Predicting Housing Prices in Santiago, Chile

Second Capstone Project Report
Robert Walker

Problem Statement

Determining the value of a home can be difficult. There are a number of factors to consider, such as size and location, as well as information about the local housing market. Such a process can be tedious and imprecise when done by humans, so it is becoming increasingly common for companies to rely on algorithms for support. However, a model developed for a certain city or country may not work well in a different part of the world. The hypothetical client for this project is a real estate agency in Santiago, Chile. They need an in-house model to help determine an appropriate price for a home, based on a set of features. In order to build such a model, the relevant data must first be acquired somehow.

Data

Source

All of the data used in this project was scraped from the website chilepropiedades.cl. This website contains thousands of listings of homes for sale in Chile. This project focused exclusively on homes located in the Metropolitan Region of Chile.

Columns in the dataset after cleaning

date	date the listing was posted
comuna	local municipality in Chile
house	binary column, 1 for house, 0 for apartment
new	binary column, 1 for new home, 0 for used home
total_area	square meters, size of the property for houses, area for apartments
built_area	square meters, only for houses, area of the footprint of the house
bedrooms	number of bedrooms in the house
bathrooms	number of bathrooms in the house
furnished	binary column, 1 for furnished, 0 for unfurnished
address	street address of the property
description	description of the property, usually a few lines
days_old	days since listing was posted, derived from date column
price_uf	price in *unidades de fomento* (UF), used for property in Chile

Methodology

Web Scraping

The web scraping involved using two libraries: **requests** and **BeautifulSoup**. The **requests** library was used to obtain the HTML source code, which was then parsed using **BeautifulSoup**. After that the relevant information was extracted by searching for tags in the HTML that contain the information. This process involved quite a bit of trial and error, much of which is shown in the Jupyter notebook. Ultimately, the extraction was accomplished by creating functions that performed the web scraping and returned the results as **pandas** DataFrames. Each DataFrame contained the search results for a particular type of home (eg. house, apartment, studio, etc.). Finally, all of the DataFrames were merged together and saved as a csv file.

Data Wrangling

In the next stage, the raw data scraped from the website had to be converted into a form that could be used. All of the scraped data was in string form, so much of it had to be converted into a numerical form. For numbers in Spanish, commas and periods are used opposite to the way they are used in English, with commas marking decimals, and periods marking every three digits to the left of the decimal. This meant that the periods needed to be removed, and then the commas had to be replaced with periods. There were also categorical features that could be turned into binary features, for example "*nueva*" and "*usada*" became a binary column called **new** with ones for new homes and zeros for used homes.

Next, there were missing and extreme values to address. This was a somewhat messy process, given that it was not always clear what values were unreasonable. There was another issue regarding the *comunas*, which are municipalities within each region. Since Santiago can refer to both the city as a whole, and the central *comuna* within it, many homes were listed as being located in the comuna of Santiago, when they are actually located in a different *comuna*, so that had to be fixed. In addition, the prices came in different units, so these all had to be converted to a common unit. The unit I chose was *unidad de fomento* (UF), which is typically used for property values in Chile (this is meant to protect against fluctuations in the national currency). 1 UF is a little less than 40 USD.

Exploratory Data Analysis

The first step was to convert the date column to the datetime format. After that, I created a **days_old** column by subtracting the date each listing was posted from the date the data was scraped. The median number of days old was 11, and 75% of the listings were 13 days old or less. The days old column turned out to not have much correlation with price. Nevertheless, knowing the turnover rate could be very useful for deciding when to re-scrape the website to

acquire more data, or how frequently the model should be re-trained if we are concerned that prices could be changing.

