Predicting housing prices

Group 17

Date: 15.11.2023

# **DESCRIBE THE PROBLEM**

## **SCOPE**

The goal of this project is to predict the housing prices by allowing the users to input certain facilities they are looking for in a house. The overarching goal is to give the users insight into how much they should expect to pay for their “dream-home”.

As there are multiple other competitors which promises the same solution, the goal of this project is to provide a “free use” utility service for the users. This will, admittedly, reduce the potential stakeholders; however, the project may still be of interest to various actors in the real-estate sphere, such as building developers. Since the primary goal is to provide a free utility, our main business goal is to earn enough for maintenance work, server costs, and hosting.

## **METRICS**

The minimal business metric for the project to be considered a success, is to maintain a user-base large enough to cover the cost of business. When it comes to the performance of the technical solution, the project’s success is based on the root mean square error to measure the performance of the machine learning model. The root mean square error is a loss-function which indicates the difference between the actual value and the predicted values. Therefore, it gives a good indication of whether the machine learning model gives an acceptable representation of reality when presented with new data.

# **DATA**

The data used in this project is tabulated data which describes the different relevant attributes one could consider when purchasing a new house. From the size of the house to the number of bathrooms. The dataset used will be gathered from the Kaggle competition <https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview>. One of the most important considerations when it comes to under-, and overfitting is to have a comprehensive dataset. The complexity of the model as well as the size of the dataset are the most important factors when it comes to getting an “accurate” prediction, or in other words, to prevent over-, and underfitting.

When it comes to the cleaning of the data, we must convert most of the features using some kind of encoding, as most of the features are objects. In this project we have elected to perform encoding using pandas inbuilt feature, that is pd.get\_dummies. Additionally, we must identify and handle null values. There are multiple ways to handle null values, such as finding the mean values, dropping the feature, or dropping the specific rows that contains the null value. We have elected to drop the column its data is comprised of more than 80% null values. As mentioned in the notebook, this does not always result in a good performance, as the absence of values may be informative in and of itself. However, we noticed an improvement in the result when doing so in this case.

Lastly, we must consider outliers, as they may dominate the other values, and thus lead to an increased performance regarding the training data, and still have a decreased performance on unseen data. To identify the outliers, we have plotted the various numerical features, identifying the features which have the largest discrepancies, and then remove the values that are outside a given quantile, in our case the 98 percentiles.

We also considered removing features with a negative correlation, however, as the adage goes, correlation does not imply causation. An experience which we shared for this project, and therefore we elected not to keep the features.

The primary privacy issue of this project is that the data is provided by an external actor, and we have no real insight into whether the data is responsibly sourced.

# **MODELING**

As we are interested in predicting a concrete value, regression is the natural consideration. There are multiple potential models to consider when performing regression, and our goal is to identify the model which reduces the loss function. We began the process by electing certain models which handles this problem differently. Regular regression using linear regression, gradient boosting by using GradientBoostingRegressor, and lastly ridge regression using ElasticNet.

After running the base models, without changing the hyperparameters, we found that the elasticnet model performed significantly better than the other alternatives. After performing hyperparameters optimization by using gird-search, we found that the “optimal” parameters for elastic net, in our case, are {'alpha': 0.1, 'l1\_ratio': 0.8, 'max\_iter': 100}, which improved the performance of the model. This can also be observed when plotting the regression graphs. In order to investigate the prediction mistakes, we plotted the regression plot of the predicted values vs. the actual values for each of the models, and for each of the models we found that the regression-line was fairly well placed, however our model does not do a great job of catching outliers.

# **DEPLOYMENT**

Regarding deployment, we have elected to allow the users to modify certain parameters, which are decided by the highest correlation. Admittedly, this is not a perfect solution, however, allowing the users to modify every single parameter, won’t necessarily lead to a better user experience, as the users may misinterpret the purpose of the parameters, or it may just be unclear what their purpose is.

We also want to avoid affecting the actual model’s performance; therefore, we will only predict on the users’ input, and not train the model on it.