1. **Introduction**
2. **First assignment**

This project aims to predict house sales prices using a Belgian real estate dataset. Since the test set lacks target values, a custom validation split was implemented to ensure reliable performance assessment. The process began with exploratory data analysis to identify missing values, outliers, relationships between features, etc. Preprocessing steps included dropping unnecessary columns, imputing missing data, and encoding categorical variables, and applying transformations where necessary to improve quality. A predictive model was then developed and evaluated using an 80-20 validation split, with the Winkler Score (α = 0.20) as the primary metric and Mean Absolute Error (MAE) as a secondary measure. Hyperparameter tuning was performed to enhance model performance. Finally, results were analysed and compared across leaderboard scores to identify potential discrepancies because of factors such as underfitting.

The objective was to minimize prediction error while ensuring model robustness. While common evaluation metrics like Root Mean Squared Error (RMSE) measure overall accuracy, this project also emphasizes the Winkler Score, which assesses the quality of prediction intervals. Instead of predicting a single value, this metric evaluates the accuracy of a predicted range, with the goal of maintaining a narrow interval that captures the true value.

**2.1 Data exploration**

To begin with, our code includes a combination of essential libraries which all play a role in the overall modeling process. This code imports the necessary libraries for data analysis, visualization, and machine learning such as Pandas, Matplotlib, or Seaborn.

We then continued with simple exploration and analysis of the data. We checked the quality of the data, missing values, distribution, outliers, correlation, and we finalized all by adding official data we found on Statbel to further pursue our data analysis. The extra data we found was related to median house prices for provinces.

*1. Analysis of missing values and other values*

In the initial stages of the data exploration process, certain columns, including 'id', 'sticker', 'price\_drop\_date', and 'added\_time', were identified as irrelevant and are dropped in the preprocessing phase. A critical step in the data exploration was identifying missing data which will be also dealed with later on. Our approach will be to maintain as much data as we can, and remove potential biases.

The dataset was divided into numeric and categorical columns to better understand the structure of the data and identify potential relationships. The numeric columns 'area', 'lat', 'lon', 'energy\_value', 'bedrooms', 'foto\_amount', 'price', 'dist\_to\_highway', 'dist\_to\_nearest\_city', and 'price\_diff' were recognized as continuous variables. Meanwhile, the categorical variables, including 'is\_appartment', 'new\_building', 'advertiser', 'subtype', 'energy\_label', 'Province', and 'postcode', were singled out for further exploration as they require handling before they can be included in machine learning models.

*3. Outliers and distributions*

A diagram of a box plot

AI-generated content may be incorrect.A graph of distribution of area

AI-generated content may be incorrect.A graph of distribution of area

AI-generated content may be incorrect.

Visual exploration (boxplot) showed an outlier for ‘area’. Since 20.000 square meters could indicate that this is not a residential property, we argued to just delete this observation. This made the initial distribution on the histogram highly skewed to the right. Deleting this value makes the distribution less skewed to the right.

We further detected 2 extremely high values for ‘bedrooms’ which are corresponding to a relatively small ‘area’ of the property (i.e. 3.21 and 5.5). Thus, we delete these two rows. Even though the other extreme values are also really high, they correspond to large property areas, which make sense and is the reason why we keep them. Additionally, the largest energy\_values are removed since they are almost 5 times as high as the others. For the other outliers we decided to not address them.

4. *Correlation*

A diagram of a graph

AI-generated content may be incorrect.

The correlation matrix confirms some expected relationships, the bigger the properties, the more bedrooms there are, the higher the price. As some features present a strong correlation between them, this could be a sign of multicollinearity. Nothing will be implemented to handle this problem. A solution could have been to suppress some features or regroup strongly correlated features together, but we did not want to suppress important variables nor did we think regrouping ‘bedrooms’ and ‘price’ made sense.

*5. Additional data analysis with extra data from the government*

We found extra data on median house prices for provinces from Statbel which during our exploration thought could reduce the Winkler score and make the whole model better. This data can be found on the following link: <https://bestat.statbel.fgov.be/bestat/crosstable.xhtml>.

