Ensemble Project

Date: 27 March 2023

ESSEC & CentraleSupélec

Our team

Mengyu LIANG
mengyu.liang@student-cs.fr

Nhat Mai NGUYEN
nhatmai.nguyen@student-cs.fr

Jinji SHEN
jinji.shen@student-cs.fr

Vanshika SHARMA,
vanshika.sharma@student-cs.fr

Overview

01 Predict Airbnb prices in New York City using ensemble models

02 Implement a decision tree from scratch in Python

Project 1

Predict Airbnb prices in New York City using ensemble models

Motivations

- Opportunity to study and understand factors impacting Airbnb prices in one of the most vibrant and diverse city
- Relevant to real-world applications with practical impact in hospitality industry: insights can lead to more informed decisions for hosts and guests
- Apply and gain deeper understanding of different ensemble methods like Random Forest, Gradient Boosting, etc. to improve accuracy and robustness of predictions

Methodology

$$\begin{tikzcd}\text{Data exploration} \arrow[r] & \text{Data pre-process} \arrow[r] & \text{Modeling \& evaluation}\end{tikzcd}$$

Data exploration

- Check duplicate, missing values
- Check format consistency
- Check outlier
- Plotting to extract insights

Data pre-process

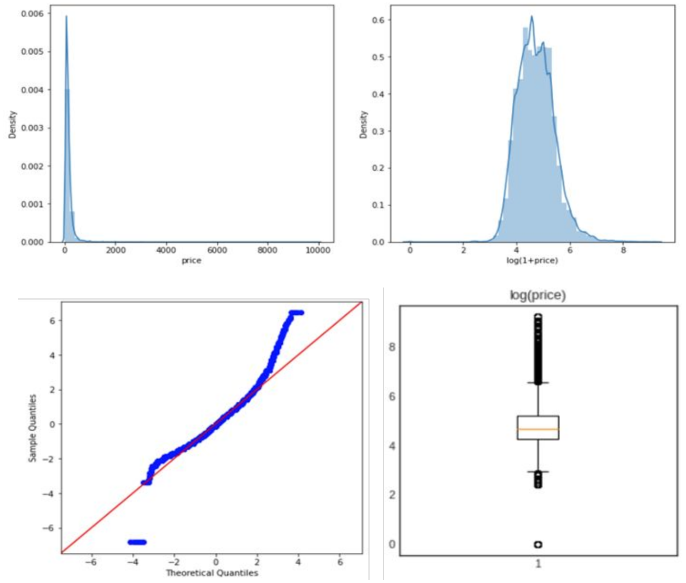
- Handle missing values & outliers
- Handle skewed data
- Encode categorical variables
- Adding features
- Rescale values

Modeling & evaluation

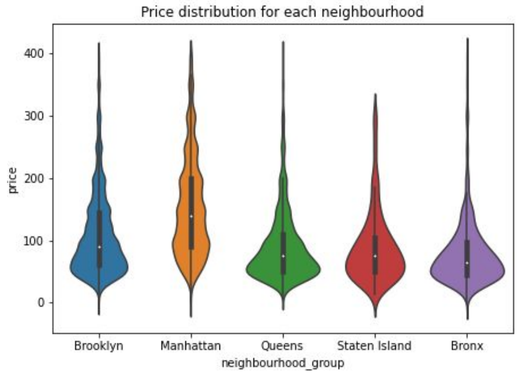
- Check feature importance
- Train, test, tune & evaluate models: Decision tree, Random Forest, CatBoost, GB, XGBoost, Bagging, AdaBoost