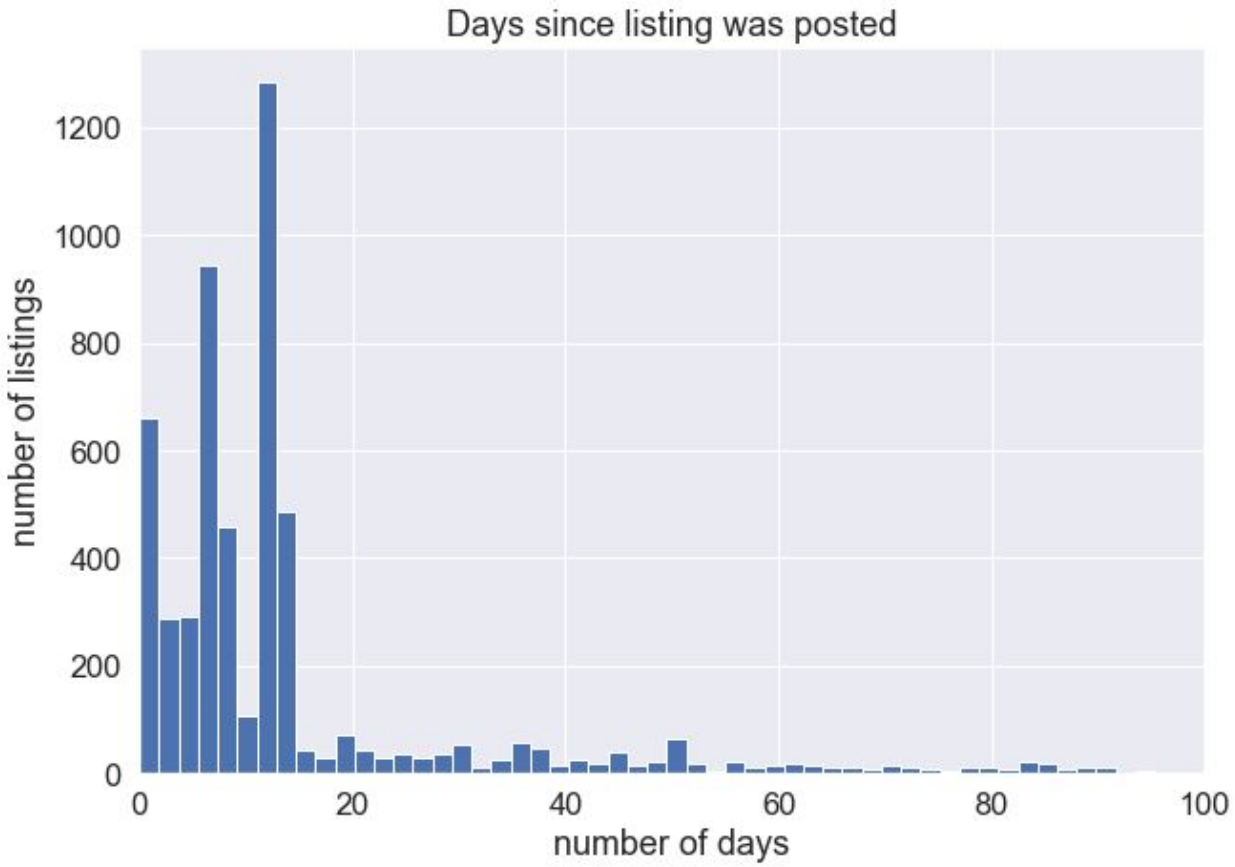

Figure 1: Histogram showing number of days since listing was posted

The next step was to convert all the different types of home into apartments or houses (for example studios are really just small apartments). Then this could also be converted to a binary column.

Once the features had been established, it was possible to make some inferences about the underlying population. Since the dataset was composed of listings that happened to be posted on the website on a particular day, confidence intervals were calculated to determine how close the sample means of various features are likely to be to the true population means. For example, the mean number of bathrooms for listings in the dataset was 2.175, but how different might this number be if we took another sample a month later when there are different listings? Calculations indicate that there is a 99% probability that the mean number of bathrooms will fall between 2.135 and 2.215.

After that, it was time to explore how the different features were related to one another, and how they related to the target variable: **price_uf**. Figure 2 shows the correlation between each pair of features in the dataset.

