Airbnb Prices Prediction

Aimilios Potoupnis
National Technical University of Athens
Athens, Greece
aimiliospot@gmail.com

Orfeas Karachalios
National Technical Univarsity of Athens
Athens, Greece
orfeaskar@pm.me

***Abstract*— This project aims to predict Airbnb house prices using machine learning algorithms based on factors such as reviews, amenities, and accommodates. The dataset used includes detailed information about each listing. Regression and classification algorithms are employed to build an accurate prediction model, and factors influencing prices are analyzed to provide valuable insights. The project has the potential to benefit both hosts and travelers on the Airbnb platform.**

Keywords—Airbnb, ML, Regression, Classification, Binning, Price Prediction

I. INTRODUCTION

The goal of this project is to use machine learning algorithms to predict Airbnb house prices based on a variety of factors, including reviews, amenities, and accommodates. As the popularity of Airbnb continues to grow, accurately predicting house prices has become increasingly important for hosts looking to maximize their earnings and for travelers seeking affordable and comfortable accommodations.

To achieve this goal, we use a dataset of Airbnb listings in Athens, Greece. This dataset contains detailed information about each listing, including the number of bedrooms, the type of property, the location, and the availability calendar.

We employ a variety of regression and classification algorithms, to build a model that accurately predict house prices. In addition to predicting house prices, we also analyze the factors that most strongly influence prices. This provides valuable insights for Airbnb hosts and travelers, and potentially informs future pricing strategies for Airbnb as a company.

Overall, this project has the potential to make a significant impact on the Airbnb ecosystem by helping hosts maximize their earnings and travelers find affordable and comfortable accommodations.

II. RELATED WORK

Authors in [1] used regression to estimate the prices for "Airbnb" in New York, essentially tackling the same problem as in this report. The features of the dataset were high in number and feature selection had to be performed. Multiple feature selection algorithms were chosen, since training on the original features produced very low scores. The results were produced for each method and the best was considered picking by the coefficients of a linear regression using Lasso regularization. Models chosen were K-means clustering, a neural net, Support Vector Regression and Gradient Boost. Evaluation criteria of results turned out to be R = 0.65. The values achieved were somewhat higher than those achieved by our models. However, hyper-tuning was performed, and we expect the results would improve in our case as well.

Authors in [2] also used regression with data collected from Beijing, China. Manual feature selection was performed, keeping only 44 features out of 116.
Models used included XGBoost and Neural Networks. Again $R^2 = 0.65$ was observed, indicating a limit with that can be achieved. The explanatory power of certain features was examined as well, especially the amenities provided by the household.

A more recent effort [3] used a dataset from Amsterdam, provided by Kaggle. Features for this dataset were already decreased to 31, which may have led to the lower performance with $R^2 = 0.63$ being achieved. The kNN and linear regression model with certain modifications were used as a benchmark and more advanced models such as Random Forest, Gradient Boost and XGBoost were tested.

Another recent effort [4] conducted experiments on two datasets, Beijing and Shanghai in China. A different approach was chosen in feature selection and preprocessing, incorporating different modes of data (textual, visual, spatial). More features were used but first they were embedded in lower dimensional vectors depending on their mode and then fed to a neural network model. Among the baselines used were XGBoost, Random Forest, SVR. The achieved metrics showed an improvement over other methodologies, reaching $R^2 = 0.68$.

In [5], a smaller dataset with only 16 features was chosen. Linear regression was used as a benchmark and more advanced models were tested, including Neural Nets, Random Forest and XGBoost. The model managed to achieve an $R^2 = 0.61$. This shows that most of the explanatory power lies in a few features, although better performance can be achieved by including more.

III. DATASET AND FEATURES

A. *Inside Airbnb Datasets*

For the purpose of this project, we use a dataset available on http://insideairbnb.com/ which includes comprehensive collections of data related to Airbnb listings in various cities around the world. These datasets provide information on individual Airbnb listings, including their location, price, availability, and amenities, as well as data on reviews, host profiles, and occupancy rates.

The datasets are designed to be user-friendly and accessible, with various formats available for download, including CSV and Excel files. The data is also easily searchable and filterable, allowing users to find the specific information they need.

In addition to the basic listing information, the datasets also include data on neighborhood demographics and geography, such as crime rates and public transportation options. This provides a broader context for understanding the Airbnb market in each city and can help inform future research and policy decisions.

