Los Angeles Housing Price Prediction Model

Prepared by: James Yokley, Joshua Chang, Bruck Negash, Anthony Edeza, and Nathan Campos

Table of Contents

1. Introduction

1.1. Description

1.1.1. Our project, the Los Angeles Housing Price Prediction Model, will be utilizing a linear regression model, decision tree model, random forest model, and knn model that will allow us to predict the median home price in Los Angeles, by zip code. We will see which model is best suited to predict future prices.

1.2. Details

1.2.1. We will be implementing the Zillow Home Value Index (ZHVI) dataset made available by zillow.com and will be focusing on the single-family home time series.

1.3. Goals

1.3.1. Our goal is to be able to predict the median value of homes in different zip codes over the next decade.

2. Dataset

2.1. Datasets Used

2.1.1. The dataset we used for our model was the Zillow Home Value Index (ZHVI) from the website Zillow.com.

2.2. Details About Dataset

2.2.1. The dataset consisted of hundreds of columns-one for each month from 2000 to now as well as zip code, size rank, region name, region type, state, metro, and county. Each row contained information about the location of the market-zip code, city, and county-with median home price being listed under each date column.

2.2.2. In order to make this data easier to analyze in a regression model, we took only the zip codes in Los Angeles county and pivoted the table. This resulted in a dataset with columns for the zipcode, date, and median price and 4641 rows of data. In order for us to utilize the date as a factor, we needed integer data types as opposed to a date type. To remedy this, we separated the date out into three columns for month, day, and year. Our final dataset that we have used to train models contains 4 feature columns and median price as our target column.

2.2.3. In addition to the dataset that we are using to train our models, we have created our own dataset that we use to make predictions on future prices. This dataset simply consists of the same columns as our training dataset, but with the month, year and day spanning from 1/1/2023 to 12/1/2033. This dataset consists of four columns and 2112 rows.

3. Development Strategies

3.1. Methods

3.1.1. Approach

3.1.1.1. Our project begins by reading a CSV file called LA housing set using the "pandas" library and displaying the data. We then defined the features to be used in a regression model and created a list of zip codes to use in the analysis.

3.1.1.2. Due to some considerations of the volatile housing market and the nature of the recent economic shifts in the past decade, the team had to consider sudden market crashes, high inflation rates, and the market fluctuations from before and after the COVID pandemic. Due to these recent changes, data has been a lot more volatile, and predictions of the short time in the future might be inaccurate, but as time increases so does the accuracy of our predictions.

3.1.1.3. The dataset also includes data from the 2000's market crash and recession, which did affect the prices, economy, and value of the dollar all together. Our predictive models would need to put this into consideration especially when attempting to predict a random price in the past.

3.1.2. Predictive Models

3.1.2.1. This section will briefly describe the predictive algorithms and models that the team has used. We will use the following algorithms and models:

Linear Regression, Decision Tree Classifier, Random Forest Model, and KNN Classification.

3.1.2.2. Linear Regression Model:

Linear Regression is a method where the data is modeled around at least one independent scalar variable, and one dependent variable. The value to be predicted is the dependent variable. The data is modeled and a "best fit" line is created using the least amount of distance between each point.

3.1.2.2.1. Many economical data is represented in data points, with however many "points" and their relative fluctuations of value over time. Due to this a predictive line to estimate a general trend of the economy would not only give us the general idea of housing prices over time, but also give us an indication of the general state of the economy at the time of the prediction, even including sudden crashes or global pandemics.

3.1.2.3. Decision Tree Model:

The Decision Tree model builds in the form of a "tree" structure, breaking large data sets into smaller subsets, and eventually results in a decision nodes and leaf nodes. Decision nodes represent values for testing, such as zip code, months, years, days, and leaf nodes would be the numerical values, or the decision made upon them. The features split the data and the different nodes make decisions upon numerical values and data.

3.1.2.4. Random Forest Model:

The Random Forest model constructs several decision trees at the training stage, and then combines the prediction results. This model also reduces/eliminates overfitting by limiting variance and instability which is seen in the Decision Tree Model.

3.1.2.5. KNN Classification:

The KNN regression model is an algorithm making a decision based on closest training samples in the feature space. To make sure data in the training set is reliable, a good K must be chosen. To do this, we first selected a zipcode that had a wide variance in pricing to represent the remaining zipcodes. We then looped through range 50, fitting a knn model with k-value of the loop instance each time and calculated the RMSE. From this list, we chose the lowest RMSE and used that as our k-value for the training of each zipcode's model.