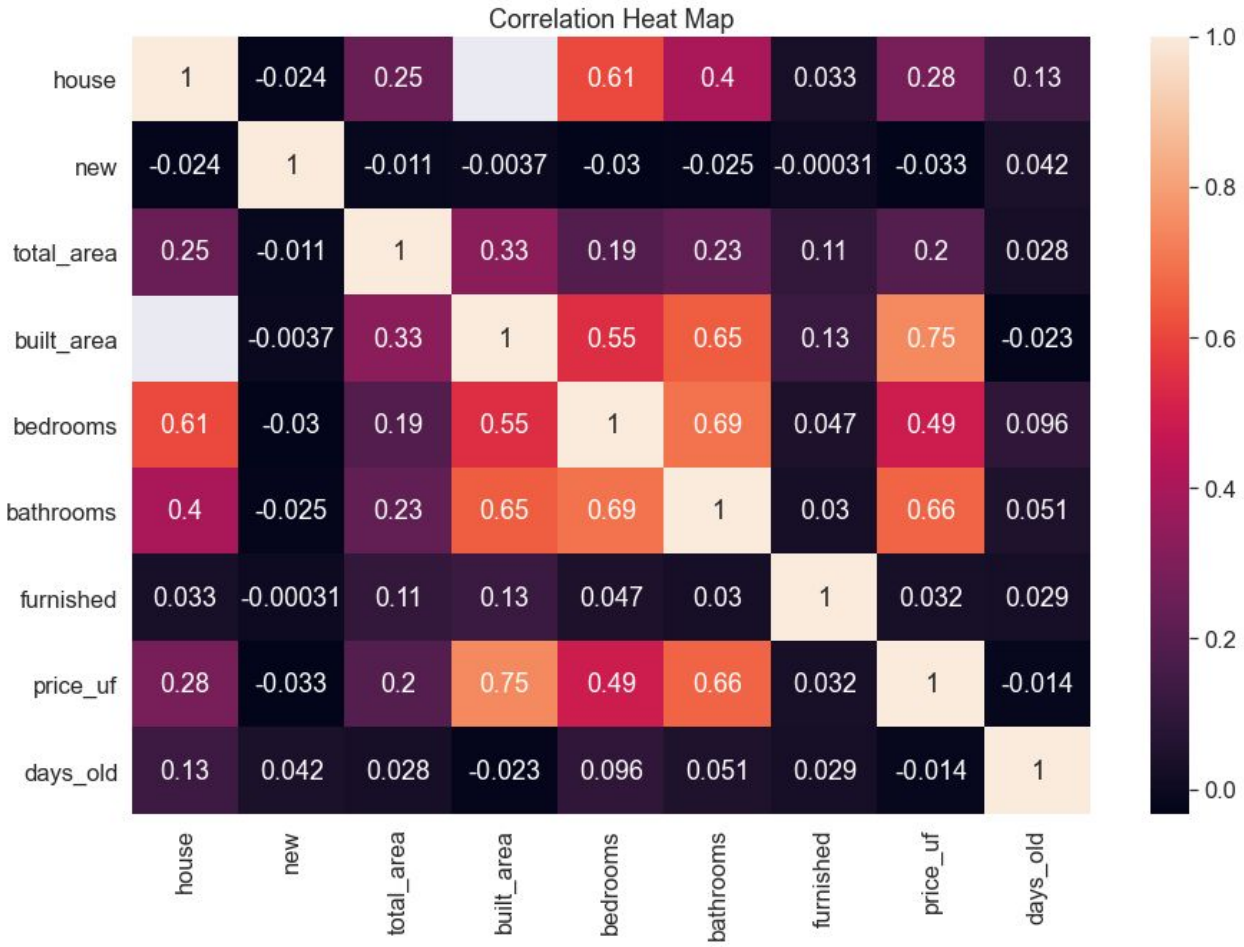

Figure 2: Correlation heatmap of the numerical features in the dataset

The features that correlate most strongly with price are **built_area** and **bathrooms**. Notably, the columns **new**, **furnished**, and **days_old** have virtually no correlation with the price, so these will not be used in the modeling stage.

The next step examined how prices differed according to *comuna*, and grouped *comunas* together to form zones, based on proximity and similar economic situation. The purpose of grouping the *comunas* together is to be able to use *comuna* information in the modeling stage of the project without having too many variables, since there are fifty or so *comunas* in the Metropolitan Region.

The final step in the EDA stage was to use geopandas to create some maps of the data. The pair of maps in Figure 3 shows mean value for price in each comuna or zone respectively, and the Jupyter notebook and slideshow contain similar maps for all of the other variables as well.

