CentraleSupélec

Airbnb Price Predictions
&
Decision Tree

GitHub link

CHAGNON Pierre
MOHAMED Shamir
SARFATI Alban
TAYLOR Thomas
TRIGANO Elie

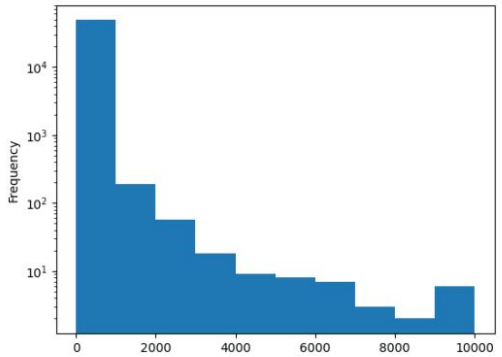
Outline

Airbnb Price Predictions

Dataset Exploration

- Dataset contains pricing and general information of AirBnB rentals in New York in 2019, the dataset lacks specific information about the nature of the AirBnBs being rented, such as luxurious or low-quality.

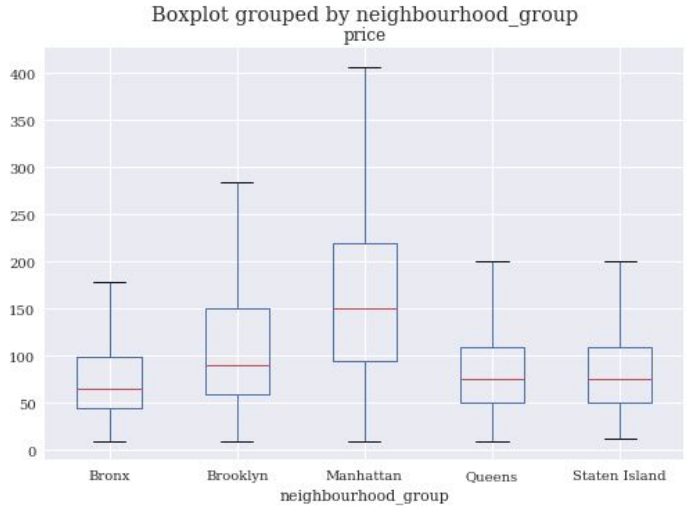
- The rental price distribution has many outliers that might affect the prediction quality and should be tested for removal, and null values for last reviews and reviews per month can be assumed as locations with no reviews and replaced with 0, while some columns can be removed due to redundant information.

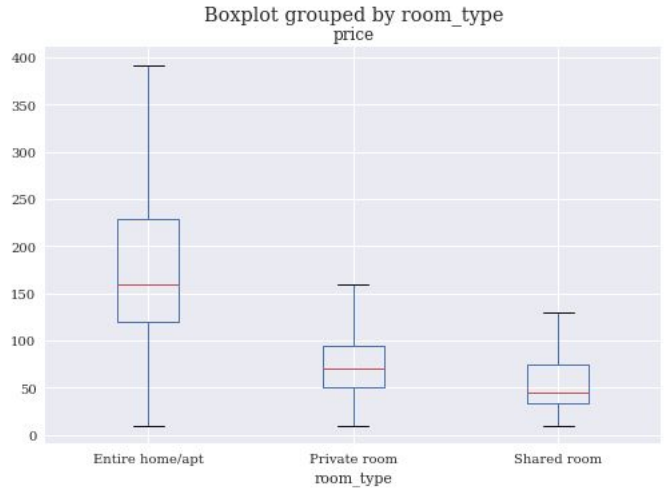


(a) Price distribution

CentraleSupélec

Dataset Exploration





- There are also interesting insights when it comes to analysing the price distribution per New York boroughs. Indeed, we observe that Manhattan has a much higher price range than the other districts. We also see that Brooklyn is higher in terms of price compared to Bronx, Queens and Staten Island that have somewhat the same prices.
- When we look at the distribution of prices in the different type of flats we can also see that they are higher priced nights in entire homes or apartments compared to a private or shared room, which makes sense to have higher prices for higher rental space.

