Andrej Slavejkov 196047

Analysis of the US national housing tracker dataset

Introduction

The US National Housing Tracker dataset is a comprehensive collection of data that provides valuable insights into the current state of the US housing market. In this project, we aim to explore the dataset through various data analysis and machine learning techniques. We will begin by preprocessing the data to ensure that it is ready for analysis. This will involve cleaning and transforming the data to remove any missing or irrelevant values.

Next, we will use a classification technique to categorize the data into different classes based on certain features. This will allow us to identify trends and patterns in the data that would otherwise be difficult to discern.

Following the classification phase, we will use regression techniques to predict the values of specific features in the data. This will allow us to make predictions about future trends in the housing market based on current data.

Finally, we will use clustering techniques to group similar data points together. This will help us to understand the relationships between different features and identify any subgroups within the data that may have unique characteristics.

In conclusion, this project will provide a comprehensive analysis of the US National Housing Tracker dataset and will provide valuable insights into the current state of the US housing market.

Data Preparation

Introduction to the data

This dataset contains 58 columns and 1452. The rows each represent the houses which were updated on the US national housing tracker on 15th January 2023. In this project we will later be classifying the data by whether its seasonally adjusted, the regression will be trying to predict the median sale price and finally when clustering we will be trying to split the data by which property type it belongs to (townhouse,condo,single-unit…).

Missing values and other data cleanup

```
median_ppsf_mom                      5
median_ppsf_yoy                     60
median_list_ppsf                     0
median_list_ppsf_mom                 5
median_list_ppsf_yoy                60
homes_sold                           0
homes_sold_mom                       5
homes_sold_yoy                      60
pending_sales                        0
pending_sales_mom                    5
pending_sales_yoy                   60
new_listings                         0
new_listings_mom                     5
new_listings_yoy                    60
inventory                          132
inventory_mom                      136
inventory_yoy                      180
months_of_supply                     0
months_of_supply_mom                 5
months_of_supply_yoy                60
median_dom                           0
median_dom_mom                       5
median_dom_yoy                      60
avg_sale_to_list                     0
avg_sale_to_list_mom                 5
avg_sale_to_list_yoy                60
sold_above_list                      0
sold_above_list_mom                  5
sold_above_list_yoy                 60
price_drops                          0
price_drops_mom                      5
price_drops_yoy                     60
off_market_in_two_weeks              0
off_market_in_two_weeks_mom          5
off_market_in_two_weeks_yoy         60
parent_metro_region               1452
parent_metro_region_metro_code    1452
```

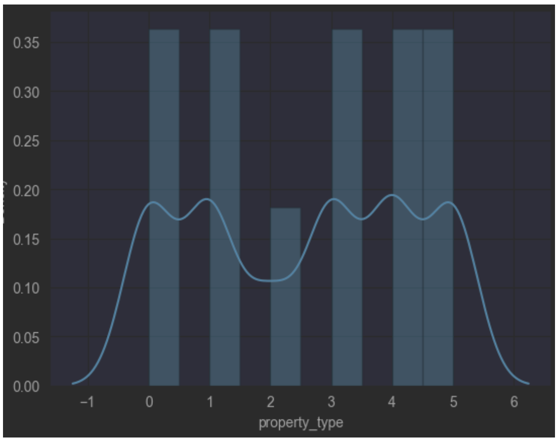
The data while not completely full of missing data does have some empty cells, while the last two columns are completely empty. To deal with this we remove all columns that are only filled with NaN values and we fill in all other missing values with the median value for that column. This way we don’t lose too much of our data from dealing with missing values.

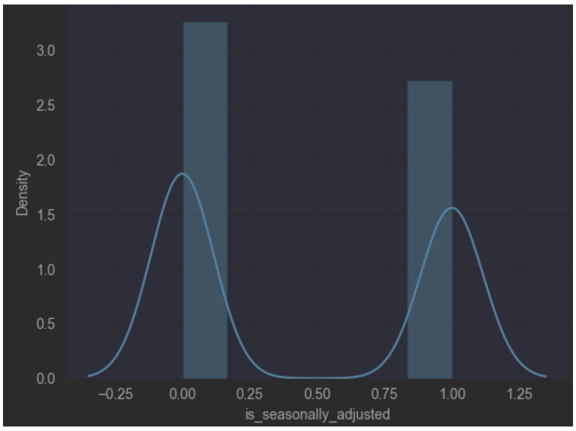
Following this we deal with columns that contain too many identical values and ones that cant contribute to the data analysis that will do later. For this reason we drop all columns that have “id” in them since they are just for identification and are unique for each row and then we also remove all columns that have just one value in repeated in them.

