Introduction to Prediction

with a final project on Airbnb prices in NYC

Mentee: Liuyixin Shao

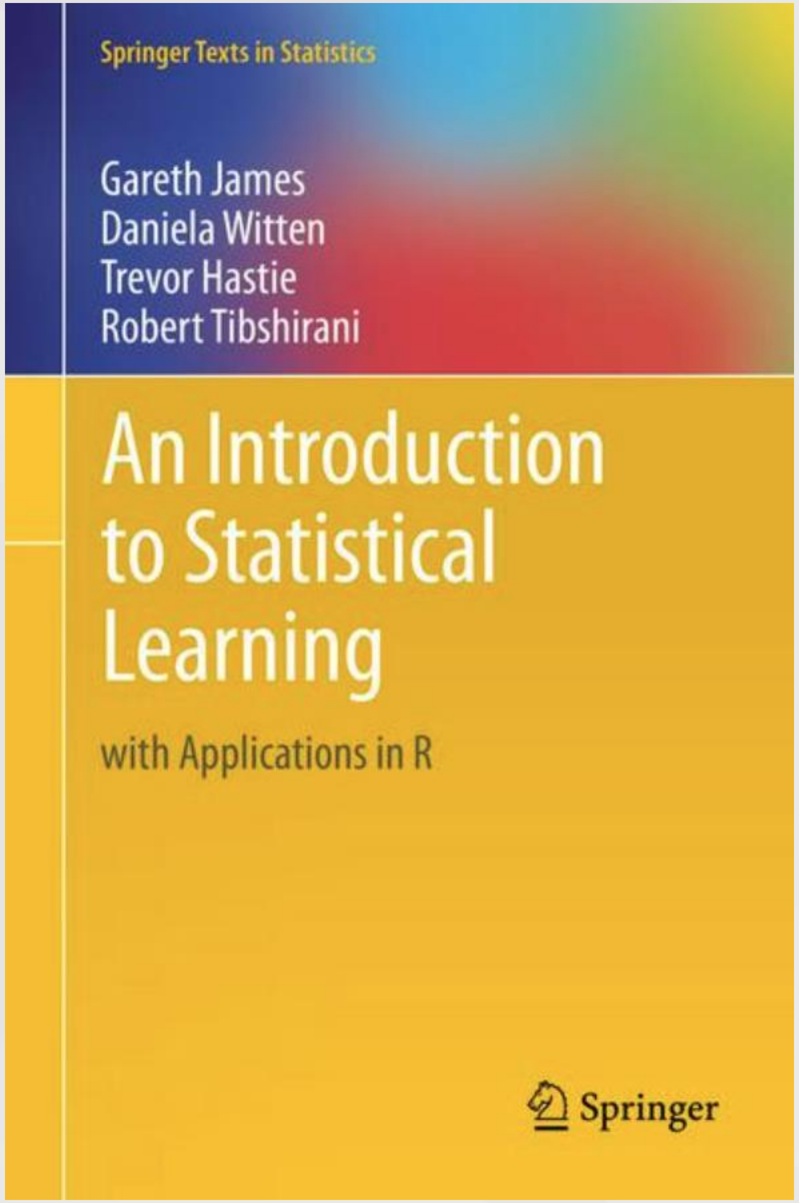
Mentor: Charlie Wolock

SPA DRP winter 2023, University of Washington

Something I've learned

1. Went through Chapter 1, 2, 3, 4, 5, 8, and 10 in the book "An Introduction to Statistical Learning".
Learned concepts such as bias-variance tradeoff, linear regression, logistic regression, resampling methods, tree models and, neural networks

