Data Analytics Report and Executive Summary

Matthew E. Heino

Data Analytics Graduate Capstone

Background

This document contains information about the degree capstone. Information contained in this document will cover various areas of the capstone project. This document will include a discussion of the research question. There will be a discussion about how the data was created or acquired. There will be a discussion of the outcomes and an analysis of what was encountered during the process of analysis and how this analysis can be used to aid in business decisions.

Research Question

The research question that will explored in this assessment will be, "To what extent does the extent that latitude and longitude of an apartment affect the price?" The business question is what areas of the country exhibit the highest mean in apartment prices. Is there a particular cluster of apartments, based on latitude and longitude, that have a higher-than-average price? Is it possible to cluster these apartments in a reliable manner using a clustering algorithm that is quick and easy to implement?

The overall hypothesis for the question would be the following:

- **Null hypothesis:** Using latitude and longitude as clustering features is it possible to produce a cluster that has a silhouette score above .60.
- **Alternate Hypothesis:** Using latitude and longitude as clustering features it is not possible to produce a cluster that has a silhouette score above .60.

The research question when taken in context is to better understand how the geographic location of the apartment plays a part in the price that the apartment fetches on the market. These findings could be interesting based on how the observations are clustered using the algorithm that will be discussed in a subsequent section of the paper. These observations can be easily

observed by looking at the statistics of the cluster. These simple statistics can help to understand the cluster and the average price as well as look at the outliers that were included within the groups.

Data Collection

The data that will be used for analysis will come from the UC Irvine Machine Learning Repository. This dataset will contain data about apartment prices along with latitude and longitude and a few other features that will be used during the analysis process. There will be 10,000 observations in the data set. There is a larger set that is available but this will not be utilized for this assessment. There are in total 22 columns in the dataset. While there are 21 features. Not all of these columns will be used to answer the question proposed in the first section of this document.

This data is stored in a standard CSV file and there will be no special measures employed to read this dataset into the application. Only standard pandas methods will be used and implemented for this component of the assessment.

While there are many freely available datasets available on the internet. There are both advantages and disadvantages to using these types of resources. The advantage of using free datasets is that there is no expense incurred for their use. The dataset that will used in this assessment is freely available under the available through the Creative Commons Attribution 4.0 International.

This license makes this dataset available to all and no fees are expected to be paid to the owner of the data. The owner of this dataset is the UC Irvine Machine Learning Repository. This is a respected repository and the data that is contained within it should be no exception.

While using a free dataset is great, there is one disadvantage that can be seen in the data. The first is the dataset is not complete. Some of the observations are missing their latitude and longitude. The website makes note that there are no missing values. This did not seem to be the case when the data was investigated. This is either a clerical oversight when the data card for the data was created or the data was not read in correctly.

The challenges that were encountered during the collection process were the following:

- Lack of documentation of the features. This was overcome by looking at the features in a spreadsheet program. This gave a better feel for what each of the features contained.
- Type of the longitude. Longitude as classified by the type is said to be categorical on the website. While this is true and false at the same time. Longitude based on research can also be seen as an integer and this is how it will be used in this analysis.

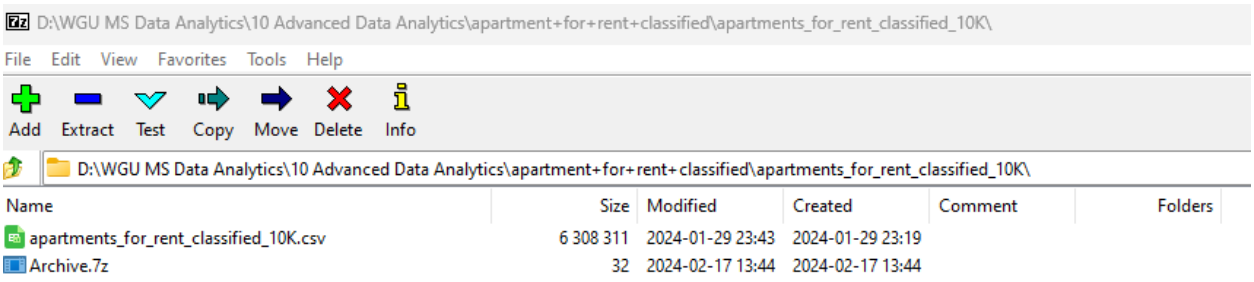
These were the challenges that were encountered using this dataset. There were no other challenges observed during this stage of the data acquisition.

Data Extraction Process and Preparation

The data extraction process begins with downloading the data from the website. The file was contained in a .zip file. After extraction, the files are in a 7-zip file so these files will need to be opened using the following utility program.

- **7-Zip File Manager** - This utility can be downloaded from the following address.
    - https://www.7-zip.org/

This file was opened using this utility and a screenshot of the utility is found below.



This should allow you to extract the CSV file that is shown in the screenshot above. You will then be able to copy this file to the appropriate directory for future use.

The next thing to do is to read the data from the CSV file into the application. The application that will be used here will be Spyder and a Jupyter Notebook. The version for Spyder IDE is 5.3.3. The version for Jupyter Notebook will be 6.4.3. The Spyder IDE was used to test code before it was copied into the Jupyter Notebook. The Jupyter Notebook will be the accumulation of all code that is tested and working. This will facilitate the code being executed one cell at a time when it is required.

Preparation. Once the data has been read from the CSV file. There will be some data exploration undertaken to see if the data has been successfully loaded into the pandas data frame. The pandas data frame is the easiest to use for this load since it provides all the methods needed to create and read data from the CSV file.

