**Milestone 4**

**DSC 324/424 Advanced Data Analysis  
Final Project: Milestone 4 Description**  
**Due:** Wednesday, June by 11:59 pm CST

**Group 8: Ayesha Ali, Shreyas P, Shalvika Mishra**

In milestone 3 our OLS model didn’t perform well, and it made us backtrack to Milestone 2 and think about whether we have dropped any significant column. Hence as a part of reverse engineering, we performed the below steps:

1. We found the variance of all variables to see which one has the highest-lowest variance.
2. In the fileds mentioned below we replaced na’s with zeros:

'Usable\_area', 'Free\_of\_Relation', 'Bedrooms', 'Bathrooms', 'Floors', 'Furnishing\_quality', 'Year\_renovated', 'Energy\_source', 'Energy\_certificate\_type', 'Energy\_consumption', 'Energy\_efficiency\_class'.

1. We did mean imputation for just Year\_built and Garages. Previously we were dropping na’s for Year\_built but on relating to real-world we thought that it might be a significant field.
2. We also decided not to convert Year\_built to date as we wanted to include it in the model.
3. We previously ran OLS on just numerical fields i.e., 'Price', 'Living\_space', 'Lot','Rooms', 'Bedrooms', 'Bathrooms', 'Floors','Garages' but now the change we created dummies for all non-numerical columns including city and places because relating to real-world it would significantly matter to the buyer which city and place the house is located.
4. Then we tried OLS regression again to check whether our second preprocessing was helpful and below are the results:

OLS Regression results from 1st Milestone and first preprocessing:

Text

Description automatically generated

OLS Regression results from 2nd Milestone and second preprocessing:

Table

Description automatically generated with medium confidence

As we can clearly see the adjusted R2 went from 67% to 73% and the R2 increased from 67% to 92%. Hence going forward with other techniques, we would be using the newly preprocessed data frame, as is evident in the previous technique we missed out on important features.

**After Professor’s Meeting:**

Our group member met the professor and showed him our data preprocessing. The challenge the group was facing was the field named City and Place appeared important when we related back to real-world scenarios. For example, if someone plans to buy a house the location of the house makes a huge significance in the decision but because this City had 534 and Place had 4762 unique records, we decided to drop it. When we tried to create dummies keeping City and Place the number of columns in total in the data frame crossed 4000 and after dropping the number of columns along with dummies are 171.

However after dropping columns City, Place, Unnamed, and Free\_of\_Relation.

We further went ahead and ran the OLS Regression, L1 Regression, and L2 Regression, and as expected the accuracy dropped (as we are removing important features from the data frame) which might be a big factor to predict the price of the house.

