Introduction to Artificial Intelligence

Project: Predicting AirBnB Prices

Mikołaj Jagielski 303860

Michał Łezka 303873

1. Introduction

The aim of this project is to predict Airbnb prices using machine learning algorithms: Linear Regression and Random Forest. Linear Regression algorithm is a simple yet effective algorithm that predicts the price of a new listing by finding relationship between Airbnb features and other independent variables and the price, which is dependent variable in this case. On the other hand, Random Forest algorithm builds a set of tree-like models of decisions where outcomes of each tree are based on the most relevant features for predicting prices, then those outcomes are aggregated into a single result. We applied these algorithms to a dataset of Airbnb listings of offers from many European cities with various features such as number of bedrooms, room type, cleanliness, and so on. After pre-processing the data, we trained and tested the models to evaluate their performance using appropriate metrics and techniques.

2. Algorithms

- The Linear Regression algorithm is a very simple regression algorithm that finds corelation between independent and dependent variables. The algorithm will attempt to find the best-fit line that minimizes the difference between the predicted prices and the actual prices in the training data. The line's equation represents the relationship between the features and the predicted price. Its biggest issue, in our context, is its weakness in working with nonnumerical data and assumption that a linear relationship between the features and the target variable, which may not always hold true in real-world scenarios.
- Gradient boosting is an iterative machine learning algorithm that decision trees, to create a strong predictive model. It works by training each new model to correct the errors made by the previous models. Its strength in comparison to other algorithms is high accuracy: and handles different types of data: It can handle a mixture of categorical and numerical features without requiring extensive preprocessing. However, it’s also very demanding and we have to use it as we may easily overfit. What is more, the performance of gradient boosting heavily relies on the proper tuning of hyperparameters, which can be a challenging task. Overall, gradient boosting is a powerful algorithm that excels in accuracy and versatility, but it requires careful parameter tuning and consideration of computational resources.
- Random Forest is an ensemble machine learning algorithm that combines multiple decision trees to improve the accuracy and generalizability of the predictions. The algorithm works as follows:

1. Randomly select a subset of the training data.
2. Randomly select a subset of the input features.
3. Build a decision tree using the selected data and features.
4. Repeat steps 1-3 to create a set number of decision trees.
5. When making a prediction for a new data point, pass it through each decision tree in the forest and take the average or majority vote of the predictions as the final prediction.

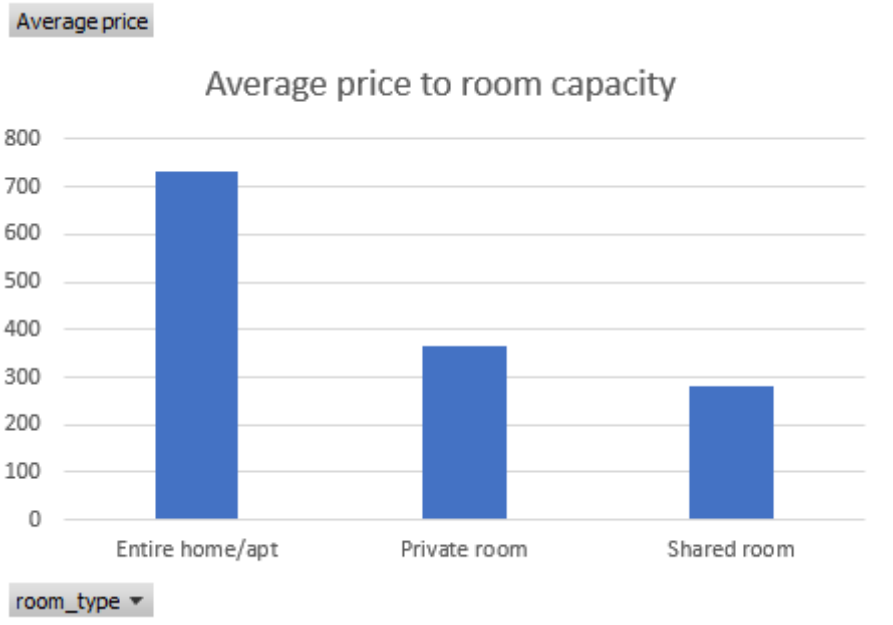
The Decision Tree algorithm is an algorithm that builds a tree-like model of decisions and their possible consequences. Each node in the tree represents a decision based on one of the features, and each branch represents one of the possible outcomes of the decision. In the context of predicting Airbnb prices, the algorithm can be used to build a decision tree based on the features that are most relevant for predicting prices. The algorithm then uses the decision tree to predict the price of a new listing by traversing the tree based on the features of the listing and the decisions made at each node.