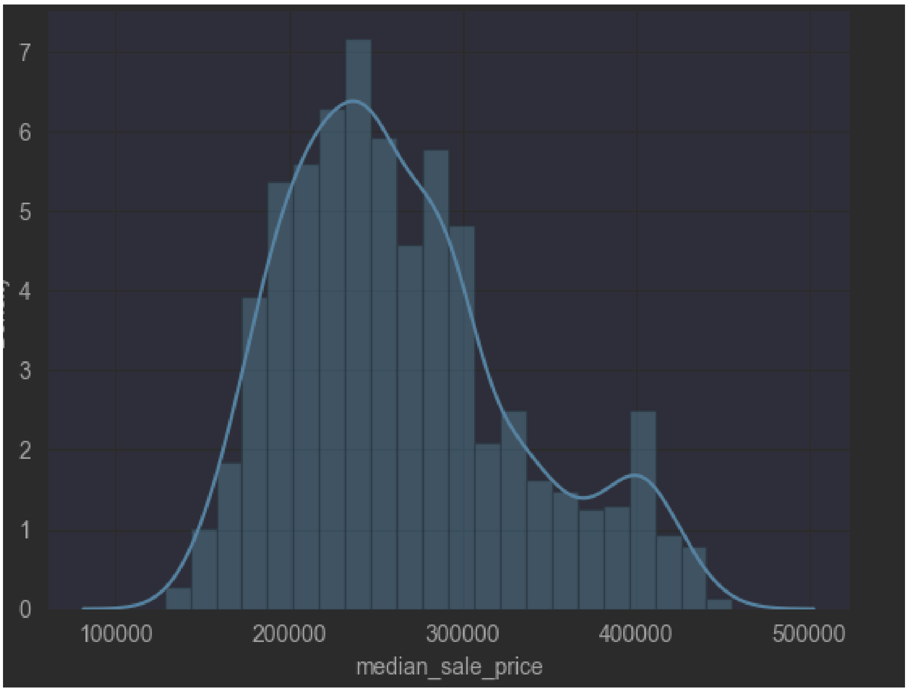
After this we can proceed to split the data into X and y depending on what our target feature is.

Visual analysis

In this step we visualize the data to get a better idea of what we are working with.