**2.2 Preprocessing**

Before discussing the preproces, we want to comment that we used the initial training dataset only for training and the initial test set for testing and we think that it could also have been possible to “mix” the datasets and separate the training/test sets afterwards.

Before applying any model, we preprocessed our data. To optimize model performance, as mentioned in our data exploration, unnecessary columns were removed. Next, missing values are handled by replacing numerical variables with their median values to avoid skewing the data. Then, the numeric columns were imputed using the K-Nearest Neighbors imputation method. Categorical variables like ’advertiser’ and ’energylabel’ are replaced with ”Unknown” to preserve missing category information rather than dropping them entirely. Since machine learning models require numerical input, categorical variables are encoded using two approaches. Label encoding is applied to ’advertiser’ and ’subtype’, where each category is assigned a unique numeric code. Additionally, one-hot encoding is used for low-cardinality categorical variables such as ’energylabel’ and ’province’, creating separate binary columns for each category while dropping one as a reference to prevent multicollinearity.

The following part consists of creating new features that measure how far each property is from the center of major cities. We did this because the distance to the city center is an important factor in property prices. We simply calculate the raw distance from each property to the nearest city center using the Haversine formula. The code computes the distance in kilometers for each property to the city centers of Brussels, Antwerp, Charleroi, Liège, Gent, Brugge, and Leuven. A new column is added to the dataset for the minimum distance to any city center (in kilometers), named dist\_to\_nearest\_city. This calculation is done for both the training and test datasets. Additionally, for each city, where the log distance is computed only for properties within a 15 km threshold from the city center; properties outside this threshold are assigned a missing value (NaN). These log-transformed distance features are added to both the training and test datasets for further analysis.

A graph of a house distribution

AI-generated content may be incorrect.

We do the same for house prices. To make sure the house price data is easier for the model to understand, we apply a log transformation. House prices usually have a few very expensive properties that can mess up predictions, so taking the logarithm of the prices helps smooth things out. This preprocessing ensures that the dataset is free from missing values and that all categorical data is appropriately encoded, making it ready for use in machine learning models.

The following process involves integrating spatial data on major highways into the property dataset to create a new feature: the distance from each property to the nearest major highway. The data on highways was sourced from OpenStreetMap and filtered to include only major highways. To ensure accurate calculations, both the highways and property data were reprojected into the same coordinate reference system (EPSG:3857), which uses metric units. Next, a spatial join was performed to compute the nearest highway for each property, and the calculated distance between each property and the nearest highway was added as a new column, "dist\_to\_highway," to the dataset.

Next, we integrated external data on the median price of houses across Belgian provinces into our primary dataset. The external data, loaded from a CSV file, was filtered to retain only entries from 2017. We cleaned the dataset by renaming dutch words in english, removing unnecessary columns, and converting non-numeric values to numeric values. First, we made sure the **province names** in both datasets looked the same. For example, we removed any extra spaces and made sure the column was named the same in both files. Then, we combined the two datasets by matching the province names. We used a method that keeps all the rows from our main dataset, and just adds extra information from the second dataset if the provinces match. The extra information we added was the median house price for each province. So now, each house in our main dataset also shows the average price in its province. Now the data is clean and complete, and we can use it for more analysis or modeling.

A graph of a number of colored squares

AI-generated content may be incorrect.

As we can see in this correlation matrix, the new variables distance to highway and distance to nearest city have a low correlation with the other variables. After that, we split the data into two parts: one for training the model and another for testing how well it performs. We randomly separate 80 percent of the data for training and keep 20 percent for validation. The model learns from the training data, and then we test it on the validation data to see if it makes good predictions. We store all the features (except for price) in X, and the log-transformed prices in Y. This gives us four datasets: 2 for training, and 2 for testing. This way, we can check if the model is learning properly before making final predictions.

**2.3 Model selection**

To build a predictive model for house prices, we use XGBoost, a powerful machine learning algorithm known for handling structured data efficiently. We start by initializing an XGBRegressor. The model is then trained on the Xtrain dataset with corresponding Ytrain values, allowing it to learn patterns in the data. Once trained, the model predicts the house prices for the validation set. However, since we originally applied a log transformation to the price variable, we reverses this transformation to get actual price predictions. To evaluate model performance, we calculate the MAE, a metric that measures the average difference between the predicted and actual house prices. Lower MAE values indicate better model accuracy. The computed MAE before hyperparameter optimization is 1,272.65.