CentraleSupélec

Model Development & Evaluation

- Outlier removal using Isolation Forest technique with a contamination score of 0.05 gave the best results compared to IQR or Z-score.

- One hot encoding was used for 'neighbourhood group' and 'room type', while Target Encoding was used for 'host id' and 'neighbourhood' due to their high cardinality.

- RMSE and MAE are the metrics to measure the error between predictions and actual values, and R^2 to assess the model fit.

Model Selection

	Model	Cross-Val Score	MSE	RMSE	MAE	R2
0	XGBoostRegressor	0.597973	2187.791253	46.773831	33.052490	0.574681
1	GradientBoostingRegressor	0.597564	2244.356671	47.374642	33.575963	0.563684
2	RandomForestRegressor	0.563950	2399.456373	48.984246	34.432058	0.533532
3	ExtraTreesRegressor	0.528926	2556.434869	50.561199	35.270560	0.503015
4	Bagging Regressor	0.534064	2574.700814	50.741510	35.613579	0.499464
5	AdaBoost Regressor	0.526538	2580.081703	50.794505	35.248159	0.498418
6	Decision Tree Regressor	0.231640	4211.875195	64.898961	44.550011	0.181188

We obtain an RMSE of 46.77 and an R2 of 57% without any tuning for our best model.

We focused on these three models :

- Decision Tree
- Random Forest
- XGBoost

Model Selection : Decision Tree

We decided to tune the following parameters :

- **`max_depth:`** the maximum depth of the tree
- **`min_samples_split:`** the minimum number of samples required to split an internal node
- **`min_samples_leaf:`** the minimum number of samples required to be at a leaf node
- **`max_features:`** the number of features to consider when looking for the best split

	Model	Cross-Val Score	MSE	RMSE	MAE	R2
0	Decision Tree Regressor	2232.608511	2419.825764	49.191725	34.649446	0.529572

Model Selection : Random Forest

We decided to tune the following parameters :

- **`n_estimators:`** the number of trees in the forest
- **`max_depth:`** the maximum depth of the tree
- **`min_samples_leaf:`** the minimum number of samples required to be at a leaf node
- **`max_features:`** the number of features to consider when looking for the best split

	Model	Cross-Val Score	MSE	RMSE	MAE	R2
0	Random Forest Regressor	2096.776144	2249.488167	47.428769	33.366679	0.562687

Model Selection : XGBoost

We decided to tune the following parameters :

- **`n_estimators:`** the number of trees
- **`max_depth:`** the maximum depth of the tree
- **`learning_rate:`** the step size shrinkage used to prevent overfitting
- **`subsample:`** the fraction of observations to be randomly sampled for each tree

	Model	Cross-Val Score	MSE	RMSE	MAE	R2
0	XGBoost	2022.330945	2154.310355	46.414549	32.65364	0.58119

Performance Evaluation

	Model	Cross-Val Score	MSE	RMSE	MAE	R2
0	XGBoost_tuned	-2021.451255	2152.036638	46.390049	32.628571	0.581632
1	XGBoostRegressor	0.597973	2187.791253	46.773831	33.052490	0.574681
2	GradientBoostingRegressor	0.597564	2244.356671	47.374642	33.575963	0.563684
3	Random Forest Regressor_tuned	2096.776144	2249.488167	47.428769	33.366679	0.562687
4	RandomForestRegressor	0.563950	2399.456373	48.984246	34.432058	0.533532
5	Decision Tree Regressor_tuned	2232.608511	2419.825764	49.191725	34.649446	0.529572
6	ExtraTreesRegressor	0.528926	2556.434869	50.561199	35.270560	0.503015
7	Bagging Regressor	0.534064	2574.700814	50.741510	35.613579	0.499464
8	AdaBoost Regressor	0.526538	2580.081703	50.794505	35.248159	0.498418
9	Decision Tree Regressor	0.231640	4211.875195	64.898961	44.550011	0.181188

- The XGBoost tuned model outperformed the other regression models in predicting the price of an Airbnb listing in New York, as it had the lowest RMSE and MSE, and the highest R^2 score of 0.58.
- The Random Forest Regressor tuned and Decision Tree Regressor tuned models also performed well, but with slightly lower accuracy compared to the XGBoost model.
- Despite having the best performing model with the XGBoost, there is still room for improvement in the prediction accuracy, as the RMSE benchmark shows an average of $46 difference from the true value.