B. *Recursive Feature Elimination Equations*

For useful features extraction we use the Recursive Feature Elimination algorithm. RFE (Recursive Feature Elimination) is a wrapper-type feature selection algorithm used in machine learning to identify the most important features in a dataset. This means that a different machine learning algorithm is given and used in the core of the method, is wrapped by RFE, and used to help select features. This is in contrast to filter-based feature selections that score each feature and select those features with the largest (or smallest) score.

Technically, RFE is a wrapper-style feature selection algorithm that also uses filter-based feature selection internally.

RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains.

This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains.

The advantage of RFE is that it takes into account the interaction between features, rather than just selecting them based on their individual importance. It also allows for the selection of an optimal number of features based on the performance of the model.

However, one potential downside of RFE is that it can be computationally expensive, especially for large datasets with many features. Additionally, it assumes that the initial model is able to accurately rank the importance of features, which may not always be the case. Overall, RFE is a powerful tool for feature selection in machine learning, but should be used with caution and in combination with other methods.

C. *Sentiment Analysis in Comments and Descriptions*

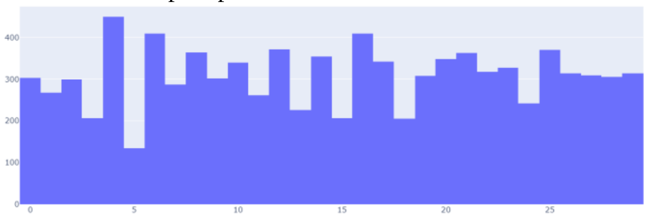
To assign numerical values in a range [-1,1] to users comments and hosts descriptions which are in text format, we use the sentiment analysis technique implemented in the nltk library. Sentiment analysis is a field of study that uses computational methods to analyze, process, and reveal people’s feelings, sentiments, and emotions hidden behind a text or interaction. It uses machine learning (ML), natural language processing (NLP), data mining, and artificial intelligence (AI) techniques to mine, extract and categorize users’ opinions on a company, product, person, service, event, or idea for various sentiments.

D. *Prices Discretization*

In order to evaluate how classification algorithms perform in this project, we discretize prices into 30 bins, each one containing approximately the same number of samples. To achieve this, we use the KBinsDiscretizer algorithm with ordinal encoding and quantile strategy.

Bins Prices Upper Bound									
10	25	29	32	35	39	40	42	45	47
50	51	54	57	60	62	65	69	71	75
80	85	92	100	110	120	134	153	184	260

Number of Samples per Bin



IV. Methods

Both classification and regression methods were applied to the dataset. These included Decision Tree, XGBoost and LGBM in the classification case and SVR, kNN, MLP and Random Forest. Algorithms belonging to the family of tree decision algorithms were used to achieve best performance, as it was seen in the scanned literature. They also perform better in terms of execution time.

Decision Tree is a supervised learning algorithm that builds a tree-like model to make decisions based on input features. The algorithm works by recursively partitioning the data into subsets based on the value of a chosen feature. At each split, the algorithm chooses the feature that best separates the data based on a criterion such as information gain, Gini impurity, or entropy. The splits continue until a stopping criterion is met, such as a maximum depth or a minimum number of samples per leaf node. In the case of classification, the algorithm produces a tree where each leaf node represents a class label. When a new data point is inputted, the algorithm traverses down the tree until it reaches a leaf node and assigns the corresponding label to the data point. They can be prone to over-fitting if not properly regularized, and they can struggle with data that has many irrelevant features.

XGBoost (Extreme Gradient Boosting) is a popular open-source algorithm for supervised learning tasks, particularly for structured and tabular data. It's based on the gradient boosting framework, which sequentially adds weak learners (decision trees) to a model to iteratively improve its performance. The algorithm starts by initializing the model with a constant value, usually the mean of the target variable. A decision tree is added to the model, which is trained to predict the errors (i.e., the difference between the predicted values and the actual values) of the previous model iteration. The depth of the tree is usually shallow to avoid overfitting. The model's predictions are updated by adding the predictions of the newly added tree. The process of adding new trees and updating the model's predictions is repeated iteratively until a stopping criterion is met. To prevent overfitting, XGBoost uses various regularization techniques such as L1 and L2 regularization (to penalize large coefficients), subsampling (to randomly sample a subset of the data for each tree), and pruning (to remove nodes with low importance). Once the training is complete, the final model is used to make predictions on new data by applying the sequence of learned decision trees. XGBoost also includes several advanced features such as handling missing data, early stopping, and feature importance calculation. It has been widely used and has achieved state-of-the-art results in many machine learning competitions and real-world applications.

LGBM is a variation of XGBoost to improve performance. LGBM uses a histogram-based approach for finding the best split points during tree construction, which can significantly reduce the time required for training and prediction.