2. Applied what I've learned to a project to predict Airbnb price in New York City



Data Analysis

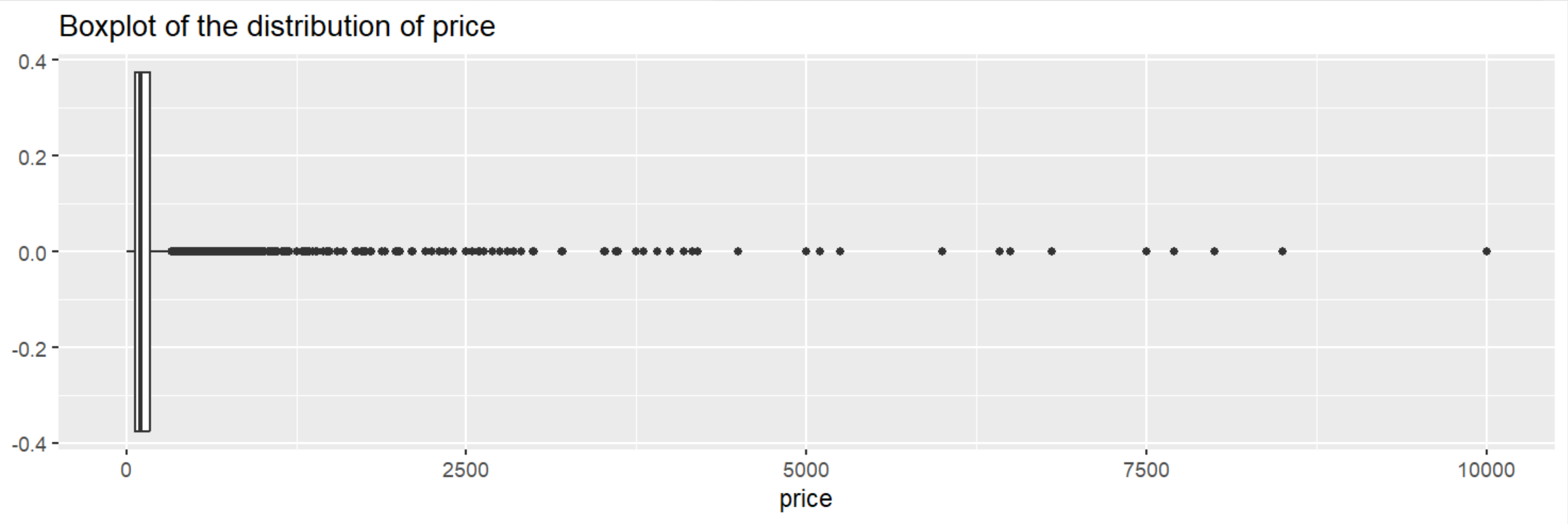
1. New York City Airbnb Open Data. This dataset contains 16 columns and 48,895 rows, documenting features for Airbnbs in NYC in 2019.

2. **Input variables** I used in the predictions are: neighbourhood_group, latitude, longitude, room_type, minimum_nights, number_of_reviews, availability_365
Predicted variable: price

Split the data into 90% training and 10% testing

```
> colnames(airbnb_org)
 [1] "id"                             "name"
 [3] "host_id"                        "host_name"
 [5] "neighbourhood_group"            "neighbourhood"
 [7] "latitude"                       "longitude"
 [9] "room_type"                      "price"
[11] "minimum_nights"                 "number_of_reviews"
[13] "last_review"                    "reviews_per_month"
[15] "calculated_host_listings_count" "availability_365"
```

Data Analysis



3 Here, I focused my project on those Airbnb with price between $0-1,000

4 I used the mean square error (MSE) of the testing data to evaluate my models.
MSE is a measure of the **average squared difference** between the predicted and actual values, thus measuring the accuracy of a regression model.