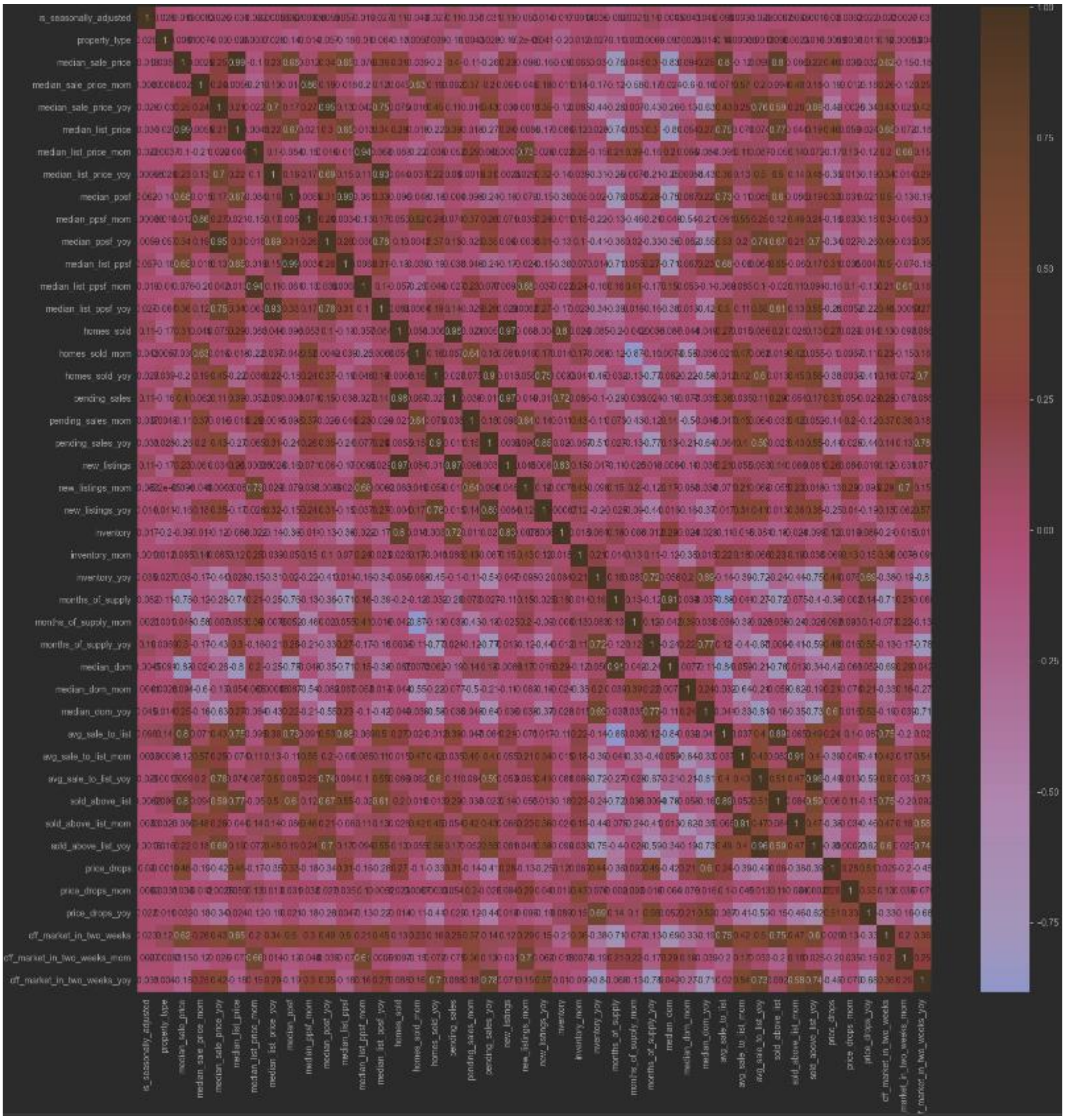






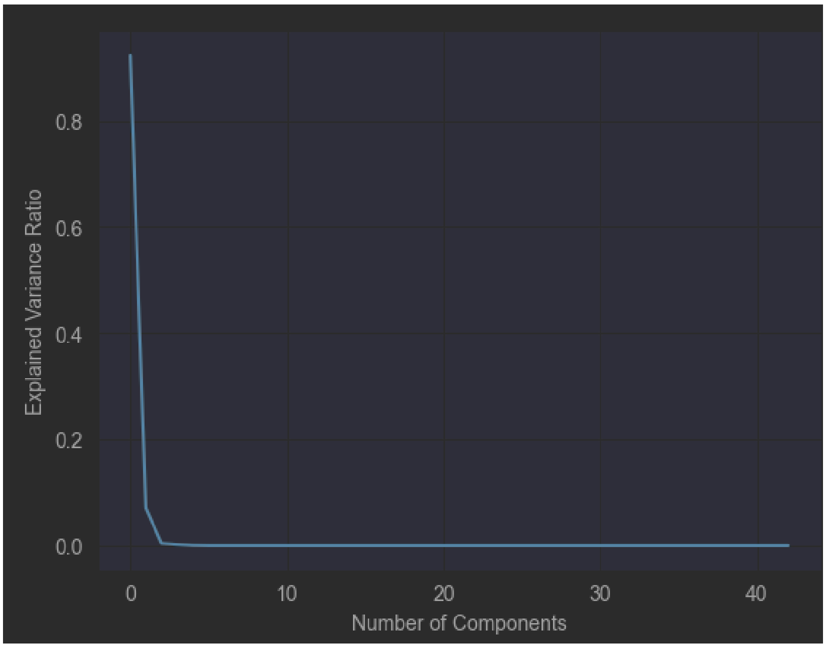
Here we have a histogram of the primary three features that we will be trying to predict with our models. The first two are more evenly spread out while the sales price as expected has a more gaussian like distribution.

Another useful method of visualizing the data is making a heatmap of the correlations between the features. Heatmaps are a pretty useful tool for visualizing the positive and negative correlations between the data we are using as well as the strength of these relationships. Its important to note that outliers and missing data can potentially heavily skew the heatmap which is why i removed the missing values. The outliers however are still present so that should be taken into consideration when looking at the correlations presented on this heatmap.



PCA reduction

PCA (Principal Component Analysis) is a dimensionality reduction technique that can be used to reduce the number of features in a dataset while retaining the most important information. We can determine to how many components we want to reduce the features by several methods. They include plotting the eigenvalues of each component, using Cross-validation to see how many components give the best results for our prediction models, or the one i will be using, calculating the explained variance ratio for each component and plotting it. The goal is to retain only the components that explain a significant portion of the variance, and ignore the rest.

