1) Describe your data set, why you chose it, and what you are trying to predict with it

Before choosing my dataset, I had the idea of predicting cryptocurrency prices, but it was advised that times series prediction was out of the scope of this course. Then, I looked into predicting Inspection Scores from Restaurants in Manhattan, and again, it was not a good choice because the dataset did not have enough features to come up with a good analysis. Finally, I took the professor's recommendation to work with the dataset from StreetEasy about Rental Prediction in Manhattan. This one was a good choice because it has enough features, and it is large enough to run the ML models introduced in the class. Also, I decided to work with it because I was curious about the fluctuations in the rent prices in Manhattan. The dataset is composed of 3,539 rows and 18 columns, and my target variable is predicting the rent prices in Manhattan.

```
manhattan = pd.read_csv('manhattan.csv')
manhattan.shape
#below we can see our dataset has 3539 rows and 18 columns
```

```
(3539, 18)
```

2) Detail what you did to clean your data and any changes in the data representation that you applied. Discuss any challenges that arose during this process.

Before proceeding with the data cleaning, I split the dataset into the training and test set as indicated in the assignment instructions. I set the training dataset to have 66% of the data, and the test dataset to have 33%. My "y" variable as I mentioned before was the "rent", and my "X" was the everything, but "rent" since this variable was dropped because it is the target variable.

Then, I used pandas.isnull().sum() to find the null values. Luckily, this dataset did not have any nulls values, and I did not have to use IMPUTER in scikilearn to value replacement. In the pictures below, we can see the features of the dataset without any null values.

```
X_train_cleaned.isnull().sum() # not missing values
```

```
rental_id           0
bedrooms            0
bathrooms           0
size_sqft           0
min_to_subway       0
floor               0
building_age_yrs    0
no_fee              0
has_roofdeck        0
has_washer_dryer    0
has_doorman         0
has_elevator        0
has_dishwasher      0
has_patio           0
has_gym             0
neighborhood        0
borough             0
dtype: int64
```

```
X_train.info() #using info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2123 entries, 1491 to 2732
Data columns (total 17 columns):
rental_id           2123 non-null int64
bedrooms            2123 non-null float64
bathrooms           2123 non-null int64
size_sqft           2123 non-null int64
min_to_subway       2123 non-null int64
floor               2123 non-null float64
building_age_yrs    2123 non-null int64
no_fee              2123 non-null int64
has_roofdeck        2123 non-null int64
has_washer_dryer    2123 non-null int64
has_doorman         2123 non-null int64
has_elevator        2123 non-null int64
has_dishwasher      2123 non-null int64
has_patio           2123 non-null int64
has_gym             2123 non-null int64
neighborhood        2123 non-null object
borough             2123 non-null object
dtypes: float64(2), int64(13), object(2)
```

After looking at the shape of the data, I decided to drop the "borough" feature since the entire dataset is about Manhattan. I dropped this column from both my training and test sets. Another feature I needed to modify was "neighborhood". When splitting the dataset, the value count of some neighborhoods was uneven. To solve this issue, I removed the neighborhoods with less than 5 values and that worked out fine. Also, I used pd.get_dummies() as a work around to eliminate the string values of the neighborhoods and break them down into valid categories. Another important step in this data exploration was using pandas.describe() to get the summary statistics of the data. It was another way to confirm that all my columns had the same number of count values. Then, I looked into the mean values of important features

```
X_train.describe() #using describe()
```