Insights from exploration

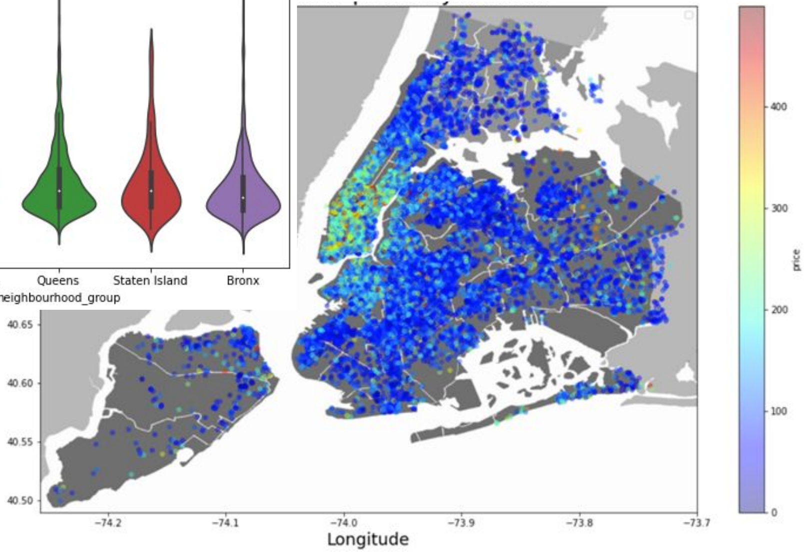
- Distribution of price is **highly skewed** and has **fat tail** on the right
- The distribution reflects the **extreme price of certain airbnbs** that are well furnished and at good location
- **Log transformation** of price follows the **normal distribution** more or less
- 2 scenarios tested:
    - Exclude 5% of the data that have extreme prices
    - Not remove any data