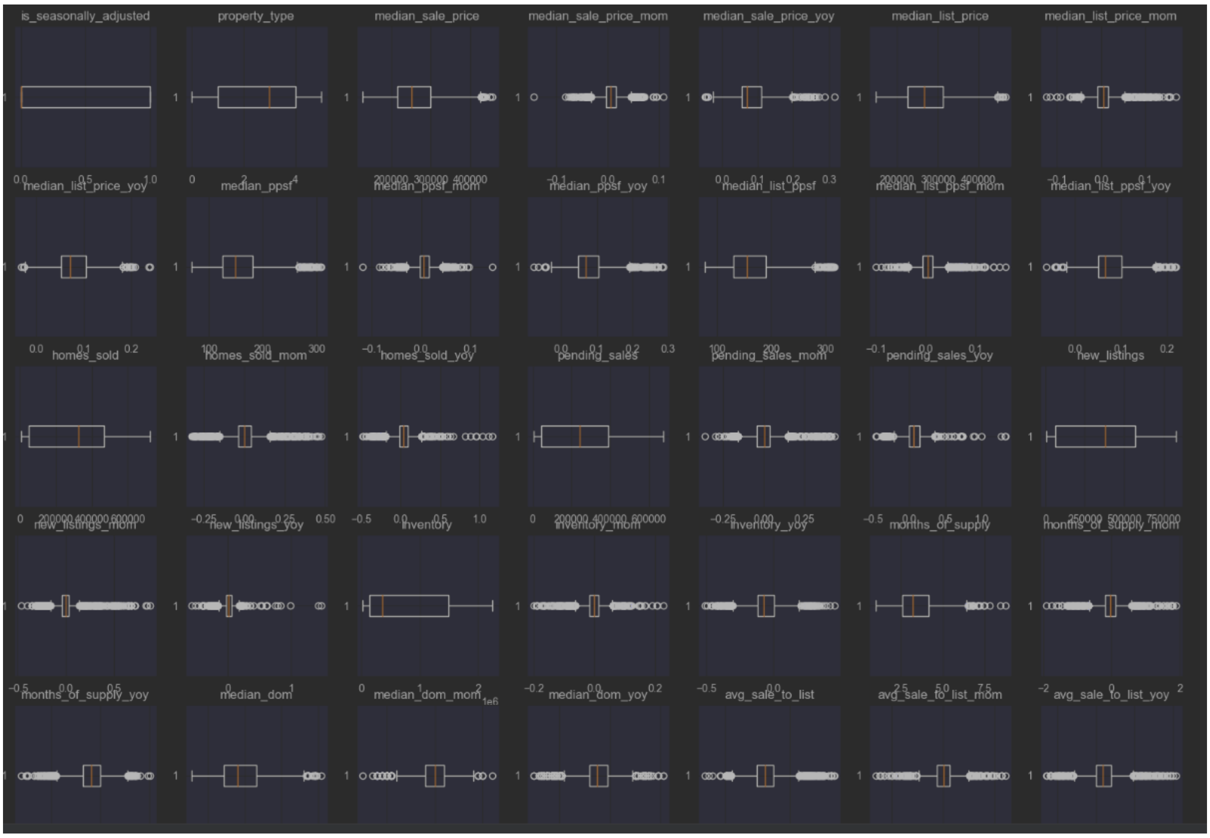


In this graph we can see that we have way too many components compared to what we would need to explain the variance in the features.

Outliers

Outliers are values in a dataset that are significantly different from other values in the same dataset. These outliers can have a significant impact on statistical analysis and modeling, as they can distort distributions and relationships between variables. One way to visualize the outliers is using boxplots for each value.

Here i will do the outlier clearnup by hand using the quartiles and the interquartile range to determine what data should be flagged as an outlier and removed. This is done to get more consistent results and prevent making biased models in the future because of the outliers. But first we visualise the outliers in each category.



These are the boxplots for most of the features we have. As we can see we have a huge amount of outliers in our data.

Classification

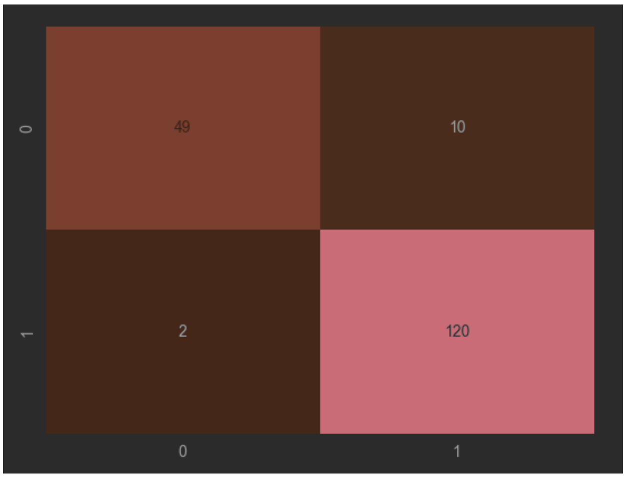
Intro

When classifying the data it is common to scale it as it can often help the models make better predictions. Then we split it into training and testing samples for both X and Y which are the training features and the target feature respectively. I use a standard scaler because it gave me better results than min max scaler and robust scaler for all of my models. As mentioned earlier I will be classifying the data into houses that are seasonally adjusted and ones that aren't.