3.1.3. Training and Testing Stage

3.1.3.1. Linear Regression (No Scaling)

1. Our first step was to take only the rows that pertained to each zipcode from our data frame list, and drop any rows in the data that contained any null values.
2. We then began looping over a list of zip codes, and for each zip code it is performing a train-test split on a dataframe, "df", with a subset of the rows that have that zipcode.

3. The features used for training and testing are specified by the "feature_cols" variable, and the target variable is the "median price" column of the data frame.
4. Finally, the linear regression model specified by "my_linreg" is fit to the training data.

3.1.3.2. RandomForest (Scaled)

The code then enters a loop over a list of zip codes. For each zip code, the code filters the original **df** DataFrame to only include rows with that zipcode, and defines new **X** and **y** variables based on the filtered DataFrame. The **X** data is then scaled using the scale function.

Next, the code splits the **X** and **y** data into training and testing sets using the **train_test_split** function from the **sklearn.model_selection module**. This function randomly splits the data into two sets, using the specified test size (in this case, 30%).

The code then creates a random forest regressor model using the RandomForestRegressor class and fitting it to the training data using the fit method. The model is then used to make predictions on the test data using the predict method.

3.1.3.3. Decision Tree (Scaled)

1. We defined the features (X) and the target variable (y) for the model, both of which are taken from the input df DataFrame.

2. Next, we loop over a list of zip codes, and for each zip code, we then filter the original df DataFrame to only include rows with that zipcode, and define new X and y variables based on the filtered DataFrame. The X data is then scaled using the scale function.
3. Next, we split the X and y data into training and testing sets using the train_test_split function from the sklearn.model_selection module. This function randomly splits the data into two sets, using the specified test size (in this case, 30%).
4. We then created two decision tree regressor models, one with a maximum depth of 5 and the other with a maximum depth of 7. The models are fitted to the training data using the fit method. The models are then used to make predictions on the test data using the predict method.

3.1.3.4. Linear Regression (Scaled)

This was similar to the train/test without scaling method, but with some additional processing on the input data.

1. The code loops through a list of zip codes, creating a new dataframe for each zip code containing only the data for that zip code.
2. Subsets the rows of the dataframe, df, to only include those with the current zip code.
3. Selects the feature columns specified by the feature_cols variable.
4. Scales the feature values using the “scale” function from “sklearn.preprocessing”

5. Converts the scaled features into a dataframe, with the same column names as the original feature columns.
6. Splits the scaled features and target variable into training and test sets using the train_test_split function.
7. Fits the linear regression model, my_linreg, to the training data and makes predictions on the test data.

3.1.3.5. KNN(Scaled)

1. We defined the features (X) and the target variable (y) for the model, both of which are taken from the input df DataFrame.
2. Next, we loop over a list of zip codes, and for each zip code, we then filter the original df DataFrame to only include rows with that zipcode, and define new X and y variables based on the filtered DataFrame. The X data is then scaled using the "scale" function.
3. Next, we split the X and y data into training and testing sets using the "train_test_split" function from the "sklearn.model_selection" module. This function randomly splits the data into two sets, using the specified test size (in this case, 30%).
4. Finally, we created a K-nearest neighbors regression model using the "KNeighborsRegressor" class with the k-value of 7, and fitting it to the training data using the "fit" method.
5. The model is then used to make predictions on the test data using the "predict" method.

3.1.4. Validation Method

3.1.4.1. No Scaling

1. After the training and testing stage, we then printed the intercept and coefficients of the linear regression model, “my_linreg”, for the current zip code being processed in the loop.
2. It then uses the model to make predictions on the test set, and calculates the root mean squared error (RMSE) between the predicted values and the true values of the “median price” for the zipcode.

3.1.4.2. Scaling

1. Using the “scale” function from “sklearn.preprocessing” we assigned the scaled features to an X variable. It then splits the data into training and test sets, fits the linear regression model to the training data, and makes predictions on the test data.
2. Prints the intercept and coefficients of the model, calculates the root mean squared error (RMSE) of the predictions, and plots the results.

3.1.5. Plotting Methods

3.1.5.1. Linear Regression:

We essentially created a separate linear regression model for each zip code and visualized the fit of the model to the data for each zip code.

