`EDA, Data Manipulation, and Modelling for Machine Learning` Results

Assign	
Status	In progress
Assignment pdf	

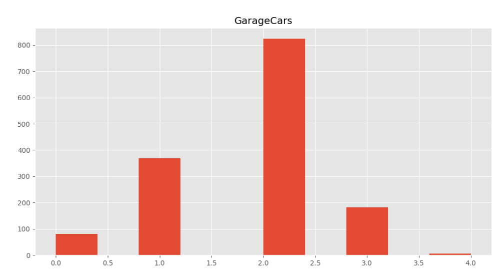
Part 1: Business and Data Understanding

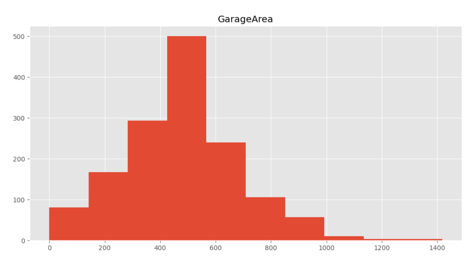
Summary

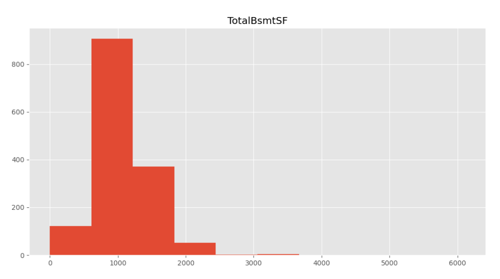
```
SUMMARY STATISTICS
NUMBER OF INSTANCES:  1460
NUMBER OF FEATURES:  81
NUMBER OF CATEGORICAL FEATURES:  55
NUMBER OF NUMERICAL FEATURES:  26
```

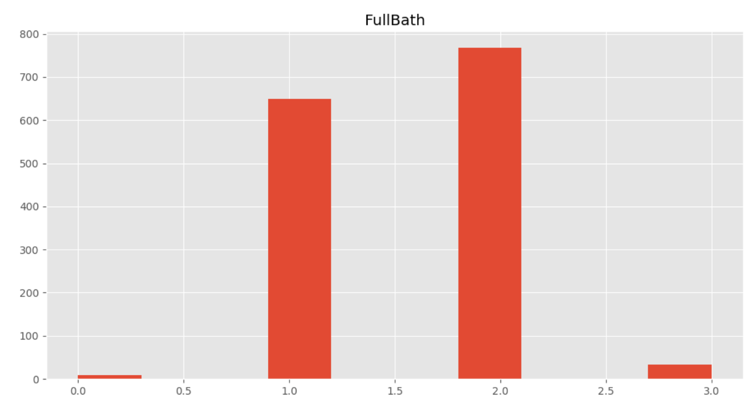
```
Top 5 numerical features

CORRELATION MATRIX
GarageCars      0.640409
GarageArea      0.623431
TotalBsmtSF     0.613581
FullBath        0.560664
TotRmsAbvGrd    0.533723
Name: SalePrice, dtype: float64
```

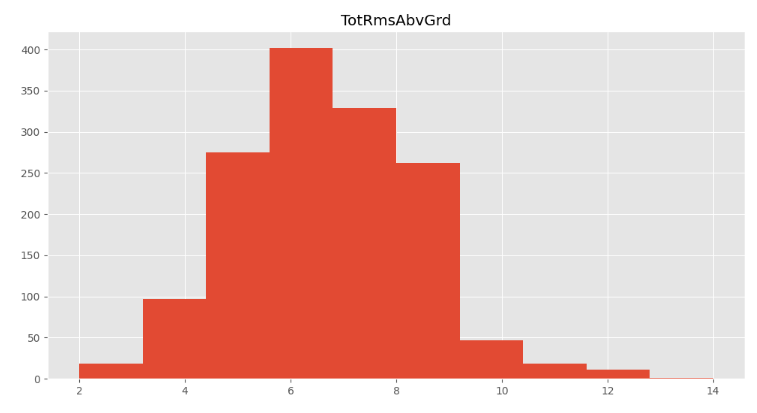
Histograms representing the distributions of these 5 numerical features found and the target variable. These have 10 bins each (one for each feature/variable). A discussion on the shape of their distributions is also below











```
FEATURE DISTRIBUTION
GarageCars skewness:  0.34 - this skewness shows us that this is approximately symmetric
GarageCars kurtosis:  0.22 - this tells us that this has a very long and flat distribution compared with the normal distribution

GarageArea skewness:  0.18 - this skewness shows us that this is approximately symmetric
GarageArea kurtosis:  0.92 - this kurtosis means that its peak is slightly sharper than the normal distribution

TotalBsmtSF skewness:  1.52 - this high positive skewness indicates that this will have longer right tails than the normal distribution
TotalBsmtSF kurtosis:  13.25 - this high level of kurtosis tells us that this has a very high and sharp and central peak

FullBath skewness:  0.04 - this skewness shows us that this is approximately symmetric
FullBath kurtosis:  0.86 - this kurtosis means that its peak is slightly sharper than the normal distribution

TotRmsAbvGrd skewness:  0.68 - this shows us the that this is slightly skewed to the right compared to the normal distribution
TotRmsAbvGrd kurtosis:  0.88 - this kurtosis means that its peak is slightly sharper than the normal distribution
```