Andrej Slavejkov 196047

Decision tree

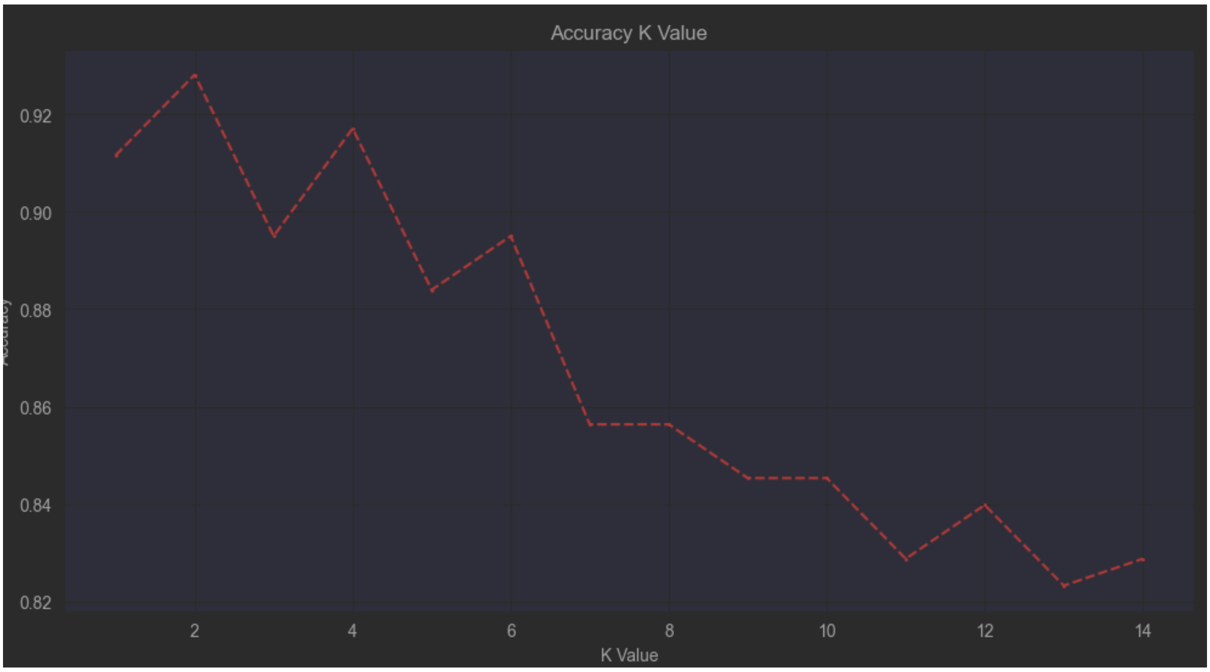
```
Accuracy: 0.9337016574585635
Precision: 0.9230769230769231
Recall: 0.9836065573770492
F1 Score: 0.9523809523809524

Number of leaves: 19
Tree Depth: 7
```



These were the results of my decision tree classifier

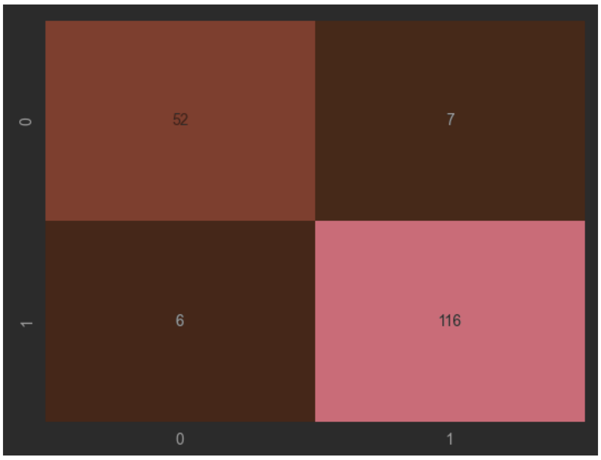
I also plotted out the entire decision tree and it is attached as an additional pdf called is_seasonally_adjusted.pdf.

KNN Classifier



After running some tests for several K values and plotting them these were the results I got. I also ran the test for error rate and it had the same optimal value for k=2

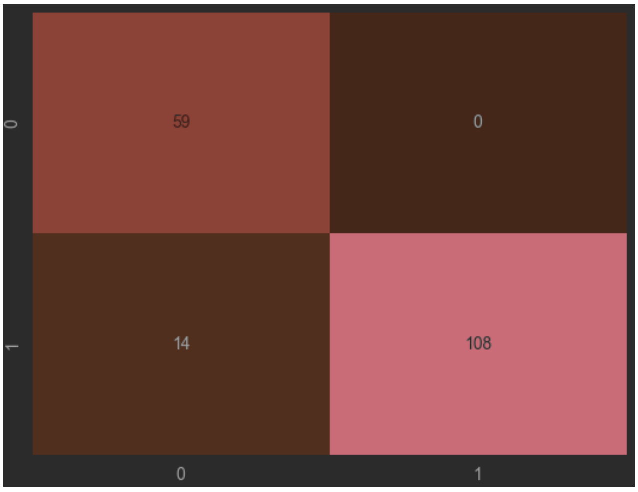
```
Accuracy: 0.9281767955801105
Precision: 0.943089430894309
Recall: 0.9508196721311475
F1 Score: 0.946938775510204
              precision    recall  f1-score   support

           0       0.90      0.88      0.89        59
           1       0.94      0.95      0.95       122

    accuracy                           0.93       181
   macro avg       0.92      0.92      0.92       181
weighted avg       0.93      0.93      0.93       181
```



These were the results of the KNN classification

Gaussian naïve bayes classifier

```
Accuracy: 0.9226519337016574
Precision: 1.0
Recall: 0.8852459016393442
F1 Score: 0.9391304347826086
              precision    recall  f1-score   support

           0       0.81      1.00      0.89        59
           1       1.00      0.89      0.94       122

    accuracy                           0.92       181
   macro avg       0.90      0.94      0.92       181
weighted avg       0.94      0.92      0.92       181
```



