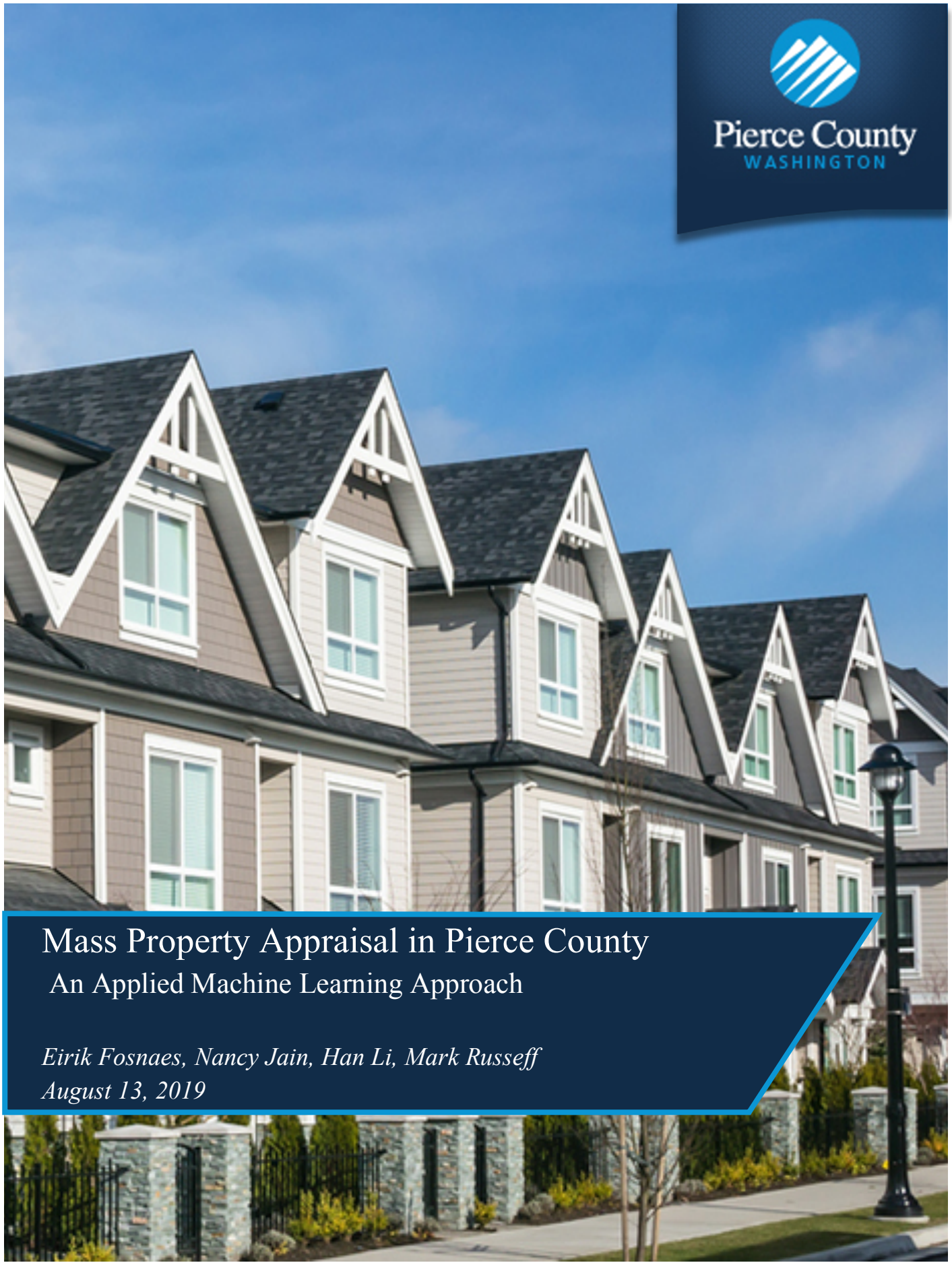


Mass Property Appraisal in Pierce County

An Applied Machine Learning Approach

Eirik Fosnaes, Nancy Jain, Han Li, Mark Russeff

August 13, 2019

TABLE OF CONTENTS

EXECUTIVE SUMMARY

In 2018, Pierce County lost approximately $12,709,635 in tax revenue due to underestimating the value of residential properties and overcharged taxpayers roughly $2,150,440 due to overestimation. Pierce County's data demonstrates the importance of accurate mass real estate appraisals not only in terms of maximizing tax revenue but also in terms of fairness. Many municipalities all over the country have begun using Automated Valuation Methods (AVM) based on machine learning models in order to provide more accurate mass appraisals at a lower cost. Academic research has found that in most cases machine learning based AVM's outperform traditional statistical approaches. (Zurada, 2011). In Wake County, NC a research paper was published using their housing data that showed that Artificial Neural Networks (ANN) do a better job of predicting house prices than traditional linear hedonic valuation models (Peterson, 2009). Today, Wake County is one of the first counties in the country using these AVM's to perform their mass appraisals. Using Pierce County data from 2018 this research paper will show that Pierce County can significantly improve the accuracy of their mass appraisals by integrating community level data into their models, clustering their data for better segmentation and implementing machine learning based AVM's. The 2018 Mean Absolute Percentage Error (MAPE) for residential property in Pierce County is 16.87% and the MAPE for the best performing approach in this paper is 9.89%--that's an almost 7% decrease in the error rate. This can be achieved by first using a data set containing community quality data like the number of crimes in the area, number of nearby parks and the average rating of schools within two square miles. Then that data is split into segments using a K-Means clustering algorithm which will give us three distinct segments of properties; older homes, newer homes, and large homes. Having these specific segmentations helps make it easier to accurately predict housing prices using random forest machine learning models. The 7% decrease in the error rate experienced with this strategy equates to an increase in tax revenue of around 880,000 dollars and a decrease in the overtaxing of property owners by almost 150,000 dollars; not to mention the lower implementation cost to taxpayer of an AVM based system.

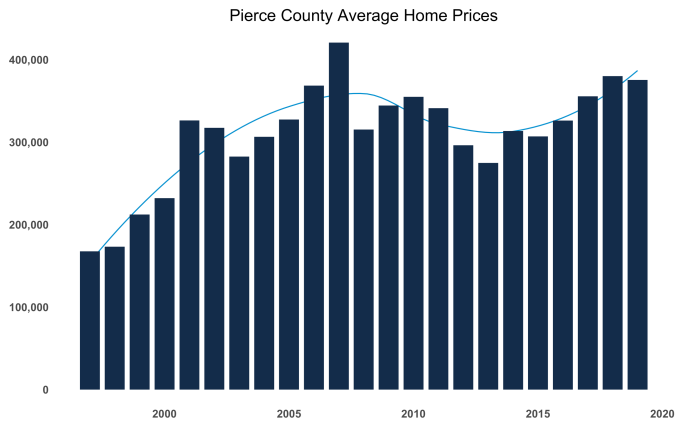
I. INTRODUCTION

Technology has become a disruptive force in many industries, everything from transportation to banking. However, the real estate appraisal industry has been particularly slow in adapting to the digital age. While we may use technology to find a home and get a mortgage, when residential home appraisals are conducted they must be conducted manually by a licensed residential appraiser. This professional uses the condition of the property, comparable home sales in the area and other statistics to come up with an opinion about the value of the property. For these appraisals a median error rate of roughly 4% would be considered within the acceptable range. (Harney, 2019) In contrast, mass appraisals of property values which are conducted by governments for tax purposes, often have even larger error rates due to the fact that, by law, they are only allowed to assess properties in groups. While these mass appraisals use large datasets and statistical methods to determine home values, it is believed that they could be improved upon by using more sophisticated Automated Valuation Methods (AVM). These AVM's are usually based on sophisticated statistical learning models and machine learning algorithms; examples of such models include estimation tools created by Zillow and Redfin. These estimators are currently reporting median error rates significantly below the mean absolute percentage error (MAPE) of 16.87% that was observed in Pierce County, WA in 2018. In fact, in Pierce County, Redfin's median error rate is currently an impressive 1.38%. (Redfin, 2019) If AVM's can be proven to outperform the current methods for mass appraisals then perhaps more sophisticated models should be implemented in order to improve the overall accuracy of real estate appraisals at a substantially lower cost than that of the current system. In fact, some municipalities are already adopting such AVM's in order to keep up with more dynamic real estate markets. Wake County, NC recently enlisted the expertise of software company SAS in order to use machine learning to assess the value of roughly 400,000 properties using 100 different market variables. The goal of Wake County is to increase its "productivity by automating repetitive tasks and increase their accuracy through quality data analysis, sound decision making and the elimination of errors." (Smith, 2018) While these more accurate appraisals may result in additional tax revenue for the county, many argue that the focus is on accuracy, fairness and cost savings.

This research paper builds upon prior research and develops several different AVM's in order to predict the sale price of real estate in Pierce County, WA based on the properties characteristics as well as local crime, nearby parks and the quality of local schools. Then the MAPE of the AVM's will be examined and compared to the error rates reported by Zillow and Redfin as well as those of the mass appraisals reported by Pierce County. The goal is to identify the model that most accurately predicts

the sale price of real estate in Pierce County, i.e. the model with the lowest MAPE. The price of real estate is notoriously difficult to predict, likely due to the myriad of hard to quantify social forces that influence price fluctuations. It can be observed in Figure 1.0 that the average price of real estate in Pierce County had increased significantly from 1997 up until the peak in 2008, then the Great Recession occurred and prices fell dramatically. However, since 2013 the real estate market in Pierce County has seen a steady increase. Such market dynamics make mass appraisals difficult, though it has been shown that AVM's using machine learning can more accurately predict market prices than manual mass appraisals alone.

FIGURE 1.0 AVERAGE YEARLY SALES PRICE



PROBLEM STATEMENT

Due to the current political climate and dynamic real estate market in Pierce County it is important that the County's mass appraisal system is both accurate and efficient; with the correct predictive machine learning models and careful variable selection Pierce County can have a more accurate system of mass appraisal at a much lower cost to the taxpayer.

II. DESCRIPTION OF DATA

PROPERTY DATA

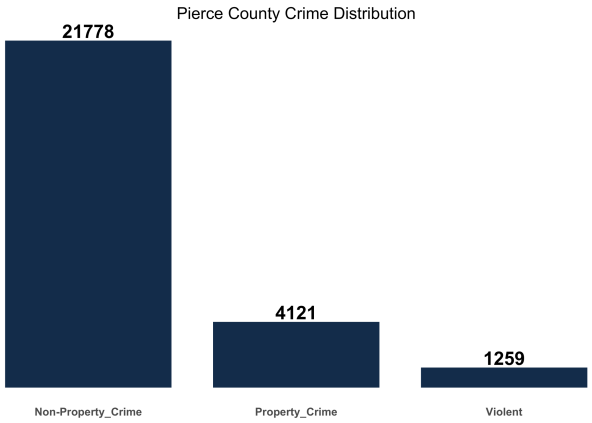
The property information data for this study was acquired from the Pierce County Assessor-Treasurer's Data Mart. This data consists of six tables: Appraisal Account, Improvement, Improvement Built-as, Land Attribute, Sale, and Tax Account. The Appraisal Account table contains 64.45MB of data distributed across 24 attributes and 331,383 rows of data. This table contains important information such as the size of the property [Land Net Acres] as well as information about the type of utilities available to the home; [Utility Electric], [Utility Sewer] and [Utility Water]. The Improvement table contains 55.39MB of data which includes 344,251 rows and 25 attributes. The attributes of most interest in this table are the [Property Type] as well as several attributes pertaining to square footage measurements such as but not limited to [Square Feet], [Balcony Square Feet], [Basement Square Feet], and [Attached Garage Square Feet]. The Improvement Built-as

table contains 56.11MB of data with 346,978 rows and 26 attributes. The most significant attributes from this table pertain to the physical structure of the building and contain many of the standard variables for a real estate price evaluation such as; [Bedrooms], [Bathrooms], [Stories] and [Year Built]. The Land Attribute table is 28.69MB and contains only three attributes but 552,549 rows of data. The [Attribute Description] variable is a possible source of text mining for additional property characteristics that may affect the value of the home such as whether the property has a pool. The Sale table is our largest table with 94.77MB of data and 524,625 rows of prior sale data from 1997 to 2019. For this analysis the most important attributes of this table are the [Sale Price] and [Sale Date] variables as they will be used to test our model's error rate. Lastly, there is the Tax Account table which contains 60.98MB of data which includes 343,673 rows of data across 28 different attributes. The most significant attributes for this analysis are the attributes related to the appraisal value of the property for the last two years, [Total Market Value- Prior Year] and [Total Market Value- Current Year], which will be used to calculate the error rate of Pierce County's current mass appraisal method. This error rate will be the benchmark that will be used to evaluate the performance of the various mass automated valuation methods employed in this study. All six of these tables can be joined by the [Parcel Number], at which point the significant attributes will be identified and other insignificant attributes will be discarded. The [Parcel Number] variable will also be used to join the property information table to additional community characteristic tables with the Tax Parcels joining table.

COMMUNITY QUALITY DATA

Additional community data pertaining to the local crime rate, parks and schools was also acquired directly from Pierce County via the Pierce County Open Data portal. This additional data consists of three tables and a joining table: Crime Data, Schools, Park Points and Tax Parcels.

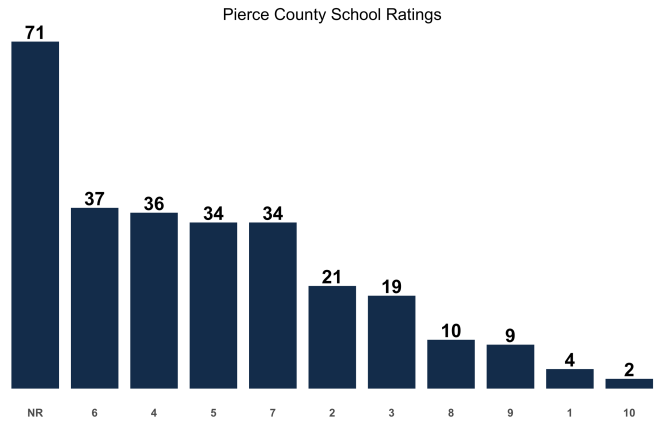
FIGURE 2.0 CATEGORICAL CRIME COUNTS



The Crime Data table contains 4.62MB of data, including 27,158 rows of reported crime activity represented by 10 attributes that include the date/time of the crime [OccuredOn], the type of crime [Public_Nam], as well as several location-based attributes. The Crime Data will be aggregated by the severity of the crime committed and categorized into three groups represented above in Figure 2.0. The Park Points table contains 120KB of data which includes 348 rows representing various locations of parks throughout the county. In the

Park Points table there are 47 different attributes, however our analysis will be focused only on the [ParkType] and location variables [X_Coord] and [Y_Coord]. The Schools table contains 64KB of data that includes 295 rows of basic information about the schools in Pierce County. The Schools table has 14 attributes including important location identifiers like the GIS coordinates; [X_Coord] and [Y_Coord]. These coordinates will be used to relate the locations of crimes and schools to the property locations from the property information data sets. In order to do this, we must use a joining table that contains the [Parcel_Number] as well as the [X_Coord] and [Y_Coord] of each property. This joining table, Tax Parcels, contains 17.5MB of location data across 327,647 rows of tax parcels and contains only the relevant location information that will be used to join the tables based on a radius of 2 miles. The Schools data will also have a [Rating] variable that was acquired from Zillow that will be manually added to the table; number of schools by rank can be observed in Figure 2.1. For each ParcelNumber in the data, a count of crimes and parks within a two square mile radius will be added as well as the average school rating within the same radius.

FIGURE 2.1 SCHOOL RATING COUNTS



III. LITERATURE REVIEW

NEURAL NETWORK HEDONIC PRICING MODELS IN MASS REAL ESTATE APPRAISAL

In this 2009 paper, published in the Journal of Real Estate Research, the researchers set out to examine the effectiveness of artificial neural networks (ANN) at predicting property values. In order to do so they constructed several different ANN's and evaluated their dollar pricing errors versus those of traditional linear hedonic pricing models. The researchers argue that neural networks are more robust to model misspecification and pricing errors than linear hedonic models and should be used more widely in mass appraisals of real estate.

The data set used in this study consists of a sample of 46,467 real estate transactions containing 18 property characteristics taken over the six-year period from 1999-2005. This data was taken from a database of over 180,000 sales observations for Wake County, NC. The data used in this study is the same sale price data Wake County uses for their

own property evaluations and taxation calculations. As mentioned above, Wake County has now adopted a machine learning approach to mass appraisals after the publication of this study.

In order to train their ANN and estimate the parameters of the linear model with OLS the researchers constructed a randomly selected training set from the data. The remaining data was used as the hold-out test sample. Model pricing performance was then presented as the differences in root mean squared errors (RMSE), mean absolute pricing errors (MAPE), and the linear model's R^2 statistics. The models were compared via the pricing error differential, which is the difference between the ANN and linear absolute forecast errors on a property by property basis. A large positive differential would indicate a performance edge for the ANN.

The study found that the size of the pricing error differential increased over time. This was due to the fact that the ANN performed better as the size of the data set increased whereas the linear model's R^2 began to flatten out at a certain point. In more recent years the researchers observed an error differential that exceeded 1.5% of the property's value per year. While this may not sound like a lot, it can be significant on a larger scale. The conclusion of the study was that linear hedonic valuation models produce avoidable valuation costs which occur primarily to nonlinearities within the model. Due to these nonlinearities the paper argues that nonlinear models such as an ANN will generate greater precision, significantly lower dollar pricing errors and will perform better in volatile pricing environments than a traditional linear model. (Peterson, 2009)

A COMPARISON OF REGRESSION AND ARTIFICIAL INTELLIGENCE METHODS IN A MASS APPRAISAL CONTEXT

This paper, another published in the Journal of Real Estate Research, is a comparative study that uses several different regression and machine learning models in order to assess property values in Louisville, Kentucky. The researchers implemented one traditional regression model, three non-traditional regression models and three artificial intelligence (AI) based models. These models were used to predict real estate prices in various simulation scenarios.

The data set used for this study contains over 309,000 properties with 143 descriptive variables and was provided to the researchers by the chief tax assessor of Louisville, Kentucky. In addition to this assessment data the researchers were also provided with a data set containing over 16,000 sales transactions. These large data sets gave them the ability to conduct a more comprehensive comparative study than had been conducted in the past.

For the regression methods the researchers used one traditional multiple regression as well as three non-traditional regression methods that included a support vector machine (SVM) which is a classification method that is based on the separation

of decision planes in multidimensional space as well as additive regression which is a method that combines many contributions from various models. Lastly, a particular form of a decision tree was used that uses linear regression at each leaf node to predict values at that node, these trees are known as M5P trees. The models used for the AI portion of the study consisted of neural networks (NN), radial basis function neural network (RBFNN) as well as memory-based reasoning (MBR).

The above models were all applied to five different simulation scenarios in order to test them based on five different error measures; MAE ($), RMSE ($), RAE (%), RRSE (%), and R^2. The study found that while the AI based models, particularly the NN, performed well when presented with data that was not very homogeneous, the additive regression and M5P tree models performed the best overall across all simulation scenarios. These findings are of particular interest to the mass appraisal community because both the additive regression and M5P models are not well known and are therefore not currently being regularly implemented in mass property valuations. (Zurada, 2011)

IMPACT OF ARTIFICIAL NEURAL NETWORKS TRAINING ALGORITHMS ON ACCURATE PREDICTION OF PROPERTY VALUES

This paper examines the use of artificial neural networks (ANN) in mass appraisals of property value via a comparative performance study. The researchers compared the performance of ANN's against more traditional linear, semi-log and log-log models. For the ANNs, various training methods were used including the Levenberg-Marquardt, back propagation, conjugate gradient, scaled conjugate gradient, and Powell-Beale Conjugate Gradient algorithms. These ANN's were then evaluated against the traditional regression models based on a test of predictive accuracy and a set of traditional model metrics.

The data set used for this study was provided by the Office of the City Valuation Officer in Cape Town, South Africa and contains 3,526 property transactions with 46 property variables. However, after cleaning the data the researchers were left with 3,232 transactions and 11 variables for their analysis.