To improve the performance of our XGBoost model, we conduct hyperparameter optimization, which involves testing different combinations of model settings to find the best configuration. This process takes around 15 minutes due to the number of combinations tested. We first define a grid of parameters to tune. We then initialize an ‘XGBRegressor‘ with a fixed random state to ensure reproducibility. We test all possible parameter combinations while optimizing for the lowest MAE. The search runs with 3-fold cross-validation, meaning the dataset is split into three parts to validate performance more reliably. After completing the grid search, the best parameter combination is extracted and used to train a final optimized model. This model is then used to predict house prices for the validation set. Finally, we compute the MAE for the optimized model, which provides a direct comparison to the pre-optimization MAE, allowing us to measure the improvement gained from tuning the parameters. The XGBoost model used for predicting house prices is a supervised learning algorithm, as it is trained on labeled data where both the input features and corresponding house prices are known. It aims to learn the relationship between these inputs and outputs in order to make accurate predictions on new data. XGBoost is a discriminative model, meaning it focuses on modeling the conditional relationship P(Y∣X)P(Y|X) rather than the joint distribution P(X,Y)P(X, Y), as generative models do. Additionally, it is considered a non-parametric model because it does not assume a fixed functional form or a set number of parameters. Instead, it builds an ensemble of decision trees whose complexity can grow with the amount and structure of the data, allowing for greater flexibility in capturing complex patterns.

To evaluate the performance of our optimized XGBoost model, we visualize how well the predicted house prices match the actual prices. This helps us understand if our model is making accurate predictions. First, we create two plots side by side. The first plot compares the actual house prices (on the x-axis) with the predicted prices (on the y-axis). Each point represents a house, and ideally, these points should align closely with the line, which represents perfect predictions (i.e., predicted price = actual price). The second plot is a residuals plot, which shows the difference between actual and predicted prices (residuals). A performing model should have residuals evenly spread around zero, meaning the model doesn’t consistently overestimate or underestimate prices.

In addition to visualizing predictions, we also analyze which features were most important in making these predictions. We extract the feature importance values from our trained XGBoost model and create a DataFrame to organize them. The features are sorted by importance and displayed in a bar chart. Features with higher importance values contributed more significantly to the model’s decisions, while less important features had a smaller impact. By combining these visualizations, we can assess both the accuracy of our model and the key drivers behind house price predictions.

**2.4 Results and interpretation**

A diagram of a distribution of actual prices

AI-generated content may be incorrect.A diagram of a distribution of the average temperature

AI-generated content may be incorrect.

With the test data properly formatted, we make predictions using our optimized XGBoost model. To provide a measure of uncertainty for our predictions, we calculate an 80 percent confidence interval. The width of the interval is determined using the standard deviation of residuals (the difference between actual and predicted values in the validation set). We approximate the interval width using a z- score of 1.28, which corresponds to an 80 percent confidence level. We then ensure that the lower bound is not negative (prices cannot be negative), by setting negative values to zero.

Finally, we calculate the Winkler Score, a measure used to assess the accuracy and reliability of the predicted house prices by incorporating both the prediction intervals and the penalties for miscalculations. The goal of the Winkler Score is to provide a balance between the prediction’s closeness to the true value and the width of the confidence intervals. To calculate the Winkler penalty, we assess the amount by which the true prices fall outside the predicted interval. If the actual value is below the lower bound or above the upper bound, a penalty is added. The penalty is scaled by a factor of 2/0.20, which amplifies the penalty for larger errors. Finally, the Winkler Score is computed as the average of the sum of the interval width (difference between upper and lower bounds) and the penalty. The final Winkler Score is then predicted, which reflects how well the model performed in predictions with reliable confidence intervals. The Score and Secondary Score listed are specific numerical values resulting from the Winkler Score calculation, indicating the model’s performance according to the given metric.

* Show the scores
* Actually interpret results
* Model evaluation?