As all mentioned algorithms have their own strengths and weaknesses, we consider this pair very good to implement and compare.

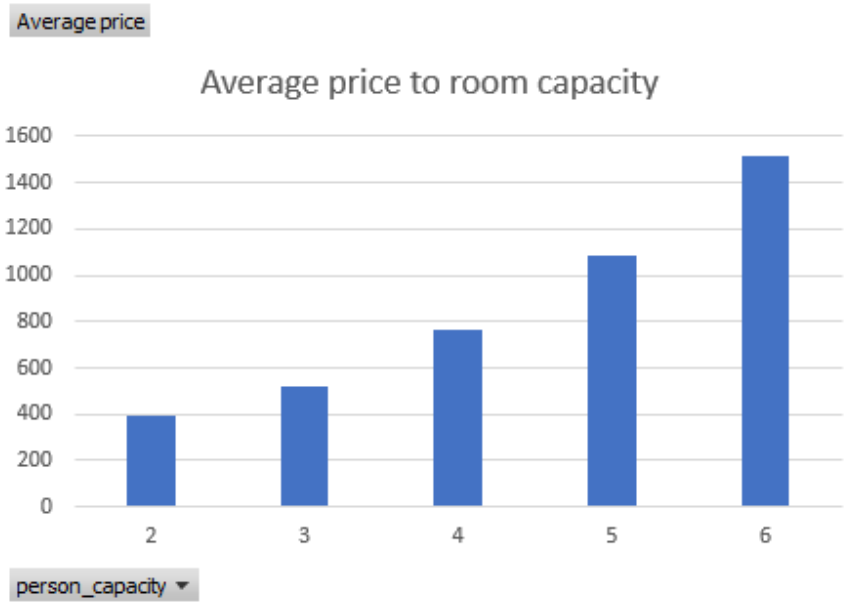
3. Data

We have noticed a few (and some rather obvious) tendencies. Some of the more obvious would be:

- The more privacy/space, the more expensive rental is

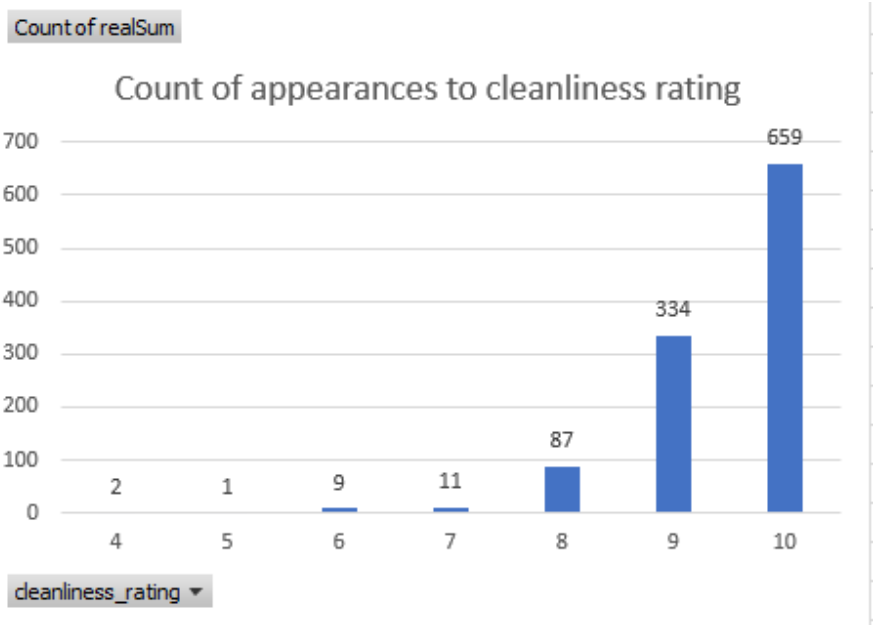


- The higher capacity, the more expensive



- Cleanliness ratings are however more interesting:

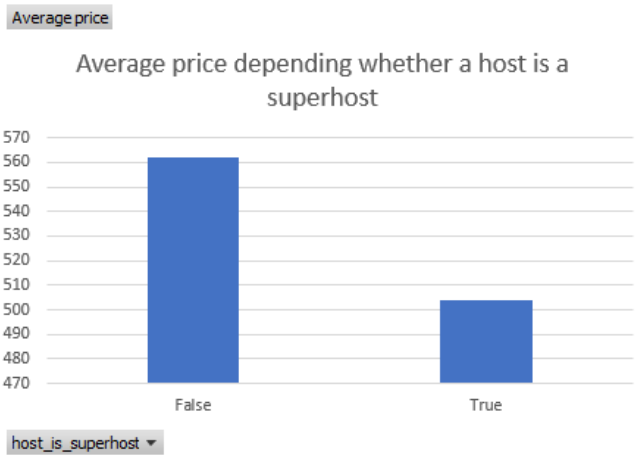
As we can see, there are some outliers, most importantly 4-star reviews having higher prices on average than 5-star reviews. When we analyse this phenomenon, we can see that it is caused by 4 and 5-star cleanliness reviews are hugely underrepresented:



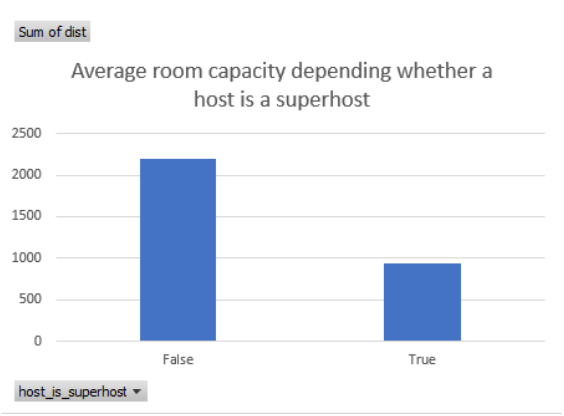
As we can see, we have a major problem in the dataset: imbalance caused by lack of samples with specific properties, like mentioned above cleanliness ratings, if host a super host, room capacities, and a huge dominance of non-business trips. Another example of such imbalance would be the number of superhosts in Amsterdam compared to regular hosts:

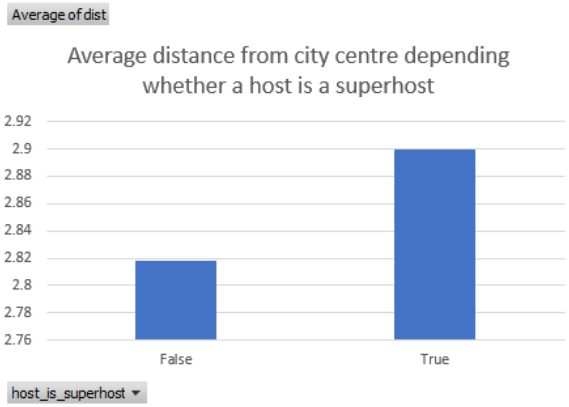
host_is_superhost	Count of Rows
False	780
True	323

Which causes some rather unexpected results:



After a deeper analysis we have observed that superhosts tend to rent smaller apartments that are further away from the city centre:





We plan to address this problem using oversampling, that is - generating synthetic samples for the minority class to balance the dataset, specifically called SMOTE (Synthetic Minority Over-sampling Technique). We can randomly generate new samples for the missing listings by interpolating between existing samples or generating new samples in regions where the missing listings are underrepresented. This can balance the dataset and prevent the model from being biased towards the majority class. We decided to split the data in chunks for each city and database to see how specific set of features, which vary in different cities, affects the results. It is also very important to walk-through the whole data and eliminate invalid data like cleanliness rating above 10 etc.

- Chosen parameters

In our opinion the combination of the features "real sum," "room type," "room shared," "room private," and "person capacity" might be sufficient for the model to correctly predict Airbnb prices. Real Sum, which is the price of the listing, is crucial feature to start with. Room type: The distinction between a room and a full apartment is important as it affects the pricing structure. Full apartments tend to have higher prices compared to individual rooms, so by considering the room type, the model can differentiate between these two categories and incorporate their pricing dynamics. The room shared and room private classification of rooms as shared or private helps capture the impact of privacy and exclusivity on pricing. Shared rooms typically have lower prices due to the shared nature and limited privacy, while private rooms are more likely to command higher prices due to increased exclusivity and personal space. The capacity of the accommodation, indicating the maximum number of people it can accommodate, is an important factor in pricing. We decided that the rest is not so important and what we have chosen is enough.