Results of the naïve bayes classifier

Neural network classifier

The neural network had the best results out of all the classifiers I tried out with a perfect accuracy precision and recall

```
Epoch 25/25
6/6 [==============================] - 0s 2ms/step - loss: 0.0157 - accuracy: 1.0000
6/6 [==============================] - 0s 1ms/step
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

Regression

In The regression I will be making models that will predict the median sale price of the houses based on the other features. To make the models more accurate I will be using a minmax scaler as this seemed to work best of the scalers I tried out.

Linear regression

```
Mean Absolute Error: 0.006484068379832413
Mean Squared Error: 7.071257714753363e-05
Root Mean Squared Error: 0.00840907706871174


                              Coefficients
is_seasonally_adjusted           -0.000870
property_type                     0.001840
median_sale_price_mom             0.014639
median_sale_price_yoy             0.055408
median_list_price                 1.071051
median_list_price_mom            -0.054399
median_list_price_yoy            -0.053820
median_ppsf                       0.860781
```

These were the results with a very small error. It should be noted that these are scaled numbers and don't exactly mean that the model could predict the price within cents of the actual value.

The coefficient list goes on for all the features.

Lasso Regression

```
Mean Absolute Error: 0.1465794446725753
Mean Squared Error: 0.030818278382656755
Root Mean Squared Error: 0.17555135539965722
```

Ridge Regression

```
Mean Absolute Error: 0.006479057337629802
Mean Squared Error: 7.064671264384408e-05
Root Mean Squared Error: 0.00840515988211075


                              Coefficients
is_seasonally_adjusted           -0.000554
property_type                     0.001867
median_sale_price_mom             0.014334
median_sale_price_yoy             0.054871
median_list_price                 1.067197
median_list_price_mom            -0.054526
median_list_price_yoy            -0.053557
median_ppsf                       0.826959
```

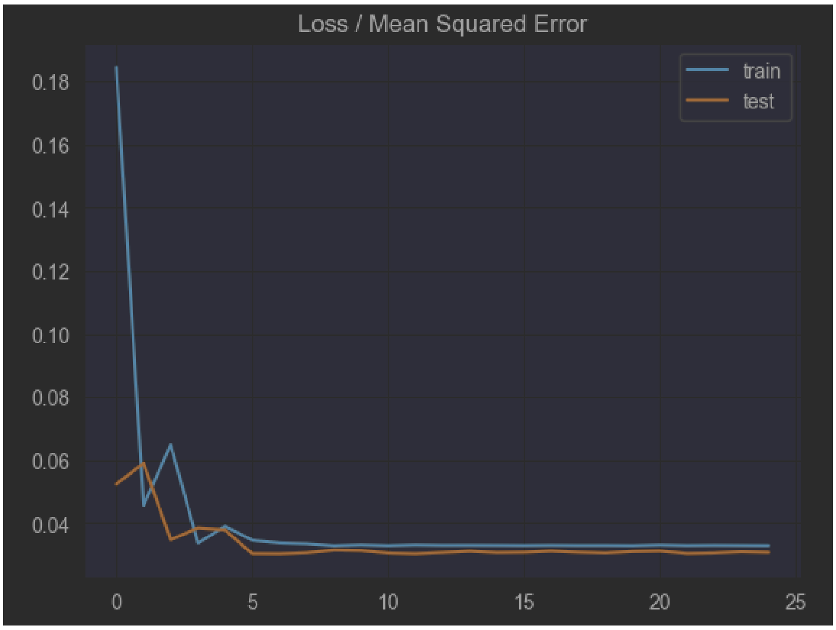
Andrej Slavejkov 196047

Neural Network Regression

The neural networks this time didn't seem to work better than everything else. Linear and ridge regressions seemed more fit to handle this prediction even though I tried with a bunch of different numbers for the layers.

```
Epoch 24/25
7/7 [==============================] - 0s 6ms/step - loss: 0.0330 - val_loss: 0.0310
Epoch 25/25
7/7 [==============================] - 0s 6ms/step - loss: 0.0329 - val_loss: 0.0308
Training error rate:  0.03291225805878639
Testing error rate:  0.030795881524682045
```

These were the end results of the neural network. I also plotted the loss/mean squared error progression for the epochs.



KNN Regression

```
Mean Absolute Error: 0.03012856696228289
Mean Squared Error: 0.002265125840069273
Root Mean Squared Error: 0.0475933381900164
```

More or less the same results as the neural network.

Support vector regression