Data Analysis

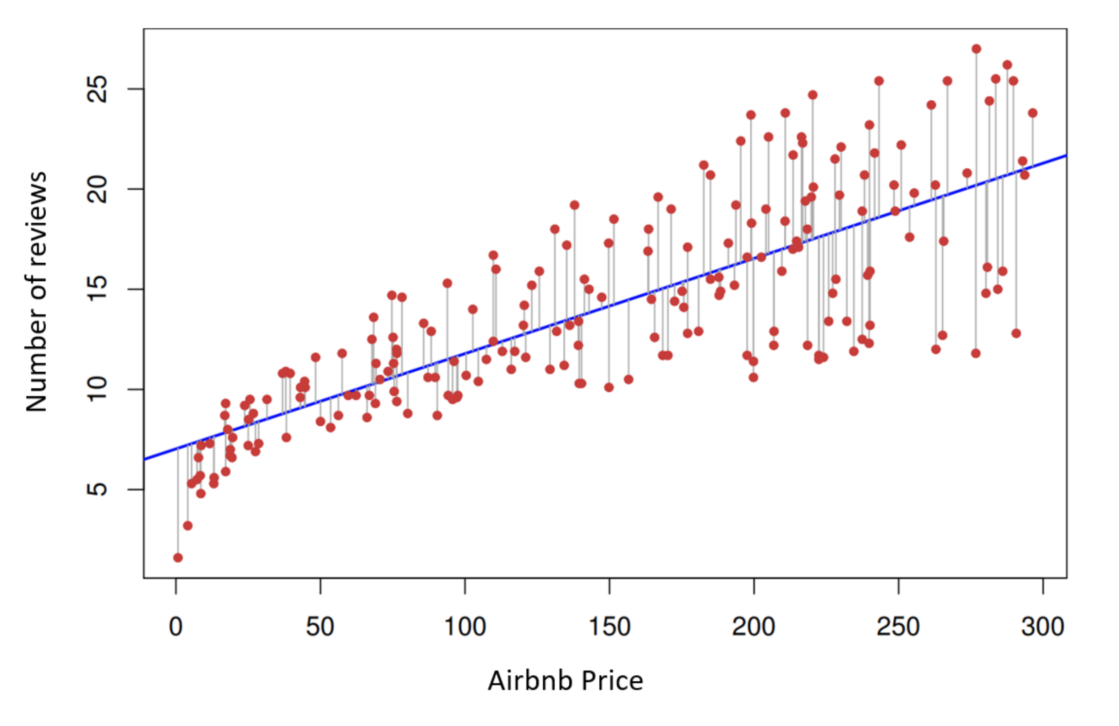
Correlation between each variable

Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

Linear regression is a simple approach to supervised learning. It assumes that the dependence of Y on X_1, X_2, ..., X_p is **linear**.

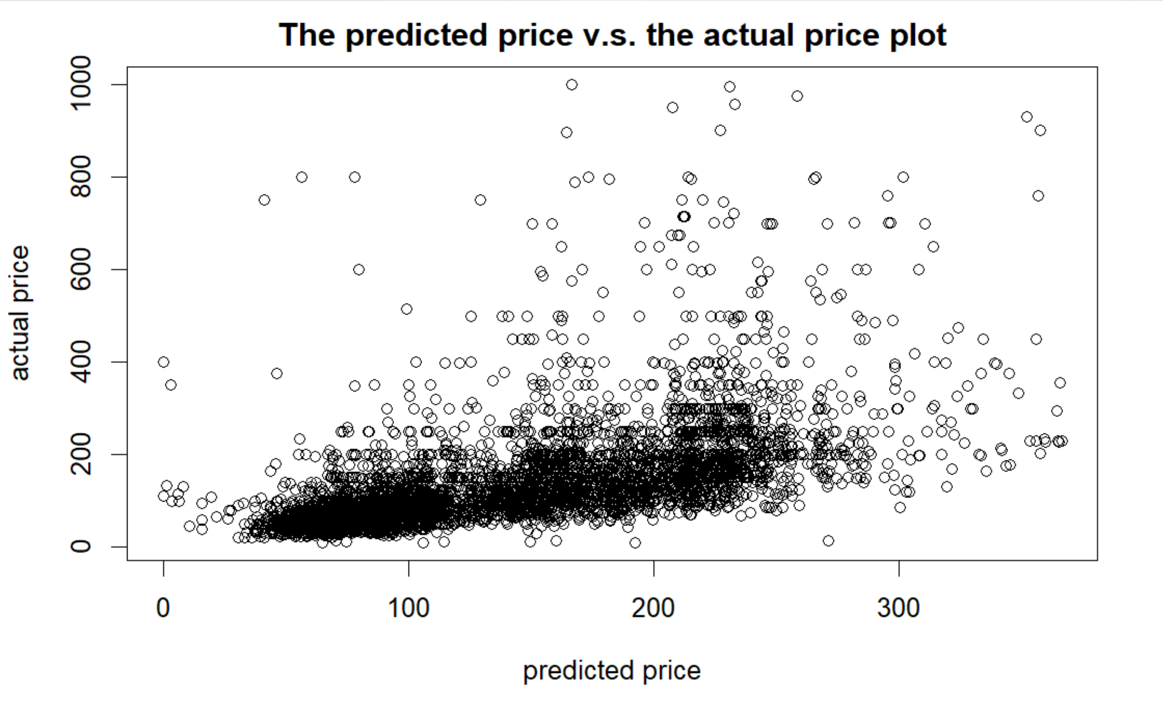
The goal of linear regression is to find the coefficients that **minimize** the squared difference between the predicted values of Y and the actual values.