1. We cleared the current figure and created a scatter plot of the year feature against the “median price” target for the training data.
2. Plotted a line showing the predicted values of “median price” for the test set against the year feature.

3.1.5.2. Decision Tree:

Our Decision model plots the predicted and true values against the year of the data, using a scatter plot and a line plot. The predicted values from the two models are plotted separately, using different colors.

3.1.5.3. Random Forest:

Our random forest regressor model plots the predicted and true values against the year of the data, using a scatter plot and a line plot.

3.1.5.4. KNN:

1. This model calculates the root mean squared error (RMSE) between the predicted and true values, using the mean_squared_error function from the sklearn.metrics module, and prints the result. It then plots the predicted and true values against the year of the data, using a scatter plot and a line plot.
2. We then filter the predicted df "pred_df" to only include rows with current zip code, scale the data, and use the trained KNN model to make predictions on the filtered data.
3. Finally, we plotted the predictions against the year of the data, using a line plot.

3.1.6. Future Prediction Methods

3.1.6.1. In order to make predictions on future data, we first created a simple dataset that consisted of our zipcodes, and month, day, and year for the next

ten years. We turned this into a dataframe and made predictions on each zip code, plotting the results.

1. We made predictions on the section of the data frame that contained the relevant zip code that we were training in the for loop.
2. Plotted a line to show these predictions and the rise of housing prices we may expect over the next ten years within each of these zip codes.

3.2. Algorithms

3.2.1. Learning Algorithm

3.2.1.1. We used Linear Regression, Decision Tree Regression, Random Tree Regression with our dataset to create a model to predict future home prices for different zip codes.

3.2.2. Predictive Models

3.2.2.1. Our data is numerical and continuous so we chose Linear Regression for our model to predict continuous values.

3.3. Tools

3.3.1. Libraries Used

3.3.1.1. For our project we used many python libraries such as: Sklearn, Pandas, Numpy, and Matplotlib.

3.3.1.2. Numpy is a python library, a collection of mathematical functions for very large arrays, matrices, and many more functions. There are many uses not only for data science, but for other mathematical uses as well.

3.3.1.3. Pandas is a library built on top of the Numpy library, and provides the project with many useful functions, such as data visualization, framing, and further analysis.

3.3.1.4. Matplotlib is a library closely associated with NumPy. It closely resembles MATLAB, which is used in many mathematical programs and projects.

3.3.1.5. Sklearn, also known as Scikit-learn, is a free machine learning library for Python. Many algorithms are included within this library, such as regression, clustering, random forest, and many more. Sklearn is designed to be used alongside the NumPy and SciPy libraries

3.3.2. Functions Used

3.3.2.1. Sklearn

Sklearn.linear_model: Used for creating a linear regression model.

Cross_Val_Score: Used to calculate the negative mean squared error.

Train_Test_Split: Used for splitting a dataset into training and testing sets

Sklearn.metrics:

1. “mean_squared_error”: function is used to calculate the mean squared error

Accuracy_Score: We need to calculate average root mean square error to find accuracy of our model.

Sklearn.preprocessing: The "scale" function is used to normalize the data.

3.3.2.2. Matplotlib.pyplot

Plt.scatter and plt.plot:

1. Used to create the plot

plt.clf():