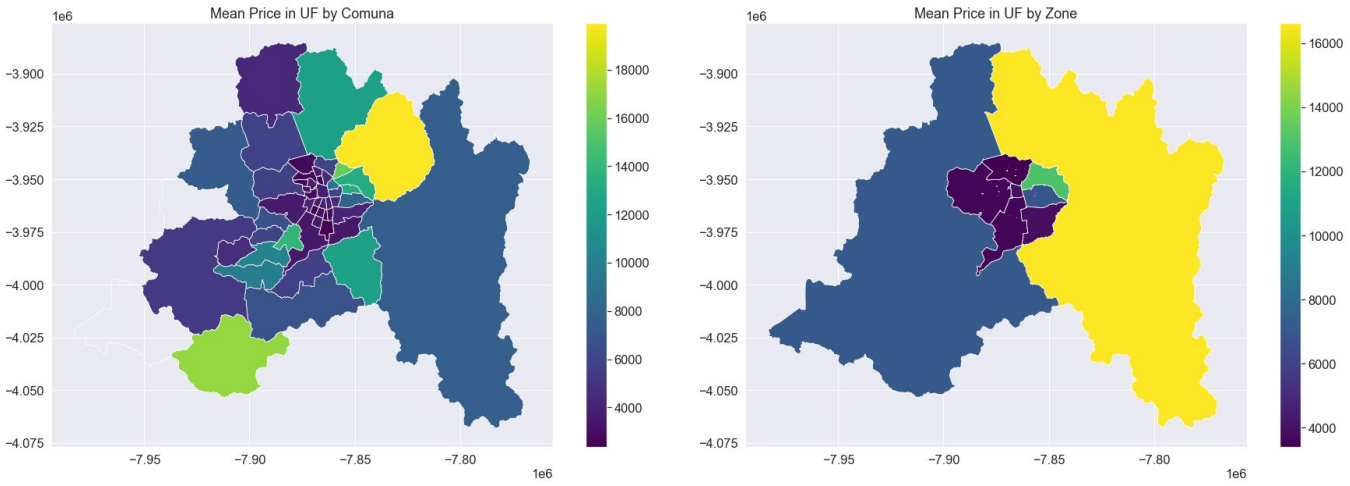

Figure 3: Maps showing average price by *comuna* and "zone"

Pre-processing

The first step of the pre-processing stage was to select which features to use in the modeling stage, based on how strongly they correlate with price. The list of features included **house**, **total_area**, **built_area**, **bedrooms**, **bathrooms**, and **zone**. The next step was to fill in the remaining missing values. For example, all of the apartments had missing values for the **built_area** column, so these values were filled using the values from the **total_area** column, since these two features could be considered synonymous for apartments. Next, the **zone** column was converted into dummy variables with one binary column for each zone. This was done because categorical data typically cannot be used in machine learning.

After that, missing values for the **bathroom** and **bedroom** column were filled using a ratio of three bedrooms for every two bathrooms (rounding down to the nearest whole number), and two bathrooms for every three bedrooms (rounding up to the nearest whole number). This was decided because the median number of bathrooms was two, and the median number of bedrooms was three, while the mean ratio of bathrooms to bedrooms was 0.79. Then, remaining apartments with missing area values were replaced with the median value, 64 square meters.

The next critical step was to split the data into training and test sets, with 30% of the data set aside for testing. The final pre-processing step was to scale the training features, since many of them had skewed distributions with a few very large values. The features were scaled logarithmically, and the scaled data as well as the original data were saved for the modeling stage.

Modeling

Finally the data was ready to be modeled with machine learning algorithms. Once the train and test data were imported, the first step was to create a dummy regressor. This initial "model" works by simply taking the mean of **y_train** (other strategies such as median can also be used) and using that value as the prediction for every listing. This approach does not even consider other features, and simply serves to establish a baseline for the performance of our model. The mean squared error for the predictions of the dummy regressor on the training set is equal to variance of **y_train**, and the R-squared is equal to zero.

Four different algorithms were used: Linear Regression, K Nearest Neighbor Regressor, Random Forest Regressor, and XGBoost Regressor. Each of these was used with both the original data and the scaled data, creating a total of eight different models. Ultimately, the best score was achieved by XGBoost using the scaled data, although the accuracy was almost identical for XGBoost and Random Forest, and using both scaled and unscaled data.

