Ensemble Learning

Group Python Project

Chenxi HE
ESSEC Business School and
Centrale Supélec, France
b00803893@essec.edu

Shiyun WANG
ESSEC Business School and
Centrale Supélec, France
b00802405@essec.edu

Yunjing JIANG
ESSEC Business School and
Centrale Supélec, France
b00794792@essec.edu

Yunqiu ZHANG
ESSEC Business School and
Centrale Supélec, France
b00794126@essec.edu

Part 1 - Airbnb Price

Abstract and Introduction

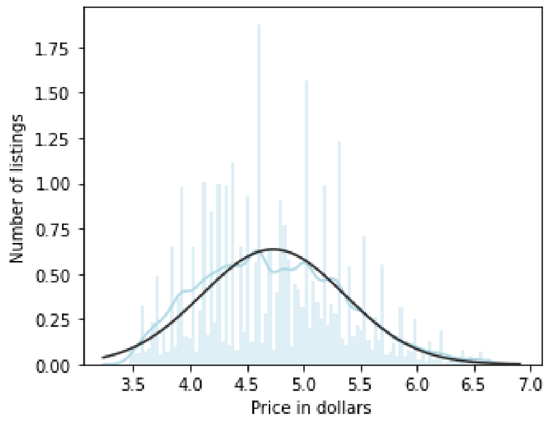
Since 2008, travelers and hosts have used Airbnb to expand the possibilities of travel and present more unique, personalized ways of experiencing the world. This dataset describes Airbnb listings and metrics of listings in New York City in 2019.

In this report, based on the data of nearly 50,000 listings in Airbnb, questions will be raised and researches will be carried out mainly around the popularity of listings and other aspects according to the information provided in the data set, such as the price, type of listings, location and minimum number of days of stay.

In this project, we will preprocess data and extract more features, perform exploratory analysis and finally implement five different ensemble learning methods to predict the price of Airbnb listings in New York City and compare performances of models using metrics.

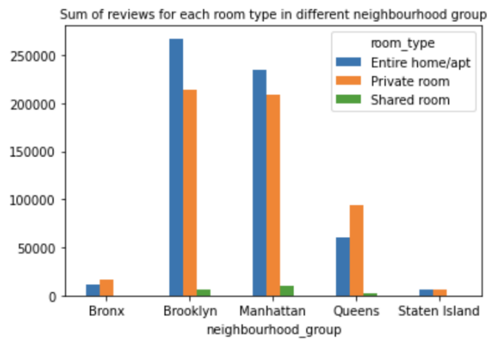
Data Preprocessing & EDA

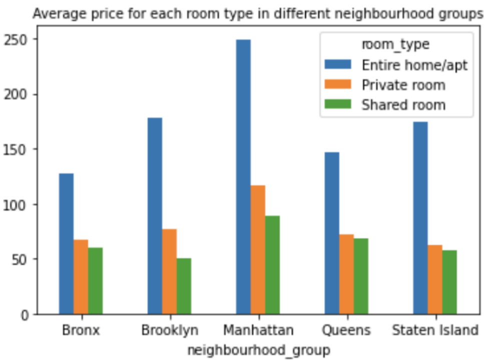
Before starting to predict the price, we did the following steps to preprocess data.
We changed the last_review format to DateTime.
We extracted some features from provided features:
'no_review' to show if the house has no review;
'count_name' to indicate the number of words in 'name'.
We checked for missing values. Missing values exist in four columns of the data set: last_review, reviews_per_month, name, host_name. And we filled them with maximum value, 0 and empty string respectively.
To improve model performance, after train-test-split, for the training set, we removed outliers from the target variable 'price' by selecting those listings whose price falls between the 1st and 99th percentiles. and then we did a logarithmic transformation to normalize the target variable. After these steps, we get the following distribution of price of the training set.