1. Used to clear current plot before creating a new one for each zip code

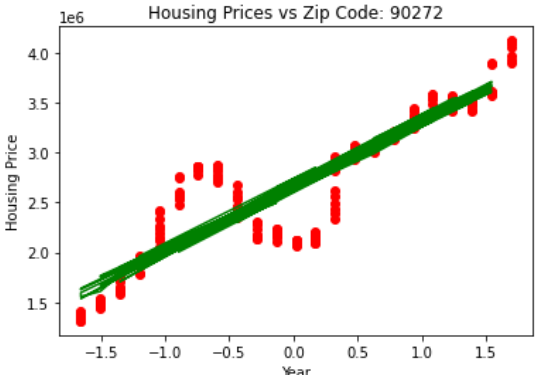
4. Results

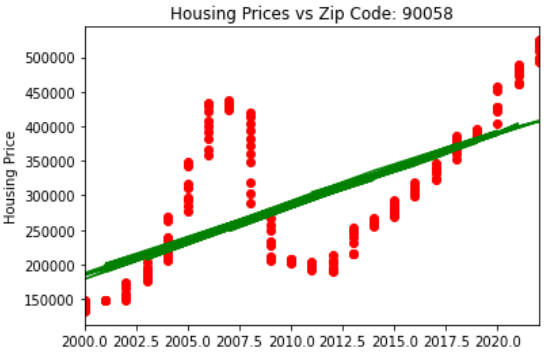
4.1. Machine Learning Algorithms

4.1.1. Linear Regression

4.1.1.1. Data visualization

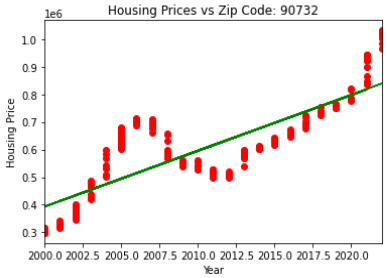
Set of randomly selected scaled linear graphs





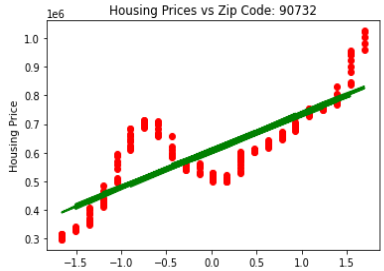
Unscaled for zip code 90732

```
Zipcode: 90732, Intercept: -40031635.49248923
Zipcode: 90732, Coefficient: [     0.         20199.94916215   871.35420686  -194.40132932]
Zipcode: 90732, RMSE: 91824.33597053889
```



Scaled for zip code 90732

```
Zipcode: 90732, Intercept: 607770.5718573175
Zipcode: 90732, Coefficient: [     0.         128160.48459971  -3048.54359047   6267.22148516]
Zipcode: 90732, RMSE: 89593.51510214912
```



4.1.1.2. Results without scaling:

The plot shows the relationship between the year and housing prices for each zip code, however the intercept is not normalized as the point was negative.

4.1.1.3. Results with scaling:

The plot is scaled and shows the relationship between the year and housing prices for each zip code. The data is normalized and now gives us an understandable price point to begin and end with.

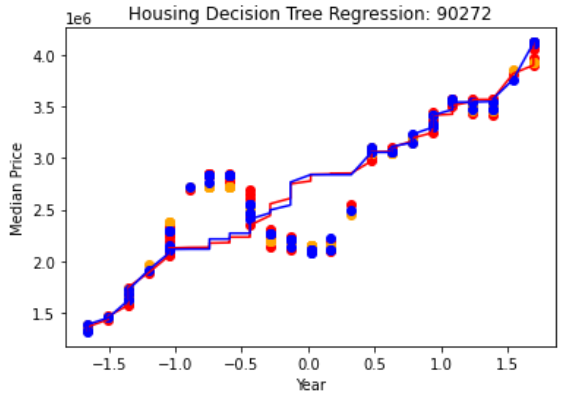
4.1.1.4. General Results:

The resulting slope of the graphs did not fluctuate much from different zip codes, which means that housing prices all across our data set in California all rose at a general rate. This would make generalizing prediction a lot more easier, and with further research we could compare the minute differences between each zip code and prices.

4.1.2. Decision Tree Regression:

4.1.2.1. Data Visualization:

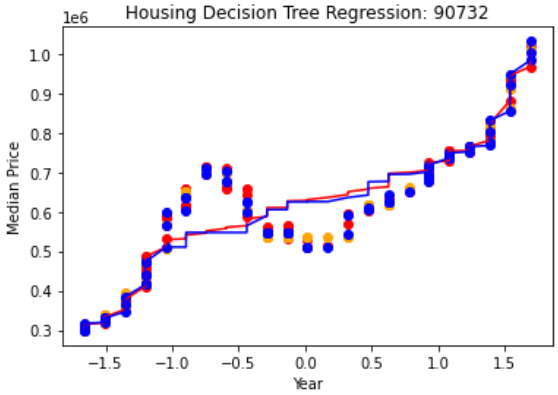
4.1.2.1.1.