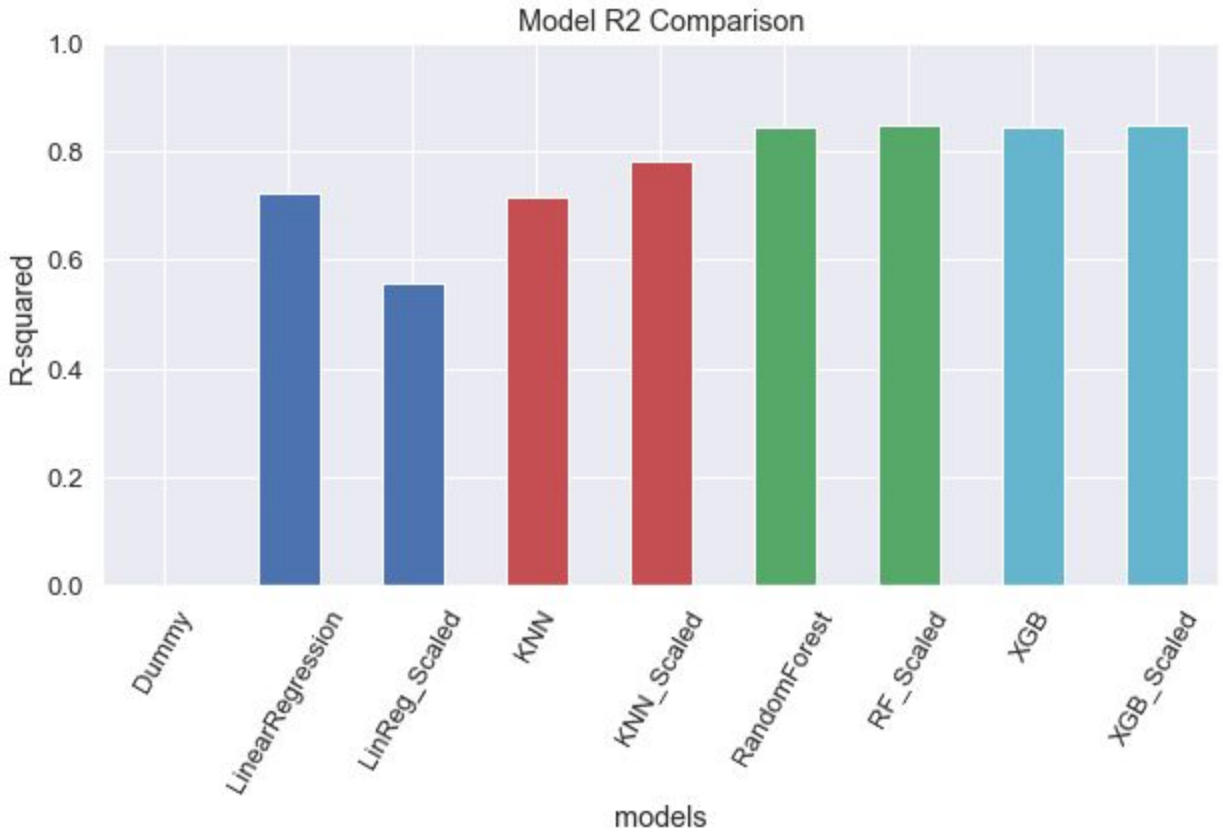

Figure 4: Bar chart showing the R-squared of each model

All of the top four models achieved an R-squared of just under 0.85, and had a mean absolute error of around 1,400 UF (a little over 53,000 USD). This was a roughly 70% reduction of MAE compared to the baseline established by the dummy regressor. Linear Regression worked much better using the original data than with the scaled data, but it still underperformed

compared to the other algorithms. K Nearest Neighbor did slightly better with the scaled data than the original data, but still was not as effective as the decision tree algorithms, Random Forest and XGBoost. One interesting thing to note is that although these two algorithms performed almost identically, they used very different methods, placing importance on different features. This suggests that there might be a natural ceiling to performance given the existing data. For the Random Forest models, the most important feature was **built_area** followed by **total_area**. For XGBoost, the most important was **zone_northeast** followed by **bathrooms**.