Decision Tree
Implementation

Best Split

$$H(X) = -\sum_{i=1}^{n} P(x_i) log P(x_i)$$

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- Implementing a decision tree involves dividing the input space into distinct regions and predicting the most common class or average of observations for each region
- Determining how to divide the predictor space requires metrics based on information theory for both classification and regression problems
- Therefore we use two metrics to evaluate the best split : Entropy and Residual Sum of Squares (RSS) : Entropy can be used to calculate the information gain of a potential split in a decision tree, and RSS is used for the regression tasks

Architecture

```
class Node:
    '''
    Helper class which implements a single tree node.
    '''
    def __init__(self, feature=None, threshold=None, data_left=None,
data_right=None, gain=None, value=None):
        self.feature = feature
        self.threshold = threshold
        self.data_left = data_left
        self.data_right = data_right
        self.gain = gain
        self.value = value

```

```
class DecisionTree:
    '''
    Class which implements a decision tree classifier algorithm.
    '''
    def __init__(self, min_samples_split=2, max_depth=5, classifier=True):
        self.min_samples_split = min_samples_split
        self.max_depth = max_depth
        self.classifier = classifier
        self.root = None

```

Algorithm 3: Build Decision Tree

Feature matrix X, target vector y, current depth *depth* Tree node

$n_{rows}, n_{cols} \leftarrow shape(X)$;

if $n_{rows} \geq min\_samples\_split$ *and* $depth \leq max\_depth$ **then**

  $best \leftarrow$ _best_split(X, y);

  if $best['gain'] > 0$ **then**

    $left \leftarrow$ **Build Decision Tree**$(best['df\_left'][:, :-1], best['df\_left'][:, -1], depth + 1)$;

    $right \leftarrow$ **Build Decision Tree**$(best['df\_right'][:, :-1], best['df\_right'][:, -1], depth + 1)$;

    return Node$(best['feature_index'], best['threshold'], left, right, best['gain'])$;

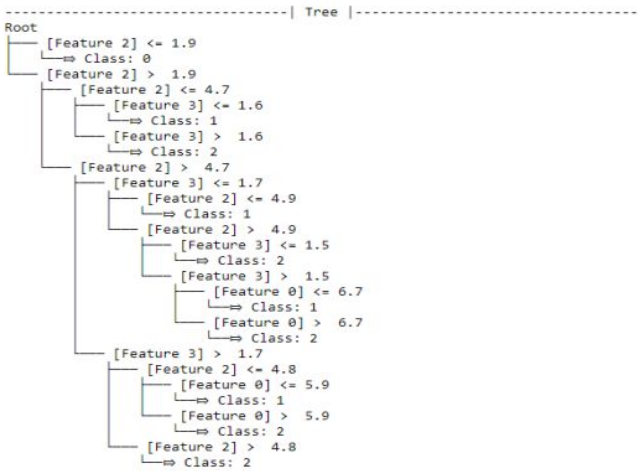
  end

end

return Node(mode(y));

- The whole recursive process is called through our fit function that calls the build function above for X and y.
- Then, all the necessary functions like entropy, information gain, and best_split can be found in the code.
- To get the prediction, we simply call the predict function that goes through the tree searching for a leaf node and gets its value.

Complementary Options



- We added features to our decision tree model to increase interpretability, including text and plot representations of splits
- Our model achieved 100% accuracy on the iris dataset for classification, but performed slightly worse than sklearn's algorithm for regression, potentially due to differences in metric penalization or splitting point definitions

```
MAE with our DT:  55.48314606741573
MSE with our DT:  5044.741573033708
MAE with sklearn DT:  55.651685393258425
MSE with sklearn DT:  5195.494382022472
```