4.1.2.1.1.1. Zip Code: 90272

Depth 5 RMSE: 70203.86376198

Depth 7 RMSE: 55322.16235385

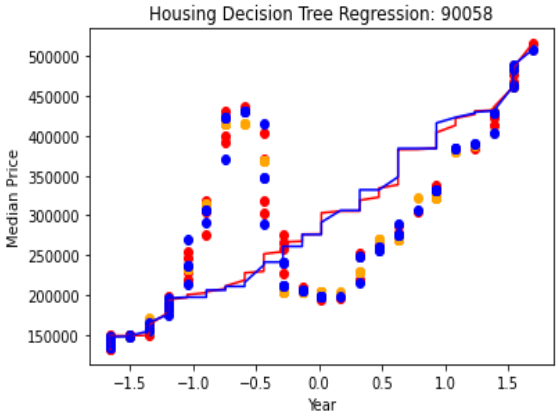


4.1.2.1.2.

4.1.2.1.2.1. Zip Code: 90732

Depth 5 RSME: 18520.5795152

Depth 7 RSME: 10955.1854972

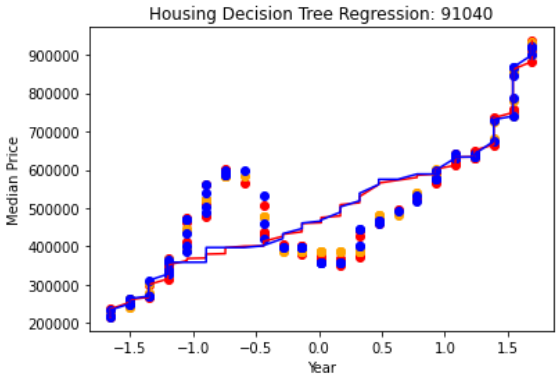


4.1.2.1.3.

4.1.2.1.3.1. Zip Code: 90058

Depth 5 RSME: 18612.3941145

Depth 7 RSME: 9161.55927064



4.1.2.1.4.

4.1.2.1.4.1. Zip Code: 91040

Depth 5 RMSE: 17881.0268797

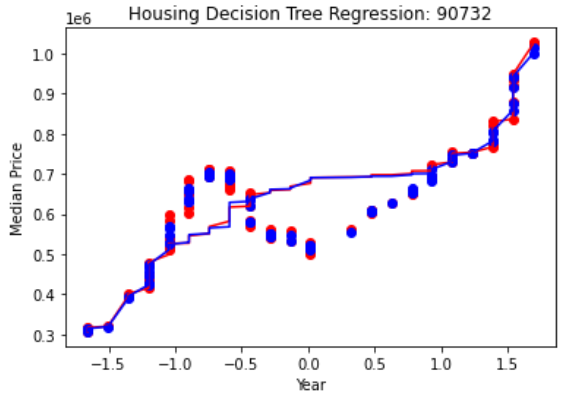
Depth 7 RMSE: 11220.5143387

4.1.2.2. General Results:

4.1.2.2.1. For our decision tree regressor model our models were trained at two maximum tree depths. The first at a tree depth of 5 and the second with a maximum tree depth of 7. The RMSE values are being printed to the console as a way to evaluate the performance of the models. A lower RMSE value indicates that the model is making predictions that are closer to the true values in the test set. In this case, the model with a maximum depth of 7 has a lower RMSE value than the model with a maximum depth of 5, suggesting that it is making more accurate predictions on the test set.

4.1.3. Random Forest Regression:

4.1.3.1. Data Visualization:



4.1.3.1.1.

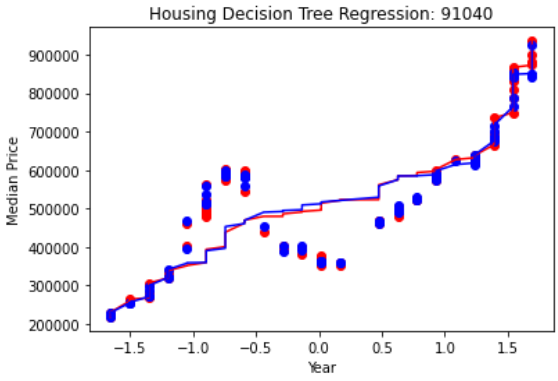
4.1.3.1.1.1. Zip Code: 90732

RSME: 11900.41392466

4.1.3.1.2.

