Predicting House Prices in Italian Cities

Francesco Iaccarino
3170051
Kaggle Username: Francesco Iaccarino

May 9, 2023

1 Introduction

In this project, the goal was to predict house prices in different Italian cities using two datasets. The first dataset, train.csv, contains data for one apartment for sale per row, and the target variable is the sale price. The second dataset, poi.csv, contains the coordinates of points of interest that we can use to further enrich the features available in our training dataset.
In this report, I will first describe the data cleaning and preparation process, followed by the feature engineering process. I will then explain the machine learning algorithms we used to make predictions and evaluate the final model's performance.

2 Explanatory analysis

To gain a comprehensive understanding of the dataset, the first step involves performing exploratory analysis. In my notebook, I utilized several methods such as head, info, describe and isna().sum() to investigate the nature of the features and evaluate the magnitude of missing values in the dataframe.

3 Data Cleaning and Preparation

Before beginning the analysis, I needed to clean and prepare the data through the definition of several functions.

- The first function, traintestdrop(df), was designed after a qualitative analysis of the null values in the dataframe, and I used it to perform, mainly, two operations:
  - Drop columns with missing values greater than 60% (garden - with a percentage of missing values of 68.1%), because a high percentage of missing values may not provide useful information to the model, or it may introduce noise or bias.
  - Drop the id column, because of the fact that unique identifiers don't provide any useful information for the model's prediction task.Including the id column in the model, in fact, would not only make the model more complex, but could also lead to overfitting, as the model may learn to recognize and rely on the unique identifiers rather than the actual features that are relevant for the prediction task.
- The second function, linear regression(df, target, predictor1, predictor2 = None, predictor3 = None), was used to fill missing values in the target variable using linear regression with one, two, or three predictor variables. Linear regression is an effective method for predicting a variable based on one or more predictor variables. In this project, I used linear regression to impute the missing values of n bathrooms and total floors.
  - As regarding n bathrooms I used surface and n rooms as predictor variables.
  - As regarding total floors I used Floor variable as predictor since the number of floors in a house could be a useful indicator of the floor where the house is located.

Moreover, these variables were chosen based on their significant correlation with the missing variables, providing a reliable basis for imputation.

- The third function, traintest subs(df), was used to fill missing values in the remaining columns with:
  - The median for latitude, longitude, expenses, floor, energy efficiency, n rooms, surface. This statistical indicator was chosen because of the fact that it is a measure of centrality that is less sensitive to extreme values or outliers, on the contrary of the mean, and because it preserves the distribution of the observations.
  - The mode for construction year, proximity to center, balcony, elevator. As regarding construction year, I thought that usually large numbers of buildings are set up in the same period (think of the period subsequent to the second world war), and so the most frequent value could have been a good measure for the imputing. As regarding the other variables, on the other and, the mode was chosen because usually the imputing on categorical is done via mode and a variable as proximity to center can be considered as a categorical variable, (1 if it is in the center and a values less than 1 if it is not in the center), while balcony and elevator are indeed categorical variable.

  I choosed to use median mostly for variables with skewed distribution, mainly numerical and mode for categorical variable
- The function traintestdum(df) was used to convert the "conditions" column into a set of binary columns using one-hot encoding.
- Finally I defined a function called traintestStandardize(df) to standardize three columns: energy efficiency, expenses and surface. In the end, I decided not to use it, because the random forest, which later on I will explain to be the best model among the one I used, is generally considered to be robust to differences in scale among the features.

4 Feature Engineering and Outliers Handling

The feature engineering is an essential step that transforms raw data into features that represent useful information inferred from the given ones to the predictive model, resulting in an improved model accuracy on unseen data. Since not all features in a dataset are equally important for predicting the target variable, by selecting the most relevant features and creating new features that capture additional information, we can reduce the dimensionality of the dataset and improve the model's performance.
Therefore, I enriched the dataset with new features such as:

- Comfortability: is a feature that analyzes each flat on a case-by-case basis and evaluates the presence of an elevator along with the floor at which the flat is located. If a flat is located above the third floor and there is no elevator, a value of 0 is assigned, otherwise, a value of 1 is assigned.
- Bathrooms per room: is a feature that simply analyses how many bathrooms there are in an house with respect to the number of rooms.
- Desirability: Through the visualization of a scatterplot I identified three different areas in which the houses are located, and after a brief analysis I identified three cities: Rome, Milan and Venice. Therefore I decided to join to the dataframe another feature that takes into account the latitude along with the longitude and divides the dataframe into three cities using KMeans algorithm. The feature "desirability" represents basically an ordinal encoding consequent to the joining of the two variables "latitude" and "longitude".
- Milan Rome and Venice: are three columns representing a one-hot encoding of the flat's position: in Rome, Venice and Milan. Later on, I will explain how these features turned out to be useful.

Handling outliers, moreover, is an important step to take in data processing because they have a significant impact on the efficiency of the model. If outliers are not handled properly, they can lead to models that are less accurate and less generalizable to new data.