Missing Data

Out of the 80 features we can see that 19 (or ~23%) contain empty values. The total amount of missing data equates to 6965 cells, or ~6% of our total data. This is actually pretty good as a rate of 15-20% is to be expected - whereas outs is around the generally accepted benchmark of 5% which means that we could, in theory, just fill out these fields with constants (either median or 0) when preprocessing. However, missing data can be for a variety of reasons beyond simply poor data quality so we are going to do a bit more of an in depth analysis (as this will be useful for the preprocessing task later)

```
NUMBER OF FEATURES WITH MISSING VALUES: 19
PERCENTAGE OF FEATURES WITH MISSING DATA:  23.46 %
TOTAL AMOUNT OF MISSING DATA:  6965
PERCENTAGE OF TOTAL DATA MISSING:  5.89 %

The following features have a high (>50%) proportion of missing features
 PoolQC         0.995
MiscFeature    0.963
Alley          0.938
Fence          0.808
dtype: float64
```

When we preprocess we can fill empty fields with the median value of their respective columns so that it can be read by our algorithms and don't sway the algorithm too much, however this is only reliable when small amounts of data (sub 5%) is missing so we need do look at what proportion of each field is missing. As can be seen above (only showing fields missing more than 50% of their data) a number of fields are missing almost all fields. On review. a number of these fields appear to be missing data simply because it is not available, such as if the house does not possess a pool.

While beyond the scope of this question, For these Values we will later fill the empty cells with 0's instead of the medians. After these fields have been filled with 0's and the data has been normalised, we can review the correspondence values again and compare it to the values that previously contained missing fields. if they have both missing fields and low correspondence, they will simply be dropped.

Objectives

Discussion on how this data could be applied to business goals and from there converted into machine learning paradigms.

Two possible business questions from this data could be "**what factors affect the house price?**" and "**how do these factors affect the house price?"/"in which way do the factors affect the house price?**".

As data science goals these could be interpreted as

Goal 1: Identify and select the features of most significance in determining output