In order to establish their baseline for their ANN models the researchers first constructed and tested the various regression models. They found that in most cases the log transformed regression models performed better than the linear regression models. This was in line with the findings of many prior studies that suggested linear model are ineffective at predicting real estate prices. (Peterson, 2009) More specifically, the semi-log model performed the best when using the traditional model evaluation metrics used in previous studies; MAE ($), RMSE ($), RAE (%), RRSE (%), and R^2. (Zurada, 2011) For the ANN's all of the training methods were evaluated and the Levenberg-Marquardt trained ANN (LMNN) was selected as the most optimal ANN. The LMNN and semi-log regression were then evaluated across two comparative tests. The first

test assessed the model's accuracy by testing if they could maintain a 50% accuracy within 10% of the assessed value of the property. The semi-log regression was not able to meet this accuracy benchmark but did manage to do better when the threshold was raised to 20% of assessed value. The LMNN was able to meet the 50% benchmark at 10%; outperforming the semi-log regression. The second test conducted by the researchers involved comparing the traditional model evaluation metrics mentioned above between the two models, once again the LMNN model performed slightly better than the semi-log regression. However, even though the LMNN model outperformed the semi-log model in both tests it did fail the important explainability test. In real estate it is often required that appraisals be explainable and defensible in legal proceedings. Due to their black box nature ANN's do not produce very transparent estimates and therefore are not currently implemented in mass appraisals. This means that due to their transparency, ease of use and explainability the semi-log regression was decided to be the preferred technique for mass appraisals in this paper. (Yacim, 2018)

HOUSING VALUE FORECASTING BASED ON MACHINE LEARNING METHODS

Big Data is becoming more and more important in almost every industry and real estate is no different. Machine learning, in particular, has recently gained in popularity and people of different industries are using machine learning to solve problems, identify trends and generate future predictions.

It is very important to be able to accurately predict housing prices so that the government can make well informed urban planning decisions. In this paper the researchers are using housing data from various Boston suburbs with 13 features, including per capita crime rate. Previous studies using Artificial Neural Networks have shown that growth of real estate prices is correlated with unemployment and consumer prices. This paper argues that ANN's are not an optimal method to use for predicting housing prices, because its accuracy is low and convergence speed is far from ideal. Since ANN has multiple equilibrium positions, ANN may be trapped in a local minimum problem in optimization, which makes it difficult to obtain global optimal value. Instead, this paper uses different models including SVM, LSSVM and PLS to analyze housing values.

The best performing model for this sample group was the Support Vector Machine, which had a mean square error estimate of 10.7373, while the other two both took longer to run and had a larger mean square of error estimate. SVM performed better than LSSVM, largely due to LSSVM being a slightly more simplified mathematical version of SVM. The was also strong linearity in the dataset, which made the PLS algorithm have a low computation accuracy.

Several machine learning models should be constructed and analyzed, as this is a very important

part of any analysis. The models should also be combined with corresponding characteristics of testing data to predict the housing values. The prediction results of the machine learning methods will vary and give different outputs to further analyze. (Mu, 2014)

BIG DATA IN REAL ESTATE? FROM MANUAL APPRAISAL TO AUTOMATED VALUATION

Real estate is one of the largest assets in the world and predicting housing prices is one of the most challenging tasks and assessing the value of real estate is worth millions of dollars.

Various measures are used to determine housing prices but they still require a lot of work. In addition, the real estate sector faces significant cost from the appraisal bureaucracy. “Better” appraisals and indexes also do not address the cost inefficiency stemming from traditional, manual appraisals. To address the imprecision and inefficiencies of property appraisal, there is significant increase in machine learning techniques and availability of data. In this paper various models and their methodology is discussed. These models can sift through millions of combinations of thousands of variables, training and testing the model on randomly selected parts of the datasets, leading to precise out-of-sample tests of predictive performance.

Hedonic Model: Hedonic pricing is a model which identifies price factors according to the premise that price is determined both by internal characteristics of the good being sold and external factors affecting it. A hedonic pricing model is often used to estimate quantitative values for environmental or ecosystem services that directly affect market prices for homes. The hedonic pricing model has many advantages, including the ability to estimate values, based on concrete choices, particularly when applied to property markets with readily available, accurate data. At the same time, the method is flexible enough to be adapted to relationships among other market goods and external factors. Hedonic pricing also has significant drawbacks, including its ability to only capture consumers’ willingness to pay for what they perceive are environmental differences and their resulting consequences. For example, if potential buyers are not aware of a contaminated water supply or impending early morning construction next door, the price of the property in question will not change accordingly. Hedonic pricing also does not always incorporate external factors or regulations, such as taxes and interest rates, which could also have a significant impact on prices.

Automated Valuation Models: AVMs have been an integral part of real estate technology since Zillow launched Zestimate that automatically analyzes various data points to produce an estimate on the current value of a home or property. Importantly, the model does not depend on the use of a capitalization rate (the cap rate, in jargon terms),

which is critical in traditional property valuation techniques. However, they are only estimates and can be misleading and they do not take human elements into consideration since they automated.

Machine learning Approach: We can apply decision tree models as well. A regression tree algorithm finds the best predictors from the set of independent variables by first minimizing the variance of a regression between each combination of the dependent and an independent variable. This yields the order of importance of the variables. Each of the explanatory variables then represents a node in the decision tree. Decision trees have several advantages over hedonic regression models and other machine learning techniques. For instance, decision trees are simple to understand and to interpret, and statistical significance can easily be calculated. Decision trees are also able to handle categorical variables, so there is no need to (manually) create dummy variables. Decision trees can make predictions in a very short computing time, even with large amounts of data. Despite these advantages, decision trees have some limitations. They are more complex and they encounter problems of overfitting and underfitting.

These models can be tested using cross validation approach. Cross-validation is a technique that is used to evaluate model performance on unseen data. In general, a dataset is divided into train and test datasets. The train dataset is used to build the prediction model, and the test dataset is used to evaluate the model. We are using AVM approach, the AVMs estimated in this article are distinct from traditional models not just because of more advanced modeling techniques but also because of the use of nonstandard explanatory variables, including hyperlocal measures of economic activity. (Kok, 2017)

HOUSE VALUE, CRIME AND RESIDENTIAL LOCATION CHOICE

Buying a house is one of the most important and significant decision in one's lifetime. Americans on average spend 30% of their income on housing. People select the place where they live for different reasons, such as a new job opportunity or to be together with their family. Two of the most important factors to consider while buying a house is schooling and crime. Crime is further divided into property and violent crime. Violent crime includes murder and non-negligent manslaughter, robbery and aggravated assault. Property crime sums up burglary, larceny-theft and motor vehicle theft. Americans are willing to pay more to move to an area with lower violent crime and higher property crime since they can spend more on security devices and construction to ensure safety. The choice set that the author defined in the paper is defined at the level of metropolitan area. Also, the author described how people are willing to pay more to move to a location with lower violent crime occurrences and are willing to pay more to move to a place with higher property crime, however, the effect of violent crime is larger than property crime. Property crime had a negative effect on housing

prices. The paper addresses the fact that this could potentially be due to the fact robberies and property damage occur in areas with more expensive real estate. The author has also emphasized on number of police in an area. The obvious evidence is that the occurrence of crimes is related closely to the number of police in a location. In reality, an increase in police number does not indicate a decrease in the crime rate, thus in this paper author use two treatments. One is the increase in police numbers of per thousand people and the other is the decrease in crime rate.

While crime is one significant factor that homebuyers may consider when thinking about a new purchase, local public-school quality is another key factor, and one that has also been the subject of significant academic research. Much like crime, school quality is highly correlated with other unobservable characteristics, including quality of the neighborhood, so it is difficult to determine causality via regression analysis. For houses not near any type of school, crime is negatively associated with housing price, an expected result. However, for houses near schools, higher levels of crime predict higher property values. Thus, the model depicted by the paper predicts that homeowners purchasing homes near schools are less sensitive to crime than those not purchasing near schools.

Due to reasons mentioned above it is fair to imply correlation between housing and crime and schools. According to the paper crime is negative when on analyzing housing that is not within a half-mile radius of a school. However, the picture becomes more complex when analyzing homes near schools. Instead of assuming that homebuyers near schools put a premium on crime when making a purchasing decision, it is more likely to assume that higher crime rates are correlated with higher housing prices due to the fact that they are more prone to nonviolent property crimes, including robberies and motor vehicle theft. We should use more sophisticated models to determine the significance of factors like crime and schools to analyze housing prices. (Zhang, 2015)

CRIME AND RESIDENTIAL CHOICE: A NEIGHBORHOOD LEVEL ANALYSIS OF THE IMPACT OF CRIME ON HOUSING PRICES

This paper looks at the effect of housing prices using housing, crime and demographic data in Columbus, Ohio using data from 1995 to 1998.

Crime serves as an important factor for determining change in socio-economic composition of areas. Crime can be an early indicator of change in housing prices of a neighborhood. Studies have shown that fear of crime leads people to move from the city to suburbs, which can leave areas of concentrated poverty which can serve as an early indicator of a decline of a neighborhood, and naturally also the house values.

Home ownership is often viewed as a way to build wealth, but threats to its value may limit the appeal of investing in a house. Crime is one of those threats that may reduce the desirability of home

ownership in neighborhoods with a high rate crime. Examining housing prices will therefore serve as an ideal measure of how desirable a neighborhood is based on change in crime. Housing markets are often localized and citywide changes in crime will distort the picture of how crime impacts housing prices for any particular neighborhood. The type of crime is also an important factor. Although property crime is the most common crime, violent crime and changes in violent crime have the largest negative impact on housing values across all neighborhoods. Socio-economic factors matter when it comes to non-victims and their choice of residential living area. Katzman (1980) found strong evidence that perception of crime plays an important role in determining where people are likely to move. Especially families with children and high-income individuals were most sensitive to crime when choosing their area of residence. This can lead to a difficult cycle of increased poverty and crime in certain areas.

Using hedonic regressions this paper measured house prices as a function of the characteristics of the house, its location and both level and changes in crime rates. Dark figures (non-reported crime) makes the results misleading according to the authors. They found sign of the coefficient to be opposite of what was expected, for instance total crime were found to actually increase housing prices. Including homicide, which is unlikely underreported gives a model that makes more sense intuitively. This concluded that higher rate of crime reduces housing values. (Tita, 2006)

CRIME AND PROPERTY VALUES: EVIDENCE FROM THE 1990S CRIME DROP?

This paper is to examine the relationship between crime and property values by exploiting the unique crime drop that occurred in the 1990s. A significant relationship between changes in crime rates and property values is estimated through this analysis. 1990s crime drop is a type of natural experiment due to its certain features including that the large and unexpected drop, substantial variation in crime decreases in multiple areas, and the stable society condition during this time period. Based on this natural experiment across the entire United States, this analysis could be regarded as a national setting.

The results indicate a strong relationship between changes in crime rates and property values although determining causality remains problematic. This robust model indicates an increase of 7.5–9.5% in housing values due to the reduction in crime for these select zip codes. It also suggests an increasing housing values of 14.5– 19.5% for the top decile of zip codes in terms of property crime reduction.

The magnitude of the crime impact on property values is quite striking. The estimated elasticities of housing values with respect to crime range from −0.15 to −0.35. The violent and property crimes, and the changes in the levels or changes in the percent of crime are associated with the value of house. This analysis also indicates that the reduction of crime translated into an average gain of around $2000 per house. Our estimates suggest that in these zip codes, the housing price gain from decreased crime was closer to $11,000 for each house. The

average price gains were even larger when calculated looking at changes in property crime. These estimates would suggest per house property value gains of between $11,000 for all zip codes and between $20,000 and $27,000 for houses in the top decile of property crime.

From this analysis, what we learnt is we could use the change of crime in a certain area, such as by zip code, when estimating the value of house. At the same time, the we could measure those crime as different levels to better understand how those crimes affecting to the value of house. It is strongly indicated that the change of crime will affect the value of house. However, in our capstone project, it might be difficult to find the data about the change of crime in Pierce County. In order to better understand the relationship between crimes and the value of house, what we could try to do is to set up a measurement to divide the crimes into different levels either by zip code or by magnitude in a certain region. (Pope, 2012)

HOUSE PRICES AND THE QUALITY OF PUBLIC SCHOOLS: WHAT ARE WE BUYING?

Per pupil expenditures, teacher-student ratios, average class sizes, and test scores by school district and often by individual school are the common information relating to the interest in the quality of local schools. Based on the various interests in the quality of school, it was commonly assumed that the value of house relates to the differences in the quality of local schools.

As next steps, we need to understand how to measure if the quality of school is high or low? Analysis has found that better reputation of schools may increase the value of house in those districts. However, it is difficult to measure the reputation of a school. It has many dimensions, including physical appearance, library facilities, quality of teachers, students' academic performance, and the range of extracurricular activities, school resources or student performance, to estimate the school premium in house prices. While school resources and the composition of the student body are two main possible reasons to influence the value of house.

In terms of school resources, expenditures per pupil are the standard measure. Analysis has found that the prices of similar houses are higher in school districts with higher expenditures per pupil. However, using expenditures as a measure of school quality does not directly measure the output of school's education level, but more input process, such as the expenditures to reduce the number of students per class, enlarge the classroom, improve the electrical equipment of education and so on. Therefore, the achievement of student seems the primary measurement for home buyers when they indicate the quality of public schools.

Composition of the student body could be regarded as a measurement of average students' performance. A school's performance could be typically

measured by how well it fulfills the immediate goals of primary and secondary education, which includes furthering artistic and vocational skills, fostering good work habits and civic awareness, and imparting academic knowledge. The scores from standardized tests, such as a given grade on some standard reading, math, or general academic test, could be generated as a common measure of student achievement by comparing academic performance across schools or school districts. Higher achievement is associated with higher house prices. The greater the improvement in average reading levels, the higher were neighborhood house prices.

More researches strongly indicate that the wide using of extra resources could result in an enhancement of the quality of education and thereby contribute to higher house prices. At the same time, the empirical evidence also shows that academic achievement can be improved by the peer group effect.

In our capstone project, we could start to find information about the average score and also the average resources input for each school and divide them into different levels. Then we could count the number of the different levels in a certain region to find the relationship between the school and the value of house in Pierce County. (Crone, 1998)

THE EFFECT OF SCHOOL QUALITY ON RESIDENTIAL SALES PRICE

When people buy a home, a number of factors influence their decision. The look of the home, as well as its size, layout, age, and proximity to amenities are all important, depending on the buyer. The local school district is a factor with significant influence. We've always known that good schools attract families with school-age children. The purpose of this research paper is to find the specific measure of school quality that is appropriate in the housing market as a proxy for school quality, and how much this measure of school quality is capitalized. Ordinary Least Squares method is used to find the significance and capitalization of school quality. Different websites offer test scores, rankings, and demographic information, including student diversity by race and gender, the percentage of students on free lunch programs, as well as the student-teacher ratio. The paper has used these statistics to develop an opinion of the schools and school districts and hence how these factors impact housing prices.

According to the paper the achievement test scores of the third grade reading level were used as the index of school quality, which had a positive effect and increase in housing prices by 5.2% in the full sample and 6.2% in the non-innercity sample. The effect of racial composition had a negative effect on housing prices but was statistically not significant when the school quality variable is included (race was statistically significant when the school quality variable was excluded from the model). Further empirical studies have confirmed that test scores are preferred to the school input measure of expenditure per pupil as a reliable measure of school quality in housing studies. The author employed a boundary

fixed effect, which assumes that neighborhood characteristics (shopping, etc.) are similar across municipal or other district borders. This approach further isolates the effect on housing price due to school quality through a comparison of housing prices on one side of the street affiliated with a specific school, to prices on the other side of the same street affiliated with a different school. However, boundary fixed effect is biased and hence it is not preferred.

This paper shows that that good schools do increase home values in some measure. Half of the home-buying population is willing to pay more than their intended budget to get into the right school district, and more than half would give up other amenities. This paper also analyzed the role of public-school quality as an influential variable in housing price. Empirical studies have shown that the various measures of school quality have a substantial impact on housing prices, but it is still debatable which measure of school quality is most appropriate. Several measures of school quality that home buyers value include school input factors (teachers' average salary and experience and expenditure per pupil), output factors(fourth grade math proficiency rate, school district designations, and performance index), value-added, comprehensive performance measures, and efficiency factors. (Seo, 2009)

IV. DATA PREPROCESSING

PROPERTY DATA

Prior to joining the property data tables, each of the tables was individually evaluated and any variables with observed Null ratios above 75% were removed. However, the variables [Waterfront Type] and [View Quality] were not removed due to the relations ship between waterfront property and home values. For these variables the Null values were assumed to represent non-waterfront property and were therefore replaced a value of "Non-Applicable". In addition to removing variables with high Null Ratios, variables with no correlation to sale prices or variables with extremely high correlation to other variables were also removed. These variables included but were not limited to: [Deed Type], [Grantor], [Grantee], [Longitude], [Latitude], [ETN] and [Business Name]. Once the initial set of important variables was identified the property data tables were joined into one master table. The master table initially consisted of 546,119 observations and 58 total variables.

Once the tables had been joined additional data cleaning was performed. This included the following numerical coding of several categorical variables:

[Waterfront Type]

- Any defined waterfront type = 1
- Non-Applicable = 0

[View Quality]

- "View Lim -" = 1
- "View Lim" = 2
- "View Lim +" = 3
- "View Avg" = 4
- "View Avg +" = 5
- "View Good" = 6
- "View Good" = 7
- "View Good +" = 8
- "View V-Good" = 9
- "View V-Good +" = 10
- Non-Applicable = 0

[Street Type]

- Paved = 1
- Unpaved = 0

[Condition]

- "Uninhabitable" = 0
- "Extra Poor" = 1
- "Very Poor" = 2
- "Poor" = 3
- "Fair" = 4
- "Average" = 5
- "Good" = 6

[Quality]

- "Low" = 0,
- "Low Plus" = 1,
- "Fair" = 2,
- "Fair Plus" = 3,
- "Average" = 4,
- "Average Plus" = 5,
- "Good" = 6,
- "Good Plus" = 7,
- "Very Good" = 8,
- "Very Good Plus" = 9,
- "Excellent" = 9

Additionally, other variables were coded in order to represent 1 if square footage was present or a 0 if not. These variables were [Attic Finished Square Feet], [Basement Square Feet], [Carport Square Feet], [Porch Square Feet], [Attached Garage Square Feet], and [Detached Garage Square Feet]. This was done to account for the presence of these additional home features but without considering their overall size.

The master table was then filtered by sale date in order to limit the data to property sales that took place only in 2018. Property type was also filtered and only residential properties were kept. This resulted in 77,796 rows of data that contained a significant number of Null values; these were removed from the data set. In addition, all variables other than [Sale Price] that were related to value were removed in order to avoid multicollinearity. In order to address any possible issues with outliers in the sale prices the final sale price of properties was limited to sales between $100,000 and $3,000,000.

Once all of the cleaning was complete and all relevant variables were selected the master property data set consisted of 15,668 observations and 28 attributes which included: 'ParcelNumber', 'SalePrice', 'Valid', 'Confirmed', 'Improved', 'TaxCodeAreaPriorYear', 'Township', 'LandNetSquareFeet', 'LandGrossFrontFeet', 'BuiltAsSquareFeet', 'Stories', 'StoryHeight',

'Bedrooms', 'Bathrooms', 'Units', 'YearBuilt', 'YearRemodeled', 'AtticFinishedSquareFeet', 'BasementSquareFeet', 'CarportSquareFeet', 'PorchSquareFeet', 'AttachedGarageSquareFeet', 'DetachedGarageSquareFeet', "WaterfrontType", "ViewQuality", "Condition", "Quality", and "Fireplaces". In order to avoid any issues with multicollinearity the correlations of the variables in the master were checked and the results can be observed in figure 4.0.

COMMUNITY QUALITY DATA

There was not a significant amount of cleaning required for the community data. The challenge in with the community data was writing the algorithm that would count the occurrences of any of the variables with a radius of the parcel number. However, there was some preliminary steps that were required with the crime data. Since there was a significant number of different crimes listed in the data, various crimes were categorized into more general residential property focused sets.

Property Crime

- Arson - Residential, Burglary - Residential, Robbery - Residential, Vandalism - Residential

Non-Property Crime

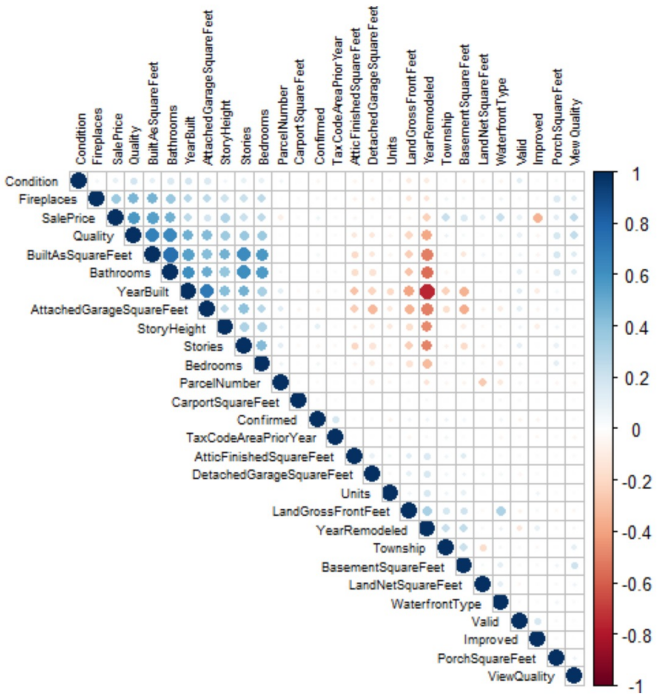
- Arson - Non-residential, Burglary - Non-residential, Criminal Traffic, Drug Possession (Methamphetamine), Drug Possession (Other), Drug Sale/Manufacture (Methamphetamine), Drug Sale/Manufacture (Other), Fraud or Forgery, Intimidation, Liquor Law Violations, Motor Vehicle Theft, Possession of Stolen Property, Telephone Harassment, Theft - Gas Station Runout, Theft - Mail, Theft - Other, Theft - Vehicle Prowl, Theft -Shop lifting, Trafficking in Stolen Property, Vandalism - Non-residential, Warrant Arrest

Violent

- Assault - Aggravated, Assault - Simple, Homicide, Robbery - Business, Robbery - Other, Robbery – Street

For each parcel number a count of each type of crime with in a two square mile radius was then calculated. This process was then repeated with the parks data were only the presence of a park was counted; the type of park was irrelevant. Lastly, with the school data, school ratings were manually added to the schools data set using information from Zillow. These rating were then used to calculate the average school rating of the schools with two square miles of the parcel number. Once the counts were completed the data set was joined with the master data set, resulting in a data set with 15,668 observations and 33 attributes. These two tables will be the source for all of the following analysis. The data with only property information will be referred to herein as model_data whereas the data with the community data will be referred to as model_data_counts.