**As seen below the R2 from 92% dropped to 74% again but the adjusted R2 increased from 67% to 73%.**

Table

Description automatically generated with low confidence

**Random Forest Results Elaboration:**

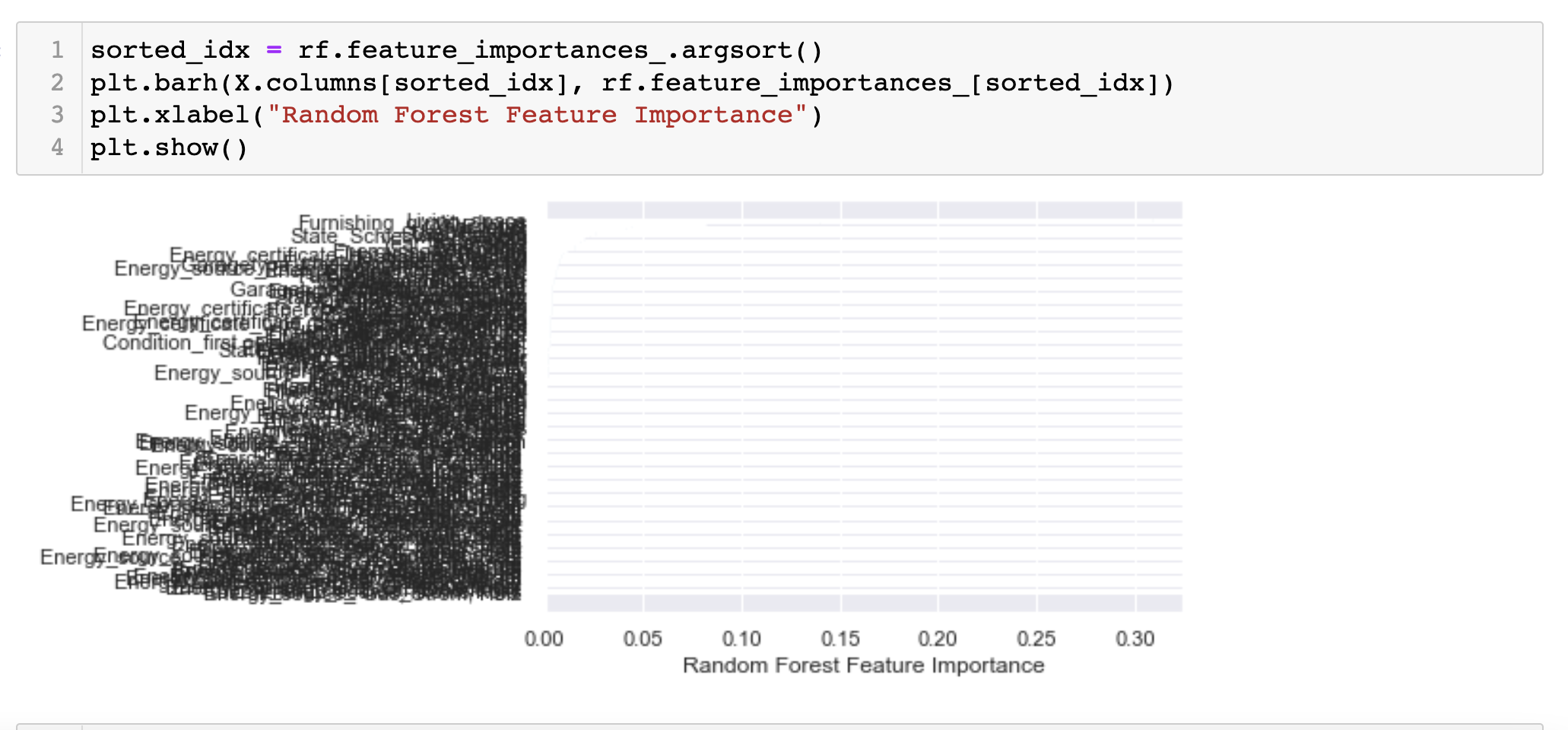
We wanted to try various supervised learning techniques for milestone 3 and keep unsupervised for milestone 4 that’s why we tried Random Forest. Another main reason for choosing random forest was we didn’t have much knowledge about the data hence after researching we found that Random Forest is used for Feature Selection as well.

Instead of trying a decision tree, we thought Random Forest would be better because Random Forest is an ensemble method that in basic terms can be explained as the prediction given by each decision tree is noted, then a majority vote is taken and that becomes the Random Forest prediction. We also used Random Forest because the model score for Random Forest was better than the decision tree (which is as expected).

The idea was to use Random Forest for feature importance to see which features will majorly affect the model. We also thought of using the var function to check the variance of features i.e., variables with a maximum variance will have the most important effect on the model (but that was necessarily not true).

**Random Forest on changed data frame:**

After we performed Random Forest on the newly preprocessed data frame the feature importance matrix came as below as there were 171 columns (including dummies for state) :



Due to a large number of features, the matrix was plotted but couldn’t help in concretely stating which features are important.