There will be an initial check for empty data in the data frame. The screenshot is shown below.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             10000 non-null  int64
 1   category       10000 non-null  object
 2   title          10000 non-null  object
 3   body           10000 non-null  object
 4   amenities      6451 non-null   object
 5   bathrooms      9966 non-null   float64
 6   bedrooms       9993 non-null   float64
 7   currency       10000 non-null  object
 8   fee            10000 non-null  object
 9   has_photo      10000 non-null  object
 10  pets_allowed   5837 non-null   object
 11  price          10000 non-null  int64
 12  price_display  10000 non-null  object
 13  price_type     10000 non-null  object
 14  square_feet    10000 non-null  int64
 15  address        6673 non-null   object
 16  cityname       9923 non-null   object
 17  state          9923 non-null   object
 18  latitude       9990 non-null   float64
 19  longitude      9990 non-null   float64
 20  source         10000 non-null  object
 21  time           10000 non-null  int64
dtypes: float64(4), int64(4), object(14)
memory usage: 1.7+ MB
None
```

As illustrated in the screenshot above, there are quite a few features that have missing data. The data that is missing will be handled using a function that was created to handle these missing values in a simple Python method. The code for this method is shown in the screenshot below[1].

```
def fill_the_column_na() -> None:

    # Replace NaN values with new value.************************************

    rent_df['pets_allowed'].fillna('No Pets', inplace=True)
    rent_df['amenities'].fillna('None', inplace=True)
    rent_df['bathrooms'].fillna(0, inplace=True)
    rent_df['bedrooms'].fillna(0, inplace=True)
    rent_df['address'].fillna('Undisclosed', inplace=True)
    rent_df['cityname'].fillna('Undisclosed', inplace=True)
    rent_df['state'].fillna('Unknown', inplace=True)

################################################################################
```

After running this function the dataframe now has the following values contained within it.

[1] The code for this can be found in the Jupyter Notebook.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             10000 non-null  int64
 1   category       10000 non-null  object
 2   title          10000 non-null  object
 3   body           10000 non-null  object
 4   amenities      10000 non-null  object
 5   bathrooms      10000 non-null  float64
 6   bedrooms       10000 non-null  float64
 7   currency       10000 non-null  object
 8   fee            10000 non-null  object
 9   has_photo      10000 non-null  object
 10  pets_allowed   10000 non-null  object
 11  price          10000 non-null  int64
 12  price_display  10000 non-null  object
 13  price_type     10000 non-null  object
 14  square_feet    10000 non-null  int64
 15  address        10000 non-null  object
 16  cityname       10000 non-null  object
 17  state          10000 non-null  object
 18  latitude       9990 non-null   float64
 19  longitude      9990 non-null   float64
 20  source         10000 non-null  object
 21  time           10000 non-null  int64
dtypes: float64(4), int64(4), object(14)
memory usage: 1.7+ MB
```

Since there were missing latitude and longitude in the data. These observations will be dropped from the data frame. The analyst does not have an easy way to fill in this data, so these empty values will be dropped from the frame. The data frame after the missing latitude and longitude were dropped is shown below.

```
<class 'pandas.core.frame.DataFrame'>
Index: 9990 entries, 0 to 9999
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             9990 non-null   int64
 1   category       9990 non-null   object
 2   title          9990 non-null   object
 3   body           9990 non-null   object
 4   amenities      9990 non-null   object
 5   bathrooms      9990 non-null   float64
 6   bedrooms       9990 non-null   float64
 7   currency       9990 non-null   object
 8   fee            9990 non-null   object
 9   has_photo      9990 non-null   object
 10  pets_allowed   9990 non-null   object
 11  price          9990 non-null   int64
 12  price_display  9990 non-null   object
 13  price_type     9990 non-null   object
 14  square_feet    9990 non-null   int64
 15  address        9990 non-null   object
 16  cityname       9990 non-null   object
 17  state          9990 non-null   object
 18  latitude       9990 non-null   float64
 19  longitude      9990 non-null   float64
 20  source         9990 non-null   object
 21  time           9990 non-null   int64
dtypes: float64(4), int64(4), object(14)
memory usage: 1.8+ MB
```

It can be observed that there are now only 9990 observations in the data frame. This is a suitable amount to create the K-means clustering model. A loss of only ten observations can be seen as acceptable for this analysis.

A quick visual will be created. This will make use of a simple scatter plot. This scatter plot is shown in the image below. If you notice the data takes on the form of the United States. Please notice that there is data included for Alaska (top left) and Hawaii (bottom left).

