Tulga Kagan Temel
3168178

Predicting Housing Prices in Certain Italian Cities with Machine Learning

Preface:

Throughout my academic studies, I have had to deal with a lot of projects and assignments. However, I don't recall doing any project or assignment as time-consuming and didactic.

Despite being already acquainted with the pandas library in Python and not having to spend a lot of time trying to get used to the syntax, completing a hands-on exhaustive task like this one created a lot of difficulties.

The two datasets that were presented contained an unusually high percentage of missing values, which led me to try out a lot of different combinations of strategies in order to come up with the best method to deal with the missing values and outliers.

As difficult as it has been, this project has definitely been the one that taught me the most. Spending hours per day just to slightly improve the existing model by a little bit, but eventually failing over and over again was, first of all, a humbling experience, and one that is most definitely important.

Introduction:

In this report, we will examine a comprehensive approach to predicting housing prices, based on several factors. My goal was to create a model capable of making predictions as accurately as possible, without overfitting ot the train data.

Below, you will find my approach toward this problem and the insights behind the strategies I have chosen.

Data Preprocessing:

First, in order to roughly understand what kind of a dataset we are dealing with, I used Python Pandas' .describe() method.

Right off the bat, I observed some unrealistic values that corresponded to the min and max values of certain columns, along with a high amount of missing data. Afterwards, I visualized the missing values by percentage in order to grasp a better understanding of the magnitude of the missing values count.

After that, Based on my domain knowledge and statistical observation, I decided to eliminate the rows with price lower than 48750(the 0.5th percentile from the bottom), and those with price higher than 3300000(the 99th percentile).

The motivation behind removing hese rows was, getting rid of the noise in the data as much as I can, in order for those outliers not to mess with the model's performance.

Data Cleaning:

After getting rid of the unrealistic rows regarding the target variable, I moved on to performing data cleaning column by column.

Starting off with balcony, we observe that it either takes the value True or None. This led me to assume the values False were not entered correctly into the dataset, and were instead counted as None values. In order for my model to consider this column effectively, I swapped the values True to 1, and the rest to 0.

The next column that we inspect is conditions. It assumes one of 4 categories: Ottimo, Nuovo, Buono stato, and Da ristrutturare. When we group our preprocessed data by conditions (df_train.groupby("conditions").mean()), we see that the average price for Ottimo is the highest, followed by Nuovo, and then Da ristrutturare and Buono Stato, respectively. I could deal with this column by transforming it into an ordinal column, or I could create dummy variables corresponding to each category. After trying both options, dummy variables performed better, so I chose with the latter option. Since there were a few rows with missing values on the conditions column, I decided to create a fifth dummy column that takes the value 1 if it's missing and 0 if it isn't. Afterwards, I dropped the conditions column.

Construction_year was one of the columns that caused me the most trouble. With roughly 30% of the data missing, and some assuming the values from the medieval times and some assuming the values from the future, it was difficult to deal with this column. Among the strategies I have tried, the one that I eventually decided to use was converting those unrealistic values to None, and then impute them with the rest of the missing values.

Latitude and longitude columns were not particularly helpful, thus I decided to convert them into the city they correspond to. First of all, I scatterplotted latitude and longitude data, which showed me that the entire data is concentrated in three clusters. After dividing them by looking at the mean values of each cluster, and checking it on the map, it was clear that they corresponded to three cities: Milan, Venice and Rome. Thus, I assigned the coordinates to three variables, and computed the euclidean distance from the row's data to estimate which city they are in. And then I created three dummy variables corresponding to each city. I then created a fourth dummy that corresponds to the latitude and longitude value being missing. Afterwards, I dropped the Latitude and longitude columns.

43% of the energy efficiency column was missing, and I didn't consider energy efficiency a particularly helpful information, therefore I dropped the column altogether.

Expenses was another column that assumed a lot of missing values. However, because this might be a very important information that could be particularly helpful when predicting the target variable price, I didn't drop this column. Instead, I swapped the unrealistic outliers with None, and imputed them later on.

The reason I am not dropping the rows that contain unrealistic outliers for any column altogether is because I want to keep the rest information, as it may prove useful for my model.

For the floor and total_floors columns, again, I transformed the unrealistic values to None, the unrealistic values being any floor higher than 31, and any floor below -1.

The garden column was just like the balcony column, so it has seen the same treatment.

N_bathrooms had only a few missing values, which seemed reasonable to fill with median value, which is 1.

I transformed the proximity to center values to values that are more explicit in their effect, named city_center_distance, after filling the missing values with 1, as there were only a few of them missing.

For the surface, it was just like expenses: a lot of missing values and unrealistic outliers, but very valuable information. Therefore, it has seen the same treatment.

Imputing the missing values:

After the preprocessing and cleaning, it was time to impute the missing values. Before doing so, I observed the correlation heatmap, to get some indication on what columns to include together in the imputation.

For the imputation algorithm, I tried Iterative imputation and KNN imputation, as I had far too many missing data to choose mean or median imputation. Iterative imputation performed better, therefore I stuck with that.

After imputing the missing values, I transformed the construction_year data to age data, as it captures better that aspect of the building.

After all this, I did the same process for test data.

Training the model:

Before training the model, I split my train data in test and validation subdatasets, which I later used to train my data, as we conduct supervised learning.

I primarily tried XGB and Random Forest algorithms for my model. I compared them after tuning the hyperparameters, and XGBoost performed slightly better. For the hyperparameter optimization part, I used Weights and Biases, and it has been quite useful. In order to run Weights and Biases on my notebook, I used Google Colab, and therefore my code is optimal for Google Colab, and may be a little troublesome for Jupyter notebook. Therefore, I advise you to run my code on Google colab in case you encounter difficulties.