**Principal Component Analysis**

We believed This technique could help in dimensionality reduction, since we have 170 columns that include the dummy variables. As it can attempt to capture most variance without losing too much information, thereby minimizing noise and redundancy.

import pandas as pd

import numpy as np

from sklearn.decomposition import PCA

from sklearn import preprocessing

import matplotlib.pyplot as plt

data = pd.DataFrame(X)

#x1= df[['Price','Living\_space','Lot','Rooms','Bedrooms','Bathrooms','Floors','Garages']]

#print(data.head())

#print(data.shape)

# First center and scale the data

scaled\_data=preprocessing.scale(data)

pca=PCA() #creating a PCA object

pca.fit(scaled\_data) # get PCA coordinates for scaled\_data

pca\_data=pca.transform(scaled\_data) # get PCA coordinates for scaled\_data

# Draw a scree plot and a PCA plot

#The following code constructs the Scree plot

per\_var= np.round(pca.explained\_variance\_ratio\_\* 100, decimals=1)

labels=['PC' +str(x) for x in range(1,len(per\_var)+1)]

plt.bar(x=range(1,len(per\_var)+1),height=per\_var,tick\_label=labels)

plt.ylabel('Percentage of explained variance')

plt.xlabel('Principal complonent')

plt.title('Scree plot')

plt.show()

#The following code constructs the Scree plot

pca\_df = pd.DataFrame(pca\_data, columns=labels)

plt.scatter(pca\_df.PC1, pca\_df.PC2)

plt.title('My PCA Graph')

plt.xlabel('PC1 - {0}%'.format(per\_var[0]))

plt.ylabel('PC2 - {0}%'.format(per\_var[1]))

for sample in pca\_df.index:

plt.annotate(sample, (pca\_df.PC1.loc[sample], pca\_df.PC2.loc[sample]))

plt.show()

## get the name of the top measurements that contribute most to pc1,pc2,pc3.

## Getting the loading scores

for i in range(4):

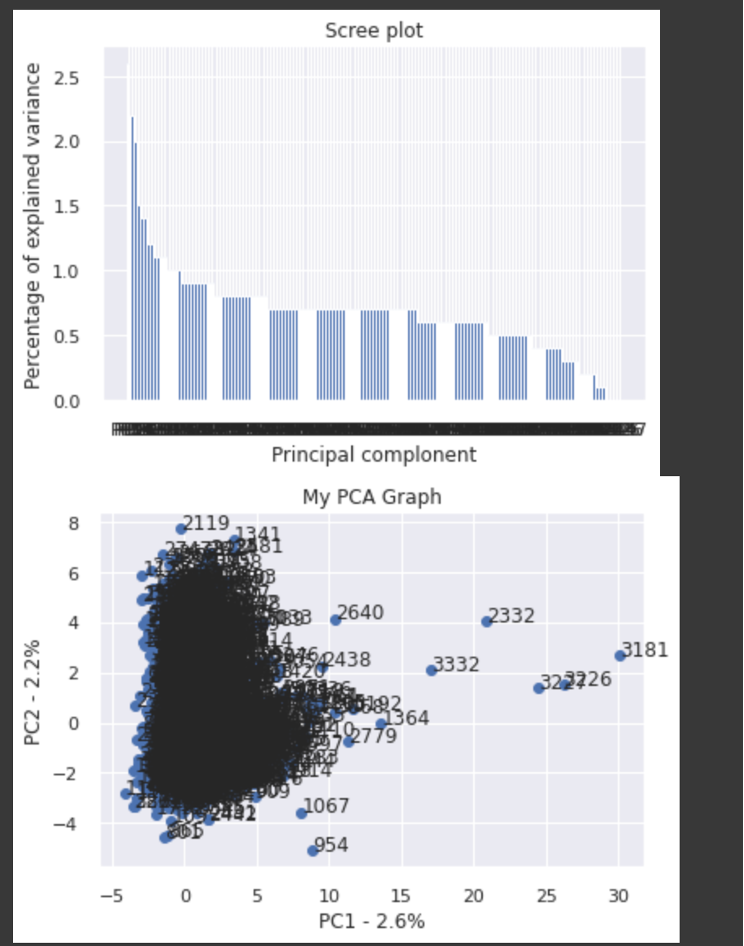
print("PC{}".format(i+1))

loading\_scores = pd.Series(pca.components\_[i])