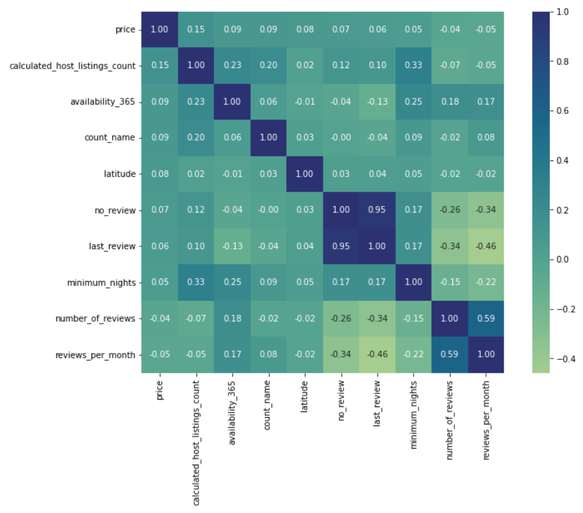
We also removed outliers from the 'minimum_nights' column in the training set by filtering out the data outside the 99th percentile value.
Then we created a preprocessor object that applied one-hot encoding for categorical features and MinMax scaling for numerical features.

We also did some data analysis about the distribution of the attributes. We tried to find out the distribution of the most reviews and the average price of different areas. By plotting the distributions, the most popular house type is Entire home/apt in Brooklyn, followed by Entire home/apt in Manhattan, and the average price of these two house types is the highest. In the Bronx and Queens, where there are relatively few reviews, the private room has a relatively high number of visits, and the price of this type of room is in the middle of the three room types. Combined with the above, Manhattan has the highest average house price, and the most house type in this area is Entire home/apt. It can be inferred that the higher house price of this house type has raised the average house price in this area. It can be seen from the above data that the most visited room type is Entire home/apt, and the average price of this type of room is the highest.





After the data processing part, we plot a correlation heatmap to help feature engineering. The final numerical features we choose are: latitude, longitude, minimum_nights, number_of_reviews, last_review, reviews_per_month, calculated_host_listings_count, availability_365, count_name. And we choose neighbourhood_group, neighborhood and room_type as the categorical variables.



Modeling

We tried 5 different ensemble models: XG Boost, Gradient Boosting, Random Forest, AdaBoost and Bagging. For each model, We built a pipeline with the preprocessor and the model and calculated the R-squared score of the model on the test data. We used GridSearchCV to tune hyperparameters. To further evaluate the performance of the models, we used cross-validation with five folds and calculated the mean R-squared score and standard deviation of the scores. The chart below is a summary of the performances obtained with the 5 different models:

Model	Mean R-squared score
XG Boost	**0.639**
Gradient Boosting	**0.631**
Random Forest	**0.642**
AdaBoost	**0.531**
Bagging	**0.629**

XG Boost

XG Boost is a type of gradient boosting model that has built-in regularization options that can help prevent overfitting and uses a distributed computing framework to train models in parallel to save computation time. After hyperparameters tuning with GridSearchCV, the best parameters were found: learning_rate: 0.1, max_dept': 7, n_estimators: 100. In our case, XGBoost was the second best model for predicting the price of Airbnb in New York City, with a mean R-squared score of 0.639 and a standard deviation of 0.008.

Gradient Boosting

Gradient Boosting is an ensemble algorithm that builds a strong predictive model by iteratively training weak learners (decision trees) on the residuals of the previous iteration, with the goal of minimizing the loss function. It can help reduce overfitting. After hyperparameters tuning with GridSearchCV, the best parameters were found: learning_rate: 0.3, n_estimator: 500. It is the third highest one with a mean R-squared score of 0.631 and a standard deviation of 0.006.

