Orion Cua

Professor Pantelis Monogioudis

CS 482 Section 102

24 April 2022

SmartEstate

Abstract:

SmartEstate combines machine learning with the real estate market. In the real estate market, there are a lot of ways to measure the potential risk of a property investment or predicting the potential growth in value of a property or even predicting the area that may produce the most profit for potential investors. You can also run formulas to determine the future value of a property. SmartEstate is concerned with the prediction aspect of the real estate industry. This applies to many areas, such as predicting the future value of a property or predicting the growth of value in an area. In this project, I have predicted the sale price of a property given about 80 variables that can be used to describe the property. The variables range from numerical values such as square footage of the garage and number of bathrooms in the house to categorial values such as roof material and type of road. To get the most accurate results, I’ve done data analysis on each of the variables and determined that some of the variables may not be necessary and could complicate the process and have therefore removed them. I’ve also removed the variables that I knew had little to no correlation to the sales price from my prior knowledge and research. Once that was completed, I transformed the categorical values into numeric to allow for cleaner analysis. After the data was cleaned and ready to use, I implemented a Random Forest classifier from scratch to predict the sale prices. The classifier worked well, and the predicted prices had a deviation of about fifteen percent off of the actual prices. It is also clear that the use of JAX was very helpful in speeding up the computation time throughout the learning process.

Introduction:

Given a complex data set, it is a challenge to understand ways to analyze the data and ultimately be able to use it in a way that makes sense. In this case, the problem was to formulate a machine learning algorithm to predict the sale prices of various properties. In the real world, every property has countless features. It is possible to describe a property in countless ways, from the material of the flooring to the height of the ceiling and even the slope of the house or street. Of course, not every one of these variables will impact the price of the property the same way. In fact, many of these variables that describe the property have little to no impact on the sale price. Therefore, even when given a countless list of variables, it is important to understand that not every variable has to be incorporated into the equation when computing the prices. This problem is very important in the real world. Every day there are property and houses being sold around the country. Each of the houses that were sold were sold at a certain price. The price of the property that was sold was calculated in some way. It could have been an estimation based on the price of other houses in the area, or it could have been trial and error, where the seller overpriced it just to test the offers that come in. Either way, the precedent for the prices of the houses was set and determined at some point. By using data analysis and machine learning, we can determine many things. First, we can find the correlation between certain variables to the sale price and find out what truly impacts the price of a house. For instance, it is easy to imagine that the total square footage of a house will impact the price of a house more than the number of bathrooms that are in the basement. Using machine learning, we are able to predict the price of houses by feeding the learner information about previous houses and the price that they sold for. Eventually, the learner will start to understand why the price of the house is what it is. Once it learns and become more accurate, we can use the algorithm to predict the price of a house given some variables. That is the problem that we are facing in this project.

For this project, I’ve decided to implement a Random Forest machine learning algorithm to predict the prices of some houses given some variables. A random forest works by combining multiple decision trees and getting the average results of each decision tree. A decision tree will split the variables over and over in order to determine the final value. In my implementation, I fed this learning algorithm over 1,000 entries to learn from. After testing the predictions of the algorithm and evaluating the results against the known prices of some houses, the learner was proven to be accurate with an average of a fifteen percent deviation from the actual price. That is to be expected, and I expect the results to be more accurate if there are more decision nodes added to the tree, as well as more data fed into the algorithm. The project was a success, and I was able to create a random forest algorithm from scratch that was able to produce accurate results.

Related Work:

The real estate market is very popular and there are many analysts and data scientists who enjoy analyzing property data and making predictions. There have also been some published work of other people analyzing the same dataset and making conclusions. There are many different ways to go about reading, cleanings and analyzing the data, and it was very interesting to learn about the different ways it has been done. My approach to cleaning and analyzing the data was similar to some implementation in the way that the correlation is calculated and displayed. It was very popular idea to find the variables that highly correlate with the sale price and to display the correlation between them on a graph. Most others cleaned the data as well, although I differ in the way I went about it. I cleaned the data using information from research as well as some minor data analytics.

There are also published works in the machine learning aspect of predicting the sales prices. Utilizing a Gradient Boosting technique was very common and seemed very practical as well. However, I decided to work on a Random Forest learner that only a few others have implemented. The other works that utilized a Random Forest model used sklearn packages that performed most the calculations for them. I decided to create one from scratch so that I am able to control the different algorithms and operations within them. I also utilized JAX in order to increase the computational speed, as some of the operations proved to be very time consuming.