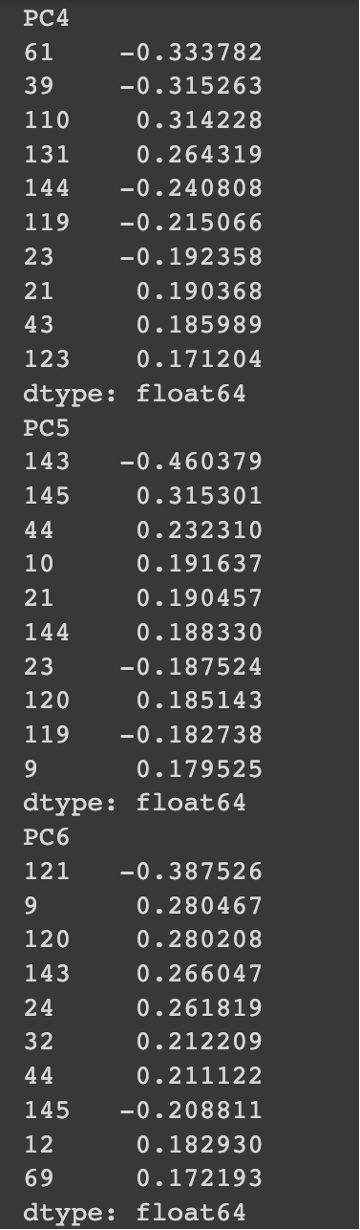
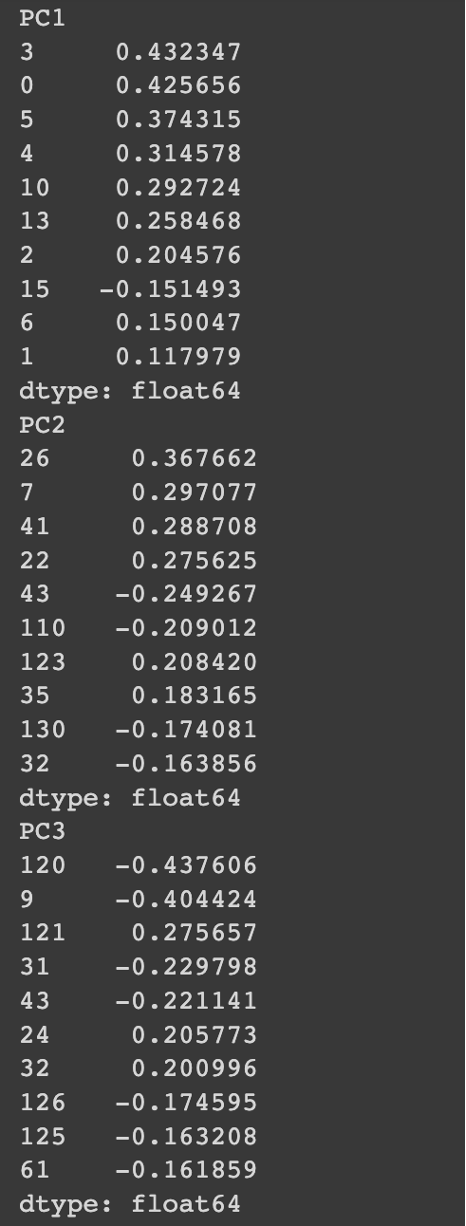
sorted\_loading\_scores=loading\_scores.abs().sort\_values(ascending=False)

top\_10=sorted\_loading\_scores[0:10].index.values

print(loading\_scores[top\_10])



The PC1 and PC2 become the new axes.



So looking at the above scree plots , this has generated 170 principal components. So if we choose 55 components based on the knee bend, giving much less dimension than the original dataset,

The first 2 components capture 2.6 % and 2.2 % variance respectively and the rest vary from 1.5% to 0.5%. This technique doesn’t capture much variance.

And interpreting loadings in each component is a hassle and time consuming, considering 170 variables.