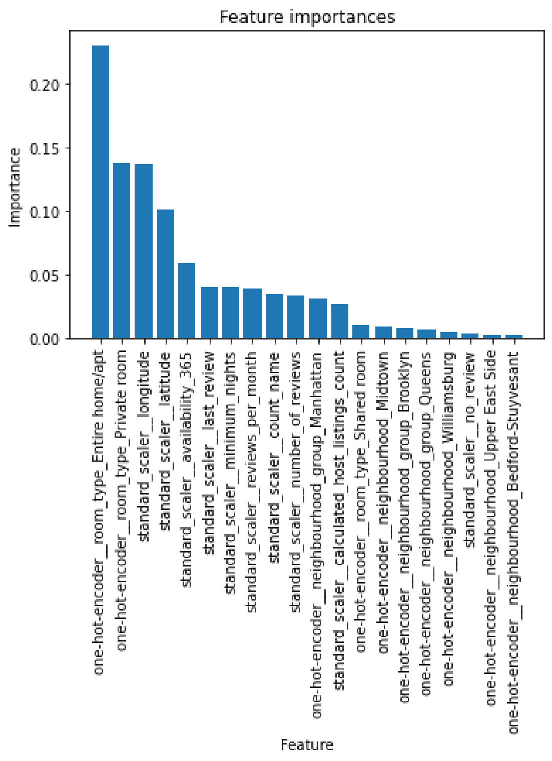
Bagging

Bagging is another ensemble method that can be used for house price prediction. It works by randomly sampling the training data with replacement to create multiple subsets of the data, and then training a base model on each subset. The final prediction is made by averaging the predictions of all the base models. Bagging can help reduce the variance of the model. In our Bagging model, the mean R-squared score is 0.629, which is the second-worst result.

Random Forest

The Random Forest model has the best performance score in our result with a mean R-squared score of 0.642 and a standard deviation of 0.010. Random Forest is a very powerful ensemble model for predicting house prices since it can handle a large number of features and can capture complex interactions between them. This is important because house prices are influenced by many factors and they interact in complicated ways. Also, Random Forest reduces overfitting by constructing multiple decision trees, where each tree is built using a random subset of the features and a random subset of the training data. Random Forest also resistant the outliers and it is easy to tune. Random Forest introduces random features on the basis of Bagging to further improve the independence of each base model. That's why Random Forest did the best performance in our model testing and has a higher score than Bagging.

We also plot the table about feature importances to see which features have the most significant impact on the output or prediction of a model. By analyzing the graph, room type_entire home/apt is the most important one and followed by room type_private room and longitude.



AdaBoost

According to the result, AdaBoost has the worst performance with a mean R-squared score of 0.531 for this dataset. The reason might be that almost all of the selected predictors have relatively low correlation coefficients, so the regularization does not work well for shrinking influences of the less important variables. Since the AdaBoost works by iteratively training weak classifiers, it may be prone to overfitting if the weak classifiers are too complex which can also lead to a bad predicting result.

Conclusion

In this report, we first clarified the problem studied in the report, then describe the way we preprocess the data, give the exploratory data analysis as well as the feature engineering, and give the models used in the study. Finally, the results of different models are validated and compared.

We have applied 5 different ensemble models-XG Boost, Gradient Boosting, Random Forest, AdaBoost and Bagging to predict the prices of Airbnb listings. Among them, random Forest shows the best result with 0.642 mean R-squared score.

While our ensemble model showed promising results, there is always room for further improvement. One area we could explore is feature engineering. By creating new features from the existing ones, we may be able to capture more of the relationship between input data and the target variable and improve the performance of our models. Additionally, we could try to use different combinations of hyperparameters to see if we can achieve even better results.

Reference

[1] Mason, L., Baxter, J., Bartlett, P., and Frean, M. (1999). Boosting algorithms as gradient descent. Advances in neural information processing systems, 12.

[2] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794.