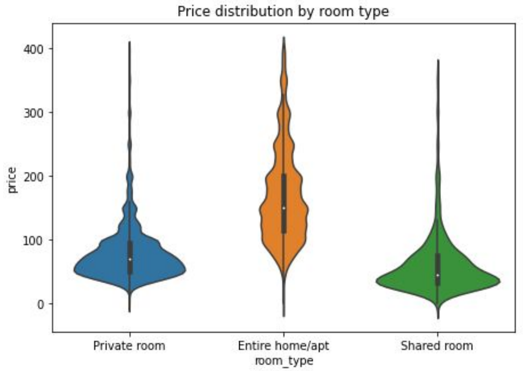


Insights from exploration



Price distribution by location



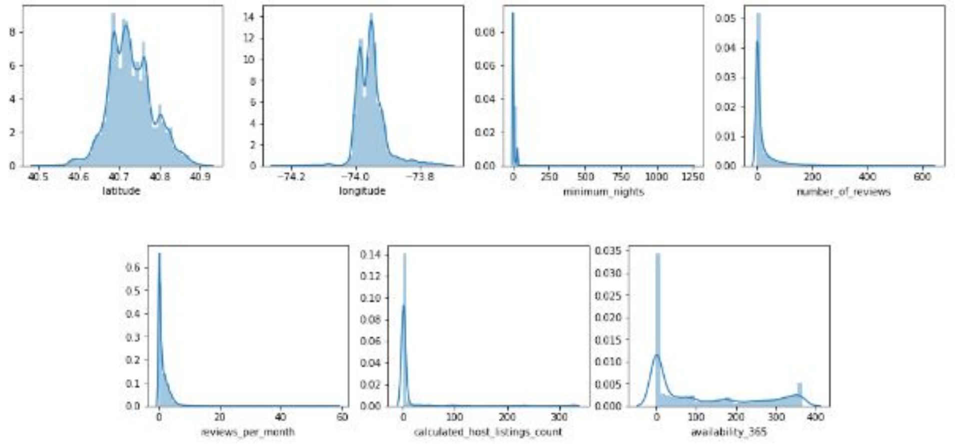


Price highly depends on whether the airbnbs are

- In the center (Manhattan & Brooklyn)
- Entire home/ apartment

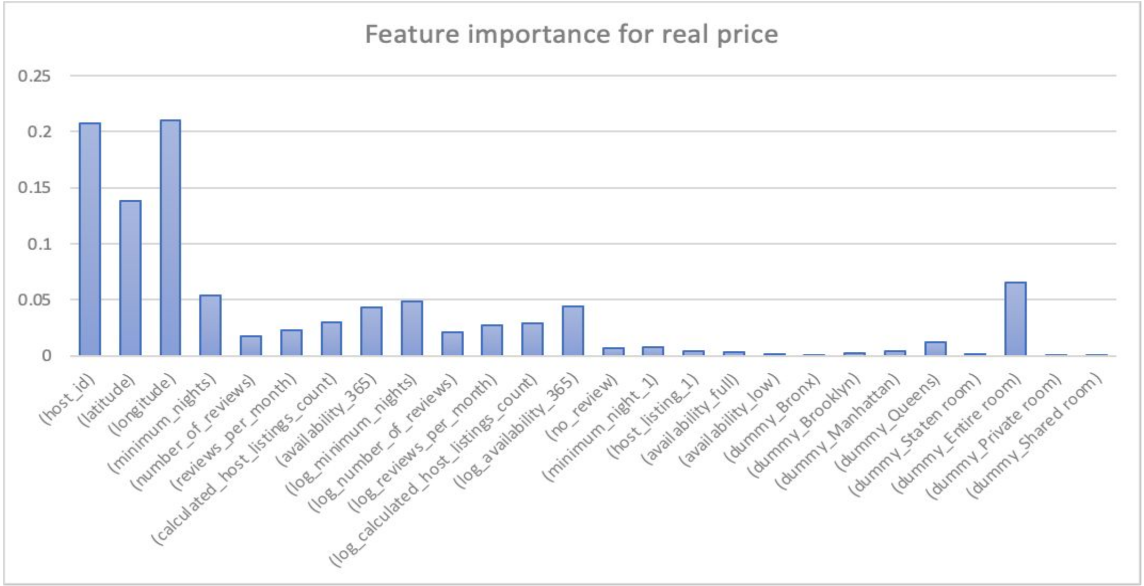
Insights from exploration

Density



- Except for longitude & latitude, other numerical variables have **highly skewed distributions**
- Due to the high concentration on certain values, **additional yes/no features are added** to test out