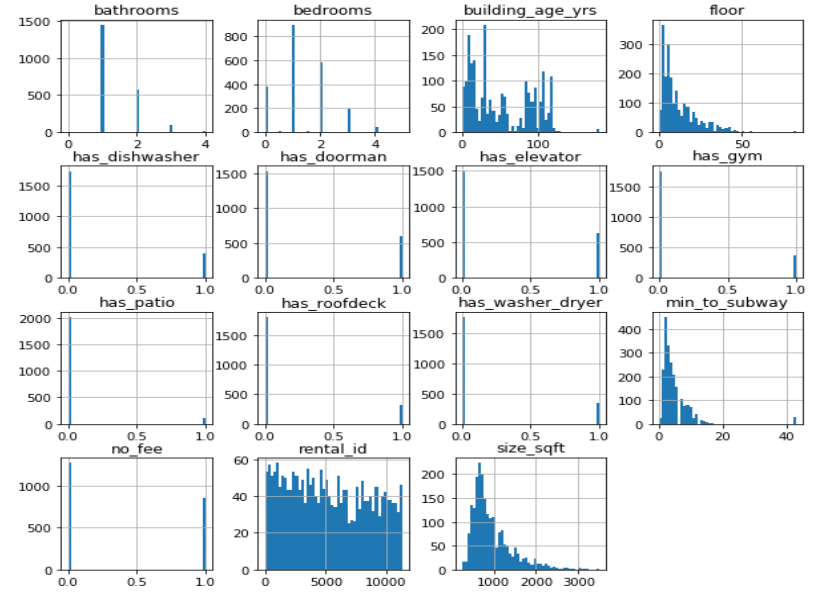
	rental_id	bedrooms	bathrooms	size_sqft	min_to_subway	floor	b
count	2123.000000	2123.000000	2123.000000	2123.000000	2123.000000	2123.000000	
mean	5280.276967	1.357984	1.375412	944.876119	4.879416	11.898022	
std	3316.279919	0.966555	0.603866	472.785072	5.285583	11.069380	

such as "bedrooms", "bathrooms", "has_elevator", "has_gym", and others because by intuition I thought they might be critical features influencing the target variable in the dataset.

To my surprise, the mean number of bedrooms and bathrooms was very close with 1.35 for bedrooms and 1.37 for bathrooms. With this information we can assume most apartments in Manhattan have 1 and 2 bedrooms and bathrooms. Also, it was interesting to see that the mean number for "min_to_subway" was 4.87. This means most apartments in Manhattan are relatively close to subway stations.

3) Discus what you learned by visualizing your data

```
X_train_cleaned.hist(bins=50, figsize=(10, 10))
plt.show() #histograms
```

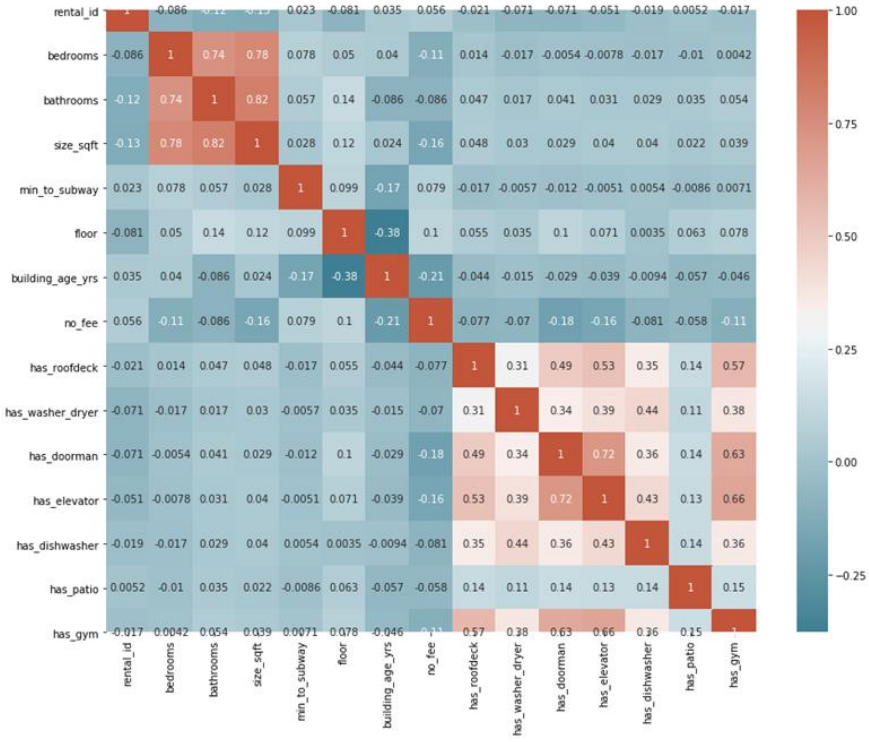


In the picture, we can observe how the data looks using pandas.hist():