[3] Link to Airbnb Price Dataset - https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data

Part 2 - Paris house price and category

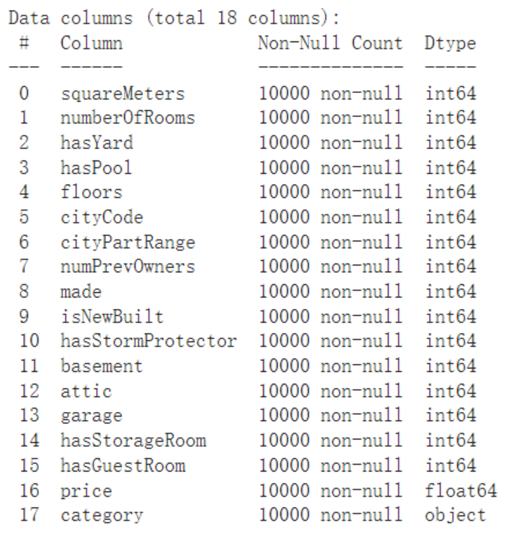
Introduction

As we all know, the area, the location, and the equipment are some important factors which will greatly influence the house price. In the condition of the fluctuation of the house price, people always try to use such features to predict the house price. Therefore, in this part we try to implement the decision tree model on a Paris housing dataset, which is a set of data created from imaginary data of house prices in an urban environment - Paris.

We build two decision tree models in total: a decision tree classification model to predict the house category(basic or luxury) and a decision tree regression model for predicting the house price, to better understand the basic logic behind the decision tree.

Data_preprocessing & EDA

The dataset we use is composed of imaginary data about house prices in Paris, which is recommended by Kaggle for practice and education purposes. There are 18 columns and all the features are numeric variables except the "category".



```
Data columns (total 18 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   squareMeters       10000 non-null  int64
 1   numberOfRooms      10000 non-null  int64
 2   hasYard            10000 non-null  int64
 3   hasPool            10000 non-null  int64
 4   floors             10000 non-null  int64
 5   cityCode           10000 non-null  int64
 6   cityPartRange      10000 non-null  int64
 7   numPrevOwners      10000 non-null  int64
 8   made               10000 non-null  int64
 9   isNewBuilt         10000 non-null  int64
 10  hasStormProtector  10000 non-null  int64
 11  basement           10000 non-null  int64
 12  attic              10000 non-null  int64
 13  garage             10000 non-null  int64
 14  hasStorageRoom     10000 non-null  int64
 15  hasGuestRoom       10000 non-null  int64
 16  price              10000 non-null  float64
 17  category           10000 non-null  object
```

Before modeling, we check the missing value and encode the categorical feature as numbers. Then we plot the Correlation Matrix to decide the selected features for our model.

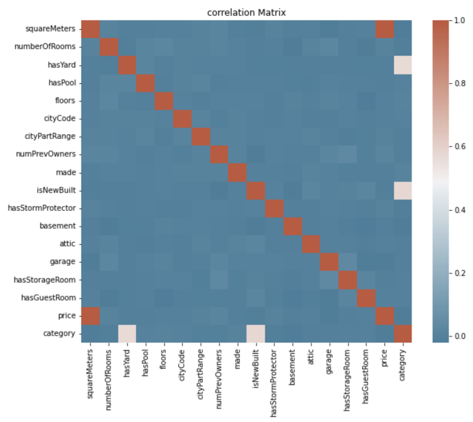
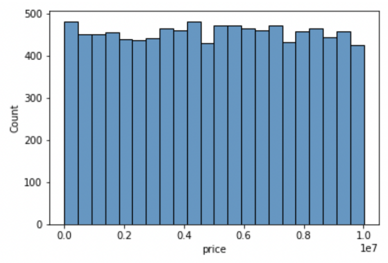


figure 1: the correlation matrix of all features

After preprocessing, we continue to visualize the dataset to gain deeper insights. When we label the price that is higher than the mean (round to 5,000,000) using the 'cat' variable, we find that nearly half of the data are cat houses, and the distribution of the price is nearly even.



We also find the number of "Basic" houses is much higher than the "Luxury" ones, using countplot.

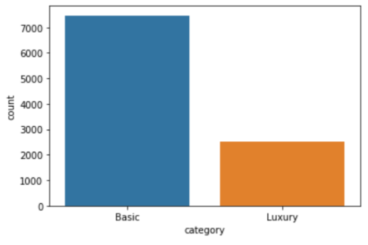


figure 2: the countplot of category

Besides, we are curious about the relationship between the year when the house was made and its price. And we observe that there is a consistent price fluctuation from 1990 to 2021. In addition, the price of the luxury house shows greater changes in value than that of the basic house, with fluctuation more significant and more often during the years analyzed.