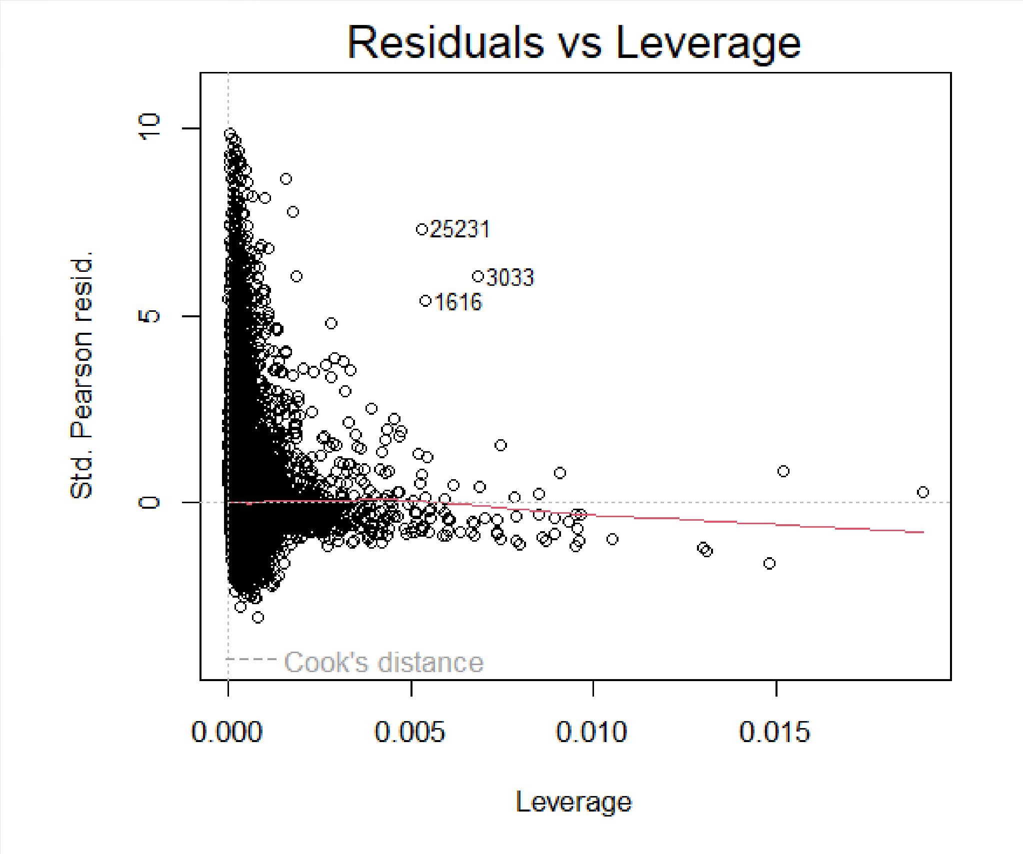


Linear Regression

Simple linear regression:

Test MSE: around 8180.25





After remove the outliers and high-leverage points:

Test MSE: around 8179.18

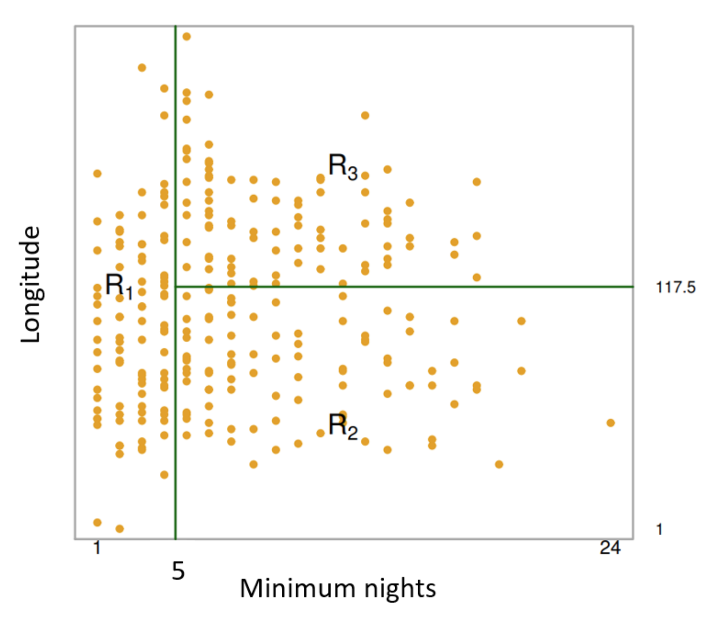
Decision Tree Model

The Decision Tree model is built by **recursively** splitting the data into smaller subsets based on the values of the input variables.

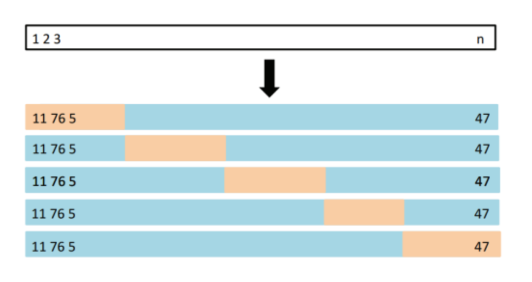
Each split corresponds to a decision on the input variables, and the resulting subsets are used to further refine the decision tree.

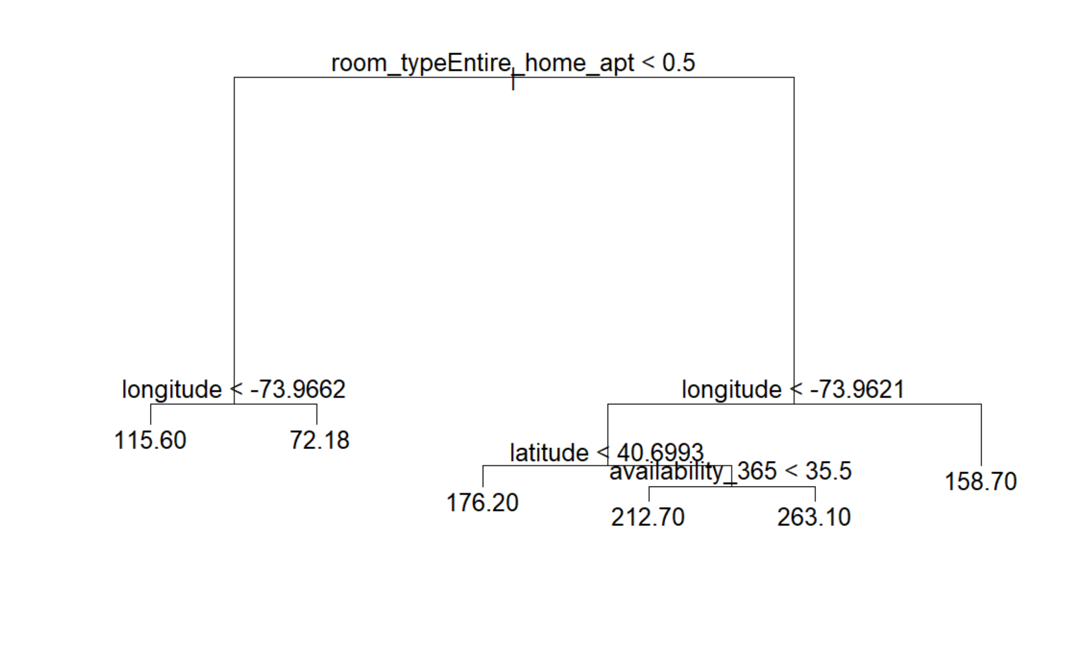
At the end of the tree, the model produces a prediction based on the values of the input variables.