and

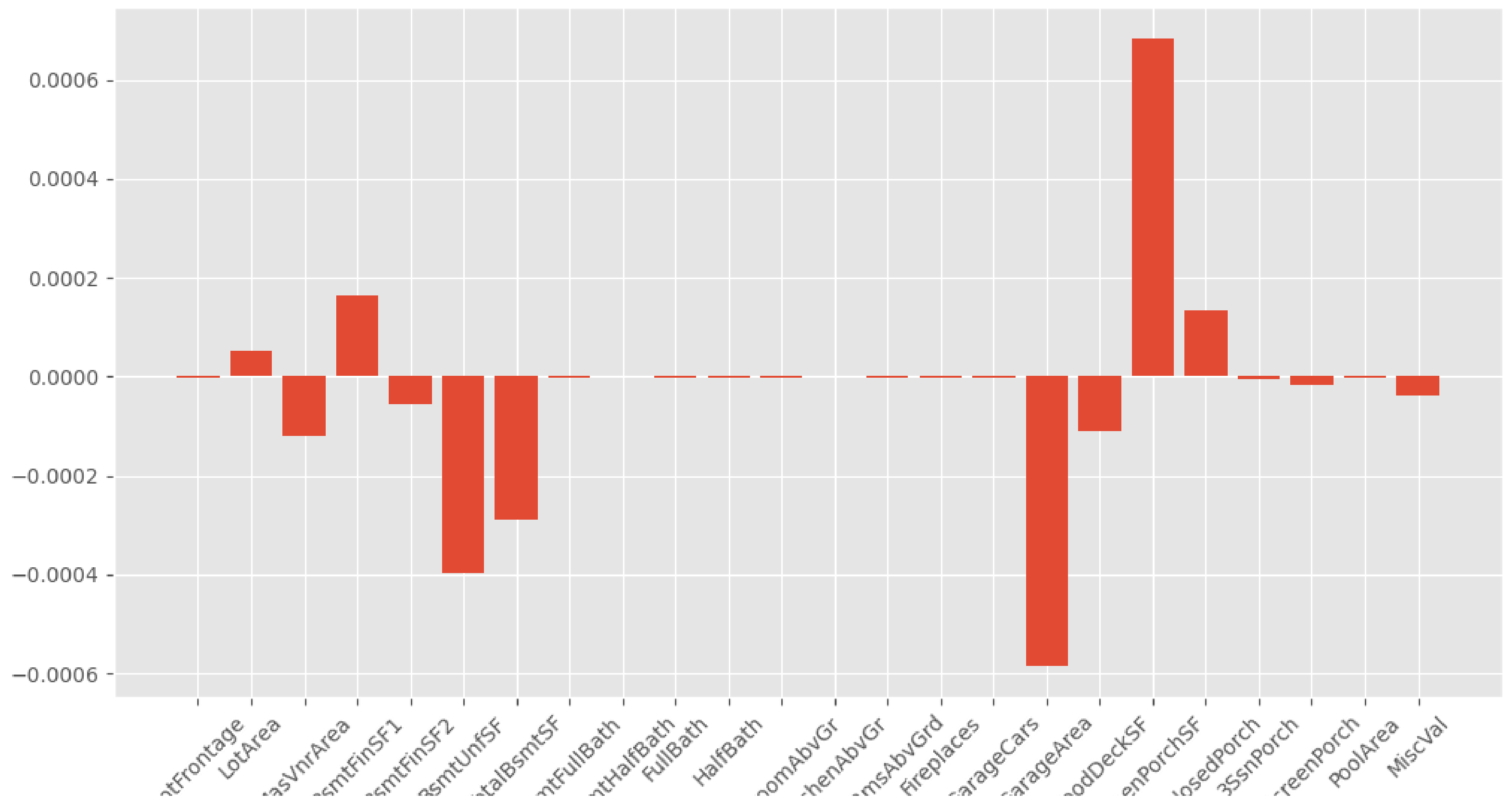
Goal 2: Identify and Extract features that have the least significance in determining output

This lends itself naturally to a Classification or Regression problem where both can be used to determine feature importance and, therefore, which features are the most and least significant.

Using logistic Regression I was able to identify the top features, the best of which are shown below:

```
FEATURE IMPORTANCE
Feature OpenPorchSF: 0.000683849111316386
Feature BsmtFinSF1: 0.0001642364119697
Feature EnclosedPorch: 0.00013303781325773228
Feature LotArea: 5.11548379260352e-05
Feature KitchenAbvGr: 2.1845269557116738e-08
```