Here, I could confirm my assumptions regarding the number of bedrooms and bathrooms in the dataset. According to the graph, most apartments have 1 and 2 bedrooms, as the same with the

number of bathrooms. Looking at the "size_sqft" histogram we can see the data is positive skew to the right, and most apartments are around 1000 sqft with a mean value of 944. Also, it is surprising to see that the frequency of "building_age_yrs" is very high close to 100. We have around 25% of the dataset with buildings going over 100 years old where the mean value for this feature is 51. Another important fact to mention, it is that very few apartments have gyms, elevators, doormen, patios, or dishwashers. Which makes sense because most of the apartments are relatively old. Years ago, having this type of luxury was not a priority since it can be a factor in reducing the usable living space for the apartments.

I printed a correlation matrix to explore the significant levels among the features and



the result was very intuitive. "Bedrooms", "bathrooms", and "size_sqft" were highly correlated among each other, which makes sense because the size will influence in the number of bedrooms and bathrooms in the apartments. Also, surprisingly "has_doorman" and "has_elevator" had a high correlation together with "has_gym" and "has_elevator".

Beside using the correlation matrix and the histogram, I also used the scatter_matrix from pandas, and the result was similar to the conclusions already mentioned above.

4) Describe your experiments with the two supervised learning algorithms you chose. This should include a brief description, in your own words, of what the algorithms do and the parameters that you adjusted. You should also report the relative performance of the algorithms on predicting your target attribute, reflecting on the reasons for any differences in performance between models and parameter settings.

The 2 supervised machine learning models I chose are the ***decision tree regressor*** and the ***lasso regression***. I will start by defining how the ***decision tree regressor*** works. Decision trees can either by used for classification and regression problems. The methodology behind it works similar to an if/else statement where the "decision nodes" are the questions and the "leaves" are the answers. Then, the data is classified in groups and the algorithm learns from these decisions and makes the group classifications. I decided to use the decision tree regressor is because of its main advantages. It is very easy to understand by non-experts analyst like myself, and by just picking one of the parameters such as max_depth, max_leaf_nodes, or min_samples_leaf it is sufficient to prevent overfitting. It is worth to mention that one of the disadvantages of this model is that it can result in poor generalizing performance even when using "pre-pruning" or choosing parameters.

To find the best parameter , in this case I picked the max_depth, I used the grid search from the sklearn library with my training set. The result was the best max_depth parameter was 4 and the best R2 was 0.75. The R2 is the metric that indicates the variance of the target variable in your model, if R2 the is close to 1, that means this it is a good model to explain the variance of your data. When using the best parameter in my test set, in this case 4, I got an R2 score of

0.77. To test the differences, I ran the model again with a max_depth of 3 and the R2 was 0.75, lower than in the previous model. This result confirms the accuracy of the grid search result. Overall, the decision tree model was not too bad in predicting the variance of my target variable with a 77% of accuracy.