FIGURE 4.0 CORRELATION MATRIX



CLUSTERING FOR SEGMENTATION

K-Means clustering is a popular unsupervised machine learning algorithm that is often used for classification problems which have unlabeled data. However, another application of clustering is in the data preprocessing step where is can be used to segment the data into more manageable and easier to predict subsets. In this case the data will be segmented into three distinct clusters that will be used in our analysis. After complete a K-Means clustering algorithm with end up with three different clusters; we can see the distribution of the clusters in Figure 4.3. Based on the summary statistics of the data below there are certain characteristics that can be garnered from the clusters.

FIGURE 4.1 CLUSTER YEAR BUILT TABLE

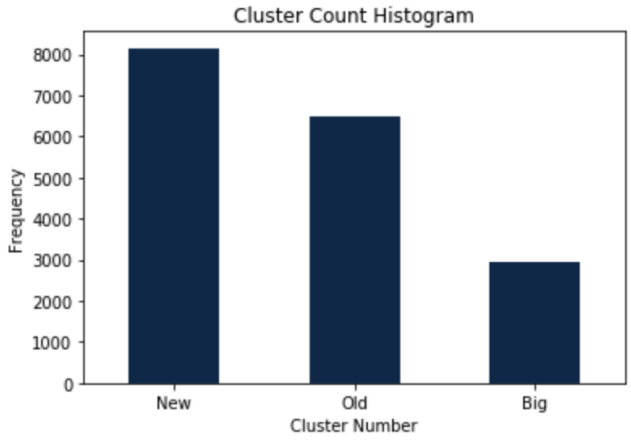
Year Built	Mean	1st quartile	3rd quartile	Std.Dev
Cluster0 Old	1947	1925	1968	25.21
Cluster1 New	1999	1992	2011	15.71
Cluster2 Larger	2000	1992	2017	23.16

FIGURE 4.2 CLUSTER SALE PRICE TABLE

Sale Price	Mean	1st quartile	3rd quartile	Std.Dev
Cluster0 Old	$303,301	$240,404	$343,434	$114,125
Cluster1 New	$361,487	$310,101	$404,040	$78,307
Cluster2 Larger	$627,613	$493,372	$690,530	$235,796

Based on Figures 4.1 & 4.2 we can see that we have a cluster of older houses, one of newer houses and one of larger more expensive houses.

FIGURE 4.3 CLUSTER DISTRIBUTION GRAPH



V. DATA MINING MODELS AND EVALUATIONS

EVALUATION METRICS

In this project, various AVM's were used in order predict the value of property; decision trees, random forests and neural networks. The metrics used for evaluation were the Mean Squared Error (MSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE).

MSE or Mean Squared Error measures the average squared error of our predictions. For each point, it calculates square difference between the predictions and the target and then average those values. The lower this value, the better the model is. MSE is a valuable tool for model evaluation because inherent in its calculation is both the bias and variance of the model; MSE results can be found in the appendix code section.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

MAE or Mean Absolute Error measures the error by calculating the average of absolute differences between the target values and the predictions. The MAE is a linear score which means that all the individual differences are weighted equally in the average. For example, the difference between 10 and 0 will be twice the difference between 5 and 0.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Mean Absolute Percent Error (MAPE) is a useful measure of prediction accuracy and is defined by the difference between actual and predicted value and is divided by the actual value. Since MAPE is a measure of error, high numbers are bad and low numbers are good.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|Actual_t - Forecast_t|}{Actual_t} * 100$$

In order to evaluate the performance of the models across various clustered data the weighted average of cluster results is calculated using the formula below where m = metric and r = # of rows for each cluster.

$$Weighted\ Avg = \frac{m_1 * r_1 + m_2 * r_2 + m_3 * r_3}{\sum r_i}$$

DECISION TREE

In a decision tree when dealing with a regression problem real valued numbers are predicted at the leaf nodes. In a standard classification tree, the idea is to split the dataset based on homogeneity of data. A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). On the other hand, in a regression tree, since the target variable is a real valued number, we fit a regression model to the target variable using each of the independent variables. Then for each independent variable, the data is split at several points.

MODEL

The following parameters were used for the decision tree regressor model: **Criterion** is used to measure the quality of a split. Criterion is set to "mse" for mean squared error which is used for variance reduction and to minimize least square errors. **Splitter** is the strategy used to choose the split at each node. The parameter "best" was chosen for the purpose of selecting the best split. **Max_depth** is the maximum depth of the desired tree. Since the target variable is a real valued number, max_depth was set as None, which means, the nodes are expanded until all the leaves are covered. **min_samples_split** is the minimum number of samples required to split an internal node. The default value has been kept, which is 2. **min_samples_leaf** represents the minimum number of samples required to be at a leaf node which was kept at 1 since the target variable is continuous. This will smoothen the regression model at each node. **max_features** is the number of features to consider when looking for the best split. In this case max_features is set to "auto", which means all the features have been considered for the best split. **random_state** is the seed used by the random number generator. For the purpose of consistency, the random_state has been kept at 0 throughout the code. **max_leaf_nodes** represent the maximum number of leaf nodes allowed. It has been set to None which means there is an unlimited number of leaf nodes allowed.

EVALUATION

Dataset Name	MAE	MAPE
model_data	*$56,581.81*	*16.31%*
model_data_counts	*$51,018.97*	*14.20%*

CLUSTERS

Dataset Name	MAE	MAPE
cluster0	*$53,764.93*	*18.55%*
cluster0_counts	*$50,224.18*	*17.80%*
cluster1	*$33,304.10*	*9.65%*
cluster1_counts	*$28,523.14*	*8.61%*
cluster2	*$107,859.21*	*17.53%*
cluster2_counts	*$93,897.91*	*14.97%*

WEIGHTED AVERAGE OF CLUSTERS

Dataset Name	MAE	MAPE
cluster_data	*$52,276.51*	*14.25%*
cluster_data_counts	*$47,425.89*	*13.06%*

RANDOM FOREST

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges as to a limit as the number of trees in the forest becomes large. They offer efficient estimates of the test error without incurring the cost of repeated model training associated with cross-validation. However, they are not easy to visualize, and they have high computational cost.

MODEL

Random Forests parameters are quite similar to that of decision trees, as random forests are nothing but combination of decision trees. **n_estimators** define number of trees in the random forest. We have set n_estimators as 1000. **Max_depth** is set to 20, otherwise the computation time will be high for large data sets.

EVALUATION

Dataset Name	MAE	MAPE
model_data	*$43,041.97*	*12.8%*
model_data_counts	*$39,648.27*	*11.31%*

CLUSTERS

Dataset Name	MAE	MAPE
cluster0	*$45,425.96*	*15.98%*
cluster0_counts	*$37,142.25*	*13.49%*
cluster1	*$25,049.42*	*7.39%*
cluster1_counts	*$21,531.39*	*6.67%*
cluster2	*$77,465.38*	*12.21%*
cluster2_counts	*$65,267.55*	*10.92%*

WEIGHTED AVERAGE OF CLUSTERS

Dataset Name	MAE	MAPE
cluster_data	*$41,305.34*	*11.37%*
cluster_data_counts	*$35,581.08*	*9.89%*

NEURAL NETWORK

We used Python to apply the two different neural networks in our output. Neural Networks are often used in business to make predictions or find patterns

that other models are less likely to identify. The inventor of neural networks, Dr. Robert Hecht-Nielsen defines this as "a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs". Neural networks are organized in layers with a pre-determined number of nodes. The nodes contain an activation method, where we tested different ones and ended up with 'relu' as our activation method. In this project we have used one and two layered neural networks, and then tested for number of nodes to find the best output of the algorithm. The data has been divided into training and test data, where 70% was training data and the remining 30% has been used as a test data to test the validation of our models. For each evaluation method we have tested for number of nodes and used MAE to find the optimal number of nodes. The prediction outputs for the neural network has proved to be overall slightly higher than the two other models. Adding the community data proved to help the model for every single test we ran, which means that our hypothesis that these variables would help the model was correct.

EVALUATION

Dataset Name	MAE	MAPE
model_data 1 hidden layer	$59,757.08	17.55%
model_data_counts 1 hidden layer	$56,580.35	16.08%
model_data 2 hidden layers	$48,679.33	14.23%
model_data_counts 2 hidden layers	$46,133.86	13.22%

CLUSTERS ONE HIDDEN LAYER

Dataset Name	MAE	MAPE
cluster0	$57,596.57	20.19%
cluster0_counts	$50,128.39	18.09%
cluster1	$38,917.08	11.19%
cluster1_counts	$35,773.74	10.82%
cluster2	$242,681.71	39.12%
cluster2_counts	$224,753.74	37.06%

WEIGHTED AVERAGE OF CLUSTERS

Dataset Name	MAE	MAPE
cluster_data	$79,743.94	19.17%
cluster_data_counts	$72,540.81	17.87%

CLUSTERS TWO HIDDEN LAYERS

Dataset Name	MAE	MAPE
cluster0	$49,232.77	17.39%
cluster0_counts	$42,179.65	15.21%
cluster1	$31,378.29	9.09%
cluster1_counts	$28,871.99	8.79%
cluster2	$91,645.63	13.97%
cluster2_counts	$86,133.86	13.22%

WEIGHTED AVERAGE OF CLUSTERS

Dataset Name	MAE	MAPE
cluster_data	$48,008.68	12.97%
cluster_data_counts	$43,321.72	11.90%

VI. INTERPRETATIONS

DECISION TREE

From the plotted decision trees, we can see that the branches extend a lot because we have not limited the maximum length. This might result in overfitting. The categorization in all the datasets started with Quality, for example for model dataset quality less than or equal to 4.5 belongs to one category and greater than 4.5 belongs to another category. For quality less than or equal to 4.5, decisions are further taken on the basis of improved, township, builtassquarefeet and bedrooms. Similarly, for quality greater than 4.5 category, decisions are further taken. These trees are further split until all the probable values are split. Samples in each node are number of observations, and values are ranges of sales values which each node of the tress falls into.

In terms of MAE and MAPE values decision tree performed best on cluster1_counts dataset, whereas, cluster2 has highest value of MAE and model_data has highest value of MAPE. We have assigned weights to the clusters and added their weighted evaluations to further decide if clusters helped us in improving the performance of the model or not. Weighted average evaluations further prove that, overall, decision tree performed better on clustered model and it improved the MAPE value by 2% approximately for both model and model_counts dataset.

RANDOM FOREST

It is difficult to interpret and visualize random forest like decision trees. Hence, we have focused only on MAE and MAPE values here. Random forest gives the best results among all the models. All values are lowest for cluster1_counts, which means random forest performed best with cluster1_counts. However, in terms of MAE cluster2 gives the highest value and MAPE is highest for cluster0. Further, weighted average values prove that, overall, random forest performed best with clustered model and it improved the MAPE value by 2% approximately for both model and model_counts dataset.

We also wanted to find out which features are most important in determining the prediction, apart from wanting to know what our model's house price prediction is. Another thing to note is that the more accurate our model is, the more we can trust feature importance measures and other interpretations. Hence, we used feature importance in case of random forest models on all the datasets since it has lowest error.

NEURAL NETWORK

We started of calculating the Mean Absolute Error (MAE), Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) for the full dataset without the community variables. We ended up using more than 100 nodes for each of our models, because this gave a better output. The highest number of nodes we used was 750, because we experienced diminishing return after this number. We mostly focused on the MAPE, because that is the number, we will use to compare our model to Pierce County's own data and the national average.

The two layered neural network model performed slightly better than the single layered neural network for out models. Without the community data the MAPE was 14.23%, while it dropped to 13.22% mean error with the community data.

This shows that our theory ahead of running these models were correct and community data does improve the accuracy of prediction Sales Price of houses in Pierce county.

After looking at the clusters`, we found that cluster 1 proved to have the lowest prediction accuracy error, while cluster 2 was by far the worst, overall this made the average weight still perform better than before clustering. Cluster 2 had a very high accuracy error for the single layered neural network but adding the second layer improved the accuracy of our model by 24%. This was by far the largest gap between the two neural network models.

VII. DISCUSSION

MODEL DISCUSSION

This research paper was conducted in two main parts in order to examine the effect of community data and clustering on a models' predictive accuracy. The data shows that the no matter the model the data set containing the community data outperformed the data set without. This is an important finding in the context of mass real estate appraisals because most organizations use only traditional property metrics in order to appraise home values. Until quite recently the idea of using community or demographic data in predicting property values was purely academic. However, with the proliferation of machine learning algorithms capable of handling extremely large amounts of data more research is being conducted on mass appraisals using both community and demographic data.

The second part of our research examined the application of clustering as a data preprocessing technique in order to segment data into more similar subsets for model building. In order to test this

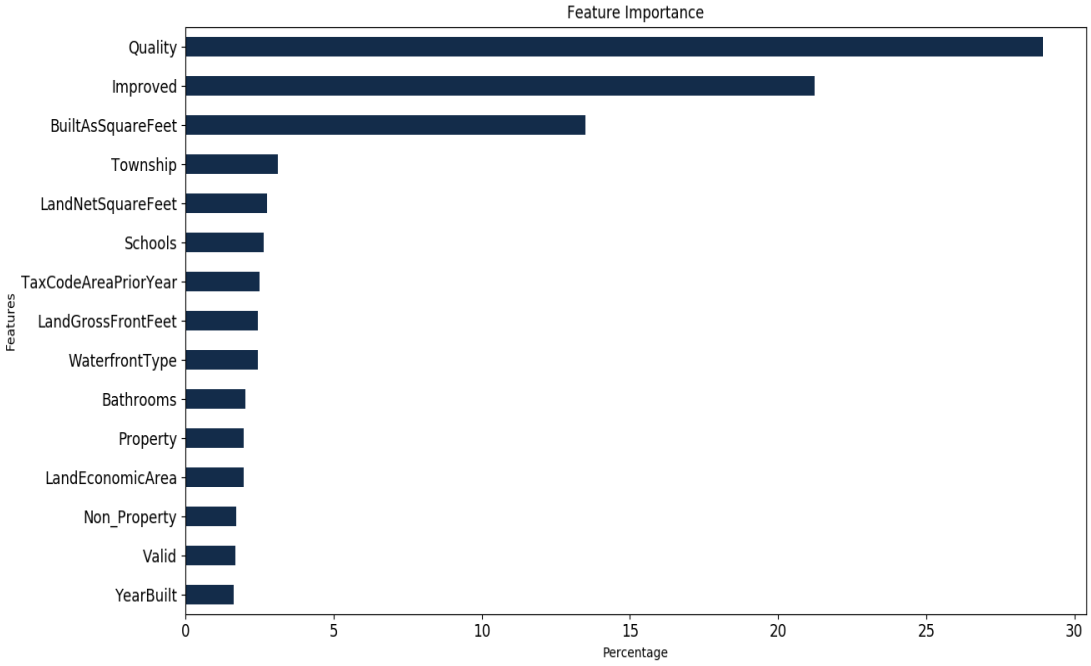
application of unsupervised learning we created three distinct clusters of data that represented older homes, newer homes and large homes. The machine learning algorithms were then applied to those data subsets and weighted averages for each were calculated. The study found that clustering does decrease the error rate of the machine learning models when used as a preprocessing technique. This application of machine learning is especially suited to unsupervised learners like K-Means clustering. By clustering the data we were also able to extract some additional insights about the data. For instance, it was observed that older houses were more difficult to accurately predict their value. This is likely due to the varying condition of older homes as well as the homogeneous nature of newer homes. Perhaps this suggests that a more powerful algorithm may be required for predictions concerning older homes.

Lastly, our research observed that no matter the data set the random forest model outperformed all of the other models. In many cases the random forest outperformed the next best model in terms of MAPE by roughly 2%. While that not seem like much, when dealing with millions of dollars like Pierce County, it can have a significant effect. These finding go against many of the finding of other studies on this subject, however, in most of those cases a random forest regressor was not considered.

FEATURE IMPORTANCE

One of the most significant benefits of a random forest model is that the determination of feature importance is straightforward. With our models we found that the most important feature was the Quality of the property. This was somewhat surprising as one would expect square footage or another proxy for size to be the most important feature. A graph visualizing the importance can be viewed in Figure 7.0. Another interesting finding in terms of feature importance is that school ratings is the most important feature of the community data. The quality of nearby schools is more important than the number of crimes near the property. Among the crime related data, property crime was the most important feature. This makes intuitive sense when thinking about the relationship between home values and property related crimes. This feature importance data will also be important for feature selection in future studies.

FIGURE 7.0 FEATURE IMPORTANCE



PIERCE COUNTY RECOMMENDATIONS

It is recommended that Pierce County use this research as a roadmap for implementing a machine learning based AVM moving forward. In 2018, Pierce County underestimated a significant amount of their appraisals which resulted in 12.7 million dollars in missed tax revenue. Even worse, tax payers overpaid 2.15 million dollars in property taxes due to overestimating property values. The MAPE for residential property in Pierce County was calculated as 16.87% in 2018 and by following our roadmap this study achieved a MAPE of 9.89%. The applied machine learning techniques in this study would result in Pierce County increasing their yearly property tax revenue roughly $880,000 and it would save tax payers almost $150,000 in overpaid property taxes. Additionally, tax payers would also save because implementing an AVM for mass appraisals has a much lower cost than conducting manual mass appraisals. For Pierce County an AVM based on random forest regressors with clustering segmentation and community data would be more accurate, fair and have a cheaper implementation cost than their current system.

VIII. CONCLUSION

This paper set out to use applied machine learning techniques in order to improve the current state of real estate mass appraisals by examining 2018 sales data from Pierce County, WA. For each step throughout this process discoveries were made and new insights discovered. First, this research demonstrated that when community level data is included in machine learning based AVM's error rates decrease. This was true for all of the different machine learning algorithms that were applied to the data. Additionally, it was observed that among the community variables included in this study the quality of the schools in the area was the most important feature. The quality of schools in the area is usually an important factor for homebuyers and based on the findings of this study it also an important factor in predicting home prices. Another area where this study observed improvements in the prediction accuracy is when clustering is used as a preprocessing tool in order to better segment the market. In this case, the segments broke up properties into older homes, newer homes and very large homes. This resulted in weighted accuracy numbers that were better than the accuracy numbers for the full data sets. Within these segments the newer homes were easier to predict, likely due to the more homogenous nature of newer homes, whereas the older homes were much more difficult to predict. Across the board, the most accurate model no matter the data set was the random forest. This goes against some of the previous research that suggested that neural networks performed better. Further research should be conducted using more sophisticated machine learning methods like deep learning or ensemble learning methods like boosting and stacking. Additional research can also examine the effectiveness of clustering based on density using a DBSCAN algorithm and the provided GIS coordinate data. This type of clustering could account for city vs. rural pricing and could lead to more accurate price predictions. Additionally, there is a significant amount of data on the Pierce County open data portal, although we were somewhat constrained by time, with more time one could include many additional community variables. Also, our research did not consider demographic information about the home owner, that information could also be used as features in future studies. If further studies can validate the findings of this study, then Pierce County could implement one of the AVM's for their mass appraisals and possible save millions of dollars in the process. The findings in this paper are both interesting and impactful. For Pierce County, it gives them a pathway to more accurate, fair and inexpensive methods of mass appraisal. For researchers, it provides a roadmap for constructing effective prediction algorithms that can be used in future research

REFERENCES

Crone, T. M. (1998). House prices and the quality of public schools: what are we buying?. *Business Review*, *9*(10), 3-14.

Harney, K. R. (2019, May 20). Who provides the more accurate home valuation: Zillow or Redfin? Retrieved from https://www.chicagotribune.com/real-estate/ct-re-0224-kenneth-harney-20190224-story.html

Kok, N., Koponen, E. L., & Martínez-Barbosa, C. A. (2017). Big Data in Real Estate? From Manual Appraisal to Automated Valuation. *The Journal of Portfolio Management*, *43*(6), 202-211.