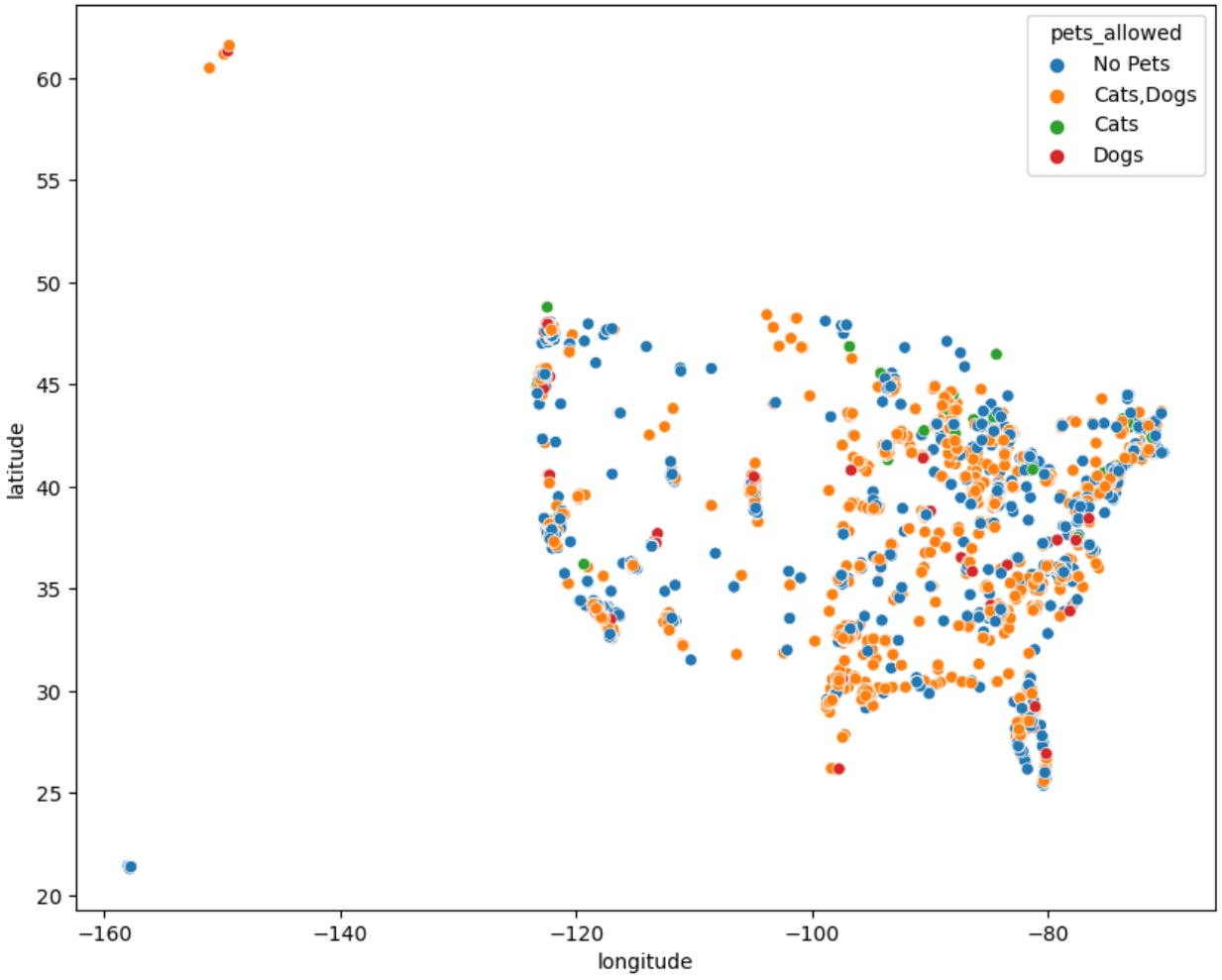


Image 1: Visual of the cleaned data.

This visual shows the data that will be used in the K-means model. There is no further need to clean the data.

Please note that the data will not have outliers removed at this time. It may be vital to look at these outliers to see how and where they are located about the other points that inhabit the same cluster. This will be discussed further in the Analysis section of this document.

Data then was split into training and testing sets. The training will have 80% of the observations while the test set will have the remaining 20% of the data. The features that will be used will be the latitude, longitude, and price. An example of the X_test split is shown below[2].

	latitude	longitude
5499	39.0744	-94.5521
4988	35.5017	-96.8974
2039	37.5444	-121.9820
304	33.5783	-111.8902
4867	43.0724	-89.4003

This data was then normalized using the method found in the **sklearn preprocessing** module. An example of the normalized **X_train** data is given below.

```
[[ 0.48337171 -0.87541521]
 [ 0.48301975 -0.87560946]
 [ 0.40713549 -0.91336777]
 ...
 [ 0.31434783 -0.94930787]
 [ 0.40934421 -0.91238003]
 [ 0.38562055 -0.92265746]]
```

The actual creation of the model will be discussed in the next section of the document.

The advantage of this technique is that the technique allows for the quick interpretation of the data. Using a visual to show the data and creating the new columns allows the data to see what other aspects may affect the apartment price. Some of these features are not going to be explored in this assessment. It is nice to have these features available at a later date.

[2] Please refer to the Jupyter Notebook for the code and other information.

The disadvantage of using the technique that was described earlier is that it is time-consuming to prepare the data for features that may not be of use in this analysis, but this preparation is necessary so this feature may be made use of in the future.

Analysis

The data analysis involved the creation of a K-means object to create the clusters that will be used to identify the apartment price clusters that are based on the longitude and latitude of the apartment listing. The number of clusters that will be created will be based on the silhouette score. The clustering seeks to obtain a score of .60 or better using the chosen features. This was obtained using the following parameters.

- **init= k-means++ -** Using this initialization strategy allows for the plotting of a randomly selected data point to be used for the initial centroid of the cluster. This will be an attempt to force the centroids as far away from each other (Mayo, 2022).
- **algorithm = 'elkan'** – This is the algorithm that seeks to avoid redundant calculations for the distance between the elements of the cluster. This will aid in creating the model with as few distance calculations as possible (VLFeat, n.d.).

The method that was chosen for the number of clusters was to look at the silhouette score and the inertia values. This was done by implementing a method. The code for the method is shown in the screenshot below.

```
def calculate_kmeans_scores (num_k: int, max_inters = 50, verb =1) -> pd.DataFrame() :


  k_val = []                # Number of clusters.
  inertia = []              # Holds the inertia scores for the Kmeans model.
  sil_score = []            # Holds the silhouette score the Kmeans model.

  scores_df = pd.DataFrame()  # Create the dataframe for the scores and values.

  for k in range(2, num_k, 1):

      k_means = KMeans(n_clusters=k, random_state=247, init='k-means++'
                       ,algorithm='elkan', max_iter=max_inters, verbose =verb)

      # Fit the model. **********************
      k_means.fit(X_train_norm)

      # Retrieve the values for each value of K. *********
      k_val.append(k)
      inertia.append(k_means.inertia_)
      sil_score.append(silhouette_score(X_train_norm, k_means.labels_))

  # Add teh values to the scores dataframe. **********************
  scores_df['k_val'] = k_val
  scores_df['inertia_value'] = inertia
  scores_df['sil_score'] = sil_score

  return scores_df
```

The output for this run is shown in the table below. Please note the total number of k-values tried was 50 (starting at 2 and stopping at 49).