As I mentioned previously, the other supervised machine learning model I used was the ***lasso regression***. I chose ***lasso*** because among the others, linear and ridge regression, I thought it was the best fit for this experiment. Linear regression is not a good fit because there are high chances of overfitting the model. It is recommended to use either ridge or lasso regression instead. The difference between ridge and lasso is the type of regularization. Ridge regression uses L2 regularization, which means the coefficients get as small as possible close to zero, but not exactly zero. In the other hand, lasso regression uses regularization L1, which mean it can reduce the coefficients to be exactly zero, by looking for the optimal or more important features in the model omitting the features that are useless or irrelevant to improve performance of the model. In terms of parameters, when using regression, we look for the optimal alpha. The optimal alpha will vary depending on the type of dataset you are working on. As I did with the previous model, I used the grid search to find the optimal alpha. I used an array with values from 0 to 5 and, and the best alpha resulted to be 1 and best R2 score was 0.81. Another important parameter is adjusting the max_iter value. When increasing the alpha, the max_iter should be reduced, and when decreasing the alpha, it should be increased. When running the model with an alpha of 1, my score value R2 value was 0.82 for both training and testing sets. Surprisingly, I ran the model with other different alphas and max_iter, and the R2 score was again 0.82 for

both sets which I found to be very strange. Overall, the lasso regression was a relatively good model predicting the variance of my target variable with 82% of accuracy.

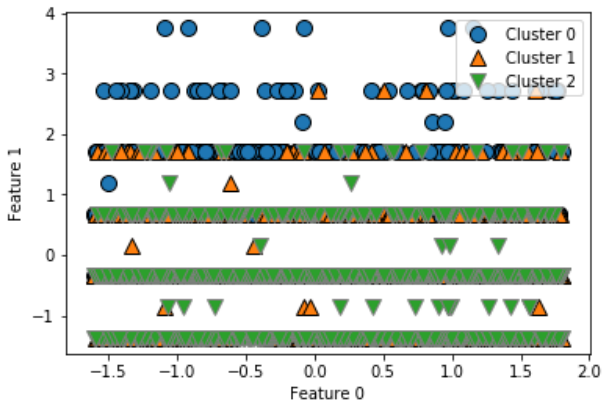
5) Describe your experiments using PCA for feature selection, discussing whether it improved any of your results with your best performing supervised learning algorithm.

Principal Component analysis (PCA) is an unsupervised method that can be used when the objective is to reduce the dimensionality of the data to improve the model performance. Another advantage is that visualizations are easy to interpret because the number of features is reduced when setting parameters. An example of this, it is when using feature selection to find the set number of principal components. For example, when using the cancer dataset, we set our model find the 2 main principal components which turn out to be if the cancer is malignant or benign.

For my experiment, I ran PCA in scaled and unscaled data to compare if it would improve the result of my lasso and decision tree regressor models. For the ***lasso regression***, when using PCA on the unscaled training and test sets the R2 score was 0.82 for both. For the scaled data, the result was very similar. The training set R2 score was 0.82 and the test set was 0.81, which was a very minimal change. When trying with the ***decision tree regressor,*** the result was pretty much the same for both, unscaled and scale data, with training R2 score of 0.80 and a test R2 score of 0.77**.** Although, the PCA did not make a significant impact in the scores, its wort to mention that after running PCA looking to capture 95% of the variance, the number of features went down from 42 to 35 reducing the number of features by 7. That was the only noticeable change in the result of using PCA for feature selection.

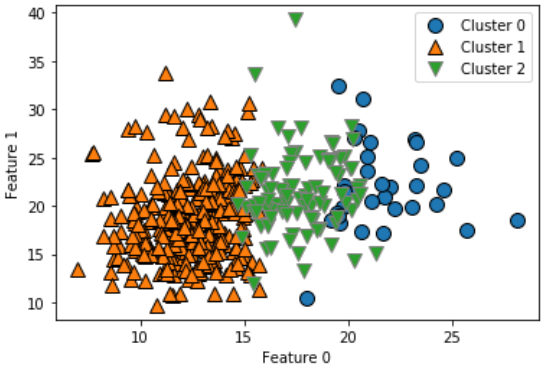
6) Discuss the results of using PCA as a pre-processing step for clustering. This should include a brief description, in your own words, of what the algorithms do. If you used the Wine dataset for this, briefly explain why your original dataset was not appropriate.