Mu, J., Wu, F., & Zhang, A. (2014). Housing value forecasting based on machine learning methods. In *Abstract and Applied Analysis* (Vol. 2014). Hindawi.

Peterson, S., & Flanagan, A. (2009). Neural network hedonic pricing models in mass real estate appraisal. *Journal of Real Estate Research*, *31*(2), 147-164.

Pope, D. G., & Pope, J. C. (2012). Crime and property values: Evidence from the 1990s crime drop. *Regional Science and Urban Economics*, *42*(1-2), 177-188.

Redfin (2019). About the Redfin Estimate. Retrieved from https://www.redfin.com/redfin-estimate

Seo, Y., & Simons, R. (2009). The effect of school quality on residential sales price. *Journal of Real Estate Research*, *31*(3), 307-327.

Smith, R. (2018, August 20). Wake County tax team enlists high-tech SAS help to tackle property assessments. Retrieved from https://www.wraltechwire.com/2018/08/20/wake-county-tax-team-enlists-high-tech-sas-help-to-tackle-property-assessments/

Tita, G. E., Petras, T. L., & Greenbaum, R. T. (2006). Crime and residential choice: a neighborhood level analysis of the impact of crime on housing prices. *Journal of quantitative criminology*, *22*(4), 299.

Yacim, J. A., & Boshoff, D. G. B. (2018). Impact of artificial neural networks training algorithms on accurate prediction of property values. *Journal of Real Estate Research*, *40*(3), 375-418.

Zhang, Z., & Hite, D. (2015). *House Value, Crime and Residential Location Choice* (No. 1375-2016-109528).

Zurada, J., Levitan, A., & Guan, J. (2011). A comparison of regression and artificial intelligence methods in a mass appraisal context. *Journal of Real Estate Research*, *33*(3), 349-387.

DATA RESOURCES

Appraisal Account: https://www.co.pierce.wa.us/736/Data-Downloads

Tax Account: https://www.co.pierce.wa.us/736/Data-Downloads

Improvement: https://www.co.pierce.wa.us/736/Data-Downloads

Improvement Built As: https://www.co.pierce.wa.us/736/Data-Downloads

Land Attribute: https://www.co.pierce.wa.us/736/Data-Downloads

Tax Parcel Data: https://gisdata-piercecowa.opendata.arcgis.com/datasets/tax-parcels-2

Crime Data: https://gisdata-piercecowa.opendata.arcgis.com/datasets/crime-data

School: https://gisdata-piercecowa.opendata.arcgis.com/datasets/schools?orderBy=PRS_ID&orderByAsc=false

Parks: https://gisdata-piercecowa.opendata.arcgis.com/datasets/parks-points

APPENDIX

DATA DICTIONARY

Dataset	Field Name	Type	Size	Description	Type of Data	Null Ratio	Domain Values	Example Values
SALE	ETN	Varchar	11	Recording number issued by the Auditor's Office when a property transaction is recorded. Excise tax must be paid before the document of conveyance can be recorded or the mobile home title transferred.	nominal	0%	0000001 ~ W993185	0094166
SALE	Parcel Count	Integer		Number of parcels associated with a sale.	nominal	0%	1 ~ 591	1
SALE	Parcel Number	Varchar	10	Unique 10 digit number assigned to each property.	nominal	0%	0019012000 ~ 9900400300	2955520120
SALE	Sale Date	Date		Date the legal document (deed) was executed.	ordinal	0%	1997-01-01 ~ 2019-06-29	2001-03-01
SALE	Sales Price	Decimal	15,2	Dollar amount recorded on the ETN.	numeric	0%	0.00 ~ 368586669	424500
SALE	Deed Type	Varchar	40	Type of document which conveyed an interest in or legal title to the property.	nominal	0%	Deed Type	Real Estate Contract

SALE	Grantor	Varchar	80	Individual(s) or company conveying ownership or interest in the property described on the deed.	nominal	2.73%	Company/Individual Conveying	Eason Scott M & Sharon M
SALE	Grantee	Varchar	80	Individual(s) or company purchasing ownership or interest in the property described on the deed.	nominal	2.66%	Company/Individual Purchasing	Miller Michael D
SALE	Valid	Varchar	7	Used to code the validity of a sale for appraisal purposes. A sale in the open market between two unrelated parties, each of whom is reasonably knowledgeable of market conditions and under no undue pressure to buy or sell would be considered a valid transaction. However, a sale can be a valid transaction but if the conditions of the sale fit one of the 27 deletion categories identified in the State Ratio RCW, the sale would be coded invalid for Assessor purposes. (i.e. sale between relatives, estate sale, percent interest, etc.)	nominal	0%	0-1	0
SALE	Confirmed	Varchar	11	Identifies if the property characteristics at the time of the sale and or circumstances of the sale were verified by Assessor-Treasurer staff.	nominal	0%	0-1	1
SALE	Exclude Reason	Varchar	30	State required description of why a sales transaction must be excluded from sales studies. Examples include non arms length sales transactions such as "Family different last names", sale transactions with undue pressure such as "Estate sale" and "Forced Sale Trans in Lieu Frcl", and sales where the property characteristics have changed since the sale, "Improved after sale".	nominal	0%	Exclude Reason	Family same last name
SALE	Improved	Varchar	8	If any parcel in the sale has any building value it is coded as improved. Vacant is vacant land with no building value.	nominal	0%	0-1	0
TAX PARCEL	Tax Parcel Number	Varchar	50	Unique 10 digit number assigned to each property.	nominal	0%	1017251003 ~ 9900400320	19012000
TAX PARCEL	Zip Code	Varchar	10	ZIP code is a number consisted of five digits from 0 to 9. The USA is divided into geographical areas and the first digit of zip code represents one of these areas.	nominal	0.86%	0 ~ W91LY	98115
TAX PARCEL	X_Coordinate	Decimal	15,4	X Coordinate in the map	numeric	0%	1059302.752 ~ 1400565.848	1078759.338

TAX PARCEL	Y_coordinate	Decimal	15,4	Y Coordinate in the map	numeric	0%	516062.4932 ~ 763055.6818	677675.0854
SCHOOL	OBJECTID	Varchar	10	Internal feature number	nominal	0%	1480 ~ 1774	1480
SCHOOL	NAME	Varchar	80	Common name of the school	nominal	0%	School Name	All Saints School - Fife
SCHOOL	ADDRESS	Varchar	80	Site Address that requires emergency response from Pierce County GIS	nominal	0%	School Address	2323 54TH AVE E
SCHOOL	CITY	Varchar	20	Name of city	nominal	0%	City of the school	FIFE
SCHOOL	ZIP	Varchar	10	Zip code	nominal	0%	98092 ~ 98580	98424
SCHOOL	DISTRICT	Varchar	20	School district	nominal	0%	District, not applicable	PENINSULA
SCHOOL	DIST_NO	Varchar	5	District number	nominal	20%	1 ~ 417, Null	10
SCHOOL	TYPE	Varchar	12	Type of school	ordinal	0%	UNIVERSITY ~ Primary	UNIVERSITY
SCHOOL	PRS_ID	Varchar	5	Id assigned to School by the Pierce Responder System	nominal	0.34%	1~479	
SCHOOL	GRADE	Varchar	20	Grade Level for Private Schools	ordinal	0%	Pre-8~Higher Education, Month-Year	Pre-6
SCHOOL	X_COORD	Decimal	15,4	X Coordinate in the map	numeric	0%	1076175.107~1262718.637	1262718.637
SCHOOL	Y_COORD	Decimal	15,4	Y Coordinate in the map	numeric	0%	522526.274~756115.9929	756115.9929
APPRAISAL ACCOUNT	Parcel Number	Varchar	10	Unique 10 digit number assigned to each property.	nominal	0%	20011002~122312012	20011002
APPRAISAL ACCOUNT	Appraisal Account Type	Varchar	15	How a parcel is classified.	nominal	0%	Account Type	Residential
APPRAISAL ACCOUNT	Value Area ID	Varchar	10	Identifies the Physical Inspection cycle for each parcel. Ranging from PI0 to PI6	ordinal	0%	PI0 ~ PI6	PI1
APPRAISAL ACCOUNT	Land Economic Area	Varchar	30	This code, also know as LEA, ties parcels to land valuation models. For Commercial parcels, it identifies the land use (commercial, industrial, multi unit) model by PI year. For Residential parcels, it is a location code identifying the appraisal area, sector and subsector. For Res Com Condos it relates directly to the account land value.	nominal	0%	1 ~ 182402, 104A, 205A, 303A, 4F8E, Null	182402
APPRAISAL ACCOUNT	Buildings	Integer		Represents the number of improvements (buildings) located on a parcel.	numeric	0.66%	0~110, Null	1
APPRAISAL ACCOUNT	Group Account Number	Varchar	30	Identifies and ties all parcels qualifying for valuation as one economic unit (contiguous assessment) constituting	nominal	89.59 %	2~830209752, Null	2

				the highest and best use of the property. Also identifies and ties all parcels in a condominium project.				
APPRAISAL ACCOUNT	Land Gross Acres	Decimal	15,4	Total acres that make up the economic unit which could include multiple parcels.	numeric	0%	0~3033.4	0.2755
APPRAISAL ACCOUNT	Land Net Acres	Decimal	15,4	Size of the individual parcel in acres.	numeric	0%	0~3033.5	0.2755
APPRAISAL ACCOUNT	Land Gross Square Feet	Decimal	15,4	Total square feet that make up the economic unit which could include multiple parcels.	numeric	0%	0~132134904	28263
APPRAISAL ACCOUNT	Land Net Square Feet	Decimal	15,4	Size of the individual parcel in square feet.	numeric	0%	0~132134905	28263
APPRAISAL ACCOUNT	Land Gross Front Feet	Decimal	15,4	For commercial parcels, the front feet of the site facing the arterial. For residential parcels, the front feet of the lot facing the address side. If a waterfront parcel, this field may represent the effective waterfront length of either an individual lot, or a combination of lots when under single ownership. The method will vary based on the valuation approach used.	numeric	0%	0~144330	90
APPRAISAL ACCOUNT	Land Width	Integer		Residential Use Only. Equal to the Net Effective Waterfront Front Feet of the individual parcel, or if using a Group Account allocation method, then the sum of Net Effective Waterfront Front Feet for the group.	numeric	0.50%	0~23232, Null	90
APPRAISAL ACCOUNT	Land Depth	Integer		Residential Use Only. Equal to the Net Effective Depth of the individual waterfront parcel, or if using a Group Account allocation method, then the sum of Net Effective Depths for the waterfront group. Effective depth is measured from the top of the bank (where bulkhead begins) to the inland property line.	numeric	0%	0~3200	90
APPRAISAL ACCOUNT	Submerged Area Square Feet	Integer		Area of a parcel that is submerged. Not included in the Gross or Net Square Feet	numeric	83.30 %	0~313638, Null	27
APPRAISAL ACCOUNT	Appraisal Date	Date		Indicates the most recent that date an appraiser physically observed the property and/or made changes to the record.	ordinal	3.90%	09-12-2001 ~ 06-20-2019	6/12/19
APPRAISAL ACCOUNT	Waterfront Type	Varchar	30	Describes the type of waterfront the property	nominal	95.70 %	Waterfront Type	WF Salt

				adjoins or has legal access to.				
APPRAISAL ACCOUNT	View Quality	Varchar	30	Assigned to reflect the market appeal of the overall view available from the dwelling or property.	nominal	89.58 %	Good, Avg, V-Good, Lim +/-	View Good +
APPRAISAL ACCOUNT	Utility Electric	Varchar	30	Identifies if power is installed, available or is not available on the property.	nominal	0%	Power Available/Installed/No-comment	Power Available
APPRAISAL ACCOUNT	Utility Sewer	Varchar	30	Identifies if sewer/septic is installed, available or not available or if the property does not support an on site sewage disposal system (no perc).	nominal	0%	Aval/Installed/No/No Perc	SEWER/SEPTIC NO
APPRAISAL ACCOUNT	Utility Water	Varchar	30	Identifies if water is installed, available or is not available.	nominal	0%	Aval/Installed/No	WATER NO
APPRAISAL ACCOUNT	Street Type	Varchar	30	Identifies if the access street is paved or unpaved. If this field is blank, the parcel could be a reference parcel or a mobile home assessed as personal property.	nominal	0%	PAVED/STREET NO ROAD/STREET UNPAVED	PAVED
APPRAISAL ACCOUNT	Latitude	Decimal	10,5	The latitude of the parcel centroid. Some parcels may not have latitude/longitude coordinates (building-only, mineral rights and reference parcels, as well as others not represented in the GIS system).	numeric	0%	46.86791~47.40339	47.15017
APPRAISAL ACCOUNT	Longitude	Decimal	10,5	The longitude of the parcel centroid.	numeric	0%	-122.83068 ~ -122.48457	-122.48457
IMPROVEMENT BUILTAS	Parcel Number	Varchar	10	Unique 10 digit number assigned to each property.	nominal	0%	19121002~217231009	19121003
IMPROVEMENT BUILTAS	Building ID	Varchar	5	Unique identifier for each improvement record on a parcel. Note, not all improvements are on a separate record and may not always be in sequence.	nominal	0%	0~822	1
IMPROVEMENT BUILTAS	Built-As Number	Integer		Unique number that identifies each Built-As on an improvement. This is a sequential number assigned by the system starting with the largest Built-As being number 1.	nominal	0%	1~7	1
IMPROVEMENT BUILTAS	Built-As ID	Integer		Number associated with the description of the purpose or style of construction of the building.	nominal	0%	1~1499	13
IMPROVEMENT BUILTAS	Built-As Description	Varchar	255	Original purpose for and or style of construction of the building. The Built As is associated with the Marshall and Swift cost and depreciation tables.	nominal	0%	Description of Built-as	1 Story

IMPROVEMENT BUILTAS	Built-As Square Feet	Integer		Sum of the square feet for the 'built as' types identified for the building. Each building may have one or more 'built as' type, though usually only one. The exceptions are 'built as' # 124 and 126 (Add on only Res & Com) or any type of Storage Tank, where it represents the count of those 'built as' as opposed to the sum of their square feet.	numeric	0%	0~1016109	1673
IMPROVEMENT BUILTAS	HVAC	Integer		Code associated with the predominant heating source for the built-as structure.	nominal	100%	Null	Null
IMPROVEMENT BUILTAS	HVAC Description	Varchar	30	Text description associated with the predominant heating source for the built-as structure i.e. Forced Air, Electric Baseboard, Steam, etc.	nominal	0.10%	Type/Description of HVAC	Electric
IMPROVEMENT BUILTAS	Exterior	Varchar	25	Predominant type of construction materials used for the exterior siding on Residential Buildings.	nominal	13.82 %	Details of Exterior Improvement	Frame Siding
IMPROVEMENT BUILTAS	Interior	Varchar	15	Predominant type of materials used on the interior walls. i.e. Sheetrock or Paneling. Collected for Residential and Mobile Homes only.	nominal	23.81 %	Drywall/Paneling, Null	Paneling
IMPROVEMENT BUILTAS	Stories	Decimal	13,2	Number of floors/building levels above grade. Stories do not include attic or basement areas.	numeric	0.00%	0~135, Null	1
IMPROVEMENT BUILTAS	Story Height	Integer		Based on property type: Commercial - determined by taking an average of the overall ceiling heights of all floors for the Blt As. Residential - determined by the majority of the story height for the first floor (main). Story height is automatically rounded by the system. (The system rounds up for .5 and higher and rounds down for anything below .5.)	numeric	0.00%	0~5010, Null	1
IMPROVEMENT BUILTAS	Sprinkler Square Feet	Integer		Total square footage covered by a sprinkler system. Sprinkler SF does not always equal the built as square footage.	numeric	0.00%	0~1029110	1
IMPROVEMENT BUILTAS	Roof Cover	Varchar	20	Material used for the roof. I.e. Composition Shingles, Wood Shake, Concrete Tile, etc.	nominal	24.02 %	Type of Roof Cover	Formed Seam Metal
IMPROVEMENT BUILTAS	Bedrooms	Integer		Number of bedrooms listed for a residential property. (Collected for informational purposes only.)	numeric	0.13%	0~21, Null	1

IMPROVEMENT BUILTAS	Bathrooms	Decimal	7,2	Number of baths listed for a residential property. The number is listed as a decimal, i.e. 2.75 = two full and one three-quarter baths. A tub/sink/toilet combination (plus any additional fixtures) is considered 1.0 bath. A shower/sink/toilet combination (plus any additional fixtures) is 0.75 bath. A sink/toilet combination is .5 bath.	numeric	0.13%	0~2075, Null	2.5
IMPROVEMENT BUILTAS	Units	Integer		Number of separate units within the building. For Commercial properties, Units may be combined to generate an income approach for the economic unit on building one.	numeric	0.00%	0~2319, Null	1
IMPROVEMENT BUILTAS	Class Code	Varchar	20	Code for one of five basic cost groups by type of framing (supporting beams and columns), walls, floors and roof structures, and fireproofing. Typically Commercial Buildings.	nominal	87.39 %	A,B,C,D,P,S,Null	A
IMPROVEMENT BUILTAS	Class Description	Varchar	50	Text description for one of five basic cost groups by type of framing (supporting beams and columns), walls, floors and roof structures, and fireproofing. Typically Commercial Buildings.	nominal	87.39 %	Fireproof Steel, Masonry,Metal Frame, Pole, Reinforced Concrate, Wood Frame	Wood Frame
IMPROVEMENT BUILTAS	Year Built	Integer		Year the building was built, as stated by the building permit or a historical record.	ordinal	0.00%	0~2019	2018
IMPROVEMENT BUILTAS	Year Remodeled	Integer		Year in which improvements are made to the existing structure that extend the life of the structure.	ordinal	0.32%	0~2019, Null	2018
IMPROVEMENT BUILTAS	Adjusted Year Built	Integer		If greater than the Year Built, it indicates improvements have been made to the original property over and above normal maintenance which would effectively reduce the age of the building.	ordinal	0.00%	0~2019, Null	2018
IMPROVEMENT BUILTAS	Physical Age	Integer		Tax year less the adjusted year built equals the physical age of the built as	numeric	0.00%	0~134, Null	24
IMPROVEMENT BUILTAS	Built-As Length	Integer		Length of the mobile/manufactured home	numeric	0.25%	0~100, Null	11
IMPROVEMENT BUILTAS	Built-As Width	Integer		Width of the mobile/manufactured home.	numeric	0.25%	0~80, Null	8
IMPROVEMENT BUILTAS	Mobile Home Model	Varchar	50	Model of the mobile/manufactured home.	numeric	93.34 %	Model of mobile home	CAMELOT

LAND ATTRIBUTE	Parcel Number	Varchar	10	Unique 10 digit number assigned to each property	nominal	0%	19012000 ~ 222324039	19012000
LAND ATTRIBUTE	Attribute	Varchar	30	Describes the category that a land attribute is assigned to: Residential (R),Commercial (C) or Res Com Condo (X) and the subcategory, such as Amenities,Economic, Functional, Neighborhood, Utilities, etc.	nominal	0%	C AMENITIES, C ECONOMIC, C FUNCTIONAL, C MA 1 WEST, C MA 3 PENINSULA, C MA 4 TACOMA N, C MA 6 EAST, C SITE DEVELOPMENT, C STREETS, C USE, C UTILITIES, C WATERFRONT, C ZONING, R AMENITIES, R ECONOMIC, R FUNCTIONAL, R NEIGHBOURHOOD, R SITE DEVELOPMENT, R SIZE, R STREETS, R UTILITIES, R VIEW, R WATERFRONT	C AMENITIES
LAND ATTRIBUTE	Attribute Description	Varchar	30	Further defines the land attribute. Land attributes may or may not contribute tovalue. i.e. Examples of attributes are View Avg, NBHD Gated, WF Bank Low,water available etc.	nominal	0%	ACCESS NO LEGAL, ACCESS POOR, ADJ FACTOR - COMMENT, BEACH ACC NO, EXTENSIVE, GROUP SIZE ADJ, LAGOON MIDPOINT, MARGINAL 20 PCT, MARGINAL 40 PCT, MARGINAL 50 PCT, POWER AVAILABLE, POWER NO - COMMENT, SALT 1 LIM -, SALT 2 LIM, SALT BANK 1 LOW, SALT BANK 3 MED, SALT BANK 4 MED+, SALT BANK 5 HIGH, SEWER/SEPTIC NO, SHAPE NEGATIVE, STREET NO ROAD, STREET UNPAVED, TOPO STEEP, WATER NO, WET	ACCESS NO LEGAL
TAX ACCOUNT	Parcel Number	Varchar	10	Unique 10 digit number assigned to each property.	nominal	0%	185005 ~ 420033037	185005
TAX ACCOUNT	Account Type	Varchar	5	REAL - Real Property STRUC - Structures PERS - Personal Property MOBIL - Mobile Home	nominal	0%	REAL, STRUC, PERS, MOBIL	REAL
TAX ACCOUNT	Property Type	Varchar	5	ASIMP - Administrative Seg Improvement LNDIM - Land and Improvements MBLHM - Mobile Home RRLEA - Railroad Leases SAPP - State Assessed Personal Property SARP - State Assessed Real Property	nominal	0%	ASIMP, LNDIM, MBLHM, RRLEA, SAPP, SARP, STRUC	ASIMP