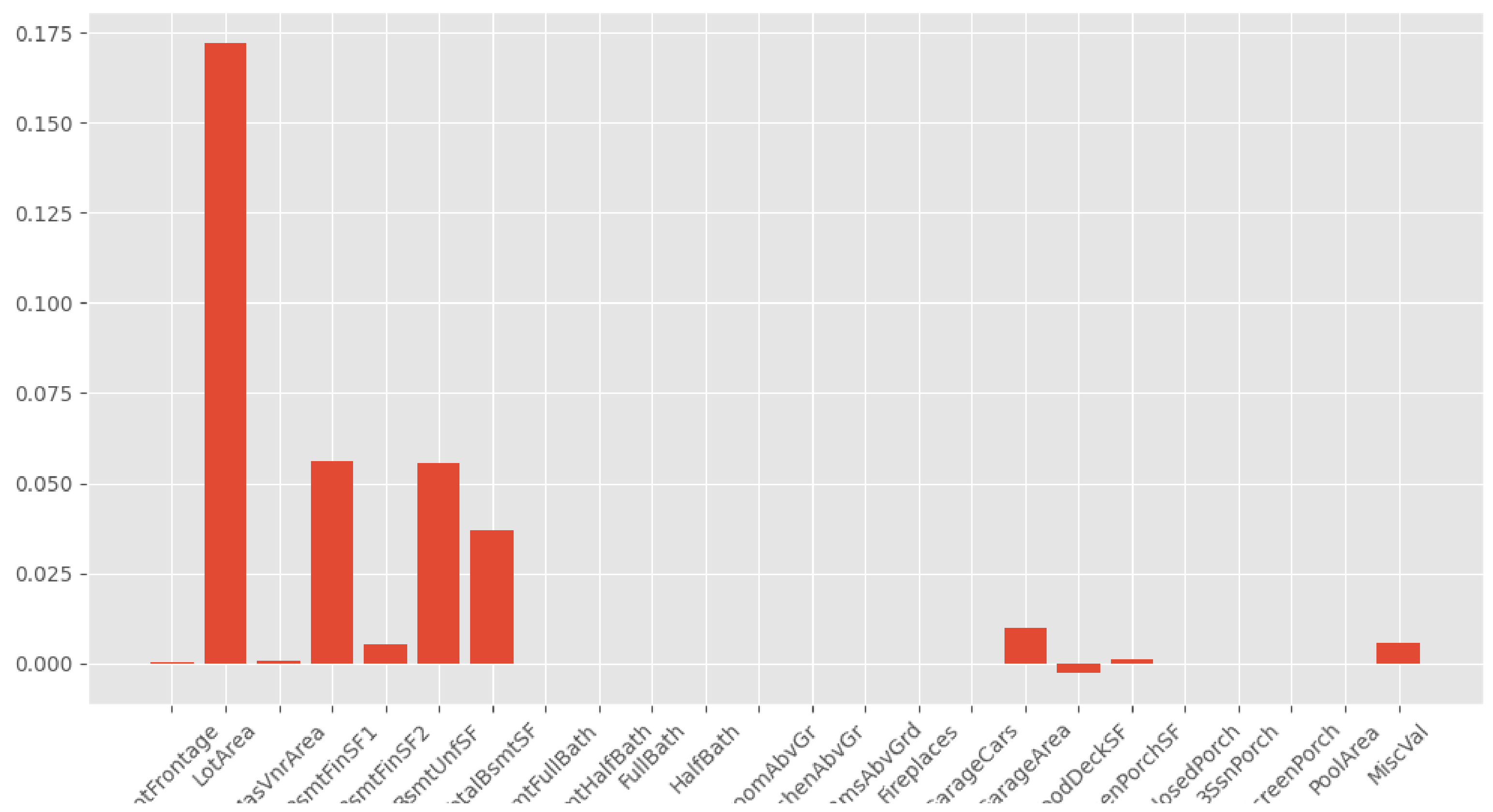
This can be compared to the features we identified earlier in terms of correspondence and can be represented on the attached graph:



I also used Classification to determine feature importance which shows slightly different results:

```
PERMUTATION CLASSIFICATION
Feature LotArea: 0.1721232876712329
Feature BsmtFinSF1: 0.055958904109589036
Feature BsmtUnfSF: 0.05568493150684932
Feature TotalBsmtSF: 0.03678082191780822
Feature GarageArea: 0.010000000000000009
Feature MiscVal: 0.005616438356164388
Feature BsmtFinSF2: 0.005136986301369867
Feature OpenPorchSF: 0.0010273972602739712
Feature MasVnrArea: 0.0008219178082191747
Feature LotFrontage: 0.00013698630136986246
Feature ScreenPorch: 6.849315068493123e-05
LotFrontage: 0.00013698630136986246
LotArea: 0.1721232876712329
MasVnrArea: 0.0008219178082191747
BsmtFinSF1: 0.055958904109589036
BsmtFinSF2: 0.005136986301369867
BsmtUnfSF: 0.05568493150684932
TotalBsmtSF: 0.03678082191780822
BsmtFullBath: 0.0
BsmtHalfBath: 0.0
FullBath: 0.0
HalfBath: 0.0
BedroomAbvGr: 0.0
KitchenAbvGr: 0.0
TotRmsAbvGrd: 0.0
Fireplaces: 0.0
GarageCars: 0.0
GarageArea: 0.010000000000000009
WoodDeckSF: -0.002602739726027395
OpenPorchSF: 0.0010273972602739712
EnclosedPorch: -0.00013698630136986246
3SsnPorch: -6.849315068493123e-05
ScreenPorch: 6.849315068493123e-05
PoolArea: 0.0
MiscVal: 0.005616438356164388
```