	k_val	inertia_value	sil_score
0	2	10.644509	0.647980
1	3	5.080237	0.627920
2	4	2.795346	0.634049
3	5	2.016834	0.622437
4	6	1.396252	0.602743
5	7	1.155275	0.601414
6	8	0.904449	0.613860
7	9	0.662381	0.613390
8	10	0.503461	0.643262
9	11	0.331967	0.668505
10	12	0.238025	0.677981
11	13	0.196159	0.683509
12	14	0.176675	0.678976
13	15	0.153288	0.684770
14	16	0.134854	0.689279
15	17	0.116457	0.686119
16	18	0.101223	0.687730
17	19	0.088058	0.696002
18	20	0.078853	0.692167
19	21	0.070237	0.696267
20	22	0.063051	0.676114
21	23	0.057355	0.682093
22	24	0.052201	0.696307
23	25	0.046577	0.692207
24	26	0.043358	0.687710
25	27	0.039717	0.679246
26	28	0.036608	0.693756
27	29	0.034266	0.676494
28	30	0.031792	0.679196
29	31	0.030006	0.663321
30	32	0.028043	0.685794
31	33	0.026472	0.671145
32	34	0.024800	0.674231
33	35	0.022525	0.668239
34	36	0.021614	0.680029
35	37	0.020458	0.676666
36	38	0.020015	0.658986
37	39	0.018462	0.659457
38	40	0.017498	0.645387
39	41	0.017105	0.660696
40	42	0.016781	0.670943
41	43	0.015484	0.679036
42	44	0.014629	0.662532
43	45	0.013908	0.669501
44	46	0.013552	0.648416
45	47	0.012577	0.668504
46	48	0.012430	0.668933
47	49	0.011950	0.655784

Another visual of the scores shows how these values changed over time regarding the number of clusters implemented in the K-means model. This graph is included below.

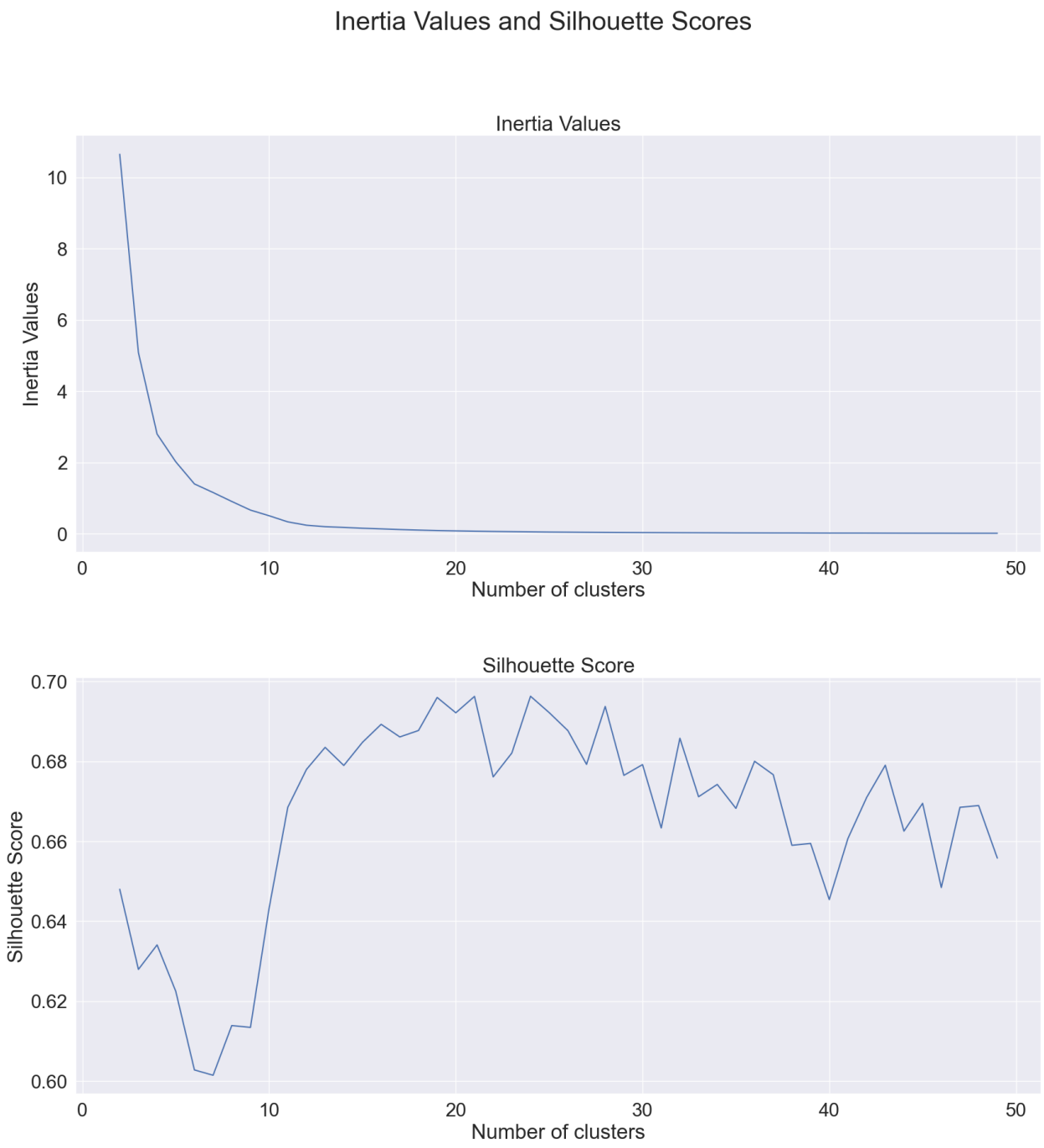


Image 2: Silhouette Score and Inertia Values.