Like the Support Vector Machine (SVM) algorithm used for classification tasks, SVR uses the concept of a maximum margin hyperplane to create a boundary between the data points. SVR aims to find a hyperplane that has the maximum distance from the closest data points, while still having the smallest possible error on the training data. The algorithm starts by loading the training data, which consists of feature vectors and corresponding target values.

A kernel function is chosen to transform the feature vectors into a higher-dimensional space, where the data may be easier to separate. The algorithm then finds the hyperplane that maximizes the margin and minimizes the error using a cost function. Unlike SVM, which seeks to maximize the margin while allowing some misclassifications, SVR aims to minimize the margin violations while keeping the margin as large as possible. The solution is found by solving a constrained optimization problem.

SVR has several advantages over other regression algorithms. It is effective in high-dimensional spaces, and it can handle nonlinear data with the use of kernel functions. It also has a good generalization ability, meaning it can perform well on unseen data. However, SVR can be sensitive to the choice of kernel function and hyperparameters, and it can be computationally expensive when dealing with large datasets.

The k-Nearest Neighbor (kNN) algorithm is a simple non-parametric machine learning algorithm used for both classification and regression tasks. The algorithm relies on the idea that similar objects are likely to share similar properties, and that objects with similar features tend to belong to the same class or have similar target values. The algorithm starts by loading the training dataset, which includes feature vectors and corresponding labels for each observation. A distance metric, such as Euclidean distance, is defined to measure the similarity between feature vectors. The number of nearest neighbors, k, is chosen. For each observation in the test set, the k nearest neighbors in the training set are identified based on the chosen distance metric. The label or target value of each neighbor is then used to make a prediction. The target value is the average of the k nearest neighbors' target values. The kNN algorithm is easy to implement, and it does not make any assumptions about the underlying data distribution. However, it can be computationally expensive and may require careful selection of k and the distance metric to achieve optimal performance. It also suffers from the curse of dimensionality, where the performance of the algorithm deteriorates as the number of features increases, unless the data is pre-processed to reduce its dimensionality.

Multilayer Perceptron (MLP) is a type of artificial neural network used for both regression and classification tasks. It consists of multiple layers of interconnected nodes or neurons that process information and learn to make predictions. The number of layers, number of neurons per layer, and activation functions for each neuron are specified. The input layer is typically the same size as the number of features, and the output layer has one neuron for regression tasks. The input features are passed through the network, and the activations of each neuron are calculated using the chosen activation function. The activations of the output layer are the predictions of the network. The difference between the predicted output and the true output is calculated using a loss function, such as mean squared error. The error is backpropagated through the network, and the weights and biases of each neuron are updated using gradient descent optimization. The gradient of the loss function with respect to each weight and bias is calculated, and the weights and biases are adjusted to minimize the loss function.

The MLP algorithm can handle nonlinear data and learn complex relationships between features and targets. It is also a universal function approximator, meaning it can approximate any function with enough hidden layers and neurons. However, MLP can be sensitive to the choice of hyperparameters, such as the number of layers and neurons, and it can overfit the training data if the network is too complex.

Random Forest is a popular machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that combines multiple decision trees to make predictions. The algorithm works by constructing a multitude of decision trees at training time and outputting the class or target value that is the mode or average of the predictions of the individual trees. A random subset of the training data is selected for each tree, and a random subset of the features is selected for each split. A decision tree is constructed for each subset of the data using a greedy algorithm that recursively splits the data based on the selected features and the best split criterion, such as information gain. The individual decision trees are combined to make a final prediction by taking the mode or average of the predictions of the individual trees.

Random Forest has several advantages over other machine learning algorithms. It is easy to use and implement, and it can handle large datasets with high dimensionality. It is also robust to outliers and missing data, and it can provide information about feature importance. However, Random Forest can be sensitive to the choice of hyperparameters, such as the number of trees and the maximum depth of each tree, and it can overfit the training data if the number of trees is too large.

V. Experiments and Results

In our experiments we split the dataset into a training set (70%) and a test set (30%) and then use Standard Scaling in order to scale numerical features to a standard range. To decide which algorithm best fits the problem, we evaluate the efficiency of regression and classification algorithms using a plethora of metrices.

After deciding which algorithm performs better, we proceed to hyperparameter tuning using HalvingGridSearch implemented in sklearn library.

A. *Standard Scaling*

Standard scaling, also known as z-score normalization, is a commonly used preprocessing technique in machine learning for scaling numerical features to a standard range. The goal of standard scaling is to transform the data so that it has a mean of zero and a standard deviation of one.