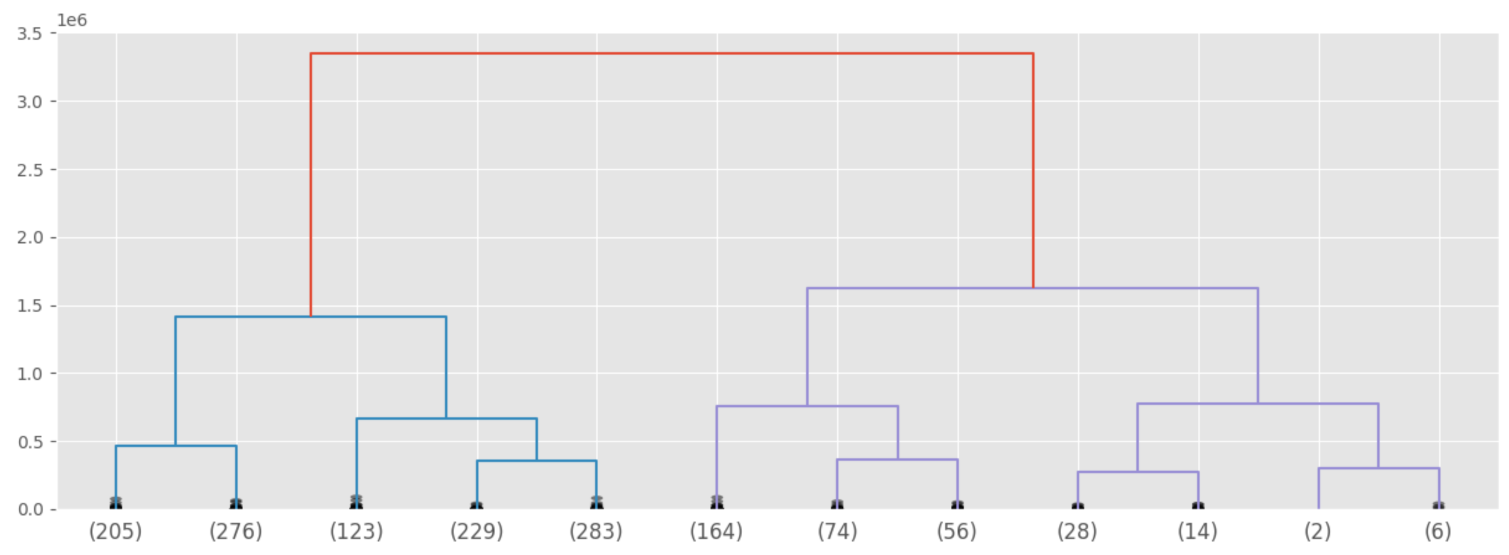
and is represented as follows:



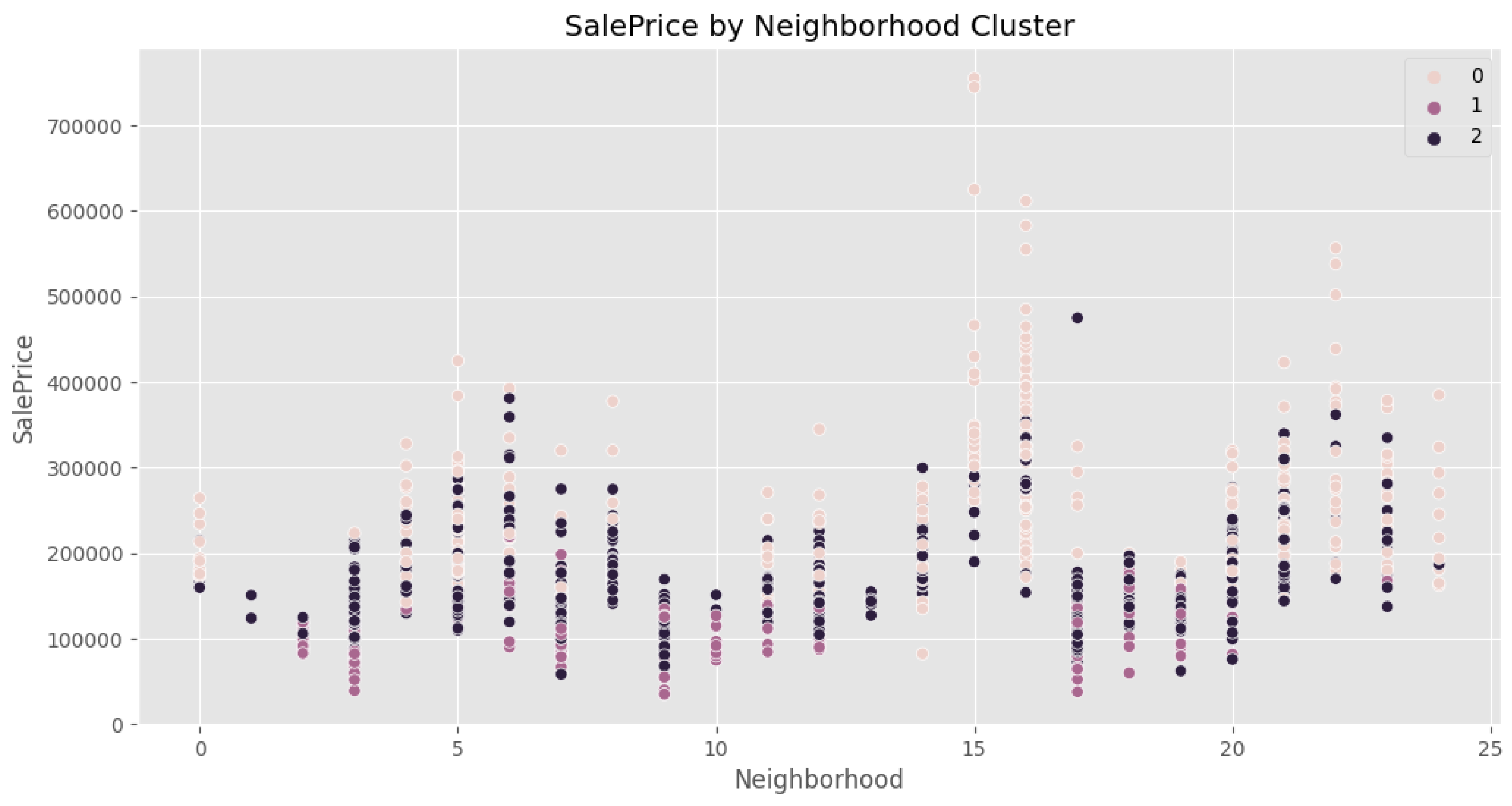
Here we can see that the majority of these features actually have very little to no impact on the prediction and so will be removed in Part 2.