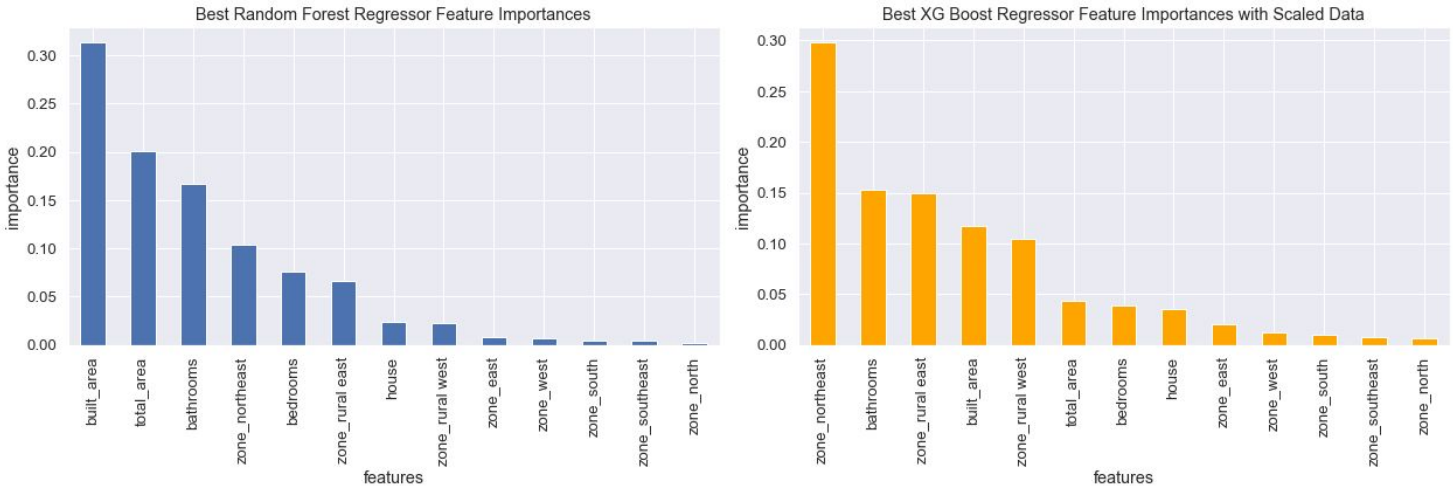

Figure 5: Bar charts showing feature importances for Random Forest and XGBoost models

Another factor to consider when selecting a model is how long it takes to train and make predictions. More complex models often take longer to train and to use, and more careful tuning of hyperparameters also adds to training time. There comes a point where tiny improvements in predictive ability no longer justify large increases in training or prediction time. Of the different models used in this project, Linear Regression and K Nearest Neighbor had the lowest training time. This was probably because they are simpler models, and also because there were fewer hyperparameters to try out. For Linear Regression, a Grid Search with cross-validation was used to determine the optimal number of features to include. (The model decided to use all of the features with both scaled and unscaled data.) For K Nearest Neighbor, a Grid Search was used to determine a value for K, a metric for distance, etc. With Random Forest and XGBoost, there were exponentially more potential combinations of hyperparameters, so Randomized Search with one hundred iterations was used instead of Grid Search to limit training time. Even so, they still required considerably more time to train.

Prediction time was lowest with Linear Regression, followed by KNN, then Random Forest, and finally XGBoost. In every case except for Linear Regression the prediction time was significantly higher when scaled data than when using the original data.