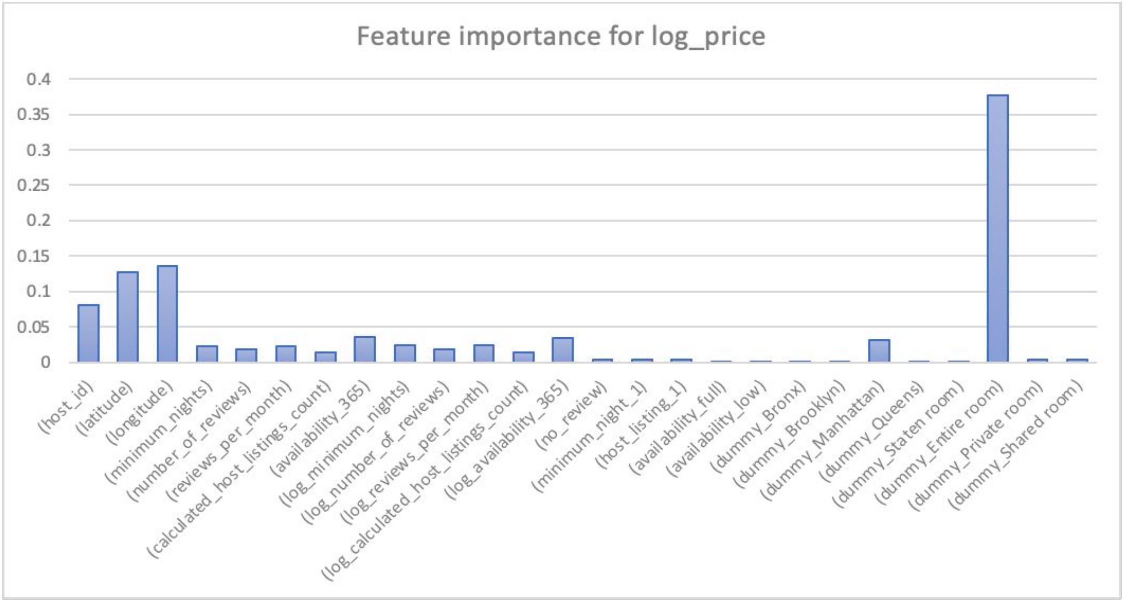
Feature importance - Real Price



Important features:

- host_id
- latitude
- longitude
- minimum_nights
- Availability_365
- dummy_Entire home/apt

Feature importance - Log Price



Important features:

- host_id
- latitude
- longitude
- dummy_Entire home/apt

Modelling - Scenarios

Scenario 1

Predicting on Real Price and All Data (i.e., Without Removing 5% Outliers)

Scenario 2

Predicting on Real Price and After Removing 5% Outliers

Scenario 3

Predicting on Log Price and All Data (i.e., Without Removing 5% Outliers)

Scenario 4

Predicting on Log Price and After Removing 5% Outliers

Within these scenarios, we trained all the models on both all features and only important features to test their performance.

Models

Decision Trees

Random Forest

Ada Boost

Gradient Boosting

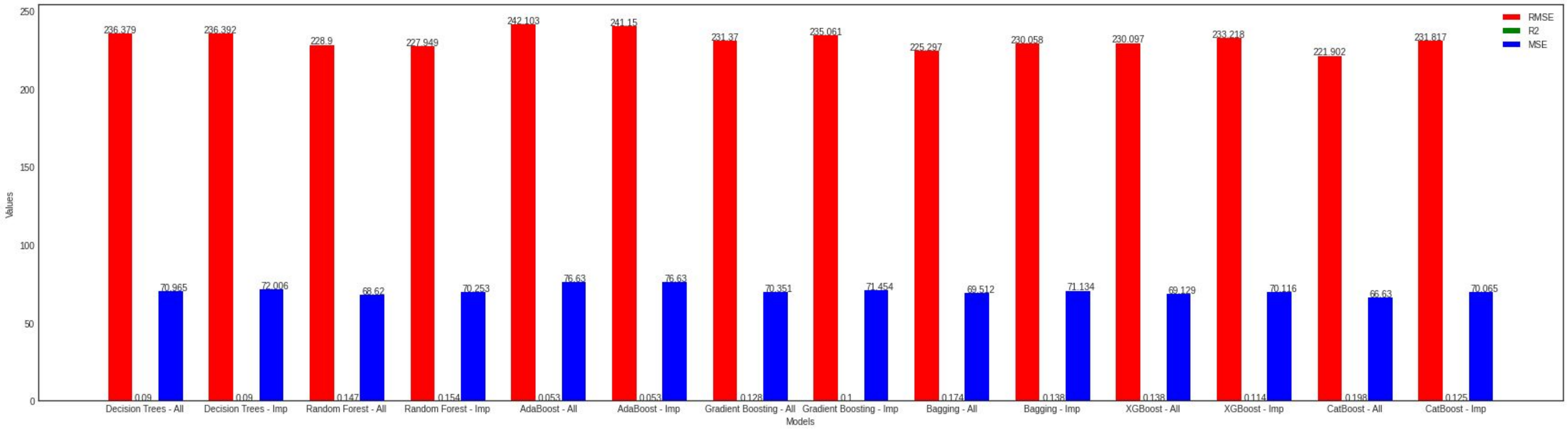
Bagging

XGBoost

CatBoost

We tested all these models on all the 4 scenarios and by training them on all features as well as only important features.