- Preprocessing

After analysing data, we have noticed that many features actually do not influence the results in any meaningful way or affect it only slightly. We have noticed a strong dependence on room type and person capacity while other parameters didn’t really affect it. There was an interesting dependency between host being superhost or not because:

Row Labels	Average of realSum
False	369.790184
True	337.333254
Grand Total	**364.3897466**

As we can see, a host being superhost actually *lowers* the cost of apartment, which is not connected to the rooms being rented by a superhost (meaning the host is highly rated, provides outstanding hospitality etc.) but to the fact that superhosts tend to rent cheaper apartments. Therefore, we have decided to definitely not include parameter to our algorithm.

Some parameters only slightly affected the price: distance from centre, attractiveness index and restaurant index, however adding them as a parameter to our algorithm actually lowered the performance so we have decided to not use them. We also thought of using SMOTER (Synthetic Minority Over-Sampling Technique for Regression) to combat unbalance in the set:

Row Labels	Count of realSum
Entire home/apt	2418
Private room	2934
Shared room	27
Grand Total	**5379**

However we have come up with worse results compared to not using SMOTE, so we decided to use the dataset as it is.

4. Training Details

- Hardware used:

CPU: AMD Ryzen 5 3600 6-Core Processor
RAM: 16 GB
Graphic card: NVIDIA GeForce RTX 2060

- Hyperparameters

For Random Forest:

MAX_DEPTH = None: Setting the maximum depth of the decision trees to "None" allows the trees to grow until all the leaves are pure or until they reach the minimum number of samples required for a leaf node. This flexibility enables the trees to capture complex patterns in the data.

MIN_IMPURITY_DECREASE = 0.05: This parameter specifies the minimum amount of impurity decrease required to split a node during tree construction. A value of 0.05 means that a split will only be performed if it results in a decrease in impurity greater than or equal to 0.05.

Gradient Boosting Parameters:

N_ESTIMATORS_GRADIENT = 100: This parameter determines the number of boosting stages or weak models (decision trees) to be combined in the strong model. A higher

number of estimators can improve the model's performance, but we didn’t notice improvement in higher values, so we decided to leave it this way to reduce resource usage.

MAX_DEPTH_GRADIENT = 10: Setting the maximum depth of the weak models (decision trees) to 10 helps control the complexity and overfitting of each individual tree. Restricting the depth prevents the trees from becoming too specialized to the training data, promoting generalization on unseen data.

MIN_SAMPLES_SPLIT_GRADIENT = 10: This parameter determines the minimum number of samples required to split an internal node during the construction of weak models. By setting it to 10, it ensures that nodes with fewer samples are not split, preventing overfitting and maintaining a reasonable level of generality.

LEARNING_RATE_GRADIENT = 0.1: The learning rate controls the contribution of each weak model to the ensemble. A lower learning rate adds more weight to subsequent models, allowing for gradual refinement. A value of 0.1 strikes a balance between preventing overfitting and achieving good convergence speed.

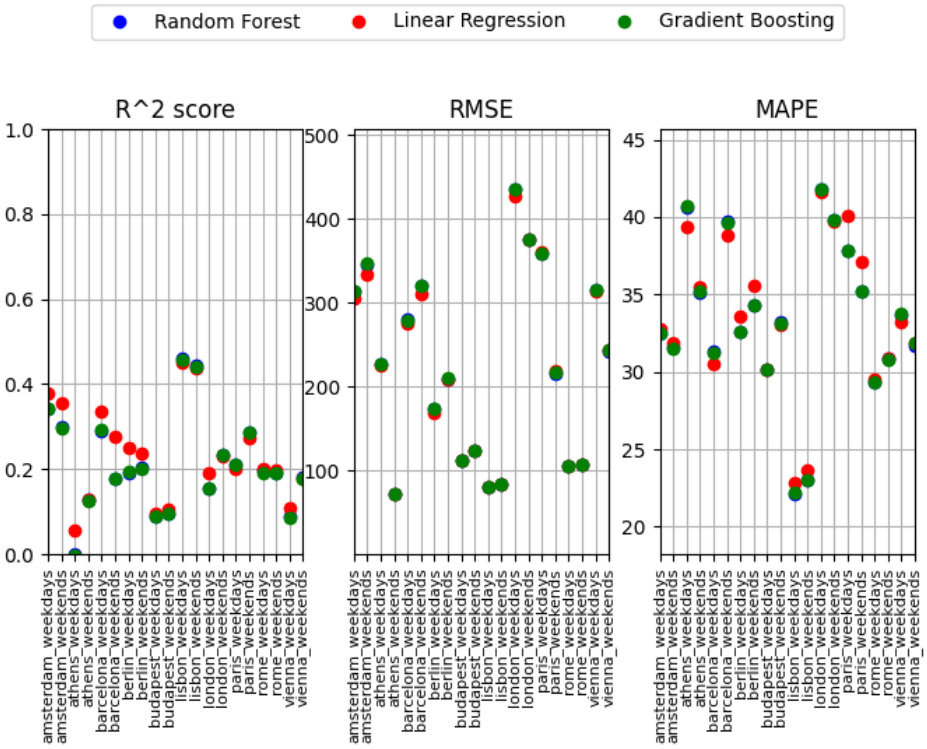
LOSS_GRADIENT = "squared_error": The squared_error loss function is commonly used in gradient boosting for regression tasks, including predicting Airbnb prices. It measures the squared difference between the predicted and actual values, emphasizing larger errors during model training.