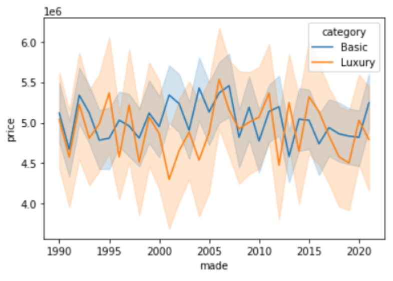


figure 3: the plot of pine on the feature "made" by category

Finally, we split the data into training & test sets and drop columns separately for classification task and regression task.

Modeling

Decision Tree Classification

In the decision tree classification task, we aim to predict the value of the category (basic or luxury).

Firstly, we conduct scaling on the training dataset, to ensure each input is on a similar scale, making it easier for the decision tree to compare and analyze. And we start with the default configuration of the model, which results in a 0.996 accuracy score and 0.003 mean square error.

Next, we specify the "max_depth" ranging from 1 to 10 and "min_samples_split" ranging from 2 to 20 and we implement the hyperparameter tuning in GridSearchCV. The best estimator score returned is 0.99, in parameters of: *{'max_depth': 2, 'min_samples_split': 2}*.
However, the real model performance keeps the same.

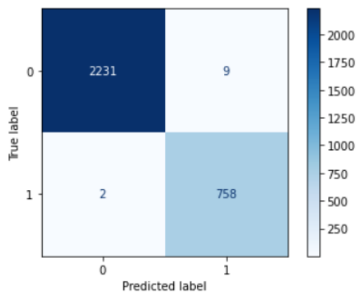


figure 4: the confusion matrix of the decision tree classification model

Finally, we look at the feature_importance and find that the most important feature is "hasYard" in this decision tree classification model.

Decision Tree Regression

In the decision tree classification task, we aim to predict the value of the price.

We also split 30% of the dataset into testset and the remaining is the train set. Then we take a look at the density of the Y variable of both datasets.

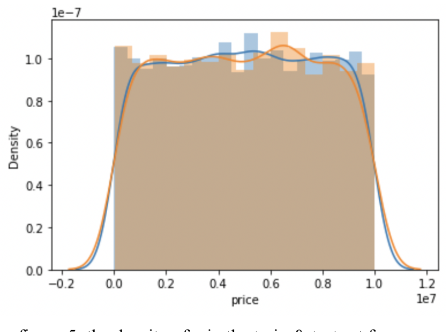


figure 5: the density of y in the train & test set for regression task

After scaling on the training dataset, the regression model gets a high r2 score of 99.6%. Because in this dataset, the correlation between price and area is very strong, the SquareMeter plays an important role in the feature importance and gets us a high accuracy forecasting.

Conclusion

In summary, decision trees can be a powerful tool for predicting house prices and categories. By analyzing various features such as location, number of rooms, and square footage, a decision tree can accurately estimate the price of a house with a high degree of confidence.

However, it is important to note that decision trees are not perfect and may not always provide accurate predictions. The dataset we use is imaginary and a few of the features have a strong correlation with the predicted variable. So the accuracy we got is high. It is crucial to gather as much relevant data as possible and fine-tune the model to achieve the best results. For example, in the classification model, we try to get the optimized parameters max_depth and min_samples_split and calculate the confusion matrix. All of these methods will help us to get a better prediction when using real-world data.

Overall, decision trees offer a straightforward and intuitive approach to predicting house prices, making them a valuable tool for real estate professionals and homeowners alike.

Reference

[1] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984).Classifification and regression trees. CRC press.
[2] Géron, A. (2017). Hands-on machine learning with scikit-learn and tensorflflow: Concepts. Tools, and Techniques to build intelligent systems.
[3] Link to Paris House Dataset - https://www.kaggle.com/datasets/mssmartypants/paris-housing-classification