Using these parameters the model was able to obtain a score of approximately 0.696 for the silhouette score. This was using 24 clusters. These clusters were then looked at in aggregate to get a description of the statistics for each cluster. Sorting the scores by silhouette score yielded the following table.

	k_val	inertia_value	sil_score
22	24	0.052201	0.696307
19	21	0.070237	0.696267
17	19	0.088058	0.696002
26	28	0.036608	0.693756
23	25	0.046577	0.692207

This model was created using the scores that can be viewed in the first row of the table above. The k-value is 24. The code for the creation of the final can be found in the Jupyter Notebook. Confirmation of the model creation is shown in the image below.

```
KMeans
KMeans(algorithm='elkan', max_iter=50, n_clusters=24, random_state=247)
```

As displayed above the model used all the parameters that were discussed earlier and used a k-value of 24. These clusters were then plotted to give a visual of the clusters. The plot below shows the clusters that were created using the model.

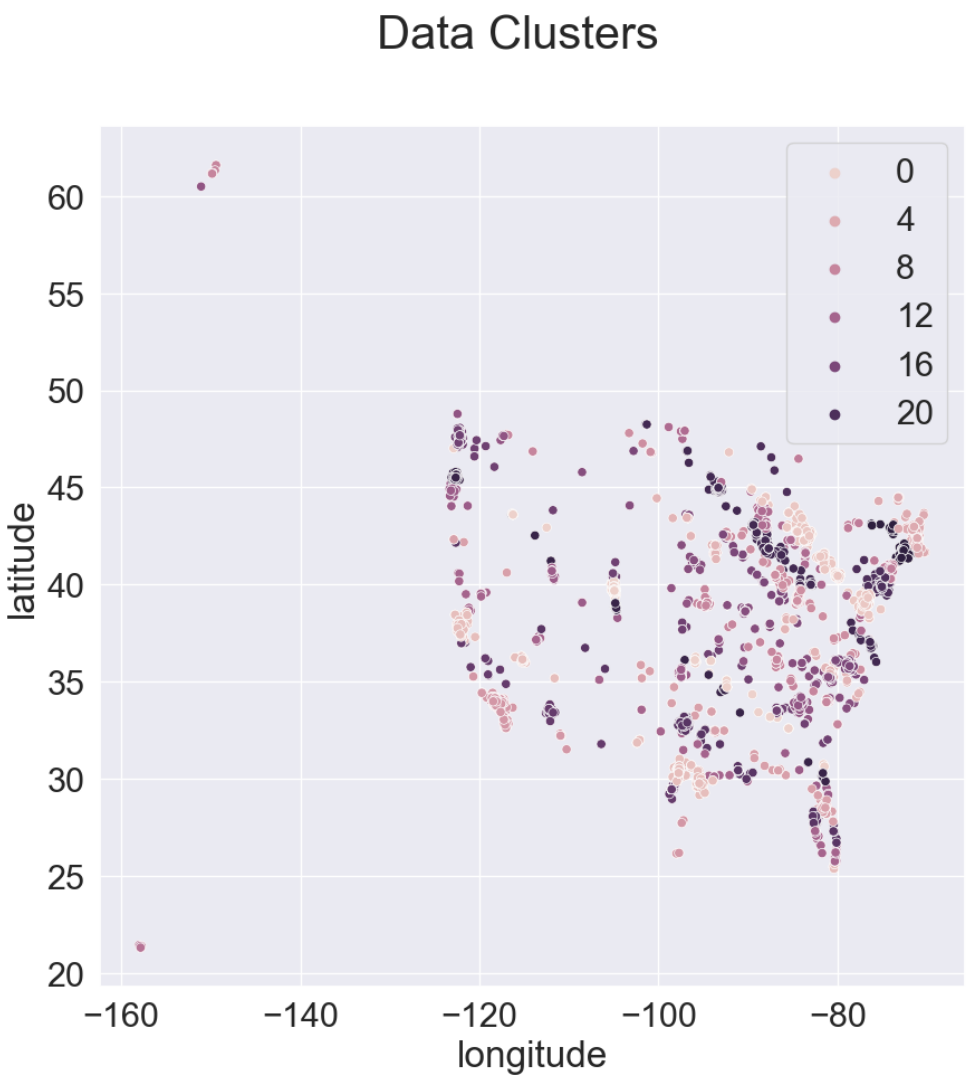


Image 3: Clusters for the data (final model).

After creating the cluster labels the labels were added to the training copy of the data. Please note that this data has not been normalized. This was to keep the data in its original state. An example of the data frame is shown in the table below.

	latitude	longitude	Cluster	apartment_price
181	39.3903	-77.1487	1	200
110	38.6487	-77.3152	1	850
5252	39.4944	-77.4535	1	1250
3627	38.8738	-77.1055	1	2105
8318	44.2146	-88.4323	1	875
...	...	...	...	...
7751	39.0984	-76.9349	1	1890
3162	39.1450	-77.1376	1	1679
6632	39.0128	-76.9812	1	1570
453	39.3284	-76.6021	1	829
9971	38.4417	-76.5173	1	2550

773 rows × 4 columns

The next part of the analysis is to get a feel for some of the statistics that each of these clusters exhibit. The boxplot below shows the results of the clusters and each cluster's percentile as well as any outliers that fall outside the normal range. The screenshot below shows the statistics as grouped by the cluster. Note: the outliers are still included in the data.