To apply standard scaling, the mean of the feature values is subtracted from each value, and the resulting difference is divided by the standard deviation of the feature values. This transformation ensures that the scaled feature values have a similar range and distribution, regardless of their original scale.

Standard scaling is particularly useful when dealing with features that have significantly different scales or units of measurement, as it helps to bring them to a common scale. This can improve the performance of many machine learning algorithms, which often assume that the features are on a similar scale.

It is worth noting that standard scaling is sensitive to outliers, which can affect the mean and standard deviation of the feature values. Therefore, it is important to handle outliers appropriately before applying standard scaling. Additionally, some machine learning algorithms, such as decision trees and random forests, are not sensitive to the scale of the features and may not benefit from standard scaling.

B. *Coefficient od Determination*

The coefficient of determination, denoted as R-squared, is a statistical measure that represents the proportion of the variance in the dependent variable (the outcome) that is explained by the independent variable(s) (the predictor(s)) in a regression model.

R-squared ranges from 0 to 1, where 0 indicates that the model does not explain any of the variability in the dependent variable and 1 indicates that the model explains all the variability in the dependent variable.

R-squared is calculated as the ratio of the explained variance to the total variance.

R-squared is a useful measure for evaluating the goodness of fit of a regression model. Higher R-squared values indicate that the model fits the data better, while lower values indicate that the model does not fit the data well. However, it is important to note that R-squared does not indicate the causal relationship between the independent and dependent variables, but only measures the goodness of fit of the model..

C. *Accuracy*

Accuracy is a commonly used metric for classification tasks and measures the percentage of correctly classified samples out of the total number of samples. In other words, it measures how often the model predicts the correct class label. However, accuracy can be misleading when the dataset is imbalanced, meaning that some classes have fewer samples than others. In such cases, the model may achieve high accuracy by simply predicting the majority class for most samples.

D. *Balanced Accuracy*

The balanced accuracy metric takes into account the class distribution and computes the average of the per-class accuracies. In other words, it measures how well the model predicts each class label, taking into account the fact that some classes may have fewer samples than others. This makes it a more suitable metric for imbalanced datasets than accuracy.

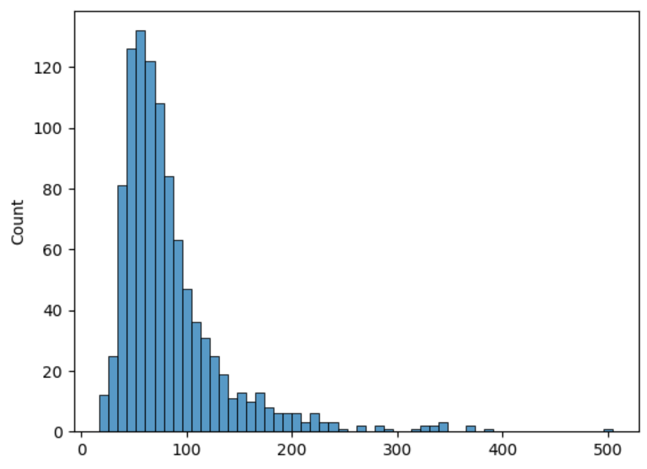
E. *F1-Score*

The F1 score is a harmonic mean of precision and recall and provides a balanced measure of both metrics. Precision measures the proportion of true positives (i.e., samples correctly classified as positive) among all samples predicted as positive, while recall measures the proportion of true positives among all actual positive samples. The F1 score balances these two metrics and is a good metric to use when the classes are imbalanced. A higher F1 score indicates better performance by the model.

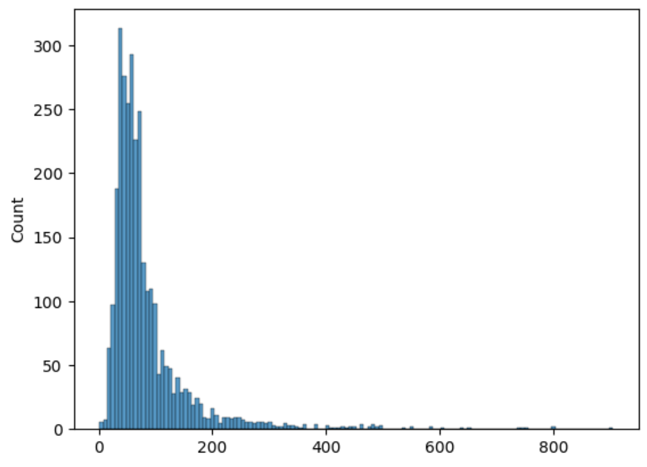
F. *Results*

After fitting and evaluating both regression and classification models in the Airbnb price prediction problem, we conclude that it is preferable to discretize prices in bins and use the Decision Trees algorithm. We have found that in regression scenarios the highest value achieved for the coefficient of determination is 0.55 whilst in classification using Decision Trees, we achieve the perfect score in Accuracy, Balanced Accuracy and F1 Score metrices.