5. Test and evaluation

To compare the how well the algorithms perform we used Root Mean Squared Error (RMSE). It is a commonly used evaluation metric for regression algorithms that measures the square root of the average squared difference between the predicted and actual values. Lower RMSE values indicate better performance. The comparison and results will be visualised - we are planning to display the results using appropriate graphs and charts from Seaborn library. MAPE (Mean Absolute Percentage Error) can be a suitable metric for evaluating a model's performance in predicting Airbnb prices due to its scale-invariance, meaning it is not affected by the magnitude of the prices, and general robustness. MAPE is also less sensitive to outliers compared to RMSE. Outliers in Airbnb prices, which can occur due to unique properties or special events, will have a smaller impact on MAPE since it focuses on relative errors rather than absolute differences. The results of each metric will be compared on charts for each city

6. Results

The results were pretty disappointing considering how much effort and time we put into this project. These are graphs of metrics evaluated from the models (Algorithms were run 50 times and these are average values):



These are numeric results of what we evaluated:

```
r_squared:
Random forest:
Average: 0.2134522718529403
Standard variation: 0.11281896422081278
Highest and lowest score: (0.45921833224126013, 0.00069487341132622
Linear:
Average: 0.23429576138300243
Standard variation: 0.10954395907565655
Highest and lowest score: (0.4504638250837128, 0.05705502478850567)
Gradient boosting:
Average: 0.2124147725864241
Standard variation: 0.11300866180704695
Highest and lowest score: (0.4583502334656833, -0.0027038454908857
rmse:
Random forest:
Average: 224.58355095263815
Standard variation: 110.8395657302569
Highest and lowest score: (434.3491334655127, 72.52879639963545)
Linear:
Average: 222.23550567495266
Standard variation: 108.74984180042689
Highest and lowest score: (426.0717690214001, 72.52995421744431)
Gradient boosting:
Average: 224.67851297414927
Standard variation: 110.85554475194907
Highest and lowest score: (434.3824283551067, 72.58965461479498)
mape:
Random forest:
Average: 33.32828211936059
Standard variation: 5.0479166703799585
Highest and lowest score: (41.7661957036883, 22.155567409141963)
Linear:
Average: 33.59850527359663
Standard variation: 4.965677884698723
Highest and lowest score: (41.65555990142807, 22.832233524786634)
Gradient boosting:
Average: 33.32253841333785
Standard variation: 5.044269448170889
Highest and lowest score: (41.76023142463466, 22.184731760037636)
```

7. Conclusions

We have noticed that some sets perform much worse than others. After analysing athenas_weekdays (on the left) and london_weekends (on the right) we can see a peculiar dependancy:

Row Labels	Average of realSum
2	146.8708515
3	126.6345944
4	150.2268148
5	170.557717
6	214.8739664
Grand Total	**155.866982**

Row Labels	Average of realSum
2	253.1772428
3	364.5807625
4	508.4616641
5	608.5940778
6	823.6525401
Grand Total	**364.3897466**

Athenas weekdays for some reason have really unintuitive pricing. We believe that it is caused by other, outside factors (similar to superhosts being cheaper on average) and we couldn't predict accurately the outcome with parameters we have chosen to train our model. In most of the datasets, the parameters we have chosen described cities the best, however there were few outliers, like Athenas or Vienna.

Another thing to notice is that Random Forest Regression and Gradient Boosting gave very similar results. It is most likely caused by the fact that those algorithms are both based on Decision Trees – therefore their performance was quite similar.

What was most interesting was that the simple Linear Regression outperformed on average both Random Forest (that is very often used as a starting baseline when making ML algorithms) and Gradient Boosting.