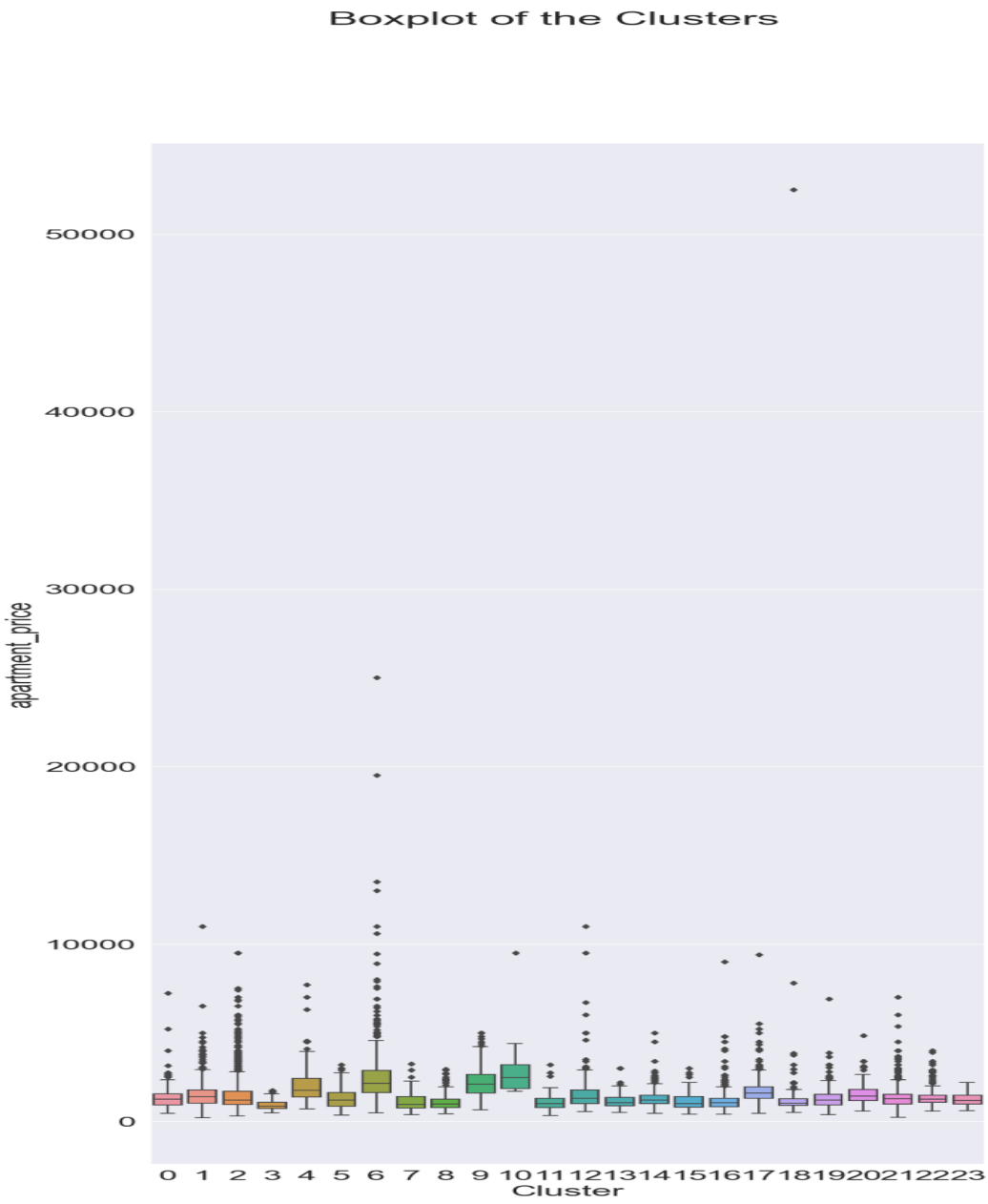


Image 4: Boxplot of the Clusters (final model).

Reviewing the boxplot above we will be examining the five largest clusters in terms of the number of observations contained within them. The clusters that will be examined further are 2, 1, 21, 6, and 19. These are the largest clusters from the 24 clusters that were created using the K-means algorithm. The table below shows the listing of the clusters sorted from the largest to the smallest number of observations in the cluster.

```
Cluster
2     1055
1      773
21     625
6      569
19     497
17     447
16     428
14     390
7      313
8      273
18     264
22     264
15     244
0      243
9      231
4      214
13     210
11     195
20     189
12     169
3      163
5      126
23     100
10      10
dtype: int64
```

As you can see the clusters account for approximately 35% of the available observations.

In cluster 2, the first cluster on the list, we see that this cluster is composed of 1055 observations. Statistics for this cluster are shown below.

```
count    1055.000000
mean     1571.665403
std      1078.881525
min       300.000000
25%       950.000000
50%      1190.000000
75%      1695.000000
max      9500.000000
Name: apartment_price, dtype: float64
```

The mean rent for this cluster is ~$1571.67. The standard deviation for this cluster is ~$1078.88. This means that there is quite a wide spread of observations within the cluster. This deviation is also shown by the spread of values for the minimum and maximum values in the cluster. The minimum was $300 while the maximum apartment rent was $9,500. This can be visualized by viewing the boxplot below.

Image 5: Boxplot of the First Cluster (2).

The next cluster, cluster 1, has the following statistics associated with it.

```
count      773.000000
mean      1509.077620
std        764.947951
min        200.000000
25%       1025.000000
50%       1399.000000
75%       1787.000000
max      11000.000000
Name: apartment_price, dtype: float64
```

The mean for this cluster is shown as the output from the describe function. The mean is ~$1509.08. The standard deviation is ~$764.95. This standard deviation is a little better than the first cluster that was examined previously. There are some extreme values in this cluster as well. The minimum value is $200 while the most expensive apartment rent was $11,000. To help visualize the extremes of the cluster there is a boxplot included below.

Image 6: Boxplot of the Second Cluster (1).

The third cluster statistics were taken from a sample of 625 observations.

```
count     625.000000
mean     1359.636800
std       656.488591
min       224.000000
25%       975.000000
50%      1275.000000
75%      1525.000000
max      6995.000000
Name: apartment_price, dtype: float64
```

The mean rent for this cluster is ~$1359.64. The standard deviation for this cluster is ~$656.49. This cluster seems to have less of a spread in the price of an apartment than the previous clusters. The minimum for this cluster is $224 and the maximum is $6995. To help visualize the spread in apartment values a boxplot is given below.