Furthermore, I decided to strike a balance between removing outliers and retaining as much data as possible, therefore I decided to remove only the two highest and the three lowest prices; the highest total floor value which resulted to be clearly an error of measurement: 31960.0; the highest construction year which again resulted to be clearly an error of measurement: 2500; the highest energy efficiency value: 3727500.0 and the observations, in which the surface resulted to be equal to 0.

5 Model running

As concerning the Machine learning models, I tried to predict house prices through several machine learning models, including linear regression, K-NN and random forests, but in the end I preferred to mainain in the notebook only the random forest which gave me the most satisfactory results in terms of performance.
Additionally, I opted to experiment with two distinct prediction approaches. The first approach consists of applying the model to the dataframe without any other major adjustment. For the second approach, on the other hand, I splitted the dataset into different parts based on the location of the houses.

- The first "model running" was based on a dataframe modelled alongside the modifications I described previously, because of the reasons I provided above.

- In the second "model running" I splitted the dataset through a mask, exploiting the one hot encoding performed on the basis of the city the house was located. Then I trained three different Random Forest models, which to me could be a useful approach to improve the prediction accuracy. This is because different cities may have different housing market trends and characteristics, such as average prices, and demand, which can affect the relationship between the features and the target variable. By training separate models for each city, I think that each model can better capture these differences and adjust the model accordingly. After the training of the models, I predicted the house prices city by city, and afterwards I have concatenated the predictions from the three models, in order to obtain a more accurate overall prediction for the entire dataset. This is because each model has learned from a different subset of the data and can provide unique insights into the prediction.

  Although this approach has solid assumptions onto which to lay, the resulting mean squared error (MSE) was not satisfactory. Despite the final output is not driven by it, in the end, I chose not to exclude it from the final report.

As I was saying before then, I decided to use Random Forest because it is able to handle a large number of features and interactions between features without overfitting. Random forest, indeed, uses a combination of many decision trees, each trained on a random subset of the features and observations, and then aggregate the predictions of these trees to make a final prediction.
Additionally, random forests are often more robust to outliers and missing values than other models, as they are based on a non-parametric approach that is less sensitive to the specific distribution of the data. This can be particularly advantageous in cases where the data is noisy or incomplete.
In order to make the predictions more accurate, furthermore, I decided to use Grid search which is a technique used to find the best hyperparameters for a machine learning model. Hyperparameters are values that are set before training the model, that affect the model's performance. In fact, the Grid search involves defining a grid of hyperparameters to test, training and evaluating the model with each combination of hyperparameters in the grid, and maintaining in the end the combination of hyperparameters that produces the best performance. Personally in the notebook I have used a form of the grid search that inspects only the maximum depth, in order not to make it too expensive in terms of computational cost, but there is also a code that examines other hyperparameters as n estimators, minimum samples split or minimum samples leaf.
Lastly, I tested also the performance of Random Forest through the Principal Component Analysis (PCA). PCA turns out to be useful since it performs a dimensionality reduction of the dataframe. It transforms the original variables into linear combinations of the original variables, selected in such a way that they explain the maximum amount of variation in the original dataset.
After several performance analysis of the models running the Random Forest using the Grid Search resulted to be the most effective technique to use, although very expensive in terms of computational cost.

6 Results and Discussion

In order to further analyse the performance of the model designed I decided also to experiment Cross Validation, which involves dividing the dataframe into a number of subsets, and then trains the model on a subset of the data and evaluates it on the remaining subset. This process is repeated multiple times and each fold is used as the validation set just once.

By using cross-validation, I found out that the Random Forest has

- Validation score: 0.2465251965129277
- Cross-validation scores: [0.25677665 0.1750791 0.21743327 0.22202477 0.26188031]
- Training score: 0.2465251965129277

Which suggest us that the model has a moderate level of accuracy in predicting the target variable on the validation set. The cross-validation scores provide a more detailed picture of the model's performance across different folds of the data. The reported scores are [0.25677665 0.1750791 0.21743327 0.22202477 0.26188031], which suggest that the model's performance varies across the different folds.
Random forest with cross validation instead has

- Validation score: 0.6734328184080626
- Cross-validation scores: [0.3981828 0.33706244 0.32249424 0.36334966 0.34463595]
- Training score: 0.6734328184080626

Which suggest us that the hyperparameter tuning process let the model having a higher level of accuracy in predicting the target variable on the validation set with respect to the previous case. As before, the reported scores are [0.3981828 0.33706244 0.32249424 0.36334966 0.34463595], and again we can affirm that the model's performance varies significantly across different folds.
The model I designed, in the end has scored a mean squared error of 962584856753.754 on the test data. However I acknowledge that there is still room for improvement.

7 Conclusion

Based on the analysis and experiments conducted in this project, we can conclude that the prediction of house prices can be improved through careful data preprocessing, feature engineering, and the use of appropriate machine learning algorithms. The feature engineering techniques used, and handling of outliers, proved to be effective in improving the performance of the models.
Furthermore, the use of statistical techniques, such as pca or hyperparameter tuning technique Grid Search, helped to identify the most suitable method for this particular dataset. The splitting of the dataset by city and training three separate models also didn't show promise in improving the accuracy of the predictions.
In conclusion, the process of predicting house prices is complex and requires careful consideration of various factors.