Hierarchical Clustering for EDA

Hierarchical clustering helps us identify how closely related different attributes are, and in the case of our top 5 features, can be represented as below:

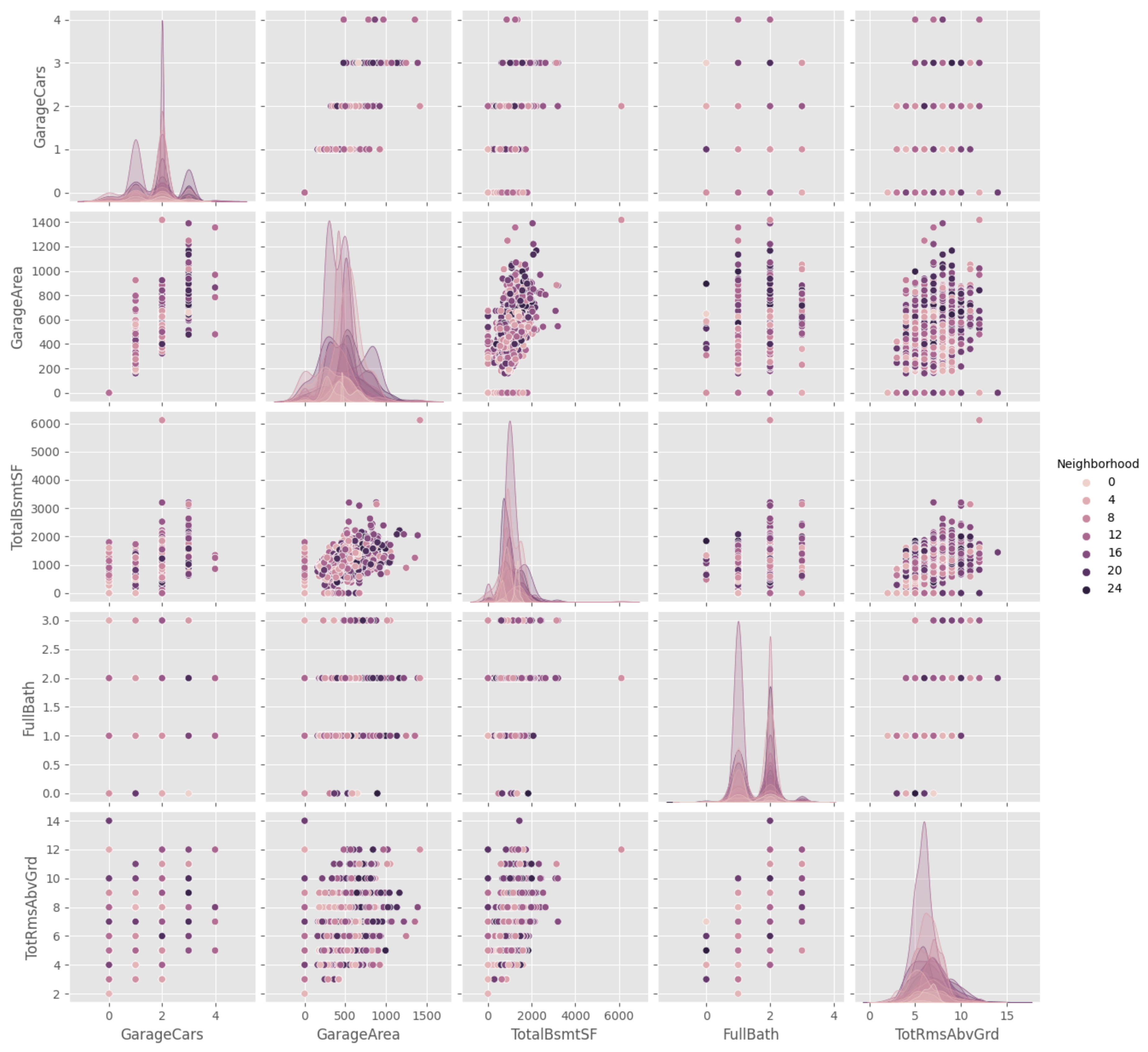


We can then use it to answer interesting questions, such as whether the houseprice varies by neighborhood. With this in mind, we can break down Salesprice by which neighborhood cluster it is a part of:



Which shows us that most of these high selling Neighborhoods were in the same clusters.

And, for good measure, a pair plot of how Every feature impacts every other feature (clustered by neighborhood) as seen here:



From which we can determine Neighborhood has a significant impact on our top 5 features, which suggests that neighborhood is a significant indicator of SalesPrice.