				STRUC - Leased Land and/or Structure				
TAX ACCOUNT	Site Address	Varchar	50	Physical location address of the property. This may be different than the mailing address. If the address contains "XXX", that means we did not have a valid address on file, so the address was estimated using our GIS (Geographic Information System).	nominal	0%	11606 150TH AV, 11704 150TH AV, XXX 157TH AVCT SW, XXX STAMFORD RD SW, etc	11606 150TH AV
TAX ACCOUNT	Use Code	Varchar	5	4 digit numeric code associated with the present highest and best use of theproperty for appraisal purposes.	nominal	0.28%	0 ~ 9900, NULL	0
TAX ACCOUNT	Use Description	Varchar	50	Text description of the numeric use code.	nominal	0.28%	APPAREL & FINISH MFG, APT/CONDO 3 STOR OR LESS, TRIPLEX 3 UNITS, SINGLE FAMILY DWELLING, UNKNOWN, NULL, etc	APPAREL & FINISH MFG
TAX ACCOUNT	Tax Year - Prior	Integer		Tax year for which all prior year values apply.	numeric	0%	2018	2018
TAX ACCOUNT	Tax Code Area - Prior Year	Varchar	10	Code which describes both a geographical area and a specific combination of taxing districts. Subject to change annually due to changing levies, bonds, incorporations and annexations.	nominal	1%	5 ~ 680	5
TAX ACCOUNT	Exemption Type - Prior Year	Varchar	40	Type of exemption, if any, applied to the property.	nominal	91.70 %	0-Unknown, Cemetery, County Owned property, Permanent Disability A, Permanent Disability B, etc	Cemetery
TAX ACCOUNT	Current Use Code - Prior Year	Varchar	5	AGRI - Agricultural CT - Current Use Timber FORDG - Forest Designated OPBRS - Open Space PBRS OPEN - Open Space	nominal	98.54 %	AGRI, CT, FORDG, OPBRS, OPEN, NULL	AGRI
TAX ACCOUNT	Land Value - Prior Year	Integer		Total fair market value of land for the account, as determined by the appraisal process.	numeric	4.34%	0 ~ 61605900, NULL	0
TAX ACCOUNT	Improvement Value - Prior Year	Integer		Total fair market value of improvements (buildings, structures) for the parcel, as determined by the appraisal process.	numeric	4.07%	0 ~ 191504800, NULL	0

TAX ACCOUNT	Total Market Value - Prior Year	Integer		Total fair market value of land and improvements for the parcel, as determined by the appraisal process.	numeric	0%	0 ~ 199734100, NULL	0
TAX ACCOUNT	Taxable Value - Prior Year	Integer		Market value of the property, minus any exemptions granted.	numeric	0%	0 ~ 199734100, NULL	0
TAX ACCOUNT	Tax Year - Current	Integer		Tax year for which all current year values apply	numeric	0%	2019	2019
TAX ACCOUNT	Tax Code Area - Current Year	Varchar	10	Code which describes both a geographical area and a specific combination oftaxing districts. Subject to change annually due to changing levies, bonds,incorporations and annexations.	nominal	0.15%	5 ~ 773	5
TAX ACCOUNT	Exemption Type - Current Year	Varchar	40	Type of exemption, if any, applied to the property.	nominal	91.68 %	County Owned Property, Government Property, Tax Title Property, Temporary Disability A, Temporary Disability B, NULL	County Owned Property
TAX ACCOUNT	Current Use Code - Current Year	Varchar	5	AGRI - Agricultural CT - Current Use Timber FORDG - Forest Designated OPBRS - Open Space PBRS OPEN - Open Space	nominal	98.52 %	AGRI, CT, FORDG, OPBRS, OPEN, NULL	AGRI
TAX ACCOUNT	Land Value - Current Year	Integer		Total fair market value of land for the parcel, as determined by the appraisal process.	numeric	0%	0 ~ 67142800, NULL	0
TAX ACCOUNT	Improvement Value - Current Year	Integer		Total fair market value of improvements (buildings, structures) for the parcel, as determined by the appraisal process.	numeric	0.01%	0 ~ 209019000, NULL	0
TAX ACCOUNT	Total Market Value - Current Year	Integer		Total fair market value of land and improvements for the parcel, as determined by the appraisal process.	numeric	0%	0 ~ 217248300	0
TAX ACCOUNT	Taxable Value - Current Year	Integer		Market value of the property, minus any exemptions granted.	numeric	0%	0 ~ 217248300	0
TAX ACCOUNT	Range	Varchar	10	Range	ordinal	7.21%	0 ~ 5, NULL	0
TAX ACCOUNT	Township	Varchar	10	Township	nominal	7.21%	15 ~ 22, NULL	15
TAX ACCOUNT	Section	Varchar	20	Section	nominal	7.21%	1 ~ 36, NULL	1
TAX ACCOUNT	Quarter Section	Varchar	20	Quarter Section	nominal	7.21%	11 ~ 44, 24 & 31, NULL	11

TAX ACCOUNT	Subdivision Name	Varchar	50	Subdivision Name	nominal	34.04 %	30TH & MERIDIAN MASTER CONDO (4A), ABANDONED MILITARY RESERVE, MANITOU PARK ADD, PRESCOTTS 2ND, NULL etc.	30TH & MERIDIAN MASTER CONDO (4A)
TAX ACCOUNT	Located On Parcel	Varchar	10	Real property parcel number on which this improvement-only parcel is located. Applies to Personal Property, Mobile Homes and (Leasehold) Structures.	nominal	93.34 %	221221019, 321211044, 5215201094, NULL	221221019
CRIME DATA	OBJECTID (PK)	Integer		Unique Identification Number of each crime	nominal	0%	586700 ~ 613857	586700
CRIME DATA	CaseNo	Varchar	10	A unique ID given to each crime case	nominal	0%	1735601778 ~ 1929601916	1735601778
CRIME DATA	District	Varchar	20	Police District ID	nominal	0%	BL02, BL03, BL04, PUYS, PC01, etc	PUYS
CRIME DATA	OccuredOn	DateTime		Date of occurrence	nominal	0%	2018-06-01 00:00:00.000Z ~ 2019-05-31 00:00:00.000Z	2018-06-01 00:00:00.000Z
CRIME DATA	LocCode	Varchar	20	General description of the location of the crime	nominal	0%	Apartment, Driveway, Parking lot, Private Vehicle, Single Family Residence	Apartment
CRIME DATA	Public_Nam	Varchar	50	General description of the type of crime	nominal	0%	Criminal Traffic, Motor Vehicle Theft, Theft-Other, Theft - vehicle prowl, etc	Criminal traffic
CRIME DATA	XCoord	Integer		X- Coordinates in Nad 83 Harn State Plane Washington South us feet	nominal	0%	1061984.796 ~ 1397345.067	1061984.796
CRIME DATA	YCoord	Integer		Y- Coordinates in Nad 83 Harn State Plane Washington South us feet	nominal	0%	517052.6447 ~ 762929.1147	517052.6447
CRIME DATA	NAT_Name	Varchar	20	Name of the Neighborhood Action Team (NAT) area where the crime occurred. A "None" value indicates that the crime did not occurred in a NAT area.	nominal	0%	Lauradel, Indian Springs, None, Crystal Gardens, Cascade Waters, etc	Lauradel
CRIME DATA	City	Varchar	20	Name of the incorporated city where the crime occurred. A "None" value indicates that the crime did not occurred in an incorporated city.	nominal	0%	Bonney lake, Eatonville, Edgewood, Gig Harbor, Pierce County, Puyallup, etc	Pierce County
IMPROVEMENT	Parcel Number	Varchar	10	Unique 10 digit number assigned to each property.	nominal	0%	0019121003 ~ 9900400320	0020237701
IMPROVEMENT	Building ID	Varchar	5	Unique identifier for each improvement record on a parcel. Note, not all improvements are on a separate record and may not always be in sequence	nominal	0%	0 ~ 99	50

IMPROVEMENT	Property Type	Varchar	15	Links the buildings on the individual records to the proper cost and depreciation tables. Options include Commercial, Duplex, Industrial, Mobile Home, Multiple Unit, Out Building, Residential, Townhouse and Triplex.	nominal	0%	Type of Property	Residential
IMPROVEMENT	Neighborhood	Varchar	10	Represents boundaries for land subject to similar social, environmental, economic and governmental forces. Neighborhoods are typically located within a geographic area defined by natural, man-made, or political boundaries. Residential Neighborhoods and LEA's are the same. Commercial Neighborhoods define the location of the property, while Commmercial LEA's define the use of the property.	nominal	0.00%	010201 ~ 906	130111
IMPROVEMENT	Neighborhood Extension	Varchar	10	Based on property type: Commercial - represents the primary occupancy or use of the property. Residential - not used, therefore they are set to zero. Mobile/Manufactured Homes - based upon the underlying land ownership. All Mobile/Manufactured homes located on land owned by the Mobile/Manufactured home owner are coded with the Neighborhood Extension 0. Exception - Mobile/Manufactured homes located within a park and owned by the park owner, will carry the same Neighborhood Extension as the park. Mobile Homes in a park have a neighborhood extension of PL 1 (P = Park, L = Leased); P2, P3, P4 et	nominal	0.00%	0 - PL1	730
IMPROVEMENT	Square Feet	Integer		Sum of the square feet for the 'built as' types identified for the building. Each building may have one or more 'built as' type (see IMPROVEMENT BUILTAS table), though usually only one. The exceptions are 'built as' # 124 and 126 (Add on only Res & Com) or any type of Storage Tank, where it represents the count of those 'built as' as opposed to the sum of their square feet.	numeric	0%	1 ~ 1078155	520

IMPROVEMENT	Net Square Feet	Integer		Used primarily to calculate income on commercial properties, where it typically represents the sum of the net rentable area(s). This number may be representative of one or more buildings. If part of a Group Account, these buildings may be located on more than one parcel. For Residential it is equal to the built as square feet or zero (0).	numeric	0%	0 ~ 1523580	404
IMPROVEMENT	Percent Complete	Decimal	15,4	Describes the stage of new construction. A completed structure is listed at 100% complete. Structures less than 100% complete are listed at the appropriate percent complete based on the appraisers observation. Depending on the duration of the construction, in some cases the appraiser may move the structure to 100% complete with an adjustment to the building for any unfinished work.	numeric	0%	0.000 ~ 1.0040	1.000
IMPROVEMENT	Condition	Varchar	15	Captures the overall depreciation of a structure. Condition is a reflection of the maintenance and upkeep of the structure.	nominal	0.09%	Condition of Structure	Very Poor
IMPROVEMENT	Quality	Varchar	15	Indication of the quality of the materials used, workmanship, architectural attractiveness, and functional design.	nominal	0.09%	Quality of Material	Fair
IMPROVEMENT	Primary Occupancy Code	Integer		Numeric code for how an improvement is being occupied. If there are multiple occupancies, the primary occupancy code is determined by the code with the largest occupancy percentage.	nominal	0.61%	26 ~ 9997	100
IMPROVEMENT	Primary Occupancy Descriptio n	Varchar	255	description for the primary occupancy code.	nominal	0.61%	Description of Occupancy	Single Family Residential
IMPROVEMENT	Mobile Home Serial number	Varchar	30	Serial number Listed on the mobile home title.	nominal	93%	02830364M ~ ZWK70149205	9079
IMPROVEMENT	Mobile Home Total Length	Integer		Length in feet taken from the mobile home title and may be updated as a result of an inspection by an appraiser.	numeric	93%	0 ~ 84	50
IMPROVEMENT	Mobile Home Make	Varchar	30	Make of the mobile home.	nominal	93.40 %	Mobile Home Brand	SKY
IMPROVEMENT	Attic Finished Square Feet	Integer		Finished living area in the attic.	numeric	94.80 %	36 ~ 3200	760

IMPROVEMENT	Basement Square Feet	Integer		Total square footage of the basement	numeric	87.50 %	49 ~ 38230	864
IMPROVEMENT	Basement Finished Square Feet	Integer		Finished square footage of the basement.	numeric	92.18 %	1 ~ 77597	456
IMPROVEMENT	Carport Square Feet	Integer		Total square footage of all carports	numeric	99.80 %	56 ~ 1224	56
IMPROVEMENT	Balcony Square Feet	Integer		Total square feet of all balcony areas. A balcony could be a veranda, lanai, or gallery. These are typically condos with exterior lanai's or church balconies.	numeric	99.60 %	30 ~17609	30
IMPROVEMENT	Porch Square Feet	Integer		Total number of square feet associated with all porches	numeric	38.23 %	0 ~ 13468	0
IMPROVEMENT	Attached Garage Square Feet	Integer		Total square footage of the attached or built in garage(s).	numeric	53.96 %	1 ~ 6896	441
IMPROVEMENT	Detached Garage Square Feet	Integer		Total detached garage(s) square footage.	numeric	94.60 %	112 - 4608	112
IMPROVEMENT	Fireplaces	Integer		Total count of single, double or PreFab stoves	numeric	42.58 %	0-8	1
IMPROVEMENT	Basement Garage Door	Integer		Indicates if the basement garage doors are single or double. The default is double if there is both a single and a double door	numeric	97.60 %	-502 ~ 760	760

CODE – DATA CLEANING

Clear Environment

Install and load Packages

```
library(tidyverse)
library(here)
library(readr)
library(sqldf)
library(corrplot)
library(kableExtra)
library(knitr)
library(blscrapeR)

opts_chunk$set(results = 'asis',      # This is essential (can also be set at the chunk-level)
               comment = NA,
               prompt = FALSE,
               cache = FALSE)

library(rpart)
library(summarytools)
```

Loading Data

```
tax_parcels <- read.csv(file = "Data/TaxParcels.csv", header=TRUE, sep=",")
tax_account <- read.csv(file = "Data/TaxAccount.csv", header=TRUE, sep=",")
schools <- read.csv(file = "Data/Schools.csv", header=TRUE, sep=",")
sale <- read.csv(file = "Data/Sale.csv", header=TRUE, sep=",")
land_attribute <- read.csv(file = "Data/LandAttribute.csv", header=TRUE, sep=",")
improvement_builtas <- read.csv(file = "Data/ImprovementBuiltas.csv", header=TRUE, sep=",")
improvement <- read.csv(file = "Data/Improvement.csv", header=TRUE, sep=",")
crime_data <- read.csv(file = "Data/CrimeData.csv", header=TRUE, sep=",")
appraisal_account <- read.csv(file = "Data/AppraisalAccount.csv", header=TRUE, sep=",")
parks <- read.csv(file = "Data/Parks_points.csv", header=TRUE, sep=",")
```

Clean Data

Selecting variables and filtering

```
tax_account <- tax_account %>%
  select(-UseCode, -PropertyType, -SiteAddress, -UseDescription, -TaxYearPrior, -ExemptionTypePriorYear, -CurrentUseCodePr
iorYear, - TaxYearCurrent, -ExemptionTypeCurrentYear, -CurrentUseCodeCurrentYear, -Range, -SubdivisionName,
         -LocatedOnParcel, -None)

tax_parcels <- tax_parcels %>%
  select(TaxParcelNumber, XCoord, YCoord)

sale <- sale %>%
  select(-ETN, -ParcelCount, -Grantor, -Grantee, -DeedType, ExcludeReason)

#Select only type and location of school.
schools <- schools %>%
  select(Type, Grade, XCoord, YCoord, Rating) %>%
  filter(Grade != "Higher Education")



#Select only the data, offense type and locatiion of a crime.
crime_data <- crime_data %>%
  select(OccurredOn, PublicName, XCoord, YCoord)



#Select and filter for appraisal.
appraisal_account <- appraisal_account %>%
  filter(AppraisalAccountType == "Residential") %>%
                      select(-c(BusinessName, ValueAreaID, Buildings, GroupAccountNumber, LandGrossAcres, LandNetAcres, Su
bmergedAreaSquareFeet, Latitude, Longitude)) %>%
```

```
        mutate(WaterfrontType = replace(WaterfrontType, WaterfrontType == "NULL", "None")) %>%
        mutate(ViewQuality = replace(ViewQuality, ViewQuality == "NULL", "None"))

#Select and filter for Improvement and ImprovenmentBuiltAs

Improvement_Cleaned <- sqldf("select ParcelNumber, BuildingID, SquareFeet, PercentComplete, Condition, Quality, AtticFinis
hedSquareFeet, BasementSquareFeet, CarportSquareFeet,  PorchSquareFeet, AttachedGarageSquareFeet, DetachedGarageSquareFeet
, Fireplaces from improvement where PropertyType = 'Residential'")

Improvement_Cleaned$CarportSquareFeet <- as.numeric(as.character(Improvement_Cleaned$CarportSquareFeet))

Improvement_Cleaned$PorchSquareFeet <- as.numeric(as.character(Improvement_Cleaned$PorchSquareFeet))

Improvement_Cleaned$BasementSquareFeet <- as.numeric(as.character(Improvement_Cleaned$BasementSquareFeet))

Improvement_Cleaned$AtticFinishedSquareFeet <- as.numeric(as.character(Improvement_Cleaned$AtticFinishedSquareFeet))

Improvement_Cleaned$AttachedGarageSquareFeet <- as.numeric(as.character(Improvement_Cleaned$AttachedGarageSquareFeet))

Improvement_Cleaned$DetachedGarageSquareFeet <- as.numeric(as.character(Improvement_Cleaned$DetachedGarageSquareFeet))

Improvement_Cleaned$Fireplaces <- as.numeric(as.character(Improvement_Cleaned$Fireplaces))

improvement_builtas$StoryHeight <- as.numeric(as.character(improvement_builtas$StoryHeight))

improvement_builtas$YearRemodeled <- as.numeric(as.character(improvement_builtas$YearRemodeled))

Improvement_final <- sqldf("select ParcelNumber, BuildingID, SquareFeet, PercentComplete, Condition, Quality, AtticFinishe
dSquareFeet, BasementSquareFeet, CarportSquareFeet,  PorchSquareFeet, AttachedGarageSquareFeet, DetachedGarageSquareFeet,
Fireplaces from Improvement_Cleaned where ParcelNumber in (select ParcelNumber from Improvement_Cleaned group by ParcelNum
ber having count(*) = 1)")



Improvement_Data <- sqldf("select i.ParcelNumber, i.BuildingID, i.BuiltAsDescription, i.BuiltAsSquareFeet, i.Exterior, i.I
nterior, i.Stories, i.StoryHeight, i.RoofCover, i.Bedrooms, i.Bathrooms, i.Units, i.YearBuilt, i.YearRemodeled, i.Physical
Age, ias.SquareFeet, ias.PercentComplete, ias.Condition, ias.Quality,  ias.AtticFinishedSquareFeet, ias.BasementSquareFeet
, ias.CarportSquareFeet,  ias.PorchSquareFeet, ias.AttachedGarageSquareFeet, ias.DetachedGarageSquareFeet, ias.Fireplaces
from improvement_builtas i inner join Improvement_final ias on i.ParcelNumber = ias.ParcelNumber and i.BuildingID = ias.Bu
ildingID and i.BuiltAsNumber = 1 where i.YearRemodeled is not Null and i.Stories is not Null")

Improvement_Data[is.na(Improvement_Data)] <- 0

parks <- parks %>% select(ï..X, Y, PropertyID, ParkType)

colnames(parks)[colnames(parks)== "ï..X"] <- "X"
```

Grouping Crime into Property Crime, Non-Property Crime and Violent

```
crime_data <- crime_data %>%
  mutate(PublicName = case_when(
    PublicName %in% c("Arson - Residential", "Burglary - Residential", "Robbery - Residential", "Vandalism - Residential")
 ~
      "Property_Crime",
    PublicName %in% c("Arson - Non-residential", "Burglary - Non-residential", "Criminal Traffic",
                      "Drug Possession (Methamphetamine)", "Drug Possession (Other)", "Drug Sale/Manufacture (Methamphetam
ine)",
                      "Drug Sale/Manufacture (Other)", "Fraud or Forgery", "Intimidation", "Liquor Law Violations",
                      "Motor Vehicle Theft", "Possession of Stolen Property", "Telephone Harassment", "Theft - Gas Station
 Runout", "Theft - Mail", "Theft - Other",
                "Theft - Vehicle Prowl", "Theft -Shoplifing", "Trafficking in Stolen Property", "Vandalism - Non-residenti
al",
                "Warrant Arrests") ~ "Non-Property_Crime",
    PublicName %in% c("Assault - Aggravated", "Assault - Simple", "Homicide", "Robbery - Business", "Robbery - Other",
                "Robbery - Street") ~ "Violent"
  ))
```