Scenario 1 - Predicting on Real Price and All Data



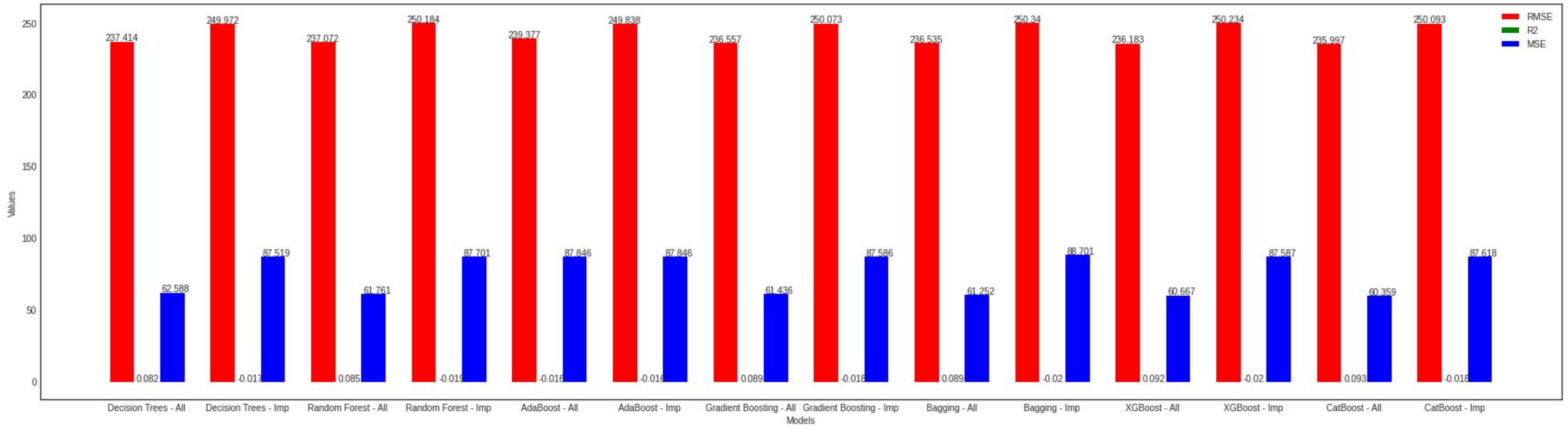
With all features:

- Based on high RMSE and MAE scores and low R-squared score, all models **perform poorly** when choosing the real price as the dependant variable to predict.
- **CatBoost**, **Bagging** and **Random Forest** are the top 3 models who perform the best.

With important features:

- All models perform **even worse when we use only important features** for training, except Random Forest which has a slightly better performance in this circumstance(R-squared: 0.154 vs. 0.147).
- **CatBoost, Bagging** and **Random Forest** are still the top 3 models with the best performance.

Scenario 2 - Predicting on Real Price and Removing 5% Outliers



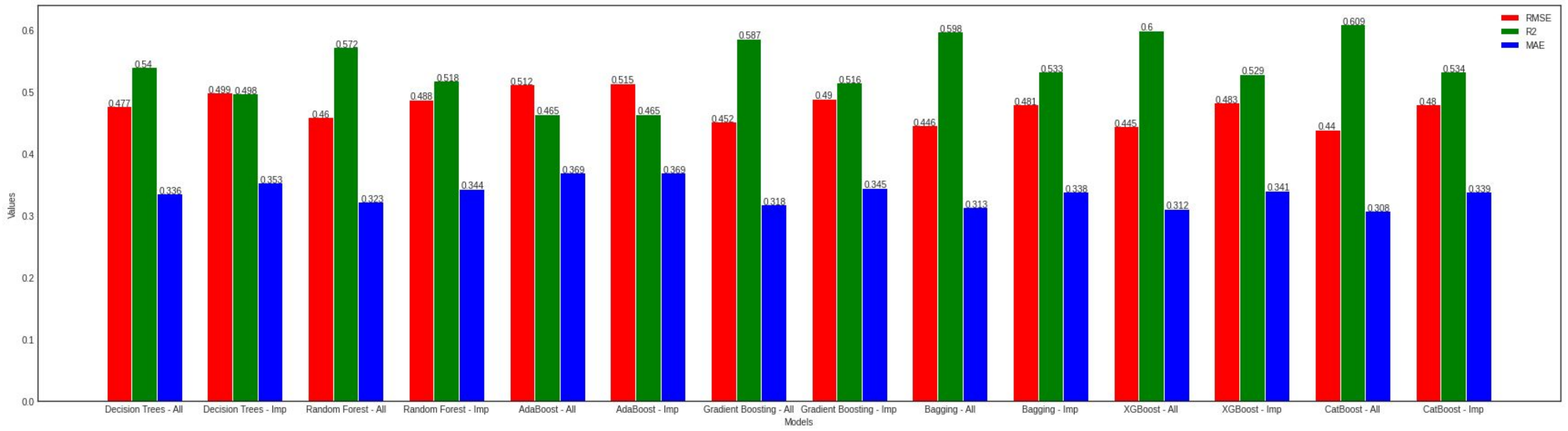
With all features:

- Removing 5% outliers from the training set obviously hurt the performance of all models, specifically AdaBoost with a negative R-squared.
- **CatBoost**, **XGBoost** and **Bagging** are the top 3 models who perform the best.

With important features:

- All models perform **even worse when we use only important features** for training.
- **AdaBoost**, **Decision Trees** and **Gradient Boosting** are the top 3 models with the best performance.

Scenario 3 - Predicting on Log Price and All Data



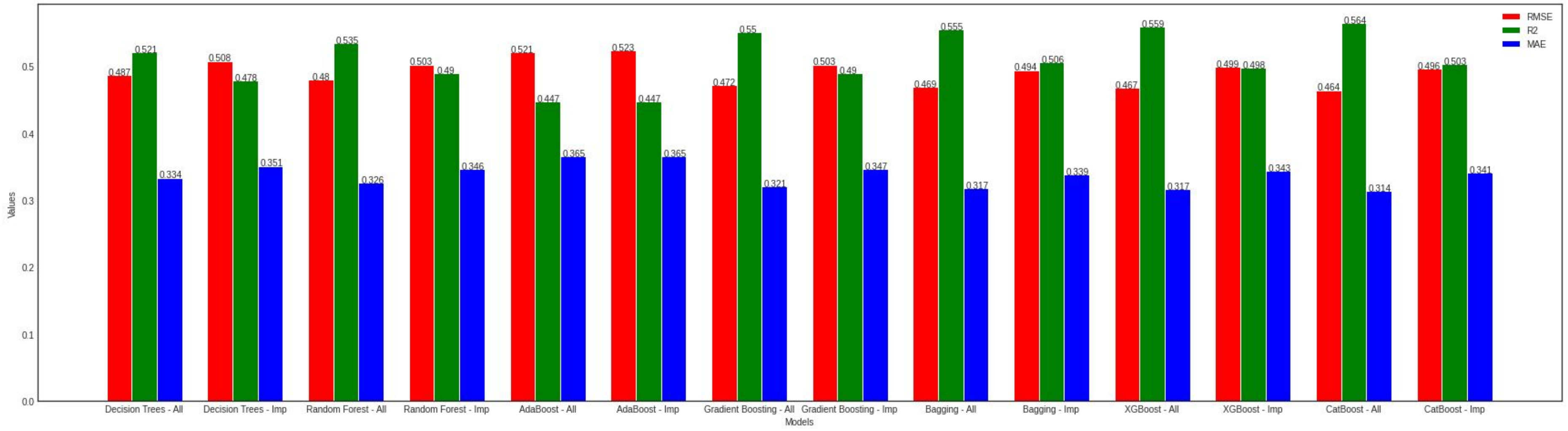
With all features:

- All models perform much better on log price.
- **CatBoost**, **XGBoost** and **Bagging** are the top 3 models who perform the best.