It's possible for the decision tree to be overfitted. Thus, we need to **prune** the tree using K-fold cross-validation



5-fold cross-validation





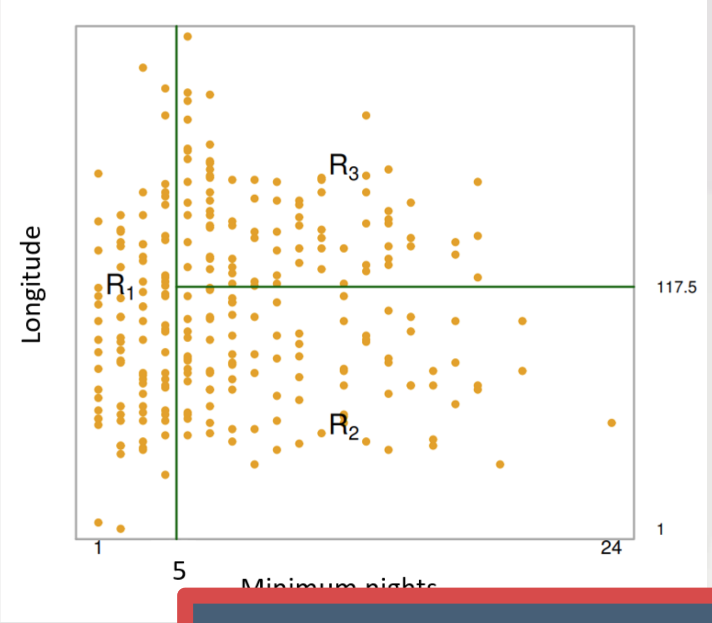
Decision Tree Model

The Decision Tree model is built by **recursively** splitting the data into smaller subsets based on the values of the input variables.

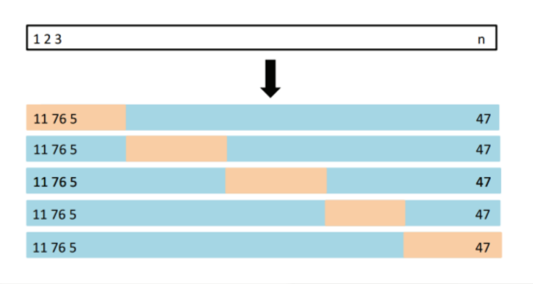
Each split corresponds to a decision on the input variables, and the resulting subsets are used to further refine the decision tree.

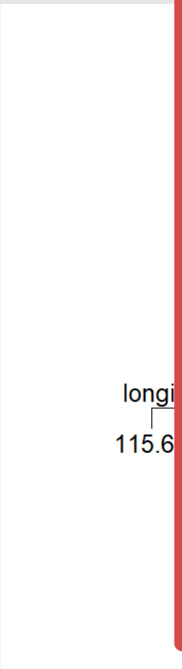
At the end of the tree, the model produces a prediction based on the values of the input variables.

It's possible for the decision tree to be overfitted. Thus, we need to **prune** the tree using K-fold cross-validation



5-fold cross-validation



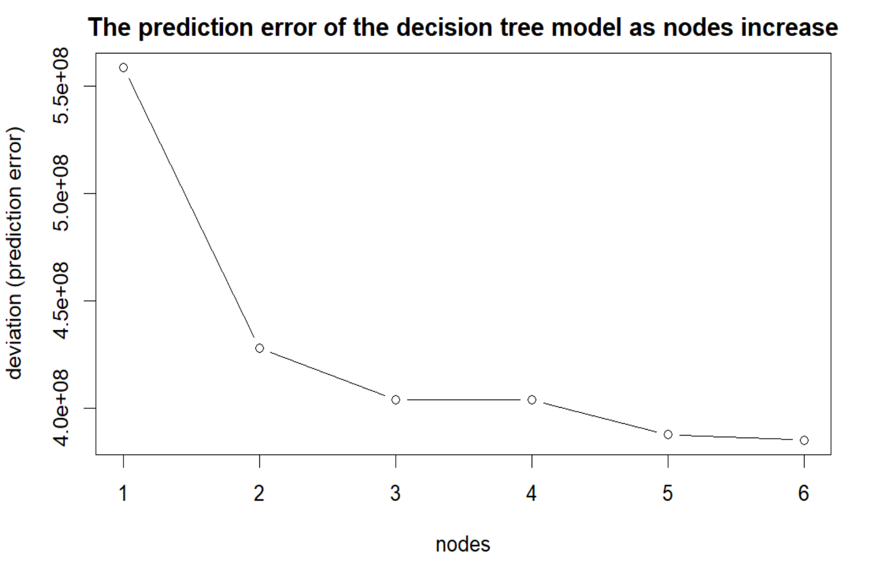


K-fold cross-validation involves dividing the data into k equal-sized subsets, or folds, and using each fold once as a validation set and the remaining k-1 folds as the training set. We use it to evaluate model performance and avoid overfitting.