When trying to work with the Manhattan dataset, my result was not good enough because continuous data does not work well for clustering. The picture below shows the result of the attempt of using k-means with this dataset:



Breast Cancer Dataset

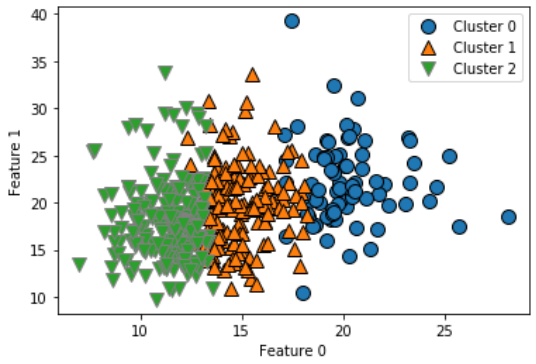
Working with the breast cancer dataset worked out well for this step. To compare the result of PCA as a pre-processing step for clustering, I split my sets into 20% for test and 80% for training. Clustering is mostly used as an initial step for exploratory data analyst and/ or data pre-processing. The first clustering method I used was k-means. K-means is one of the simplest unsupervised learning algorithms. How it works, "k" is defined which refers to the number of clusters in which you want to divide the data. The "mean" refers to the average of the data points which form the boundaries defining the cluster centers. To define the number of clusters,



Unprocessed Dataset

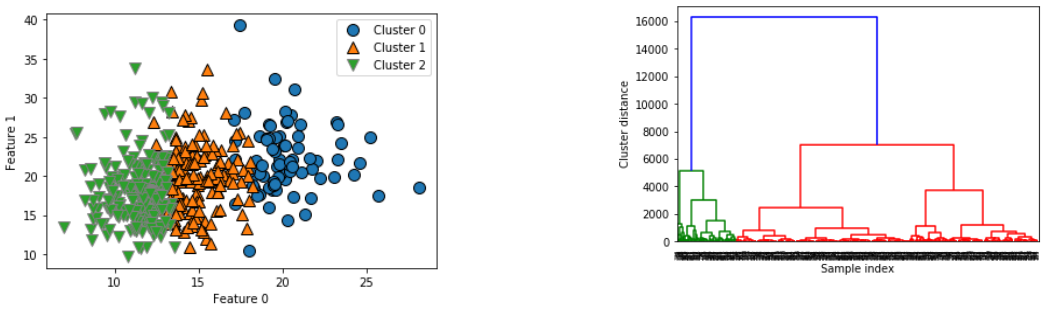
I followed to suggestion of the elbow visualization and assigned "n_clusters = 3". When running the model on the unprocessed data, I was able to generate clear clusters with an adjusted ran index (ARI) score of 0.54 and a Silhouette score of 0.33. These numbers are not very good because are very far from zero. This means our clusters are somewhat unrelated.

For the PCA processed dataset the clusters were in different positions. Cluster 1 is in the middle, cluster 2 is the first group, and cluster 0 is less compact. It is a fact that it can be challenging interpreting the meaning of these clusters. When trying to get the ARI score using the processed data, I got some errors. Sadly, I was not able to figure out the reason causing this error.