Image 7: Boxplot of the Third Cluster (21).

In the fourth of the largest clusters, the statistics that were derived from a sample of

```
count      569.000000
mean      2564.231986
std       1899.774283
min        485.000000
25%       1615.000000
50%       2160.000000
75%       2875.000000
max      25000.000000
Name: apartment_price, dtype: float64
```

This cluster has the highest mean of the observations for a cluster so far, and it will have the highest mean of the clusters but has the second lowest number of observations within it. The minimum does not seem to be out of the ordinary. This minimum seems to be in line with the other observations that are in the cluster as well as the clusters that were examined ahead of it.

The maximum apartment price is $25,000. This is the highest seen, and this may be why the mean is quite high. Looking at the number of observations one could imply that other observations are nearly as high as the maximum for this cluster.

Image 8: Boxplot of the Fourth Cluster (6).

Looking at the boxplot we can see that there are quite a few observations that fall outside of the maximum of the boxplot. These points would be of interest to see what constitutes their high price. This will not be investigated in this document but could be an avenue for future research in the future.

The last cluster has the following statistics. The table below shows the statistics for 497 observations in this cluster.

```
count     497.000000
mean     1277.038229
std       521.655496
min       390.000000
25%       930.000000
50%      1200.000000
75%      1515.000000
max      6900.000000
Name: apartment_price, dtype: float64
```

The mean is ~$1277. 04 and the standard deviation is ~$521.66. The minimum is $390 and the maximum value in the cluster is $6900. The boxplot for this cluster is given below.

Image 8: Boxplot of the Fifth Cluster (19).

To help illustrate the clusters the visual below shows the top five clusters as they are depicted in a simple scatter plot of the data This scatter plot has been color-coded using the labels.

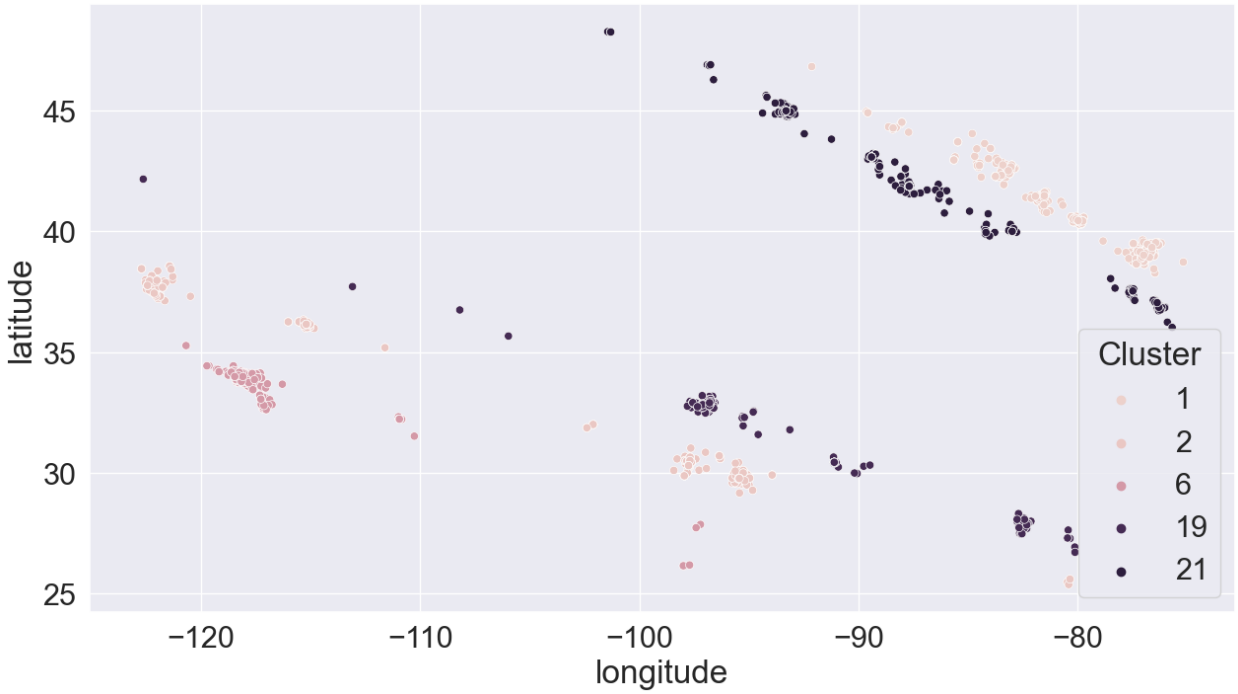


Image 9: Scatter plot of all the clusters.

While this plot is useful for showing the clusters in one visual it does not offer any way to view these clusters within the United States, from which this data is derived. Using **geopandas** to plot the top five clusters we can get a better feel of where these clusters are. The visual below shows the clusters but the observations have been overlaid on top of a map of the United States.