Predicted Price Distribution in Regression Scenario

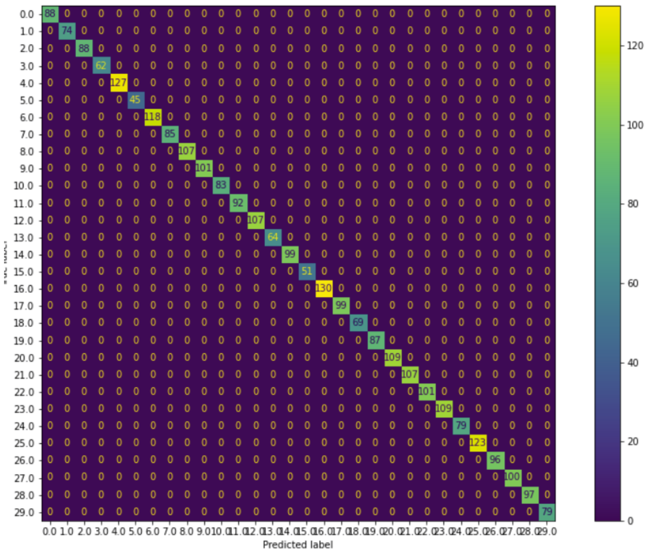


Real Price Distribution



Decision Tree Classifier Metrices			
	Accuracy	Balanced Accuracy	F1 Score
	1	1	1

Decision Tree Confusion Matrix



VI. Conclusion and Future Work

In conclusion, Decision Tree algorithm outperformed the rest of classification and regression algorithms and this can happen due to several reasons. Firstly, decision trees are easy to understand and interpret, as they provide a clear, graphical representation of the decision-making process. This makes them useful for exploratory data analysis and for identifying important features in the dataset. After that, decision trees are non-parametric, which means they make no assumptions about the underlying distribution of the data. This makes them more flexible than other classification algorithms, which may require the data to follow a specific distribution. Moreover, decision trees are able to model non-linear relationships between features and the target variable, making them suitable for datasets with complex patterns. Furthermore, decision trees are robust to outliers and missing data, as they can still make accurate predictions even when some data points are missing or deviate significantly from the rest of the data. Finally, decision trees can also be combined with other models using ensemble methods such as random forests and boosting, which can further improve their performance. It is worth noting that decision trees may not always outperform other classification algorithms, particularly on datasets with a large number of features or when the relationships between features and the target variable are complex and difficult to model. In such cases, other algorithms such as support vector machines or neural networks may be more appropriate. Additionally, decision trees are susceptible to overfitting, which can lead to poor generalization performance on new, unseen data. Regularization techniques such as pruning can help to mitigate this issue.

For further steps, we would consider deploying our model and make its predictions available to users and future investors, giving them the ability by paying a small fee to decide in which location to buy a new house to turn it into an Airbnb and calculate the expected future returns.

VII. References

[1] Kalehbasti, Pouya Rezazadeh et al. “Airbnb Price Prediction Using Machine Learning and Sentiment Analysis.” _International Cross-Domain Conference on Machine Learning and Knowledge Extraction_ (2019).

[2] S. Yang, "Learning-based Airbnb Price Prediction Model," _2021 2nd International Conference on E-Commerce and Internet Technology (ECIT)_, Hangzhou, China, 2021, pp. 283-288, doi: 10.1109/ECIT52743.2021.00068.

[3] Y. Liu, "Airbnb Pricing Based on Statistical Machine Learning Models," _2021 International Conference on Signal Processing and Machine Learning (CONF-SPML)_, Stanford, CA, USA, 2021, pp. 175-185, doi: 10.1109/CONF-SPML54095.2021.00042.

[4] L. Jiang, "A Multi-Source Information Learning Framework for Airbnb Price Prediction", arxiv.org, https://arxiv.org/abs/2301.01222 doi: 10.48550/ARXIV.2301.01222.

[5] A. Zhu, R. Li and Z. Xie, "Machine Learning Prediction of New York Airbnb Prices," _2020 Third International Conference on Artificial Intelligence for Industries (AI4I)_, Irvine, CA, USA, 2020, pp. 1-5, doi: 10.1109/AI4I49448.2020.00007.