House Price Prediction - ML Challenge 30412

Francesco Vacca
Student ID: 3041929
Kaggle username: francescovacca

May 9, 2023

Abstract

This is the report on the House price prediction ML challenge for the course 30412. The goal of the project is to predict house prices in different italian cities using machine learning methods.

1 Introduction

The dataset 'train.csv' contains records for 46312 houses sold in Italy, with respective prices and other 15 features, while the 'poi.csv' dataset contains information about points of interest that can be used for feature engineering. The aim is to explore the train data and develop a machine learning model to predict the prices of the 'test.csv' dataset, eventually exploiting some information from the poi dataset.

2 Data Transformation

2.1 Exploratory Data Analysis

After having loaded some useful libraries, such as numpy, pandas for data pre-processing, and some scikit-learn packages, I performed some basic EDA (exploratory data analysis) on the train data, in order to gain an overview on the target variable distribution and on the correlations among features, with size related variables (surface and number of rooms/bathrooms), which seemed to be the most positively correlated with price, while construction year was the most strongly negatively correlated one. Moreover, I had an overview on the missing data (NaNs), which were numerous and could be a serious problem for many of the features.

2.2 Data pre-processing: missing data and outliers handling

First of all, I removed some of the most extreme outliers in price, which would have skewed the model. Then I analyzed feature by feature, in order to detect the outliers, fill the missing data, and removing the column if it was the case.

For some of the features, it made sense to drop the column, as they had more than 30 % of missing values, and didn't correlate in a meaningful way to prices; in particular, these features were: 'garden', 'balcony', 'expenses', 'total_floors', 'elevator', and 'energy_efficiency'. Note that for 'garden', 'balcony', and 'elevator', values were either 'True' or NaN, and it could be argued that the missing values are simply falses. I initially tested this, replacing nans with falses, but since this is a huge assumption, and since this didn't improve the correlation with price, I decided to discard the features.

For latitude and longitude, the dataset contained a few missing values, which could however have large impact on the model, so I removed the 13 houses of which the position was unknown.

About 'conditions', this is a categorical variable with 4 different values, according to the condition of the house. I decided to replace the categorical values with ordinal encoding, with values representing ascending quality of condition, while for the null values I filled the entries with 0, representing unknown conditions.

Dealing with number of rooms and bathrooms, I decided to fill the null values with the median of the train dataset, as it is a statistics less sensible to outliers compared to the mean.

The surface regressor deserves a specific analysis. Firstly, it must be seen that there are 96 entries with zero surface, which seem as record errors. While they could be transormed into nans, and then treated equally to the other nans, it must be noted that their average price is around 6 million (with median 3.8 million), while for the missing values, the price mean and median are around 300k. Therefore, for the null values, it could be the case to regress the surface from the information at our disposal, namely, the number of rooms and bathrooms. On the other hand, I left the zero surface records as they are, as the model used (Random Forest) seems to recognize this 'flag', and computes better predictions on the test records with zero surface.

Finally, regarding 'floor' and 'construction year', it must be noted that both have some no-sense value, specifically an house at the 56th floor (not existent in Italy), and an house built in the year 2500. After having removed these outliers, I imputed the null values using the scikit-learn's KNNImputer, which compares the most similar houses across the dataset and assigns the most likely values given those.

2.3 Feature Augmentation

After having cleaned the train dataset, I proceeded by creating new features which could be useful for price prediction.

Firstly, plotting the houses on a map (thanks to the information provided by the latitude and longitude features), it is possible to notice, after a quick visual inspection, that all of the houses in the train dataset are clustered around three main cities, namely Milan, Rome, and Venice. Therefore, I added the feature **'City'**, associating each entry with the respective urban area (See Fig. 1).

Afterwards, once again exploiting the position of the house, I calculated the distance in kilometers of each house from the respective city's center, using the haversine formula, and thus adding the feature **'distance_from_center'**, which, as predictable, is negatively correlated with the prices in the train data.

To further distinguish between neighborhoods in the different cities, I performed another clustering inside each of them, then adding a feature **'neighborhood'** representing the cluster of the house (See Fig. 2). I then stored the center of each cluster in an array, in order to use it to cluster the test dataset houses.

Finally, I tried to take advantage of the POI dataset, adding regressors such as the distance from the train station or the distance from university. In order to do so, I used a function which computed the distance of the house from each POI of the specific type, and then selected the distance from the nearest one, adding the information as a new variable. However, these features were not really correlated with price, while being usually correlated with another information, i.e., distance from center. Hence I decided not to add them to the dataset to perform the predictions, in order to avoid multicollinearity.

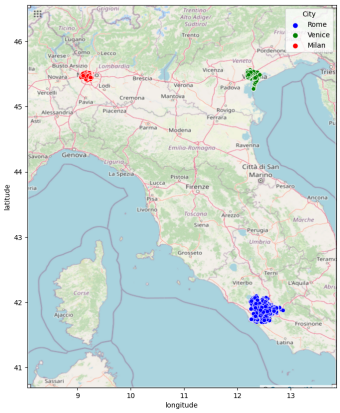

Figure 1: Train data urban areas

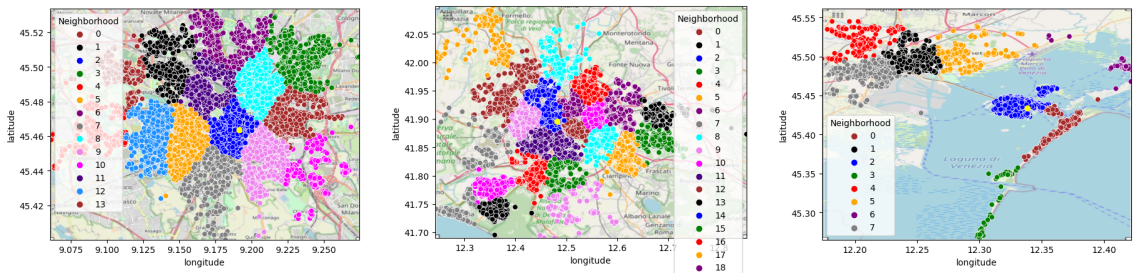

Figure 2: Cities' neighborhoods (from left to right: Milan, Rome, and Venice.)