Data:

The data given in the Kaggle competition was provided in a CSV format. It included 2 separate data files, one for the training set and one for the test set. The training set included 1460 instances, each with 80 total columns, including the sales price. The test set included 1460 instances as well, however, did not include the sales price, as that is the variable that is to be predicted. The variables included integers, floats, and objects. Data cleaning was necessary in order to analyze the data more efficiently. I first removed the columns that were over ninety percent null or consisted of ninety-five percent of redundant values. Then, I removed numerical columns that had close to zero correlation to the sales price. I also converted the categorial values into numeric values using one-hot encoding.

Methods:

For predicting the sale price, I decided to implement a Random Forest model. I created a Random Forest model from scratch so that I will be able to control the operations involved in creating the decision trees in the forest. I chose this approach so that I can take measures to increase the computing speed and improve accuracy in the results. I also wanted to ensure that I incorporate JAX into the computations within each decision tree. I also considered other methods such as using neural networks or gradient boosting methods. The computations in the decision trees were chosen and created in a way that will produce the most accurate results. With each decision node in a tree, the node will be split on a certain variable and threshold. I’ve implemented an algorithm that will parse through each and every variable and each possible threshold within each variable in order to find the combination that will result in the most productive split. The accuracy of the potential variance and threshold combination was calculated using the Residual Sum of Squares (RSS) algorithm. Since this algorithm will be computed thousands of times, it was important to minimize the computation time of this algorithm. I also created a dictionary for each DecisionTree object in order to navigate the architecture of each tree. The random forest was used to construct a selected number of trees and find the average prediction of all the trees. Within the random forest implementation, I incorporated a bootstrap to select the sample that will be used to create each decision tree.

Experiments:

To analyze the results of my random forest implementation, I used the Mean-Squared Error (MSE) algorithm, a Root Mean-Squared Error (RMSE) algorithm and found the average deviation percentage of the predictions compared to the actual prices. The initial random forest model that I created incorporated 10 decision trees in the forest, as well as a bootstrap of 0.9 and a max depth of 3 for each tree in the forest. The results were as follows:

Text

Description automatically generated

Given a small number of trees in the forest, it is clear that there is room for improvement. There have been some published works that have calculated the impact on the accuracy of their Random Forest implementation by toggling the number of trees in the forest and plotting the resulting MSE of each instance. The results indicated that there was no direct correlation between the number of trees in the forest and the overall accuracy of the results. A similar experiment was conducted to see the impact of the depth of each decision tree to the results. In this case, there did seem to be a correlation that indicated that the lower the tree depth, the higher the MSE would be, which means the less accurate the results are. However, at a certain point the deeper the depth of a tree, the less accurate the results become. There is no definite conclusion that can be created based on these experiments.

Conclusion:

Through building a Random Forest model from scratch, I was able to learn the different ways to calculate and split the nodes to determine the architecture of a decision tree. The results indicate that the algorithm that I implemented within the decision trees were accurate, but there definitely is room for improvement. It was also very interesting to be introduced to JAX and learn about the different packages within JAX that can be used in a machine learning environment. I was able to incorporate JAX functionality into my project and used it to increase the computation speed and efficiency. For future extensions, it would be possible to further increase the efficiency and processing speed of the random forest if the algorithms within the splitting operations were fully vectorized and computed with JAX. This project can also be extended to incorporate other learning methods such as Gradient Boosting and comparing the results and efficiency to that of the Random Forest learner. I am also interested in learning more libraries included with JAX that I will be able to take advantage of in machine learning. I have learned a lot through this project, and I feel that this has been a great step into the incredible world of machine learning.

Sources:

<https://jax.readthedocs.io/en/latest/notebooks/quickstart.html>

<https://jax.readthedocs.io/en/latest/jax.numpy.html>

<https://analyticsindiamag.com/how-to-speed-up-machine-learning-operations-with-jax/>

<https://www.kaggle.com/code/abdelrahmantarek13/houseprice-step-by-step>

<https://towardsdatascience.com/how-to-detect-constant-quasi-constant-features-in-your-dataset-a1ab7aea34b4>

<https://www.kaggle.com/code/lildatascientist/decision-tree-and-random-forest-from-scratch>

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques>

<https://medium.com/hanman/data-modeling-building-a-house-price-prediction-model-1450f825073b>