4.1.3.1.2.1. Zip Code: 90272

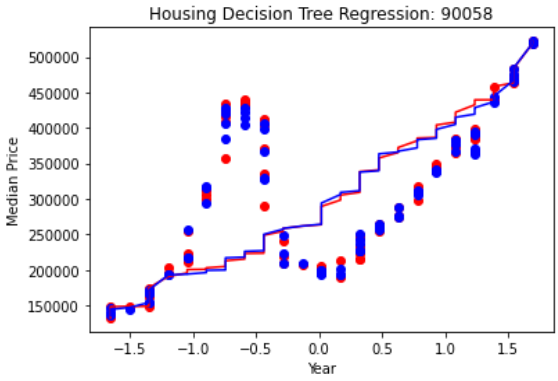
RSME: 45655.101250315



4.1.3.1.3.

4.1.3.1.3.1. Zip Code: 91040

RSME: 12650.91979721



4.1.3.1.4.

4.1.3.1.4.1. Zip Code: 90058

RSME: 8893.76150517

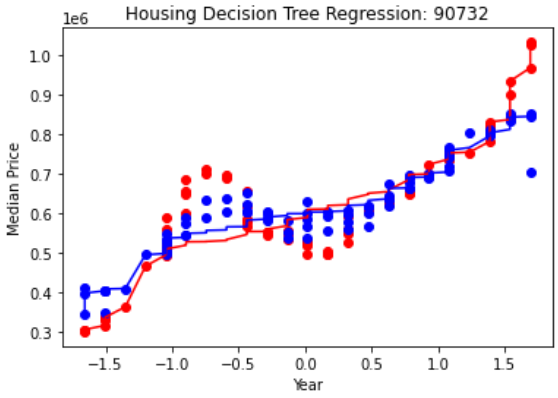
4.1.3.2. General Results:

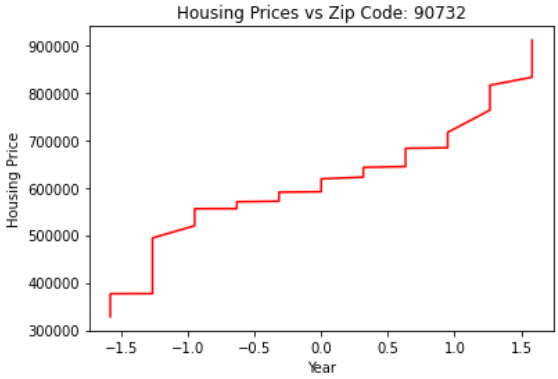
The predicted values were all very close to our real values, and were only off by about one to three percent, which gives a very good general idea for housing prices in the future.

4.1.4. K-Nearest Neighbor:

4.1.4.1. Data Visualization:

4.1.4.1.1.

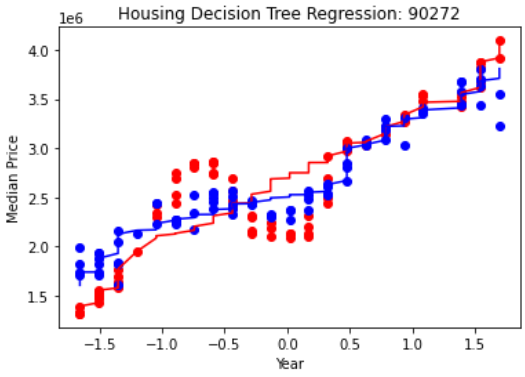


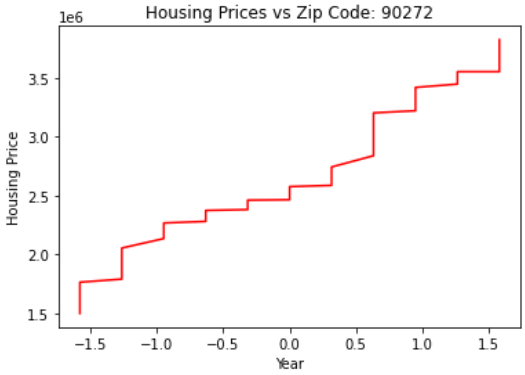


4.1.4.1.1.1. Zip Code: 90732

RSME: 64378.13808437

4.1.4.1.2.