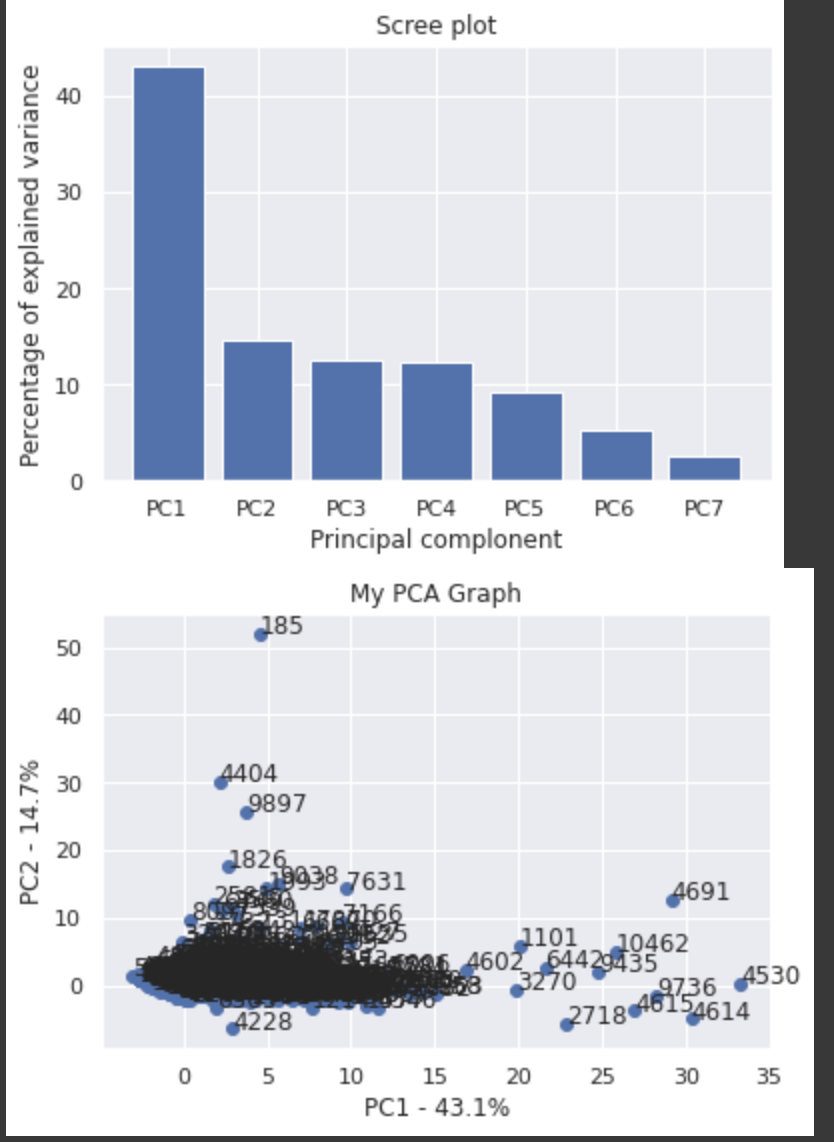
But we achieved a lower dimensional projection of the dataset.

So decided to pursue with numerical variables only and perform feature extraction and find hidden patterns to analyze using this technique.

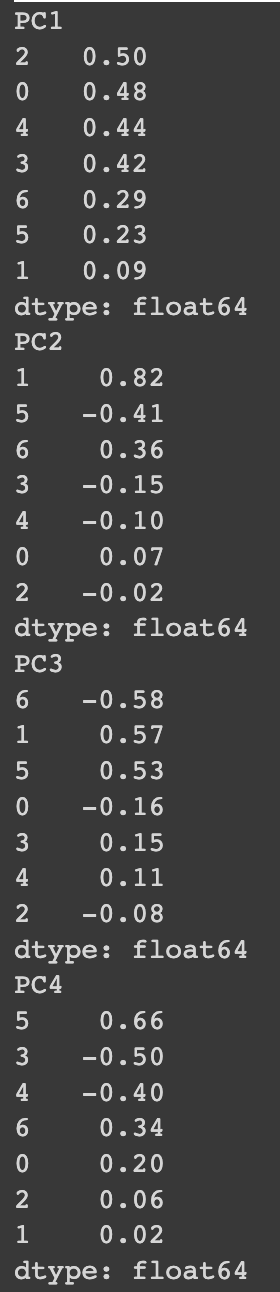
We have done Principal component analysis technique on the dataset that includes numerical variables, Living\_space, Lot , Rooms , Bedrooms, Bathrooms, Floors, Garages. We used sklearn package to import PCA and used Matplotlib to plot the graphs.