Decision Tree

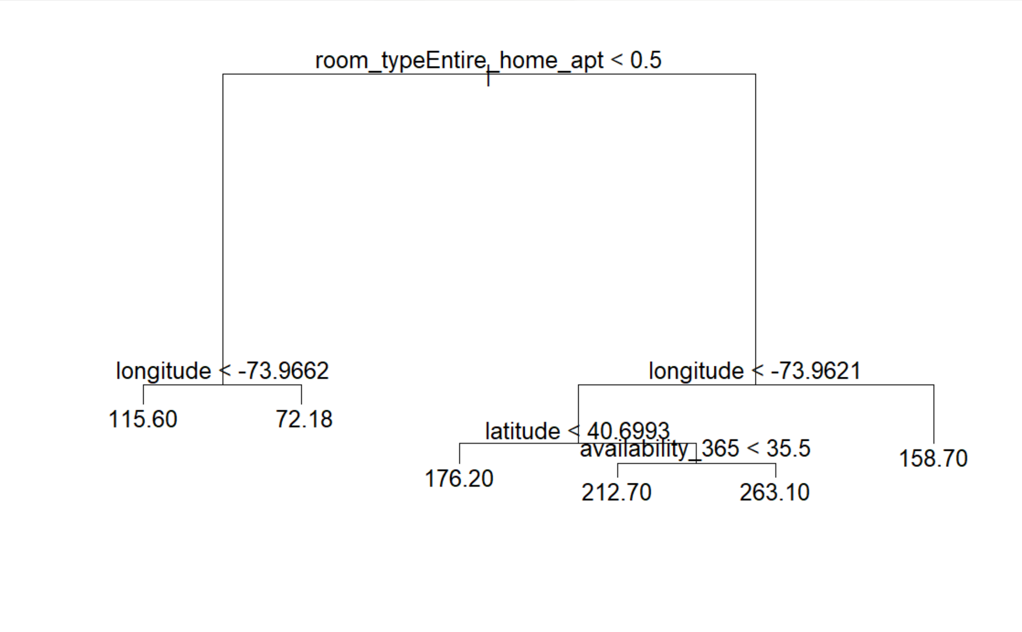
Default
Nodes: 6
Test MSE: around 8519.61