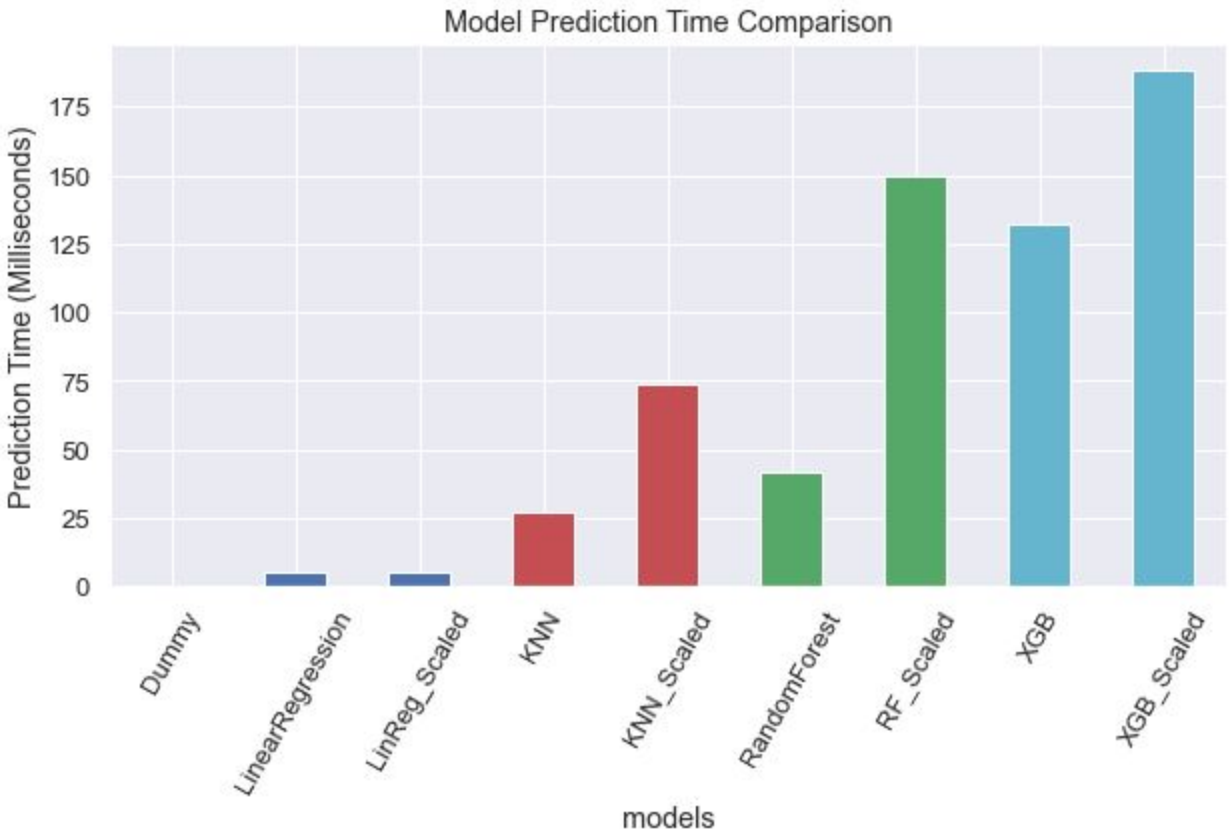

Figure 6: Bar chart showing prediction time in milliseconds for each model

Conclusions and Further Research

With the existing data, the decision tree algorithms (Random Forest and XGBoost performed the best. Of these, the Random Forest Regressor using the original data would probably be the best choice, due to its lower training time and prediction time. The model should be used to establish an initial idea for a price, which could then be refined based on other factors such as the condition of the home, the specific location (the zones used in the project were quite general), the proximity to public transportation, supermarkets, etc.

The model could be retrained with new data scraped from chilepropiedades.cl on a weekly or biweekly basis to ensure that it is up to date. It might also be useful to track how much price or other features change over time. In addition, one might re-scrape periodically or look for data from different sources in order to build a larger dataset, which could then be used to make more robust predictions. Many of the models seemed to be overfitting, which might be avoided by using more data.

The data initially scraped from the website also included an address column as well as a description column. Both of these columns could certainly yield valuable information for

predicting a price. For example, there are Python packages for parsing street addresses. This could help verify that the home is really located in a given comuna, something that was an issue in this project. It may also be possible to develop some sort of desirability score for a given location based on its proximity to parks, hospitals, supermarkets, public transport, etc. This would save the time required for a human to make such an assessment. As for the description column, it may be possible to apply sentiment analysis, or to identify key words such as *piscina* ("pool" in Spanish) which could affect the value of a home. Overall, this project should be seen as a valuable first step, on top of which more sophisticated and precise models can be built.