Also we first scaled the data. scaled\_data=preprocessing.scale(data), using preprocessing() function from sklearn python package.

The screeplot has the components sorted based on their eigen values with its proportional variance percentages all accumulating to 100%. Based on the screeplot , we can choose 4 principal components based on the bend, capturing 80.5% variance. First component responsible for 43% of variance and PC2 15%, PC3 capturing 11% ,PC4 10% approximately.



The below are the loading scores corresponding to each principal component that are sorted. Each component contains variable loadings that influence each component.



'Living\_space','Lot','Rooms','Bedrooms','Bathrooms','Floors','Garages'[0 1 2 3 4 5 6]

The first PC1 comprises all positive loadings, the most influencing of Rooms, living space, Bathrooms and not so much influence of Lot variable. This can comprise of spacious houses. Typically representing most of the houses.

The second component has the most influencing features of Lot, and least influencing of Floors. The lot variable features dominate this component. This can typically feature outdoor spaces.

The third component has the most influencing features of Lot, Floors and least influencing of Garages.

The fourth component has the most influencing feature of Floors and least influencing of Bedrooms. This might be the unfurnished or the empty floors, that can be perfect for the designers and architects to design on their own liking.

PC1= 0.5(Rooms)+ 0.48(living)+0.44(Bathrooms)+0.42(Bed)+0.29(Garages)+ 0.23(Floors)+0.09(Lot)

PC2= 0.82(Lot)-0.41(Floors)+0.36(Garages)-0.15(Bed)-0.10(Bathrooms)+ 0.07(Living space)-0.02(Rooms)

PC3= -0.58(Garages)+ 0.57(Lot)+0.53(Floors)-0.16(Living space)+0.15(Bedrooms)+ 0.11(Bathrooms)-0.08(Rooms)

PC4= 0.5(Floors)+ 0.48(Bedrooms)+0.44(Bathrooms)+0.42(Garages)+0.29(Livingspace)+ 0.23(Rooms)+0.09(Lot)

This feature extraction helps us to discover patterns that affect the prices,

We found this technique is better than dimensionality reduction to analyze the dataset and its patterns and also capturing 80% of variability of these variables.

**Clustering:**

After Data pre-processing we performed K-means clustering, the k-means algorithm tries to minimize distortion, which is defined as the sum of the squared distances between each observation vector and its dominating centroid. We further used the elbow method to determine the optimal K value.

We tried normalizing data before clustering. Standard scaler, MinMax Scaler and Normalizer were performed separately on the data before clustering. Since we could not get even a single well defined cluster, we decided to perform clustering without normalization. These were the results that we got.

Chart, line chart

Description automatically generated

From the above diagram, the elbow is at K= 3 hence we did clustering taking the K value as 3. We got 3 clusters and as depicted below i.e., PRICE vs CLUSTER:

We can see from the below figure that we got 1 cluster separate considering the sale price aspect which is cluster 0 (red) and it is discussed further below.

Chart, histogram

Description automatically generated

labels= 0-red , 1-green ,2-blue

In the above diagram, we can clearly see that the cluster with green color has the highest SALE PRICE and red has the least. We further went on to find in-depth cluster analysis: Cluster Information:

{No of value in each cluster}

A picture containing text

Description automatically generated

We can see that cluster 0 has the most values and we can consider them as the low-priced houses and cluster 2 as medium-priced and cluster 1 as high priced. Hence because of clustering, we could find out which set of records would fit in which cluster.

Chart, histogram

Description automatically generated

We evaluated clustering results by calculating the silhouette score and it was approximately 67%.