```
Mean Absolute Error: 0.032251227179070724
Mean Squared Error: 0.0016750697228812088
Root Mean Squared Error: 0.040927615651O6388
```

Andrej Slavejkov 196047

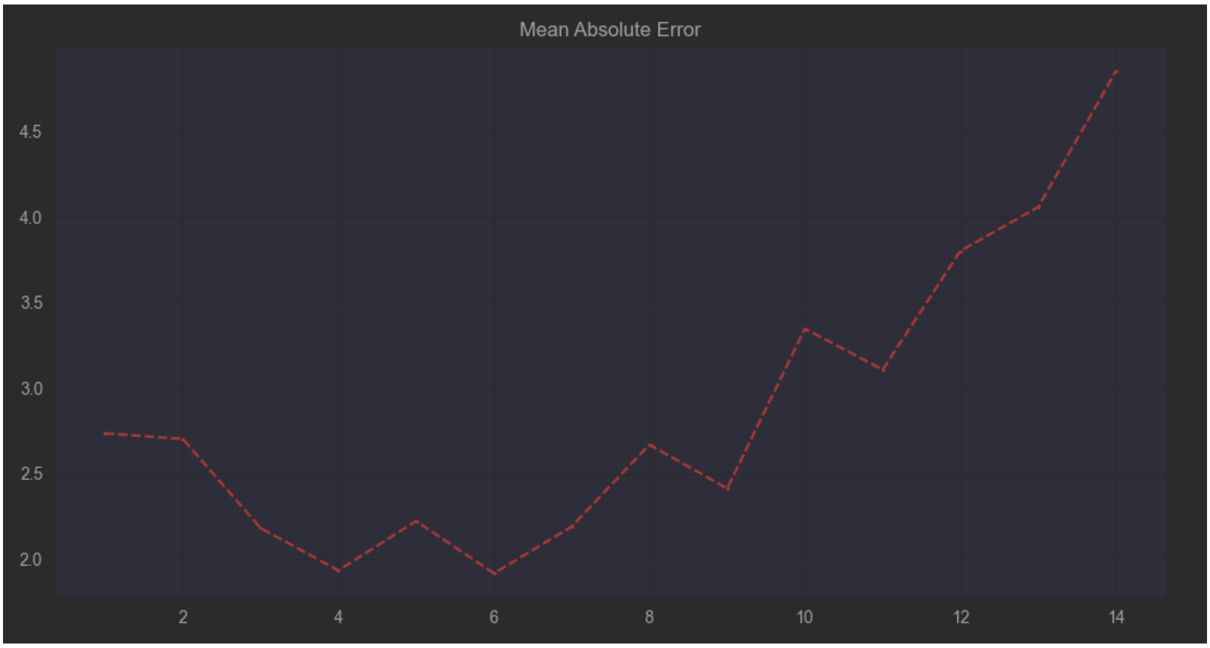
Clustering

Intro

For the clustering I will be classifying the data by the property type the house belongs to. There are 6 classes. I will also be doing a PCA reduction to get down to two components in order to be able to plot the results of the clustering so that its easier to intuitively understand the clusters. This is because without reduction it would be 43 dimensional clusters

K-means clustering

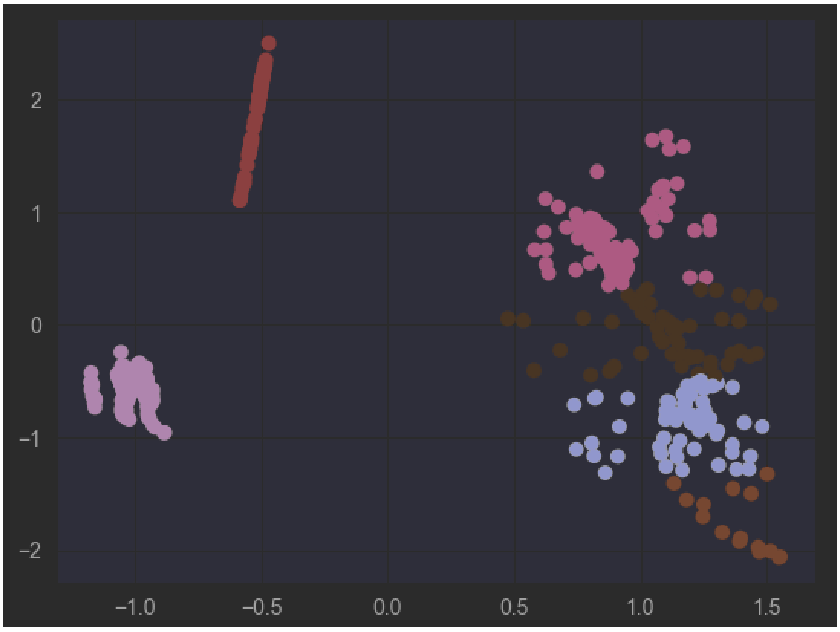
I ran the same method as with the knn classifier to find out for what number of clusters the error rate is lowest. This is the resulting graph.



The model seems to accurately be able to predict there are 6 classes.

Andrej Slavejkov 196047

this was the resulting scatter plot with different colors for each of the clusters to see how it split the data

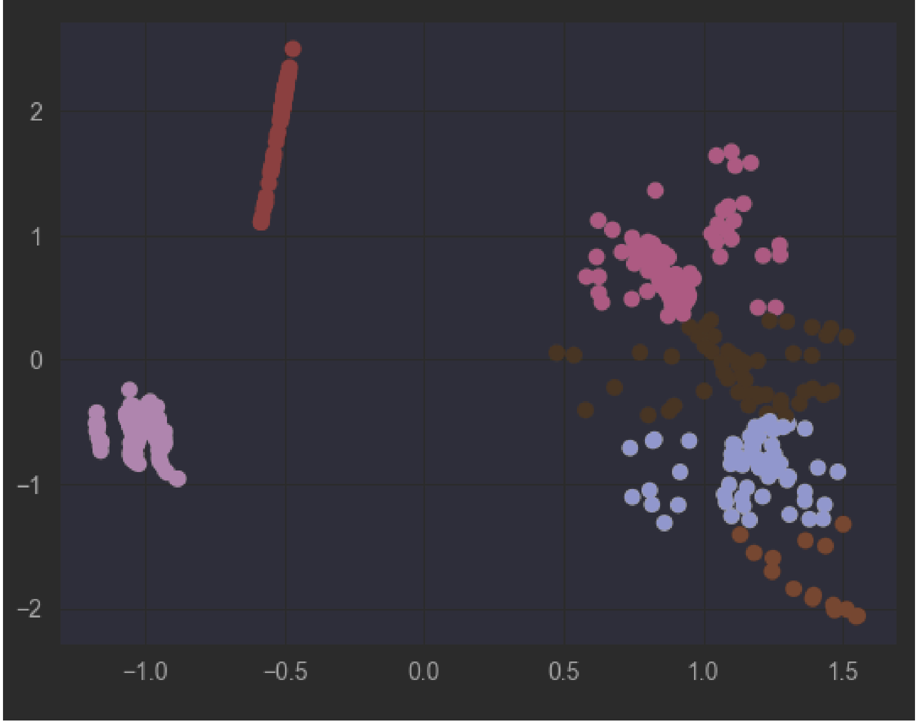