Join the data

```r
#Join sale and tax tables
sale_tax <- left_join(sale, tax_account, by = "ParcelNumber")

#Join with appraisal tables
sale_tax_appraisal <- left_join(sale_tax, appraisal_account, by = "ParcelNumber")

#Align column names and join to tax parcels.
tax_parcels <- tax_parcels %>% dplyr::rename(ParcelNumber = TaxParcelNumber)
sale_tax_appraisal_parcel <- left_join(sale_tax_appraisal, tax_parcels, by = "ParcelNumber")

#Join improvment aggregated table
master <- left_join(sale_tax_appraisal_parcel, Improvement_Data, by = "ParcelNumber")

str(master)
```

Dropping NA for TaxCodeArea, these have NAs in many more columns and are not useful

```r
master <- master %>%
  drop_na(TaxCodeAreaPriorYear) %>%
  drop_na(AppraisalAccountType) %>%
  select(-AppraisalAccountType) %>%
  select(-AccountType) %>%
  drop_na(BuildingID) %>%
  select(-BuildingID) %>%
  drop_na(XCoord)
```

Convert factor to character

```r
master <- master %>%
  mutate_if(is.factor, as.character)
```

Replacing NA with Not Applicable in View Quality and Waterfront

Converting into rating

```r
master$WaterfrontType[which(is.na(master$WaterfrontType))] <- "Not_Applicable"
master$ViewQuality[which(is.na(master$ViewQuality))] <- "Not_Applicable"


master <- master %>%
  mutate(WaterfrontType = case_when(
    WaterfrontType %in% c("WF Salt", "WF Stream/Creek", "WF Lake", "WF River") ~ 1,
    WaterfrontType == "Not_Applicable" ~ 0
  )) %>%
  mutate(ViewQuality = case_when(
    ViewQuality == "Not_Applicable" ~ 0,
    ViewQuality == "View Lim -" ~ 1,
    ViewQuality == "View Lim" ~ 2,
    ViewQuality == "View Lim +" ~ 3,
    ViewQuality == "View Avg" ~ 4,
      ViewQuality == "View Avg +" ~ 5,
      ViewQuality == "View Good" ~ 6,
      ViewQuality == "View Good" ~ 7,
          ViewQuality == "View Good +" ~ 8,
            ViewQuality == "View V-Good" ~ 9,
          ViewQuality == "View V-Good +" ~ 10
  )) %>%
  mutate(StreetType = case_when(
    StreetType == "Street Unpaved" ~ 0,
    StreetType == "Paved" ~ 1
  )) %>%
  mutate(Condition = case_when(
    Condition == "Uninhabitable" ~ 0,
    Condition == "Extra Poor" ~ 1,
    Condition == "Very Poor" ~ 2,
    Condition == "Poor" ~ 3,
    Condition == "Fair" ~ 4,
    Condition == "Average" ~ 5,
    Condition == "Good" ~ 6
  )) %>%
  mutate(Quality = case_when(
```

```
    Quality == "Low" ~ 0,
    Quality == "Low Plus" ~ 1,
    Quality == "Fair" ~ 2,
    Quality == "Fair Plus" ~ 3,
    Quality == "Average" ~ 4,
    Quality == "Average Plus" ~ 5,
    Quality == "Good" ~ 6,
    Quality == "Good Plus" ~ 7,
    Quality == "Very Good" ~ 8,
    Quality == "Very Good Plus" ~ 9,
    Quality == "Excellent" ~ 9
  )) %>%
  mutate(AtticFinishedSquareFeet = case_when(
    AtticFinishedSquareFeet > 0 ~ 1,
    AtticFinishedSquareFeet <= 0 ~ 0
  )) %>%
  mutate(BasementSquareFeet = case_when(
    BasementSquareFeet > 0 ~ 1,
    BasementSquareFeet <= 0 ~ 0
  )) %>%
  mutate(CarportSquareFeet = case_when(
    CarportSquareFeet > 0 ~ 1,
    CarportSquareFeet <= 0 ~ 0
  )) %>%
  mutate(PorchSquareFeet = case_when(
    PorchSquareFeet > 0 ~ 1,
    PorchSquareFeet <= 0 ~ 0
  )) %>%
  mutate(AttachedGarageSquareFeet = case_when(
    AttachedGarageSquareFeet > 0 ~ 1,
    AttachedGarageSquareFeet <= 0 ~ 0
  )) %>%
  mutate(DetachedGarageSquareFeet = case_when(
    DetachedGarageSquareFeet > 0 ~ 1,
    DetachedGarageSquareFeet <= 0 ~ 0
  ))
```

Convert to numeric

```
master$LandValuePriorYear <- as.numeric(master$LandValuePriorYear)
master$ImprovementValuePriorYear <- as.numeric(master$ImprovementValuePriorYear)
master$TotalMarketValuePriorYear <- as.numeric(master$TotalMarketValuePriorYear)
master$TaxableValuePriorYear <- as.numeric(master$TaxableValuePriorYear)

master$LandValueCurrentYear <- as.numeric(master$LandValueCurrentYear)
master$ImprovementValueCurrentYear <- as.numeric(master$ImprovementValueCurrentYear)
master$TotalMarketValueCurrentYear <- as.numeric(master$TotalMarketValueCurrentYear)
master$TaxableValueCurrentYear <- as.numeric(master$TaxableValueCurrentYear)

master$LandNetSquareFeet <- as.numeric(master$LandNetSquareFeet)
master$LandWidth <- as.numeric(master$LandWidth)

master$Bedrooms <- as.numeric(master$Bedrooms)
master$Bathrooms <- as.numeric(master$Bathrooms)
master$Stories <- as.numeric(master$Stories)
master$StoryHeight <- as.numeric(master$StoryHeight)
master$Units <- as.numeric(master$Units)
master$PhysicalAge <- as.numeric(master$PhysicalAge)
```

Convert character to factor

```
master <- master %>%
  mutate_if(is.character, as.factor)
```

Drop NA value of LandWidth & ImprovementValuePriorYear

```
master <- subset(master, (!is.na(LandWidth)))
master <- subset(master, (!is.na(ImprovementValuePriorYear)))
```

Drop remaining NA from Stories

```
master <- master %>%
  drop_na(Stories)
```

Drop remaining NA from XCoord

```
master <- master %>%
  drop_na(XCoord)
```

Removing unreasonable high numbers of Bathrooms, Stories and Storyheight

```
master <- master %>%
  filter(Bathrooms <= 5)

master <- master %>%
  filter(Stories < 10)

master <- master %>%
  filter(StoryHeight <= 10)
```

Converting SaleDate to date

```
master$SaleDate <- as.Date(master$SaleDate, format = "%m/%d/%Y")
```

Filter for Sale Date in 2018

```
master <- master %>%
  filter(SaleDate >= "2018-01-01" & SaleDate <= "2018-12-31")
```

Selecting only SalePrice between $100,000 and $3,000,000

```
master <- master %>%
  filter(between(SalePrice, 100000, 3000000))

summary(master$SalePrice)
```

Min. 1st Qu. Median Mean 3rd Qu. Max. 100000 275000 340000 380099 432591 2883000

Creating the columns we want to do analysis with

```
model_data <- subset(master, select = c('ParcelNumber', 'SalePrice', 'Valid', 'Confirmed', 'Improved',
                                        'TaxCodeAreaPriorYear', 'Township', 'LandNetSquareFeet',
                                        'LandGrossFrontFeet', 'BuiltAsSquareFeet', 'Stories', 'StoryHeight',
                                        'Bedrooms', 'Bathrooms', 'Units', 'YearBuilt', 'YearRemodeled',
                                        'AtticFinishedSquareFeet', 'BasementSquareFeet', 'CarportSquareFeet',
                                        'PorchSquareFeet', 'AttachedGarageSquareFeet', 'DetachedGarageSquareFeet',
                                        "WaterfrontType", "ViewQuality", "Condition", "Quality", "Fireplaces"
                                        ))
```

Converting two columns to numeric from factor

```
model_data$TaxCodeAreaPriorYear <- as.numeric(model_data$TaxCodeAreaPriorYear)
model_data$Township <- as.numeric(model_data$Township)
```

Creating Counts data

Commented out, due to length of run time. Ran it once

```
# radius <- 5280 * 2
#
# schools$Rating = as.numeric(schools$Rating)
#
# count_GIS <- function(X, Y, Name) {
#   count <- nrow(crime_data %>%
#   filter(PublicName == Name & XCoord <= X + radius & YCoord <= Y + radius & XCoord >= X - radius & YCoord >= Y - radius)
)
# }
#
```

```
# parks_GIS <- function(XCoord, YCoord) {
#   count <- nrow(parks %>%
#   filter(X <= XCoord + radius & Y <= YCoord + radius & X >= XCoord - radius & Y >= YCoord - radius))
# }

# schoolcount_GIS <- function(X, Y) {
#  count <- ifelse(is.null(nrow(schools %>%
#   filter(XCoord <= X + radius & YCoord <= Y + radius & XCoord >= X - radius & YCoord >= Y - radius))), 0, nrow(schools %
>%
#    filter(XCoord <= X + radius & YCoord <= Y + radius & XCoord >= X - radius & YCoord >= Y - radius)))
#  }
#
#  schoolranksum_GIS <- function(X, Y) {
#    sum <- ifelse(nrow(schools %>% filter(XCoord <= X + radius & YCoord <= Y + radius & XCoord >= X - radius & YCoord >=
Y - radius) %>% select(Rating))==0, 0 , (sum(schools %>%
#      filter(XCoord <= X + radius & YCoord <= Y + radius & XCoord >= X - radius & YCoord >= Y - radius) %>% select(Rating
) %>% summarise(rate_school = sum(Rating)))))
#
#  }
#
#
#  parcel_counts <- tax_parcels %>%
#    group_by(ParcelNumber) %>%
#    mutate(Violent = count_GIS(XCoord, YCoord, "Violent")) %>%
#    mutate(Property = count_GIS(XCoord, YCoord, "Property_Crime")) %>%
#    mutate(Non_Property = count_GIS(XCoord, YCoord, "Non-Property_Crime")) %>%
#    mutate(Parks = parks_GIS(XCoord, YCoord)) %>%
#    mutate(Schools = ifelse(schoolranksum_GIS(XCoord, YCoord) == 0, 0, (schoolranksum_GIS(XCoord, YCoord)/ schoolcount_GI
S(XCoord, YCoord))))
#
#
# parcel_counts
```

Creating counts data to join with model_data

```
parcel_counts <- read.csv(file = "EDA/counts_data.csv", header=TRUE, sep=",")

parcel_counts <- parcel_counts %>%
  select(-X, -XCoord, -YCoord)

#Join the model data with the counts data so that we can include community variables.
model_data_counts <- left_join(model_data, parcel_counts, by = "ParcelNumber")

model_data_counts <- model_data_counts %>%
  drop_na(Property)
```

Write count data and model_data to csv

```
#Write to csv so that we can use in python if necessary.

model_data <- model_data_counts %>%
  select(-Violent, -Property, -Non_Property, -Parks, -Schools)

write.csv(model_data, "model_data.csv")
write.csv(model_data_counts, "model_data_counts.csv" )
```

Join clusters with count data

```
cluster0 <- read.csv(file = "EDA/cluster0.csv", header = TRUE, sep = ",")
cluster1 <- read.csv(file = "EDA/cluster1.csv", header = TRUE, sep = ",")
cluster2 <- read.csv(file = "EDA/cluster2.csv", header = TRUE, sep = ",")

cluster0_counts <- left_join(cluster0, parcel_counts, by = "ParcelNumber")
cluster1_counts <- left_join(cluster1, parcel_counts, by = "ParcelNumber")
cluster2_counts <- left_join(cluster2, parcel_counts, by = "ParcelNumber")

cluster0_counts <- cluster0_counts %>%
  drop_na(X) %>%
  select(-X)
```

```
cluster1_counts <- cluster1_counts %>%
  drop_na(X) %>%
  select(-X)

cluster2_counts <- cluster2_counts %>%
  drop_na(X) %>%
  select(-X)

cluster0 <- cluster0_counts %>%
  select(-Violent, -Property, -Non_Property, -Parks, -Schools)

cluster1 <- cluster1_counts %>%
  select( -Violent, -Property, -Non_Property, -Parks, -Schools)

cluster2 <- cluster2_counts %>%
  select( -Violent, -Property, -Non_Property, -Parks, -Schools)
```

Write clusters to csv

```
write.csv(cluster0, "cluster0.csv")
write.csv(cluster0_counts, "cluster0_counts.csv")
write.csv(cluster1, "cluster1.csv")
write.csv(cluster1_counts, "cluster1_counts.csv")
write.csv(cluster2, "cluster2.csv")
write.csv(cluster2_counts, "cluster2_counts.csv")
```

Kmeans_Cluster

August 13, 2019

0.1 Clustering Code

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

[2]:
```
# Load files
model_data = pd.read_csv("model_data.csv", sep=',', header = 0)
```

[4]:
```
model_data.columns
```

[4]:
```
Index(['Unnamed: 0', 'ParcelNumber', 'SalePrice', 'Valid', 'Confirmed',
       'Improved', 'TaxCodeAreaPriorYear', 'Township', 'LandNetSquareFeet',
       'LandGrossFrontFeet', 'BuiltAsSquareFeet', 'Stories', 'StoryHeight',
       'Bedrooms', 'Bathrooms', 'Units', 'YearBuilt', 'YearRemodeled',
       'AtticFinishedSquareFeet', 'BasementSquareFeet', 'CarportSquareFeet',
       'PorchSquareFeet', 'AttachedGarageSquareFeet',
       'DetachedGarageSquareFeet', 'WaterfrontType', 'ViewQuality',
       'Condition', 'Quality', 'Fireplaces'],
      dtype='object')
```

[5]:
```
final_data = model_data.drop(['Unnamed: 0'], axis = 1)
```

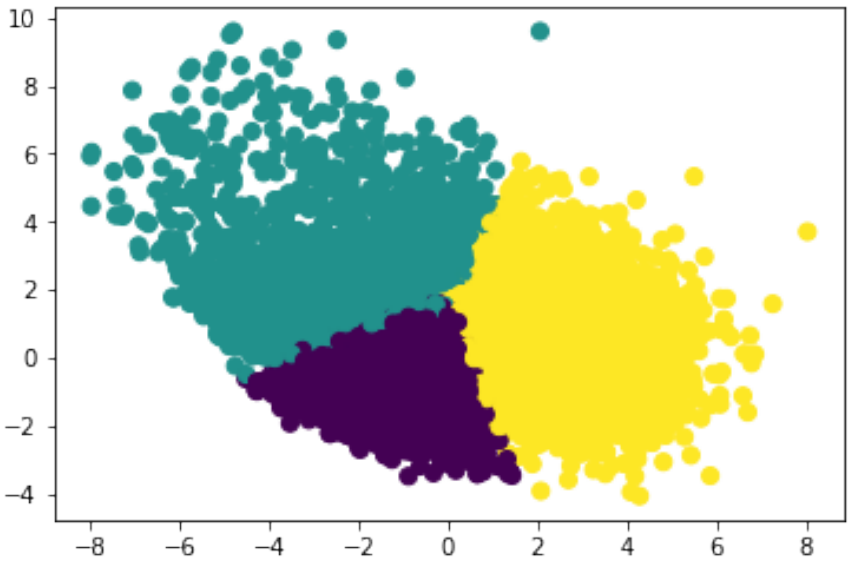
[7]:
```
scaler = StandardScaler()
scaler.fit(final_data.iloc[:, 1:])
X1 = scaler.transform(final_data.iloc[:,1:])

pca = PCA(n_components=3)
X1 = pca.fit_transform(X1)

# Run kmeans cluster
k_means = KMeans(3, random_state=0).fit_predict(X1)

# Cluster groups
c0 = final_data[k_means == 0]
c1 = final_data[k_means == 1]
c2 = final_data[k_means == 2]
```

```
#Cluster Plot
plt.scatter(X1[:, 0], X1[:, 1], c=k_means, s=50, cmap='viridis')
plt.show()
```



[8]:
```
# Save the cluster result
"""
c0.to_csv("~/cluster0.csv")
c1.to_csv("~/cluster1.csv")
c2.to_csv("~/cluster2.csv")
"""
```

[8]:
```
'\nc0.to_csv("/Users/han/Downloads/cluster0.csv")\nc1.to_csv("/Users/han/Downloads/cluster1.csv")\nc2.to_csv("/Users/han/Downloads/cluster2.csv")\n'
```

[1]:
```
# Load Packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn import tree
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn import utils
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.utils import check_array
from sklearn.ensemble import RandomForestRegressor
```

[2]:
```
# Load cleaned data files
model_data = pd.read_csv("model_data.csv", sep=',', header = 0)
model_data_counts = pd.read_csv("model_data_counts.csv", sep=',', header = 0)
```

[3]:
```
# Drop the unused column from the data
model_data = model_data.drop(['Unnamed: 0'], axis = 1)
model_data_counts = model_data_counts.drop(['Unnamed: 0'], axis = 1)
```

[4]:
```
# Prep data for use in models
model_data_x = model_data.drop(['ParcelNumber', 'SalePrice'], axis=1)
model_data_y = model_data["SalePrice"]

model_data_counts_x = model_data_counts.drop(['ParcelNumber', 'SalePrice'],␣
 ↪axis=1)
model_data_counts_y = model_data_counts["SalePrice"]
```

[5]:
```
# Function to calculate the mean absolute percentage error
def mean_absolute_percentage_error(y_test, y_pred):
    y_test = y_test.values
    y_test = y_test.reshape(-1,1)
    y_pred = y_pred.reshape(-1,1)
    y_test = check_array(y_test)
    y_pred = check_array(y_pred)
    return np.mean(np.abs((y_test - y_pred)/y_test)) * 100
```

0.1 Non-Clustered Data Without Community

0.1.1 Decision Tree

[6]:
```
# Defined a function to calculate mae
def get_mae(max_leaf_nodes, train_X, train_y,test_X, test_y):
    model_2 =␣
 ↪DecisionTreeRegressor(max_leaf_nodes=max_leaf_nodes,random_state=0)
    model_2.fit(train_X,train_y)
    pred_y = model_2.predict(test_X)
    mae = mean_absolute_error(test_y,pred_y)
    return mae
```

[7]:
```
# Split into training and test set.
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(model_data_x,model_data_y,test_size =0.3,random_state=0)

# Determine the best model
for max_leaves in␣
 ↪[2,3,5,50,150,200,250,300,500,1000,3000,5000,10000,20000,50000]:
    my_mae = get_mae(max_leaves, X_train, y_train, X_test, y_test)
    print("max leaf nodes : %d \t and Mean Absolute error :␣
 ↪%d"%(max_leaves,my_mae))
```

```
max leaf nodes : 2       and Mean Absolute error : 94865
max leaf nodes : 3       and Mean Absolute error : 94486
max leaf nodes : 5       and Mean Absolute error : 83548
max leaf nodes : 50      and Mean Absolute error : 63223
max leaf nodes : 150     and Mean Absolute error : 59006
max leaf nodes : 200     and Mean Absolute error : 58389
max leaf nodes : 250     and Mean Absolute error : 57563
max leaf nodes : 300     and Mean Absolute error : 57258
max leaf nodes : 500     and Mean Absolute error : 56581
max leaf nodes : 1000    and Mean Absolute error : 56878
max leaf nodes : 3000    and Mean Absolute error : 59461
max leaf nodes : 5000    and Mean Absolute error : 60623
max leaf nodes : 10000   and Mean Absolute error : 61023
max leaf nodes : 20000   and Mean Absolute error : 61026
max leaf nodes : 50000   and Mean Absolute error : 61026
```

[8]:
```
# Build the best tree model.
tree_model = tree.DecisionTreeRegressor(max_leaf_nodes = 500,␣
 ↪max_features='auto', splitter='best', random_state = 0)
tree_model = tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)

## export estimated tree into dot graphic file
dot_tree = tree.export_graphviz(tree_model, out_file='DtreeModel.dot',␣
 ↪feature_names=model_data_x.columns)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
18151889087.612602
56581.81124280858
16.30520296607561
```

0.1.2 Random Forest

[9]:
```
# Defined a function to calculate mae for RF
def get_mae_rf(n_estimators, max_depth, train_X, train_y,test_X, test_y):
    model = RandomForestRegressor(bootstrap=True, n_estimators=n_estimators,␣
 ↪max_depth=max_depth, criterion = 'mse',
                                  random_state=0)
    model.fit(train_X,train_y)
    pred_y = model.predict(test_X)
    mae = mean_absolute_error(test_y,pred_y)
    return mae
```