Contents lists available at ScienceDirect

North American Journal of Economics and Finance

journal homepage: www.elsevier.com/locate/najef

Predicting the direction of stock market prices using tree-based classifiers

Suryoday Basak[a], Saibal Kar[b,c], Snehanshu Saha[a,*], Luckyson Khaidem[a], Sudeepa Roy Dey[a]

[a] *Department of Computer Science and Engineering, and Center for Applied Mathematical Modeling and Simulation (CAMMS), PESIT South Campus, Bangalore, India*

[b] *Centre for Studies in Social Sciences, Calcutta, India*

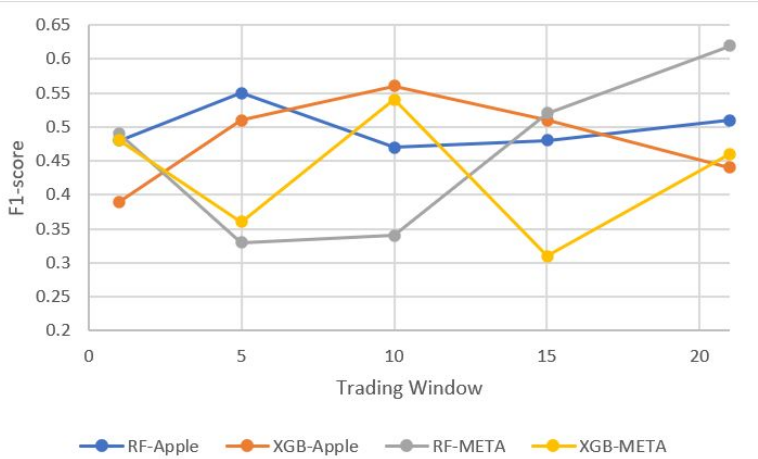
[c] *IZA, Bonn, Germany*

Abstract

- Design a framework using tree-based method to learn from the market data and predicts the direction in which a stock price will change at the closing time everyday.
- The aim is to generate models, RandomForest and XGBoost, that minimize the forecasting error, hence minimizing investment risk, for a short (1 day) to a medium (21 days) trading window.
- We evaluate our approach and report the testing performances for two technological stocks, Apple and Meta.

CentraleSupélec

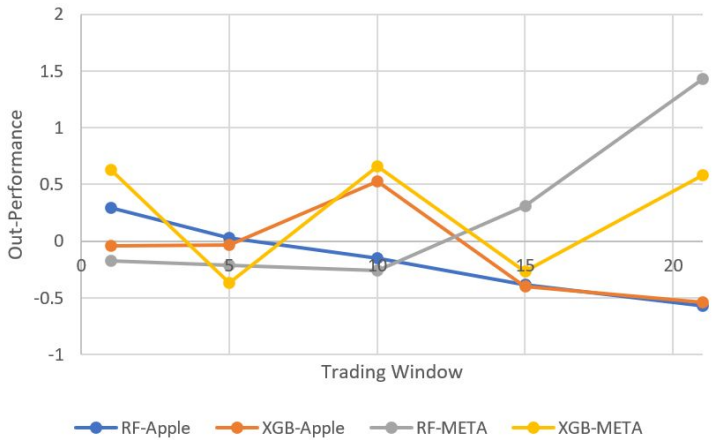
Trend of F1-score against the trading width



- Overall, the F1-score range from 30% to 60% depending on the stock and the trading window.
- F1-score doesn't increases with the increase in window-width, the trend is non-linear for both stocks.
- Random Forest has higher F1-score on 21 days window while XGBoost has higher F1-score on 10 days window-width.

CentraleSupélec

Trend of out-performance strategy



- As for F1-score, there isn't a generalized trend.
- For high out-performance on a 10 days window, XGBoost are to be preferred, while the Random Forest' out-performance on Meta 21 days window is substantial.

CentraleSupélec

Trading indications

- The model suggests trading decisions based on n-days intervals, with red dots indicating a predicted drop in prices and green dots indicating a predicted rise.
- This information can guide buying or selling decisions.
- We can observe the out-performance of the RF model strategy compared to the market (+143%).
- The framework allowed to improve the financial performances by providing significant long-short predictive signals.

CentraleSupélec

Thank You for your attention !

CentraleSupélec