Part 2: Data Preparation and Machine Learning

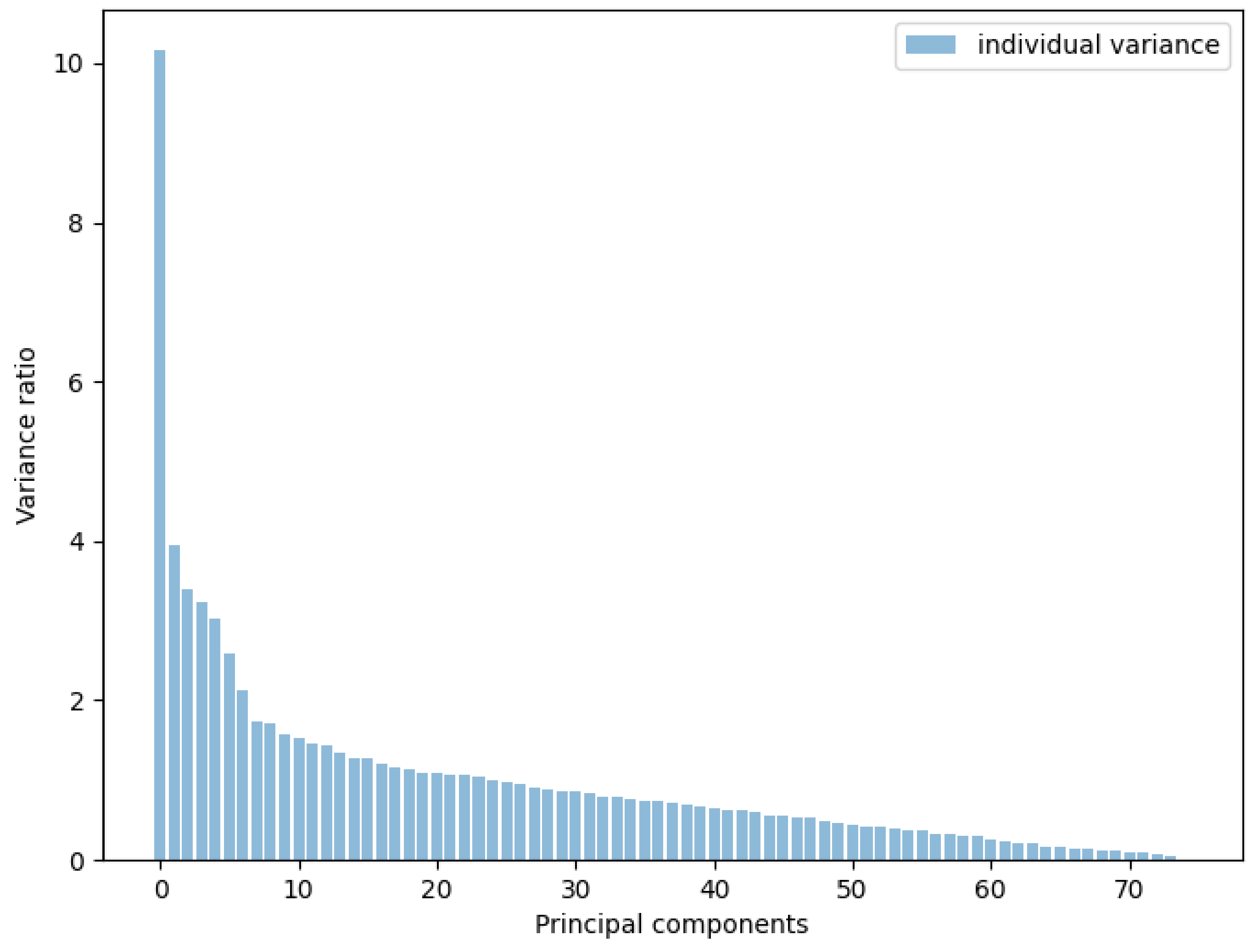
Preprocessing

Based on the Data analysis performed above we know we need to perform the following preprocessing steps

1. The first step was to strip out the metadata (Id) from from the dataset.
2. Following this, columns identified to be possessing a high proportion of missing fields were removed, after which the same was performed for rows.
3. The remaining null values were then filled with 0 values
4. The numeric data was then normalized and the categorical data transformed into numerical
5. Duplicate data and outliers were then dropped
6. the data was then standardized
7. What was returned was this data split into it's body and target components

After this we can use 2 dimensionality reduction tools (**Principal component analysis** and **Locally Linear Embedding)** to identify which features are irrelevant and/or redundant to predicting SalesPrice.

I used Principal component analysis to perform a variance analysis on the features which shows a sharp drop off in the impact of these features. This tells us that when we perform our dimensionality reduction we can likely reduce the number of features by 50% without any noticeable drop off in accuracy.



Next, using Isomap and LocallyLinearEmbedding, I performed the same analysis, reducing the number of features to just 5. As we can see from the above graph that there is a sharp drop off in variance score with a long tail.

A comparison of ML models

With the two groups of features selected from the analysis performed, I used the ordinary linear regression and ridge regression (with alpha=0.5) to predict the house prices. I also applied a Random Forest Regeression (a more powerful ensemble regression method) model to this task. Comparing the the mean squared errors generated by each model on the training and the test sets we see the following:

```
Initial Accuracy (negative mean squared errors):
Accuracy Linear Regression: -3.4496812505592683e+34
