA. The R^2 values when backward selection is used were higher than that of the PCA approach and the MSE values were lower than the PCA method. Interestingly we found that after the 3rd feature the change in MSE is minimal for the backward selection and a potential optimal dimension could be at 3.

Based on the above results, we found that KNN was a better model than the linear model; regardless of the dimension reduction approach. This may imply that the data is somewhat nonlinear, hence the reason why KNN was better suited to capture the pattern of the dataset.