3 Model Selection and Predictions

After that the train dataset was processed and the new features added, the following step was to find a suitable model to predict the prices of the houses in the 'test.csv' dataframe. In order to do so, I randomly divided the train data into a train and a cross validation set, so that I could compare and evaluate the performance of the different models and adjust the parameters.
In particular, I tested these different approaches:

1. Linear Regression
2. Random Forest
3. KNN Regression
4. Neural Network

I'm now going first to briefly discuss the discarded models, and then explain the final model I decided to use.

3.1 Discarded Models

Speaking about discarded models, the first one I tried was the most basic one for this kind of task, namely, **Linear Regression**, which however proved to perform poorly, probably because unable to capture the non-linear relations between price and the other variables. Then I made an attempt with **KNN Regression**, a non-parametric algorithm that makes predictions based on the average of the k-nearest neighbors of a data point, based on the assumption that similar houses should have similar prices; however also this model performed badly. Beyond these, I also tried to implement Neural Network with Keras and TensorFlow, but the model performed badly, and was also hugely expensive computationally speaking (especially for a computer without a great GPU), so I didn't even keep the code in the final notebook. The last model I discarded was a Random Forest regression differentiated per city,meaning that three models were trained, each on one city's observations. However, this model proved to generalize worse than the unique model trained on all of the observations (the one I used), as it probably had higher variance and tended to overfit.

3.2 Final Model: Random Forest

The final model I decided to use for prediction was Random Forest Regression, implemented through the scikit-learn package. This is an ensemble learning method that combines multiple decision trees to make predictions. Each tree in the forest independently predicts the target variable, and the final prediction is obtained by aggregating the predictions of all trees. This ensemble approach helps to reduce overfitting and to improve the generalization ability of the model.

This model proved to be the best performing among the others, mainly because it can capture complex relationships between the input features and the target variable, being able to handle both linear and non-linear relationships, as well as interactions and feature dependencies. This is important in predicting house prices since the relationship between various features (e.g., location, surface, number of rooms, floor, etc.) and the price can be complex and non-linear, reason why simpler models such as linear regression perform poorly.

After a first trial using all of the features in the model, I exploited the fact that Random Forest provides a measure of feature importance, ranking the features based on their contribution to the overall predictive power of the model, and hence I discarded the less important features, 'City' and 'proximity_to_center', to obtain the final model. The feature City was probably not particularly meaningful, being the three urban areas comparable from a prices' point of view, but was useful to calculate the distance from center of the houses; this latter variable is precisely the reason why 'proximity_to_center' turned out to be useless, as they are strongly correlated.
Removing these two features led to a slight improvement in the model's performance on cross validation set.

Finally, I trained the Random Forest on the entire train dataframe, in order to take advantage of all of the information at my disposal. The model was now ready for the test data.

Table 1: Models performances on Cross Validation set

Model	MSE	R^2
Linear Regression	6.2933e+11	0.257
KNN Regression	5.34e+11	0.368
Random Forest Regression	4.7744e+11	0.436
Random Forest Regression per City	6.0942e+11	0.395
Random Forest Regression (after feature selection)	**4.7054e+11**	**0.446**

3.3 Test data processing and predictions

Before actually computing the predictions, I had to perform some operations on the test data.

Firstly, I had to deal with the missing values, for which I used the same approach as for the train data. Notice that the imputing on the test missing data was computed using the train data statistics and models. For instance, instead to replace the NaNs of 'n_rooms' with the median of test dataframe, the median of train dataframe was used. The same can be said about the linear regression model used for the surface imputation and the KNNImputer used for floor and construction year, both trained on the train dataframe.
This was done this way so that the imputed values could be more consistent with the model, preserving the patterns and relationships observed during training, making the test dataset more realistic and representative.

After the missing data, the following step was to add the new features, i.e., 'City', 'distance_from_center', and 'neighborhoods', also to the test records, after having checked that the test data locations were consistent with the train data ones. Here I used the previously saved array of centers of clusters to find the nearest one to each house, thus assigning the neighborhood to which they belonged.

Both these processes where encoded in two functions ('preprocessing' and 'city_and_distance') in order to automate the process and avoid to run many cells.

To conclude, the Random Forest model was applied to the test data to obtain the prices' predictions, which were then joined in a dataframe with the respective house id.

4 Conclusions

The model implemented is by no means an incredibly reliable and highly accurate method for predicting the price of a house, and I wouldn't recommend to use it for investment decisions. Indeed, there are infinitely many factors which determine the price of an house, such as the neighborhood's criminal rate, the presence of green spaces, the rent prices, the tourist attractions nearby, and many other variables which would have been interesting to add to the model. However, the model developed seems to be able to give an overall good prediction of an house's price given the information provided.