```
Mean Absolute Error: 1.9173553719008265
Mean Squared Error: 6.87603305785124
Root Mean Squared Error: 2.622219109428356
```

And this was the resulting error rate

Agglomerative clustering

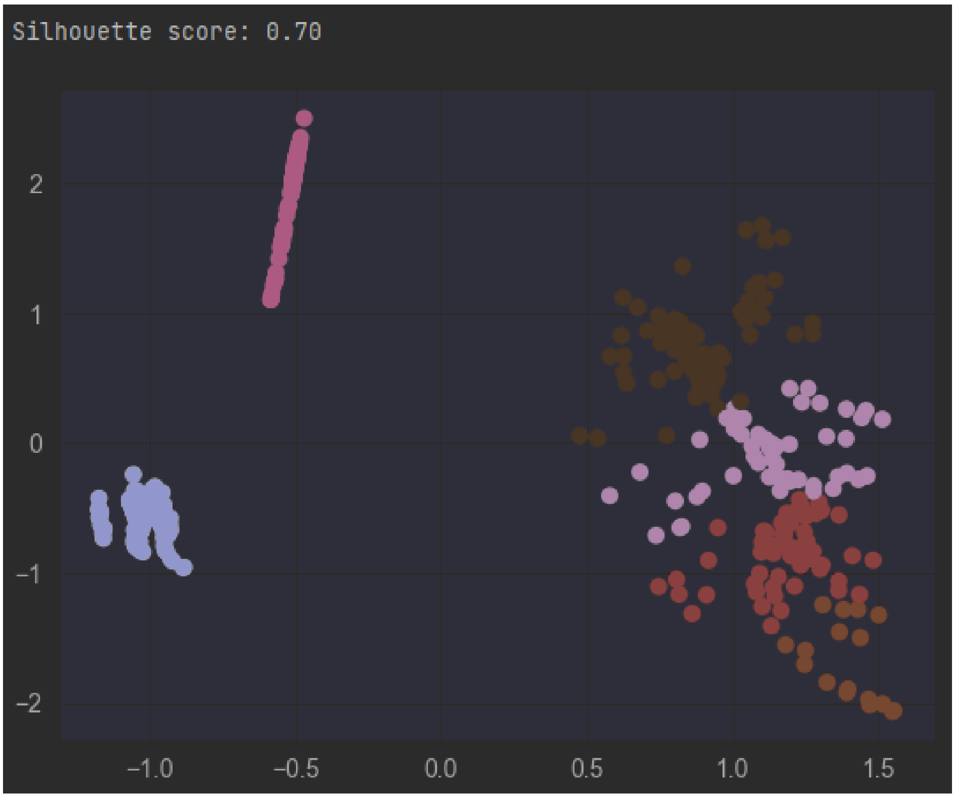
```
Silhouette score: 0.71
```



Andrej Slavejkov 196047

The best way to determine how well an agglomerative cluster model works is using a silhouette score. It goes from -1 to 1 with 1 being the best accuracy.

Gaussian Mixture Model Clustering



I similarly use a silhouette score to evaluate the accuracy of the GMM. They both seem to do pretty well.