[10]:
```
# Determine the best model
for n_estimators in [1000, 1500]:
    for max_depth in [25,50,100, 200]:
        my_mae = get_mae_rf(n_estimators, max_depth, X_train, y_train, X_test,␣
 ↪y_test)
        print("n estimators : %d \t and max depth : %d \t and Mean Absolute␣
 ↪error : %d"%(n_estimators, max_depth, my_mae))
```

```
n estimators : 1000      and max depth : 25      and Mean Absolute error : 43041
n estimators : 1000      and max depth : 50      and Mean Absolute error : 43058
n estimators : 1000      and max depth : 100     and Mean Absolute error : 43058
n estimators : 1000      and max depth : 200     and Mean Absolute error : 43058
```

```
n estimators : 1500      and max depth : 25      and Mean Absolute error : 43054
n estimators : 1500      and max depth : 50      and Mean Absolute error : 43073
n estimators : 1500      and max depth : 100     and Mean Absolute error : 43073
n estimators : 1500      and max depth : 200     and Mean Absolute error : 43073
```

[11]:
```
# Build the best random forest model
forest_model = RandomForestRegressor(bootstrap=True, n_estimators=1000,␣
 ↪max_depth=25, criterion = 'mse',
                                random_state=0)
forest_model.fit(X_train, y_train)

y_pred = forest_model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
9493817890.824959
43041.965602357035
12.798611277808384
```

0.1.3 Neural Network - One Hidden Layer

[12]:
```
#Create scaled data
scaler = StandardScaler()
scaler.fit(X_train)

X_train_s = scaler.transform(X_train)
X_test_s = scaler.transform(X_test)
```

[36]:
```
# Defined a function to calculate mae
def get_mae_nn(hidden_layer_sizes, train_X, train_y,test_X, test_y):
    model = MLPRegressor(hidden_layer_sizes=(hidden_layer_sizes),␣
 ↪max_iter=1000, activation='relu', random_state=0)
    model.fit(train_X,train_y)
    pred_y = model.predict(test_X)
    mae = mean_absolute_error(test_y,pred_y)
    return mae
```

[34]:
```
# Determine the best model
for hidden_layer in [250,500,750]:
    my_mae = get_mae_nn(hidden_layer, X_train_s, y_train, X_test_s, y_test)
```

```
    print("Hidden layer : %d \t and Mean Absolute error :␣
↪%d"%(hidden_layer,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Mean Absolute error : 63502

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Mean Absolute error : 59757
Hidden layer : 750       and Mean Absolute error : 58799

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[35]:
```
# Build the best single layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(500), max_iter=1000, activation='relu',␣
 ↪random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
14764764317.131102
59757.079659852585
17.552168090998993
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.1.4 Neural Network- Two Hidden Layers

[37]:
```
# Defined a function to calculate mae
def get_mae_nn2(hidden_layer_sizes, hidden_layer_2, train_X, train_y,test_X,␣
↪test_y):
    model = MLPRegressor(hidden_layer_sizes=(hidden_layer_sizes,␣
↪hidden_layer_2), max_iter=1000, activation='relu', random_state=0)
    model.fit(train_X,train_y)
    pred_y = model.predict(test_X)
    mae = mean_absolute_error(test_y,pred_y)
    return mae
```

[ ]:
```
# Determine the best model
for hidden_layer in [250]:
    for hidden_layer2 in [250, 150, 50]:
        my_mae = get_mae_nn2(hidden_layer, hidden_layer2, X_train_s, y_train,␣
↪X_test_s, y_test)
        print("Hidden layer : %d \t and Hidden Layer2 : %d \t Mean Absolute␣
↪error : %d"%(hidden_layer, hidden_layer2,my_mae))
```

[46]:
```
# Build the best two layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(250,250), max_iter=1000,␣
↪activation='relu', random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
```

```
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
9878253309.38182
48679.32994009733
14.23490906739196
```

0.2 Non-Clustered Data With Community

0.2.1 Decision Tree

[48]:
```
# Split into training and test set.
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(model_data_counts_x,model_data_counts_y,test_size =0.
 ↪3,random_state=0)

# Determine the best model
for max_leaves in␣
 ↪[2,3,5,50,150,200,250,300,500,1000,3000,5000,10000,20000,50000]:
    my_mae = get_mae(max_leaves, X_train, y_train, X_test, y_test)
    print("max leaf nodes : %d \t and Mean Absolute error :␣
 ↪%d"%(max_leaves,my_mae))
```

```
max leaf nodes : 2       and Mean Absolute error : 97541
max leaf nodes : 3       and Mean Absolute error : 94069
max leaf nodes : 5       and Mean Absolute error : 90044
max leaf nodes : 50      and Mean Absolute error : 60631
max leaf nodes : 150     and Mean Absolute error : 55193
max leaf nodes : 200     and Mean Absolute error : 54176
max leaf nodes : 250     and Mean Absolute error : 53616
max leaf nodes : 300     and Mean Absolute error : 52945
max leaf nodes : 500     and Mean Absolute error : 51607
max leaf nodes : 1000    and Mean Absolute error : 51018
max leaf nodes : 3000    and Mean Absolute error : 52675
max leaf nodes : 5000    and Mean Absolute error : 53335
max leaf nodes : 10000   and Mean Absolute error : 53551
max leaf nodes : 20000   and Mean Absolute error : 53552
max leaf nodes : 50000   and Mean Absolute error : 53552
```

[50]:
```
# Build the best tree model.
tree_model = tree.DecisionTreeRegressor(max_leaf_nodes = 1000,␣
 ↪max_features='auto', splitter='best', random_state = 0)
tree_model = tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)

## export estimated tree into dot graphic file
dot_tree = tree.export_graphviz(tree_model, out_file='DtreeModel2.dot',␣
 ↪feature_names=model_data_counts_x.columns)
```

```
MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
9579693845.901884
51018.96739094695
14.204571559070347
```

0.2.2 Random Forest

[51]:
```
# Determine the best model
for n_estimators in [1000, 1500]:
    for max_depth in [25,50,100, 200]:
        my_mae = get_mae_rf(n_estimators, max_depth, X_train, y_train, X_test,␣
 ↪y_test)
        print("n estimators : %d \t and max depth : %d \t and Mean Absolute␣
 ↪error : %d"%(n_estimators, max_depth, my_mae))
```

```
n estimators : 1000      and max depth : 25      and Mean Absolute error : 39676
n estimators : 1000      and max depth : 50      and Mean Absolute error : 39648
n estimators : 1000      and max depth : 100     and Mean Absolute error : 39648
n estimators : 1000      and max depth : 200     and Mean Absolute error : 39648
n estimators : 1500      and max depth : 25      and Mean Absolute error : 39674
n estimators : 1500      and max depth : 50      and Mean Absolute error : 39672
n estimators : 1500      and max depth : 100     and Mean Absolute error : 39672
n estimators : 1500      and max depth : 200     and Mean Absolute error : 39672
```

[52]:
```
# Build the best random forest model
forest_model = RandomForestRegressor(bootstrap=True, n_estimators=1000,␣
 ↪max_depth=50, criterion = 'mse',
                                random_state=0)
forest_model.fit(X_train, y_train)

y_pred = forest_model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)
```

```
MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
5581437560.6362705
39648.27364546162
11.313672666545123
```

0.2.3 Neural Network - One Hidden Layer

[53]:
```
#Create scaled data
scaler = StandardScaler()
scaler.fit(X_train)

X_train_s = scaler.transform(X_train)
X_test_s = scaler.transform(X_test)
```

[54]:
```
# Determine the best model
for hidden_layer in [250,500,750]:
    my_mae = get_mae_nn(hidden_layer, X_train_s, y_train, X_test_s, y_test)
    print("Hidden layer : %d \t and Mean Absolute error :␣
 ↪%d"%(hidden_layer,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Mean Absolute error : 65450

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Mean Absolute error : 58703
Hidden layer : 750       and Mean Absolute error : 56580

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[55]:
```
# Build the best single layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(750), max_iter=1000, activation='relu',␣
 ↪random_state=0)
```

```
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
12695343935.578144
56580.34953988233
16.083654434889247
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.2.4 Neural Network - Two Hidden Layers

[56]:
```
# Determine the best model
for hidden_layer in [250]:
    for hidden_layer2 in [250, 150, 50]:
        my_mae = get_mae_nn2(hidden_layer, hidden_layer2, X_train_s, y_train,␣
↪X_test_s, y_test)
        print("Hidden layer : %d \t and Hidden Layer2 : %d \t Mean Absolute␣
↪error : %d"%(hidden_layer, hidden_layer2,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 250       and Hidden Layer2 : 250         Mean Absolute error :
46133
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 250       and Hidden Layer2 : 150         Mean Absolute error :
46222
Hidden layer : 250       and Hidden Layer2 : 50          Mean Absolute error :
50326

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[57]:
```
# Build the best two layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(250,250), max_iter=1000,␣
 ↪activation='relu', random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
6870437257.282856
46133.86361316045
13.222850429660049

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.3 Clusters

[62]:
```
cluster0 = pd.read_csv("cluster0.csv", sep=',', header = 0)
cluster0_counts = pd.read_csv("cluster0_counts.csv", sep=',', header = 0)
cluster1 = pd.read_csv("cluster1.csv", sep=',', header = 0)
cluster1_counts = pd.read_csv("cluster1_counts.csv", sep=',', header = 0)
cluster2 = pd.read_csv("cluster2.csv", sep=',', header = 0)
cluster2_counts = pd.read_csv("cluster2_counts.csv", sep=',', header = 0)
```

[63]:
```
cluster0_x = cluster0.drop(cluster0.columns[[0, 2]], axis=1)
cluster0_y = cluster0["SalePrice"]

cluster0_counts_x = cluster0_counts.drop(cluster0_counts.columns[[0, 1, 3]],␣
 ↪axis=1)
cluster0_counts_y = cluster0_counts["SalePrice"]

cluster1_x = cluster1.drop(cluster1.columns[[0, 2]], axis=1)
cluster1_y = cluster1["SalePrice"]

cluster1_counts_x = cluster1_counts.drop(cluster1_counts.columns[[0, 1, 3]],␣
 ↪axis=1)
cluster1_counts_y = cluster1_counts["SalePrice"]

cluster2_x = cluster2.drop(cluster2.columns[[0, 2]], axis=1)
cluster2_y = cluster2["SalePrice"]

cluster2_counts_x = cluster2_counts.drop(cluster2_counts.columns[[0, 1, 3]],␣
 ↪axis=1)
cluster2_counts_y = cluster2_counts["SalePrice"]
```

0.4 Cluster 0 - Old Homes

[72]:
```
# Split into training and test set.
X_train, X_test, y_train, y_test = train_test_split(cluster0_x,␣
 ↪cluster0_y,test_size =0.3,random_state=0)

# Determine the best model
for max_leaves in␣
 ↪[2,3,5,50,150,200,250,300,500,1000,3000,5000,10000,20000,50000]:
    my_mae = get_mae(max_leaves, X_train, y_train, X_test, y_test)
    print("max leaf nodes : %d \t and Mean Absolute error :␣
 ↪%d"%(max_leaves,my_mae))
```

```
max leaf nodes : 2       and Mean Absolute error : 69654
max leaf nodes : 3       and Mean Absolute error : 66695
max leaf nodes : 5       and Mean Absolute error : 66654
max leaf nodes : 50      and Mean Absolute error : 55234
max leaf nodes : 150     and Mean Absolute error : 53764
max leaf nodes : 200     and Mean Absolute error : 54354
max leaf nodes : 250     and Mean Absolute error : 54830
max leaf nodes : 300     and Mean Absolute error : 55185
max leaf nodes : 500     and Mean Absolute error : 57518
max leaf nodes : 1000    and Mean Absolute error : 59525
max leaf nodes : 3000    and Mean Absolute error : 60887
max leaf nodes : 5000    and Mean Absolute error : 60864
max leaf nodes : 10000   and Mean Absolute error : 60864
```

```
max leaf nodes : 20000   and Mean Absolute error : 60864
max leaf nodes : 50000   and Mean Absolute error : 60864
```

[74]:
```
# Build the best tree model.
tree_model = tree.DecisionTreeRegressor(max_leaf_nodes = 150,␣
 ↪max_features='auto', splitter='best', random_state = 0)
tree_model = tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)


MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
8283670074.111674
53764.92627923287
18.54822269754711
```

0.4.1 Random Forest

[75]:
```
# Determine the best model
for n_estimators in [500, 1000]:
    for max_depth in [10,25,50,100]:
        my_mae = get_mae_rf(n_estimators, max_depth, X_train, y_train, X_test,␣
 ↪y_test)
        print("n estimators : %d \t and max depth : %d \t and Mean Absolute␣
 ↪error : %d"%(n_estimators, max_depth, my_mae))
```

```
n estimators : 500       and max depth : 10      and Mean Absolute error : 47820
n estimators : 500       and max depth : 25      and Mean Absolute error : 45546
n estimators : 500       and max depth : 50      and Mean Absolute error : 45524
n estimators : 500       and max depth : 100     and Mean Absolute error : 45524
n estimators : 1000      and max depth : 10      and Mean Absolute error : 47711
n estimators : 1000      and max depth : 25      and Mean Absolute error : 45425
n estimators : 1000      and max depth : 50      and Mean Absolute error : 45428
n estimators : 1000      and max depth : 100     and Mean Absolute error : 45428
```

[77]:
```
# Build the best random forest model
forest_model = RandomForestRegressor(bootstrap=True, n_estimators=1000,␣
 ↪max_depth=25, criterion = 'mse',
                             random_state=0)
forest_model.fit(X_train, y_train)
```

```
y_pred = forest_model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
6469160074.115412
45425.95836811525
15.976575208662226
```

0.4.2 Neural Network - One Hidden Layer

[78]:
```
#Create scaled data
scaler = StandardScaler()
scaler.fit(X_train)

X_train_s = scaler.transform(X_train)
X_test_s = scaler.transform(X_test)
```

[79]:
```
# Determine the best model
for hidden_layer in [100,250,500,750]:
    my_mae = get_mae_nn(hidden_layer, X_train_s, y_train, X_test_s, y_test)
    print("Hidden layer : %d \t and Mean Absolute error :␣
 ↪%d"%(hidden_layer,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 100       and Mean Absolute error : 155350

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Mean Absolute error : 78475
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Mean Absolute error : 63771
Hidden layer : 750       and Mean Absolute error : 57596

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[80]:
```
# Build the best single layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(750), max_iter=1000, activation='relu',␣
 ↪random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
8839544626.130396
57596.572528149874
20.194371018469134

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.4.3 Neural Network - Two Hidden Layers

[82]:
```
# Determine the best model
for hidden_layer in [250, 100]:
    for hidden_layer2 in [100, 50, 25]:
        my_mae = get_mae_nn2(hidden_layer, hidden_layer2, X_train_s, y_train,␣
↪X_test_s, y_test)
        print("Hidden layer : %d \t and Hidden Layer2 : %d \t Mean Absolute␣
↪error : %d"%(hidden_layer, hidden_layer2,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 250       and Hidden Layer2 : 100         Mean Absolute error :
49528
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 250       and Hidden Layer2 : 50          Mean Absolute error :
49398
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 250       and Hidden Layer2 : 25          Mean Absolute error :
49534
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 100       and Hidden Layer2 : 100         Mean Absolute error :
49474
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 100       and Hidden Layer2 : 50          Mean Absolute error :
49232
Hidden layer : 100       and Hidden Layer2 : 25          Mean Absolute error :
49387

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[84]:
```
# Build the best two layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(100,50), max_iter=1000,␣
 ↪activation='relu', random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
7563978804.583362
49232.77270906934
17.38966353780262

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.5 Cluster 1 - New Homes

0.5.1 Decision Tree

[85]:
```
# Split into training and test set.
X_train, X_test, y_train, y_test = train_test_split(cluster1_x,␣
 ↪cluster1_y,test_size =0.3,random_state=0)

# Determine the best model
```

```
for max_leaves in␣
 ↪[2,3,5,50,150,200,250,300,500,1000,3000,5000,10000,20000,50000]:
    my_mae = get_mae(max_leaves, X_train, y_train, X_test, y_test)
    print("max leaf nodes : %d \t and Mean Absolute error :␣
 ↪%d"%(max_leaves,my_mae))
```

```
max leaf nodes : 2       and Mean Absolute error : 50884
max leaf nodes : 3       and Mean Absolute error : 47066
max leaf nodes : 5       and Mean Absolute error : 44512
max leaf nodes : 50      and Mean Absolute error : 36379
max leaf nodes : 150     and Mean Absolute error : 34385
max leaf nodes : 200     and Mean Absolute error : 34210
max leaf nodes : 250     and Mean Absolute error : 33414
max leaf nodes : 300     and Mean Absolute error : 33696
max leaf nodes : 500     and Mean Absolute error : 33304
max leaf nodes : 1000    and Mean Absolute error : 33765
max leaf nodes : 3000    and Mean Absolute error : 34655
max leaf nodes : 5000    and Mean Absolute error : 34674
max leaf nodes : 10000   and Mean Absolute error : 34673
max leaf nodes : 20000   and Mean Absolute error : 34673
max leaf nodes : 50000   and Mean Absolute error : 34673
```

[86]:
```
# Build the best tree model.
tree_model = tree.DecisionTreeRegressor(max_leaf_nodes = 500,␣
 ↪max_features='auto', splitter='best', random_state = 0)
tree_model = tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)


MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
2611214074.244445
33304.09969593975
9.646638511094276
```

0.5.2 Random Forest

[87]:
```
# Determine the best model
for n_estimators in [500, 1000]:
    for max_depth in [10,25,50,100]:
        my_mae = get_mae_rf(n_estimators, max_depth, X_train, y_train, X_test,␣
↪y_test)
        print("n estimators : %d \t and max depth : %d \t and Mean Absolute␣
↪error : %d"%(n_estimators, max_depth, my_mae))
```

```
n estimators : 500      and max depth : 10      and Mean Absolute error : 28140
n estimators : 500      and max depth : 25      and Mean Absolute error : 25071
n estimators : 500      and max depth : 50      and Mean Absolute error : 25055
n estimators : 500      and max depth : 100     and Mean Absolute error : 25055
n estimators : 1000     and max depth : 10      and Mean Absolute error : 28149
n estimators : 1000     and max depth : 25      and Mean Absolute error : 25063
n estimators : 1000     and max depth : 50      and Mean Absolute error : 25049
n estimators : 1000     and max depth : 100     and Mean Absolute error : 25049
```

[88]:
```
# Build the best random forest model
forest_model = RandomForestRegressor(bootstrap=True, n_estimators=1000,␣
↪max_depth=50, criterion = 'mse',
                                     random_state=0)
forest_model.fit(X_train, y_train)

y_pred = forest_model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
1494635610.4229672
25049.423053717095
7.385039308461962
```

0.5.3 Neural Network - One Hidden Layer

[89]:
```
#Create scaled data
scaler = StandardScaler()
scaler.fit(X_train)

X_train_s = scaler.transform(X_train)
```

```
X_test_s = scaler.transform(X_test)
```

[90]:
```
# Determine the best model
for hidden_layer in [100,250,500,750]:
    my_mae = get_mae_nn(hidden_layer, X_train_s, y_train, X_test_s, y_test)
    print("Hidden layer : %d \t and Mean Absolute error :␣
 ↪%d"%(hidden_layer,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 100       and Mean Absolute error : 134061

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Mean Absolute error : 68852

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Mean Absolute error : 45300
Hidden layer : 750       and Mean Absolute error : 38917

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[91]:
```
# Build the best single layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(750), max_iter=1000, activation='relu',␣
 ↪random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)
```

```
MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
2849912418.6158223
38917.08298002663
11.190152161890854

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.5.4 Neural Network - Two Hidden Layers

[92]:
```
# Determine the best model
for hidden_layer in [250, 100]:
    for hidden_layer2 in [100, 50, 25]:
        my_mae = get_mae_nn2(hidden_layer, hidden_layer2, X_train_s, y_train,␣
↪X_test_s, y_test)
        print("Hidden layer : %d \t and Hidden Layer2 : %d \t Mean Absolute␣
↪error : %d"%(hidden_layer, hidden_layer2,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Hidden Layer2 : 100         Mean Absolute error :
31378

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Hidden Layer2 : 50          Mean Absolute error :
31792
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Hidden Layer2 : 25          Mean Absolute error :
32004

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 100       and Hidden Layer2 : 100         Mean Absolute error :
31915

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 100       and Hidden Layer2 : 50          Mean Absolute error :
32247
Hidden layer : 100       and Hidden Layer2 : 25          Mean Absolute error :
32663

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[93]:
```
# Build the best two layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(250,100), max_iter=1000,␣
 ↪activation='relu', random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)
```

```
MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
1971650115.892438
31378.292685691016
9.09300835801417