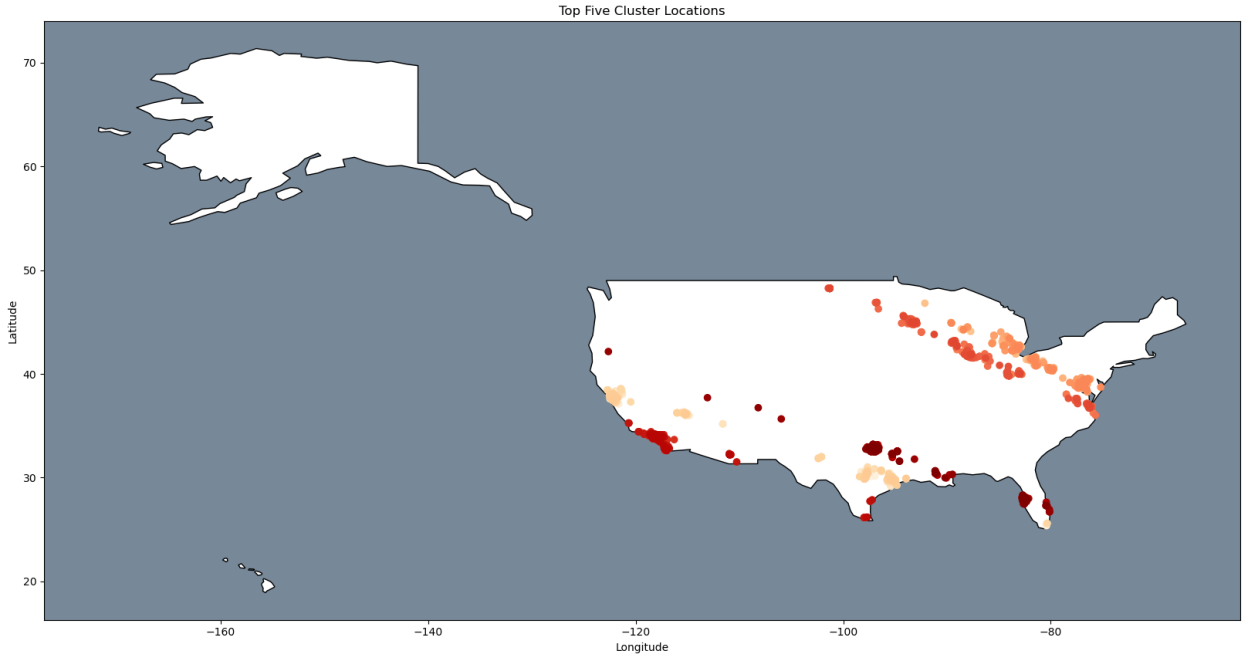


Image 10: Geopandas plot of all the clusters.

Since the legend does not appear[3], the clusters are the following (starting from bottom left):

- 6 – the red color
- 2 - is the light orange
- 19 - is the burnt red color
- 21 – dark orange
- 1 – medium orange

To revisit the cluster that had the highest apartment price, cluster 6, we now can see that this cluster resides in what is southern California. This makes sense if you are familiar with the housing costs that are associated with the area of California in question.

The disadvantage of using this method is that the K-means cluster has the following major disadvantage – this method is susceptible to outliers. As discussed earlier, these outliers will be left in as these are legitimate observations and should be studied as they relate to the other data that are found in the cluster. While this particular model will include the outliers, it

[3] The code to produce this legend would not work.

helps to keep in mind that this inclusion of the outliers will skew the position of the centroids of the K-means clusters.

This inclusion of the outliers is useful because, in some areas of the United States, there are outliers where looking at this information can be beneficial (Frost, n.d.). For example, it might be interesting to look at the amenities that are offered by this particular apartment.

Data Summary and Implications

After reviewing the data and the model, it is prudent to say that there are areas that have a wide range of prices when it comes to apartment prices. Looking at the clusters we can surmise that using latitude and longitude to determine the price of the apartment is not practical. Looking at the statistics of the top clusters.

It is easy to assume that we can not assume the price of the apartment. This was interesting because it shows that based on geospatial location is not enough to determine the price of an apartment. If we were to figure out the price of an apartment we would need to look at other attributes that may affect the price of an apartment. We may want to look at the state in which the apartment is located, or we could look at the size or the amenities.

This information was included in the dataset that was the focus of the analysis, but it was more interesting to see if it was possible to see if there was a way to cluster using the features that were stated earlier. This does leave open an avenue that could be explored further. These additional avenues are discussed in the following paragraphs.

Using this model there can be two directions that this data analysis can take. These are listed below.

1. Since there was an inclusion of the outliers, we can look to remove them to see what the clusters and the statistics show. Why would that be a direction to take? First, we may want to look what the "exact" statistics barring any outliers from these calculations. Second, we may want to look at how the inclusion of the outliers affected the creation of the clusters when it came to the positions of the centroids. This change in the positioning of the centroids may have changed in the composition of the clusters. The change in the composition of the clusters may have an impact on the statistics of the observations within the cluster. For example, the mean price may change or the percentiles may change for the cluster.
2. The next direction may be to use the larger dataset that is available. Using the larger dataset may expose the clustering algorithm to more samples and may make the model more reliable. Since K-means does seem to scale well as the size of the dataset increases. Choosing the K-means algorithm for clustering was a good choice since this algorithm can scale to the larger datasets that may become available in the future.

The model was able to separate the data points into well-defined clusters based on the silhouette score. The model with its current set of features could not adequately determine the price. There would be too much variation in the price solely based on the location of the property. This location would only be based on the latitude and the longitude.

To be a better fit this model would need to be trained on data that either uses more appropriate features or has more granularity when it comes to information about the states. This data is not available in the set at this time. Although this model would be perfectly adequate to group properties that inhabit the same geographic location. Other means would need to be employed to deduce the price of an apartment. This is possible.

To close out the model will excel at grouping properties that are "located near " each other, but the model cannot be utilized to answer the question that was proposed earlier. The model clustering would need to be employed on a dataset that had the data based on a single geographic area like a state or a city. This would be a plausible use of the K-means model in this situation.

References

F. In-text Citations:

Frost, J. (n.d.). *Guidelines for Removing and Handling Outliers in Data*. Statistics by Jim. Retrieved February 18, 2024, from https://statisticsbyjim.com/basics/remove-outliers/

Mayo, M. (2022, May 13). *Centroid Initialization Methods for k-means Clustering - KDnuggets*. KDnuggets. Retrieved February 18, 2024, from https://www.kdnuggets.com/2020/06/centroid-initialization-k-means-clustering.html

VLFeat. (n.d.). *VLFEAT - Documentation > C API*. Retrieved February 18, 2024, from https://www.vlfeat.org/api/kmeans-fundamentals.html