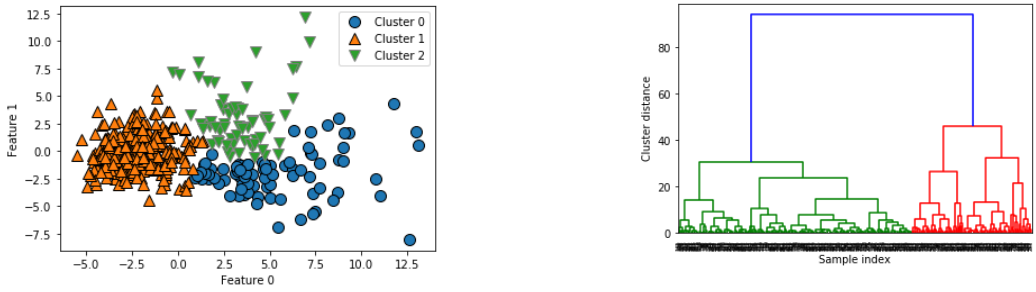


PCA Processed Dataset

The second clustering algorithm I used was ***agglomerative clustering***. This clustering mechanism finds points of data that are closest to each other, and successively groups them together. As with k-means, we need to specify the number of clusters we would like to generates. Agglomerative clustering is hierarchical because it performs operations sequentially. It would be a good choice when trying to make decisions about how well you want to group your data. When running the algorithm with unprocessed and process data the result was different as well, as the same with k-means. The order of the clusters changes, but in this case, I was able to generate the hierarchical visualization to see the ordering of the clusters.



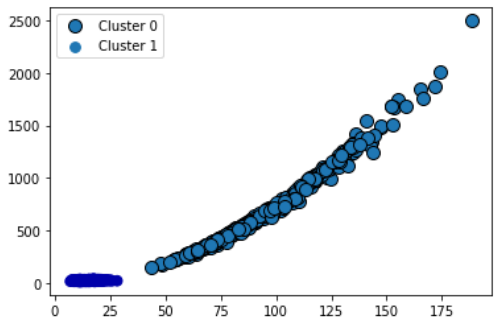
Unprocessed Dataset



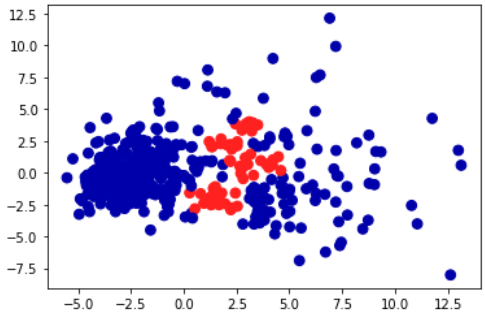
Processed Dataset

In the pictures we can observe how the clusters are split into 1 and 2 groups in the hierarchical visualization, but the scaling and order of the data point is very different. The ARI score for the unprocessed data was 0.42 and the silhouette score was 0.34. These number are lower than in the k-mean experiment, and they are not very good because the are far from 1 indicating our clusters are not very compact. I was not able to generate ARI and silhouette scores for the processed data.

The third clustering algorithm I used was the **DBSCAN**. This is a clustering method that is used to separate clusters of high density from clusters of low density. In this instance, we do not have to specify the number of clusters as is needed for k-means and agglomerative clustering. Personally, I found this one to be the most difficult to work with because I was not able to get good clusters.



Unprocessed Data



Processed Data

In the pictures we can observe the algorithm works better with the processed data, the clusters are not very clear in both cases. The ARI score for the unprocessed data was zero, meaning our clusters are highly unrelated. I was not able to generate ARI score for the processed dataset. For some unknown reason, I was not able to run the silhouette score for neither of the dataset. All these issues are the reason why I find this clustering algorithm very difficult to interpret. Even when trying to plot different features did not help me to come up with a better result.

7) Summarize what you learned across the three projects, including what you think worked and what you would do differently if you had to do it over.

I learned a lot while doing this deep exploratory data analysis experiment. I learned that one of the most important aspects before running any ML algorithm is to the make sure your data is in a good shape. Having messy data can make the entire analysis very inaccurate and it can produce bias results. If I had to do this project over again, I would choose a different dataset with better features for classification. Regression was interesting, but I think a dataset with more categorical features would have given me more options to run models in both supervised and unsupervised learning. I found this class to be very interesting, but I think I would have learned

more by taking it in a regular semester. I think having basic statistical knowledge should be required before taking this class. Sometimes, it can feel like you are guessing if you do not understand the meaning of certain key statistical concepts. Overall, I feel I took advantage of this class during the summer term, and I feel ready to keep expanding my ML knowledge at my own pace from now on.