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.6 Cluster 2 - Large Homes

0.6.1 Decision Tree

[94]:
```
# Split into training and test set.
X_train, X_test, y_train, y_test = train_test_split(cluster2_x,␣
↪cluster2_y,test_size =0.3,random_state=0)

# Determine the best model
for max_leaves in␣
↪[2,3,5,50,150,200,250,300,500,1000,3000,5000,10000,20000,50000]:
    my_mae = get_mae(max_leaves, X_train, y_train, X_test, y_test)
    print("max leaf nodes : %d \t and Mean Absolute error :␣
↪%d"%(max_leaves,my_mae))
```

```
max leaf nodes : 2       and Mean Absolute error : 147054
max leaf nodes : 3       and Mean Absolute error : 139225
max leaf nodes : 5       and Mean Absolute error : 125939
max leaf nodes : 50      and Mean Absolute error : 111908
max leaf nodes : 150     and Mean Absolute error : 108681
max leaf nodes : 200     and Mean Absolute error : 108481
max leaf nodes : 250     and Mean Absolute error : 109183
max leaf nodes : 300     and Mean Absolute error : 107859
max leaf nodes : 500     and Mean Absolute error : 108536
max leaf nodes : 1000    and Mean Absolute error : 109151
max leaf nodes : 3000    and Mean Absolute error : 109000
max leaf nodes : 5000    and Mean Absolute error : 109000
max leaf nodes : 10000   and Mean Absolute error : 109000
max leaf nodes : 20000   and Mean Absolute error : 109000
max leaf nodes : 50000   and Mean Absolute error : 109000
```

[96]:
```
# Build the best tree model.
```

```
tree_model = tree.DecisionTreeRegressor(max_leaf_nodes = 300,␣
 ↪max_features='auto', splitter='best', random_state = 0)
tree_model = tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)


MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
44238667370.31389
107859.211311511
17.53086341011697
```

0.6.2 Random Forest

[98]:
```
# Determine the best model
for n_estimators in [100, 500, 1000]:
    for max_depth in [10,25,50,100]:
        my_mae = get_mae_rf(n_estimators, max_depth, X_train, y_train, X_test,␣
 ↪y_test)
        print("n estimators : %d \t and max depth : %d \t and Mean Absolute␣
 ↪error : %d"%(n_estimators, max_depth, my_mae))
```

```
n estimators : 100      and max depth : 10      and Mean Absolute error : 85281
n estimators : 100      and max depth : 25      and Mean Absolute error : 77813
n estimators : 100      and max depth : 50      and Mean Absolute error : 77465
n estimators : 100      and max depth : 100     and Mean Absolute error : 77465
n estimators : 500      and max depth : 10      and Mean Absolute error : 85225
n estimators : 500      and max depth : 25      and Mean Absolute error : 77492
n estimators : 500      and max depth : 50      and Mean Absolute error : 77521
n estimators : 500      and max depth : 100     and Mean Absolute error : 77521
n estimators : 1000     and max depth : 10      and Mean Absolute error : 85286
n estimators : 1000     and max depth : 25      and Mean Absolute error : 77566
n estimators : 1000     and max depth : 50      and Mean Absolute error : 77662
n estimators : 1000     and max depth : 100     and Mean Absolute error : 77662
```

[99]:
```
# Build the best random forest model
forest_model = RandomForestRegressor(bootstrap=True, n_estimators=100,␣
 ↪max_depth=50, criterion = 'mse',
                             random_state=0)
forest_model.fit(X_train, y_train)
```

```
y_pred = forest_model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
23706055893.16915
77465.38449770093
12.210480656978474
```

0.6.3 Neural Network - One Hidden Layer

[100]:
```
#Create scaled data
scaler = StandardScaler()
scaler.fit(X_train)

X_train_s = scaler.transform(X_train)
X_test_s = scaler.transform(X_test)
```

[102]:
```
# Determine the best model
for hidden_layer in [250,500,750,1000]:
    my_mae = get_mae_nn(hidden_layer, X_train_s, y_train, X_test_s, y_test)
    print("Hidden layer : %d \t and Mean Absolute error :␣
 ↪%d"%(hidden_layer,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Mean Absolute error : 502894

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Mean Absolute error : 390702
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 750       and Mean Absolute error : 302668
Hidden layer : 1000      and Mean Absolute error : 242681

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[103]:
```
# Build the best single layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(1000), max_iter=1000, activation='relu',␣
 ↪random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
94969726971.85104
242681.71296869867
39.1156077372017

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.6.4 Neural Network - Two Hidden Layers

[105]:
```
# Determine the best model
for hidden_layer in [500, 250]:
    for hidden_layer2 in [100, 50, 25]:
        my_mae = get_mae_nn2(hidden_layer, hidden_layer2, X_train_s, y_train,␣
↪X_test_s, y_test)
        print("Hidden layer : %d \t and Hidden Layer2 : %d \t Mean Absolute␣
↪error : %d"%(hidden_layer, hidden_layer2,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 500       and Hidden Layer2 : 100         Mean Absolute error :
91800
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 500       and Hidden Layer2 : 50          Mean Absolute error :
91645
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 500       and Hidden Layer2 : 25          Mean Absolute error :
92475
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 250       and Hidden Layer2 : 100         Mean Absolute error :
91657
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 250       and Hidden Layer2 : 50          Mean Absolute error :
93104
Hidden layer : 250       and Hidden Layer2 : 25          Mean Absolute error :
94318

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[106]:
```
# Build the best two layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(500,50), max_iter=1000,␣
 ↪activation='relu', random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
27808452154.432747
91645.62718001976
13.966358820900759

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.7 Cluster 0 - Community - Old Homes

0.7.1 Decision Tree

[107]:
```
# Split into training and test set.
X_train, X_test, y_train, y_test = train_test_split(cluster0_counts_x,␣
 ↪cluster0_counts_y,test_size =0.3,random_state=0)

# Determine the best model
```

```
for max_leaves in␣
 ↪[2,3,5,50,150,200,250,300,500,1000,3000,5000,10000,20000,50000]:
    my_mae = get_mae(max_leaves, X_train, y_train, X_test, y_test)
    print("max leaf nodes : %d \t and Mean Absolute error :␣
 ↪%d"%(max_leaves,my_mae))
```

```
max leaf nodes : 2       and Mean Absolute error : 67392
max leaf nodes : 3       and Mean Absolute error : 64618
max leaf nodes : 5       and Mean Absolute error : 64588
max leaf nodes : 50      and Mean Absolute error : 54080
max leaf nodes : 150     and Mean Absolute error : 51319
max leaf nodes : 200     and Mean Absolute error : 50353
max leaf nodes : 250     and Mean Absolute error : 50452
max leaf nodes : 300     and Mean Absolute error : 50224
max leaf nodes : 500     and Mean Absolute error : 51626
max leaf nodes : 1000    and Mean Absolute error : 51708
max leaf nodes : 3000    and Mean Absolute error : 51704
max leaf nodes : 5000    and Mean Absolute error : 51530
max leaf nodes : 10000   and Mean Absolute error : 51530
max leaf nodes : 20000   and Mean Absolute error : 51530
max leaf nodes : 50000   and Mean Absolute error : 51530
```

[108]:
```
# Build the best tree model.
tree_model = tree.DecisionTreeRegressor(max_leaf_nodes = 300,␣
 ↪max_features='auto', splitter='best', random_state = 0)
tree_model = tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)


MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
9252967622.68576
50224.178581149485
17.8027127246439
```

0.7.2 Random Forest

[109]:
```
# Determine the best model
for n_estimators in [500, 1000]:
    for max_depth in [10,25,50,100]:
        my_mae = get_mae_rf(n_estimators, max_depth, X_train, y_train, X_test,␣
↪y_test)
        print("n estimators : %d \t and max depth : %d \t and Mean Absolute␣
↪error : %d"%(n_estimators, max_depth, my_mae))
```

```
n estimators : 500      and max depth : 10      and Mean Absolute error : 41741
n estimators : 500      and max depth : 25      and Mean Absolute error : 37256
n estimators : 500      and max depth : 50      and Mean Absolute error : 37183
n estimators : 500      and max depth : 100     and Mean Absolute error : 37183
n estimators : 1000     and max depth : 10      and Mean Absolute error : 41745
n estimators : 1000     and max depth : 25      and Mean Absolute error : 37142
n estimators : 1000     and max depth : 50      and Mean Absolute error : 37153
n estimators : 1000     and max depth : 100     and Mean Absolute error : 37153
```

[110]:
```
# Build the best random forest model
forest_model = RandomForestRegressor(bootstrap=True, n_estimators=1000,␣
↪max_depth=25, criterion = 'mse',
                                     random_state=0)
forest_model.fit(X_train, y_train)

y_pred = forest_model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
4524567238.605582
37142.24903008234
13.491965868742698
```

0.7.3 Neural Network - One Hidden Layer

[111]:
```
#Create scaled data
scaler = StandardScaler()
scaler.fit(X_train)

X_train_s = scaler.transform(X_train)
```

```
X_test_s = scaler.transform(X_test)
```

[112]:
```
# Determine the best model
for hidden_layer in [250,500,750,1000]:
    my_mae = get_mae_nn(hidden_layer, X_train_s, y_train, X_test_s, y_test)
    print("Hidden layer : %d \t and Mean Absolute error :␣
 ↪%d"%(hidden_layer,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Mean Absolute error : 76252

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Mean Absolute error : 60571

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 750       and Mean Absolute error : 54462
Hidden layer : 1000      and Mean Absolute error : 50128

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[113]:
```
# Build the best single layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(1000), max_iter=1000, activation='relu',␣
 ↪random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)
```

```
MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
6632070491.350225
50128.38718016801
18.09158291501098

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.8 Neural Network - Two Hidden Layers

[115]:
```
# Determine the best model
for hidden_layer in [500, 250]:
    for hidden_layer2 in [250,100]:
        my_mae = get_mae_nn2(hidden_layer, hidden_layer2, X_train_s, y_train,␣
↪X_test_s, y_test)
        print("Hidden layer : %d \t and Hidden Layer2 : %d \t Mean Absolute␣
↪error : %d"%(hidden_layer, hidden_layer2,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Hidden Layer2 : 250         Mean Absolute error :
42179

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Hidden Layer2 : 100         Mean Absolute error :
42897
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Hidden Layer2 : 250         Mean Absolute error :
42500
Hidden layer : 250       and Hidden Layer2 : 100         Mean Absolute error :
42925

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[116]:
```
# Build the best two layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(500,250), max_iter=1000,␣
 ↪activation='relu', random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
5284053303.372593
42179.65188964617
15.205906511776007

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.9 Cluster 1 - Community - New Homes

0.9.1 Decision Tree

[117]:
```
# Split into training and test set.
X_train, X_test, y_train, y_test = train_test_split(cluster1_counts_x,␣
 ↪cluster1_counts_y,test_size =0.3,random_state=0)

# Determine the best model
for max_leaves in␣
 ↪[2,3,5,50,150,200,250,300,500,1000,3000,5000,10000,20000,50000]:
    my_mae = get_mae(max_leaves, X_train, y_train, X_test, y_test)
    print("max leaf nodes : %d \t and Mean Absolute error :␣
 ↪%d"%(max_leaves,my_mae))
```

```
max leaf nodes : 2       and Mean Absolute error : 51644
max leaf nodes : 3       and Mean Absolute error : 48050
max leaf nodes : 5       and Mean Absolute error : 45495
max leaf nodes : 50      and Mean Absolute error : 35686
max leaf nodes : 150     and Mean Absolute error : 31651
max leaf nodes : 200     and Mean Absolute error : 30914
max leaf nodes : 250     and Mean Absolute error : 30543
max leaf nodes : 300     and Mean Absolute error : 30137
max leaf nodes : 500     and Mean Absolute error : 29475
max leaf nodes : 1000    and Mean Absolute error : 28597
max leaf nodes : 3000    and Mean Absolute error : 28638
max leaf nodes : 5000    and Mean Absolute error : 28523
max leaf nodes : 10000   and Mean Absolute error : 28523
max leaf nodes : 20000   and Mean Absolute error : 28523
max leaf nodes : 50000   and Mean Absolute error : 28523
```

[118]:
```
# Build the best tree model.
tree_model = tree.DecisionTreeRegressor(max_leaf_nodes = 5000,␣
 ↪max_features='auto', splitter='best', random_state = 0)
tree_model = tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)


MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
2373551673.0864406
```

```
28523.137360996167
8.610516190717481
```

0.9.2 Random Forest

[119]:
```
# Determine the best model
for n_estimators in [500, 1000]:
    for max_depth in [10,25,50,100]:
        my_mae = get_mae_rf(n_estimators, max_depth, X_train, y_train, X_test,␣
↪y_test)
        print("n estimators : %d \t and max depth : %d \t and Mean Absolute␣
↪error : %d"%(n_estimators, max_depth, my_mae))
```

```
n estimators : 500      and max depth : 10      and Mean Absolute error : 25395
n estimators : 500      and max depth : 25      and Mean Absolute error : 21531
n estimators : 500      and max depth : 50      and Mean Absolute error : 21568
n estimators : 500      and max depth : 100     and Mean Absolute error : 21568
n estimators : 1000     and max depth : 10      and Mean Absolute error : 25354
n estimators : 1000     and max depth : 25      and Mean Absolute error : 21550
n estimators : 1000     and max depth : 50      and Mean Absolute error : 21575
n estimators : 1000     and max depth : 100     and Mean Absolute error : 21575
```

[120]:
```
# Build the best random forest model
forest_model = RandomForestRegressor(bootstrap=True, n_estimators=500,␣
↪max_depth=25, criterion = 'mse',
                                     random_state=0)
forest_model.fit(X_train, y_train)

y_pred = forest_model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
1325666712.116515
21531.3939649972
6.674285939659905
```

0.9.3 Neural Network - One Hidden Layer

[121]:
```
#Create scaled data
scaler = StandardScaler()
scaler.fit(X_train)

X_train_s = scaler.transform(X_train)
X_test_s = scaler.transform(X_test)
```

[122]:
```
# Determine the best model
for hidden_layer in [250,500,750,1000]:
    my_mae = get_mae_nn(hidden_layer, X_train_s, y_train, X_test_s, y_test)
    print("Hidden layer : %d \t and Mean Absolute error :␣
 ↪%d"%(hidden_layer,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Mean Absolute error : 55844

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Mean Absolute error : 41838

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 750       and Mean Absolute error : 37646
Hidden layer : 1000      and Mean Absolute error : 35773

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[123]:
```
# Build the best single layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(1000), max_iter=1000, activation='relu',␣
 ↪random_state=0)
```

```
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
2657052882.1652417
35773.74391234945
10.816869616556502
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.9.4 Neural Network - Two Hidden Layers

[124]:
```
# Determine the best model
for hidden_layer in [500, 250]:
    for hidden_layer2 in [100, 50, 25]:
        my_mae = get_mae_nn2(hidden_layer, hidden_layer2, X_train_s, y_train,␣
↪X_test_s, y_test)
        print("Hidden layer : %d \t and Hidden Layer2 : %d \t Mean Absolute␣
↪error : %d"%(hidden_layer, hidden_layer2,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 500       and Hidden Layer2 : 100         Mean Absolute error :
28871
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 500       and Hidden Layer2 : 50          Mean Absolute error :
29170

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Hidden Layer2 : 25          Mean Absolute error :
29780

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Hidden Layer2 : 100         Mean Absolute error :
28894

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Hidden Layer2 : 50          Mean Absolute error :
29353
Hidden layer : 250       and Hidden Layer2 : 25          Mean Absolute error :
30352

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[125]:
```
# Build the best two layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(500,100), max_iter=1000,␣
 ↪activation='relu', random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)
```

```
MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
1980681042.655062
28871.99255664342
8.793104334677786
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.10 Cluster 2 - Community - Large Homes

0.10.1 Decision Tree

[126]:
```
# Split into training and test set.
X_train, X_test, y_train, y_test = train_test_split(cluster2_counts_x,␣
 ↪cluster2_counts_y,test_size =0.3,random_state=0)

# Determine the best model
for max_leaves in␣
 ↪[2,3,5,50,150,200,250,300,500,1000,3000,5000,10000,20000,50000]:
    my_mae = get_mae(max_leaves, X_train, y_train, X_test, y_test)
    print("max leaf nodes : %d \t and Mean Absolute error :␣
 ↪%d"%(max_leaves,my_mae))
```

```
max leaf nodes : 2       and Mean Absolute error : 137505
max leaf nodes : 3       and Mean Absolute error : 126922
max leaf nodes : 5       and Mean Absolute error : 118679
max leaf nodes : 50      and Mean Absolute error : 100660
max leaf nodes : 150     and Mean Absolute error : 96498
max leaf nodes : 200     and Mean Absolute error : 97212
max leaf nodes : 250     and Mean Absolute error : 97427
max leaf nodes : 300     and Mean Absolute error : 96459
max leaf nodes : 500     and Mean Absolute error : 95979
max leaf nodes : 1000    and Mean Absolute error : 94576
max leaf nodes : 3000    and Mean Absolute error : 93897
max leaf nodes : 5000    and Mean Absolute error : 93897
max leaf nodes : 10000   and Mean Absolute error : 93897
max leaf nodes : 20000   and Mean Absolute error : 93897
max leaf nodes : 50000   and Mean Absolute error : 93897
```

[127]:
```
# Build the best tree model.
tree_model = tree.DecisionTreeRegressor(max_leaf_nodes = 3000,␣
 ↪max_features='auto', splitter='best', random_state = 0)
tree_model = tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)


MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
42125650339.976364
93897.91128163482
14.970183912085858
```

0.10.2 Random Forest

[128]:
```
# Determine the best model
for n_estimators in [500, 1000]:
    for max_depth in [10,25,50,100]:
        my_mae = get_mae_rf(n_estimators, max_depth, X_train, y_train, X_test,␣
 ↪y_test)
        print("n estimators : %d \t and max depth : %d \t and Mean Absolute␣
 ↪error : %d"%(n_estimators, max_depth, my_mae))
```

```
n estimators : 500      and max depth : 10      and Mean Absolute error : 74802
n estimators : 500      and max depth : 25      and Mean Absolute error : 65471
n estimators : 500      and max depth : 50      and Mean Absolute error : 65428
n estimators : 500      and max depth : 100     and Mean Absolute error : 65428
n estimators : 1000     and max depth : 10      and Mean Absolute error : 74724
n estimators : 1000     and max depth : 25      and Mean Absolute error : 65343
n estimators : 1000     and max depth : 50      and Mean Absolute error : 65267
n estimators : 1000     and max depth : 100     and Mean Absolute error : 65267
```

[129]:
```
# Build the best random forest model
forest_model = RandomForestRegressor(bootstrap=True, n_estimators=1000,␣
 ↪max_depth=50, criterion = 'mse',
                                random_state=0)
forest_model.fit(X_train, y_train)

y_pred = forest_model.predict(X_test)
```

```
MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
20346787127.528152
65267.54755389047
10.920858770150627
```

0.10.3 Neural Network - One Hidden Layer

[130]:
```
#Create scaled data
scaler = StandardScaler()
scaler.fit(X_train)

X_train_s = scaler.transform(X_train)
X_test_s = scaler.transform(X_test)
```

[131]:
```
# Determine the best model
for hidden_layer in [250,500,750,1000]:
    my_mae = get_mae_nn(hidden_layer, X_train_s, y_train, X_test_s, y_test)
    print("Hidden layer : %d \t and Mean Absolute error : %d"%(hidden_layer,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Mean Absolute error : 481370

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Mean Absolute error : 361541

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Hidden layer : 750       and Mean Absolute error : 279638
Hidden layer : 1000      and Mean Absolute error : 224753

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[132]:
```
# Build the best single layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(1000), max_iter=1000, activation='relu',␣
 ↪random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
83180622293.53993
224753.73653948793
37.06356365280802

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

0.10.4 Neural Network - Two Hidden Layers

[133]:
```
# Determine the best model
for hidden_layer in [500, 250]:
    for hidden_layer2 in [100, 50, 25]:
        my_mae = get_mae_nn2(hidden_layer, hidden_layer2, X_train_s, y_train,␣
 ↪X_test_s, y_test)
        print("Hidden layer : %d \t and Hidden Layer2 : %d \t Mean Absolute␣
 ↪error : %d"%(hidden_layer, hidden_layer2,my_mae))
```

```
/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Hidden Layer2 : 100         Mean Absolute error :
81183

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Hidden Layer2 : 50          Mean Absolute error :
82166

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 500       and Hidden Layer2 : 25          Mean Absolute error :
82861

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Hidden Layer2 : 100         Mean Absolute error :
81376

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Hidden layer : 250       and Hidden Layer2 : 50          Mean Absolute error :
82940
Hidden layer : 250       and Hidden Layer2 : 25          Mean Absolute error :
89477

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[134]:
```
# Build the best two layer Neural Network
mlp = MLPRegressor(hidden_layer_sizes=(500,100), max_iter=1000,␣
 ↪activation='relu', random_state=0)
mlp.fit(X_train_s, y_train)


## predict test set
y_pred = mlp.predict(X_test_s)

MSE = mean_squared_error(y_test, y_pred)
print(MSE)

MAE = mean_absolute_error(y_test, y_pred)
print(MAE)

MAPE = mean_absolute_percentage_error(y_test, y_pred)
print(MAPE)
```

```
23884666364.348392
81183.05968627369
13.350655434040076

/Applications/anaconda3/lib/python3.7/site-
packages/sklearn/neural_network/multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[ ]:
```

```