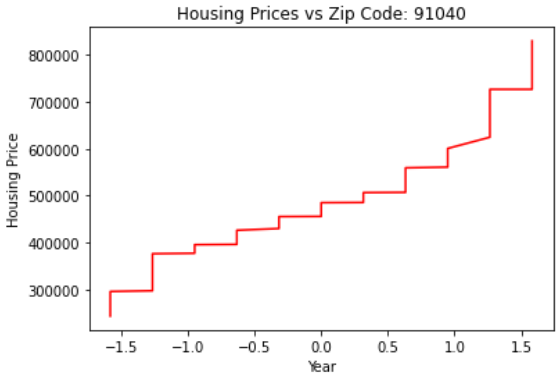




4.1.4.1.2.1. Zip Code: 90272

RSME: 288718.8412256295

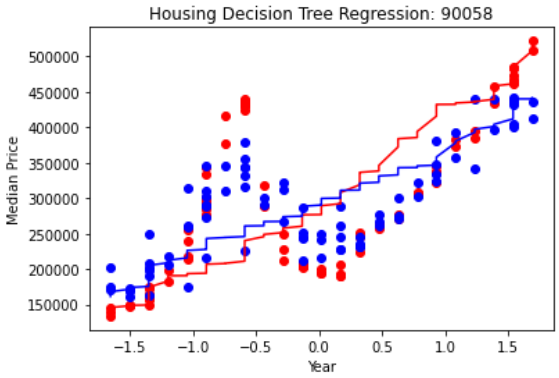
4.1.4.1.3.

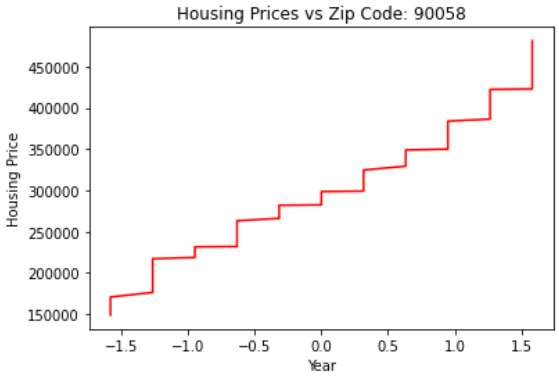


4.1.4.1.3.1. Zip Code: 91040

RSME: 70367.57183122

4.1.4.1.4.



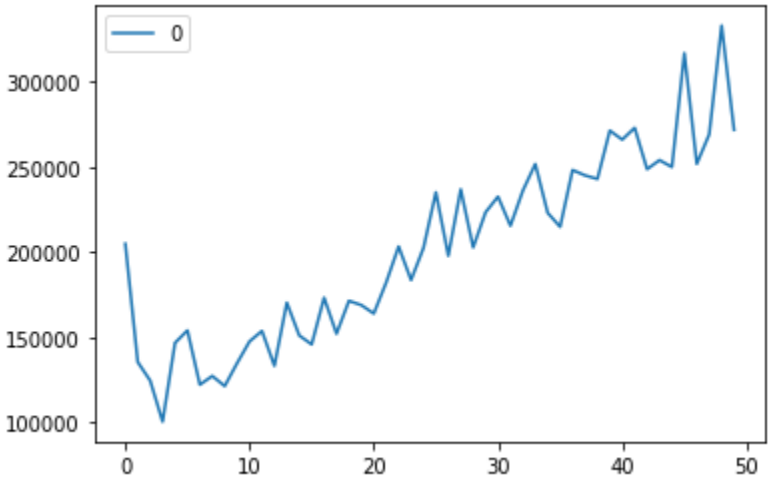


4.1.4.1.4.1. Zip Code: 90058

RSME: 52824.9460249

4.1.4.2. General Results:

RMSE versus K plot



This is a graph of the RMSE value with a given K, and as one can obviously see, K values from around 5 seemed to be more effective. For our final K value we used 7, as it gave the lowest RMSE value of approximately 122176.

For our general results, we saw some similarities with a linear increase of pricing, but from the perspective of a house-buyer (or maybe even a real

estate agent) the results clearly say one thing: The longer one waits to buy, the more prices increase. There are some fluctuations with growth and actual value that do affect the predictions, but since we were looking for a more general prediction and not an economical prediction machine, these fluctuations will not affect our goal.

4.2. Comparison of the Algorithms

4.2.1. General Comparison

4.2.1.1. Each algorithm gives us a similar general result in that we know home prices will be increasing over the next decade.

4.2.1.2. From the graphs, we can see that a few of the models are far more accurate in predicting median home prices on the test dataset. In particular, the decision tree, random forest, and KNN models had overall lower RMSE values.

4.2.1.3. We feel that scaling our values played an important factor in training our models, and resulted in much better results overall compared to our un-scaled linear regression model.