With important features:

- All models perform **slightly worse when we use only important features** for training but better than predicting on real price.
- **CatBoost, XGBoost** and **Random Forest** are the top 3 models with the best performance.

Scenario 4 - Predicting on Log Price and Removing 5% Outliers



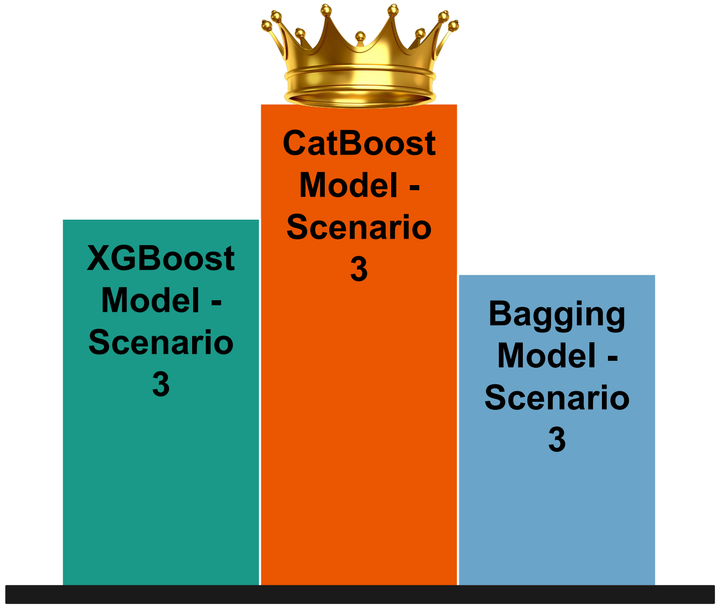
With all features:

- All models perform better in this scenario when compared to predicting on real prices (scenario 1 and 2) but it is not better than Scenario 3.
- **CatBoost**, **XGBoost** and **Bagging** are the top 3 models who perform the best.

With important features:

- All models perform **slightly worse when we use only important features** for training.
- **CatBoost, XGBoost** and **Random Forest** are still the top 3 models with the best performance.

Conclusion


CatBoost Model - Scenario 3
XGBoost Model - Scenario 3
Bagging Model - Scenario 3

From this one can also conclude, that predicting on log price and using all the data (i.e., not removing any outlier) is the best way. We can change the prediction from log price to price by simply applying the exponential function.

The details of the code can be found on our GitHub page: https://github.com/nguyen-nhat-mai/ensemble_project/blob/main/Airbnb_Price_Prediction.ipynb

Project 2

Implement a decision tree from scratch in Python

Project 2 - Dataset and Algorithms:

Dataset:

- Load a wine dataset for classification (scikit-learn)
- Generated synthetic data for regression

Steps of the decision tree algorithm:

- Determine the splitting criterion (Gini impurity or MSE) for the current node
- Compute the impurity or MSE for each possible split of the data based on each feature
- Choose the feature that minimizes the impurity or MSE and split the data based on its threshold value
- Recursively build the tree by repeating steps 1-3 for each child node until a stopping criterion is met

Project 2 - Results

- Classification task: achieve an accuracy of 91.67% on the test set
- v.s. the scikit-learn decision tree classifier- accuracy of 94.44%

- Regression task: achieve a RMSE of 27.65 on the test set
- v.s. the scikit-learn decision tree regressor - RMSE of 18.24

Project 2 - Conclusion

- Our decision tree implementation in python can be used for both classification and regression tasks with flexibility in customization of stopping and splitting criteria.
- Our implementation provides a starting point for further exploration and experimentation with decision trees
- The details of the code can be found on our GitHub page: https://github.com/nguyen-nhat-mai/ensemble_project/blob/d62ca4137e1934dbe5812bf10bf7e2b4dc64e3f5/Decision_tree_classifier_regressor.ipynb

Thank you.