Accuracy Ridge Regression: -931,122,448.4451574
Accuracy Random Forest Regression: -1,034,678,906.8430752

Isomap Accuracy (negative mean squared errors):
Accuracy Linear Regression: -2,485,312,067.1863275
Accuracy Ridge Regression: -2,485,310,906.604135
Accuracy Random Forest Regression: -2,184,652,053.621549

LLE Accuracy (negative mean squared errors):
Accuracy Linear Regression: -5,816,260,985.733909
Accuracy Ridge Regression: -6,118,144,304.606129
Accuracy Random Forest Regression: -5,061,896,356.4454775
```

When performing Linear Regression on the original dataset, it performs abysmally, especially when compared to both the Isomap and LLE datasets. However, this is completely turned on its head for Ridge Regression which saw no noticeable improvements over the original dataset, and in fact, especially in the case of LLE, performed significantly worse. The same is true for the Random Forest Regression.

Out of all of these methods it seems that Random Forest performs best on average, unsurprisingly.

Part 3: Further Analysis

For further comparison I used both Logistic Regression and Gaussian Regression models for a point of difference. However, none of these techniques seemed able to outperform the Random Forest. The compared performance is shown below:

```
Initial Accuracy (negative mean squared errors):
Accuracy Logistic Regression Regression: -2,457,316,838.5215945
Accuracy Gaussian Process Regression Regression: -39,289,943,447.41868

Isomap Accuracy (negative mean squared errors):
Accuracy Logistic Regression : -3,049,541,192.5112247
Accuracy Gaussian Process Regression : -39,289,943,457.962105

LLE Accuracy (negative mean squared errors):
Accuracy Logistic Regression : -8,341,131,187.571087
Accuracy Gaussian Process Regression : -39,289,943,611.02766
```

Interstingly Gaussian regression stands out as particularly bad, however has very little to no variance in its result. This leads me to believe that through some tinkering with kernels (about which I understand very little), it could perform very well.