4.2.1.4. We relied on RMSE values to determine the efficacy of each model and compare them to one another. We analyze these values and give our final conclusion next.

4.2.2. Final Results

4.2.2.1. The variance in home prices being broken down by zipcode make it difficult to declare a direct front runner for the models as a whole. To better evaluate our models, we created a dataframe consisting of the RMSE values for each of the zipcodes based on each algorithm.

	zipcodes	Linear Regression	Linear Regression Scaled	Decision Tree Depth 5	Decision Tree Depth 7	Random Forest	KNN
0	90732	99200.108761	86358.673373	25387.049465	12969.562735	10333.737608	70941.068675
1	90272	330363.591777	352375.507209	77285.269608	56492.784044	69554.261430	200093.193128
2	91040	97714.014850	90689.530395	23379.956006	10080.874967	8186.924112	90611.164226
3	90068	114834.368152	114298.624932	41786.945389	28822.824927	21367.283531	114070.044604
4	91602	138913.760404	142432.591570	32974.631768	21750.591783	14839.494347	103623.009983
5	91345	95590.513605	106924.368053	24130.707575	15519.690527	14873.801975	96193.679322
6	91436	183697.626455	190248.248598	68332.446437	34503.319713	20148.288133	169917.343266
7	90293	185786.189308	173096.469969	43548.200050	20768.662432	25833.085202	173753.628773
8	90077	257646.477721	262610.802986	70007.396270	43226.570268	40130.490406	263549.279692
9	90058	79885.704146	81156.849861	19497.323665	9338.647161	8986.199107	46560.752811
10	90011	89313.768417	87344.707402	16615.033866	9090.897138	7493.814100	66218.475662
11	91331	96109.746988	94523.249967	19866.349481	10210.506290	8926.909494	75068.097607
12	90044	85563.512996	92346.217207	19696.373798	8523.633273	7164.140281	71222.160362
13	91342	99973.008364	97885.847287	19612.366648	10680.861516	9297.014064	70310.516212
14	91335	101290.762763	108072.444785	29344.129136	13073.464369	12752.528097	76545.375845
15	90003	77713.637311	81889.575979	16264.230644	7202.705538	9446.482853	66323.584681

4.2.2.2. Looking at the final RMSE values, we can see that the most consistent top performing algorithm is random forest. There are a few instances where decision tree with depth 7 give us lower values, and overall they are quite close in performance. We have listed the best performing algorithm for each zipcode below.

4.2.2.3. For 90732 - Random Forest.

4.2.2.4. For 90272 - Decision Tree with depth 7.

4.2.2.5. For 91040 - Random Forest.

4.2.2.6. For 90068 - Random Forest.

4.2.2.7. For 91602 - Random Forest.

4.2.2.8. For 91345 - Random Forest.

4.2.2.9. For 91436 - Random Forest

4.2.2.10. For 90293 - Decision Tree with depth 7.

4.2.2.11. For 90077 - Random Forest.

4.2.2.12. For 90056 - Random Forest.

4.2.2.13. For 90011 - Random Forest.

4.2.2.14. For 91331 - Random Forest.

4.2.2.15. For 90044 - Random Forest.

4.2.2.16. For 91342 - Random Forest.

4.2.2.17. For 91335 - Random Forest.

4.2.2.18. For 90003 - Decision Tree with depth 7.

4.2.2.19. In conclusion, we got some interesting results when comparing these machine learning algorithms against each other when attempting to predict future median home prices. We think that the accuracy of Random Forest in particular shows a lot of promise in delivering a general idea of where housing prices will be across LA county over the next 10 years. We are well aware that changes to the economy, inflation, mortgage interest rates, and more will play a very large factor in these future prices and we look forward to seeing how these predictions and models hold up over time.

5. Team Member Responsibility

5.1. Data Extraction

5.1.1. James Yokley worked on retrieving the dataset and started a dataframe for all of us to work on.

5.2. Data Mining

5.2.1. We all worked on analyzing data to see results and double check.

5.3. Prediction Model Accuracy

5.3.1. James Yokley, Bruck Negash, and Nathan Campos worked to check the accuracy of the model.

5.4. Data Visualization

5.4.1. Anthony Edeza and Joshua Chang doubled checked chart visualization to see if it worked.

5.5. Final Report and Slides

5.5.1. James Yokley, Bruck Negash, Anthony Edeza, Nathan Campos, and Joshua Chang all worked on the final report and slides.