Pruned default tree
Nodes: 6
Test MSE: around 8519.61



After using the 5-fold cross-validation, I found the optimal nodes value: 72

Complex tree model
Nodes: 72
Test MSE: around 7222.99

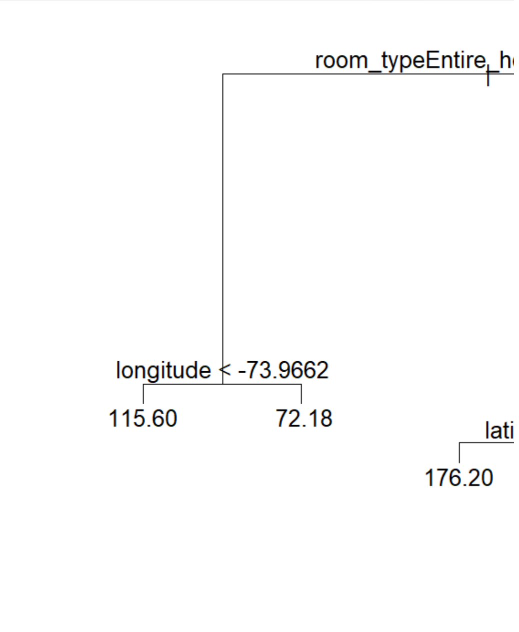


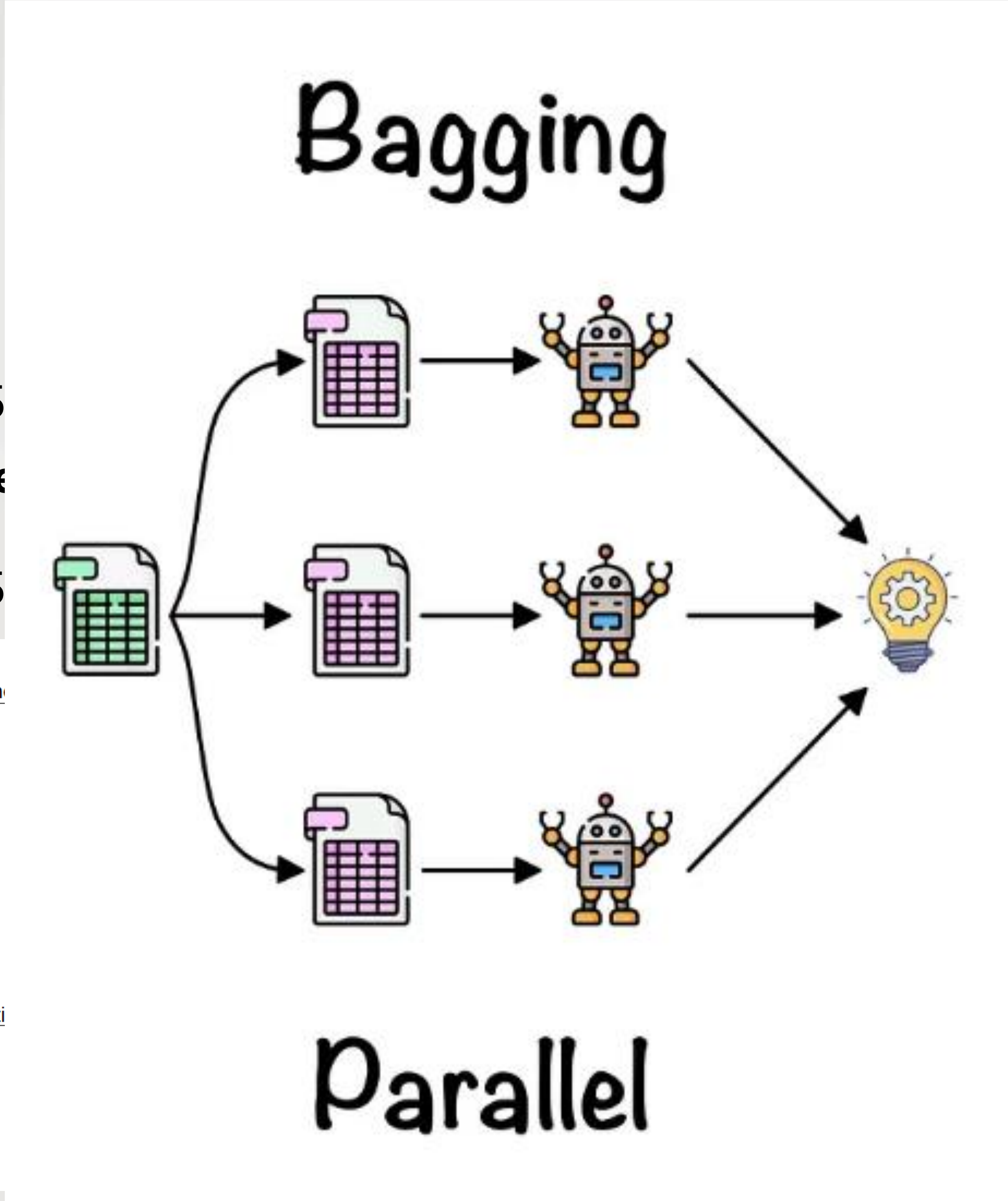
mindev <dbl>	minsize <dbl>	save <dbl>
5e-03	8	8511.453
1e-03	8	7957.647
5e-04	8	7846.225
1e-04	8	8777.368
1e-01	10	9784.506
1e-02	10	8763.865
5e-03	10	8511.453
1e-03	10	7963.344
5e-04	10	7780.320
1e-04	10	8525.603

Decision T

Default
Nodes: 6
Test MSE: around 85

Pruned default tree
Nodes: 6
Test MSE: around 85





fter using the 5-fold cross-
alidation, I found the
ptimal nodes value: 72

Complex tree model
Nodes: 72
est MSE: around 7222.99

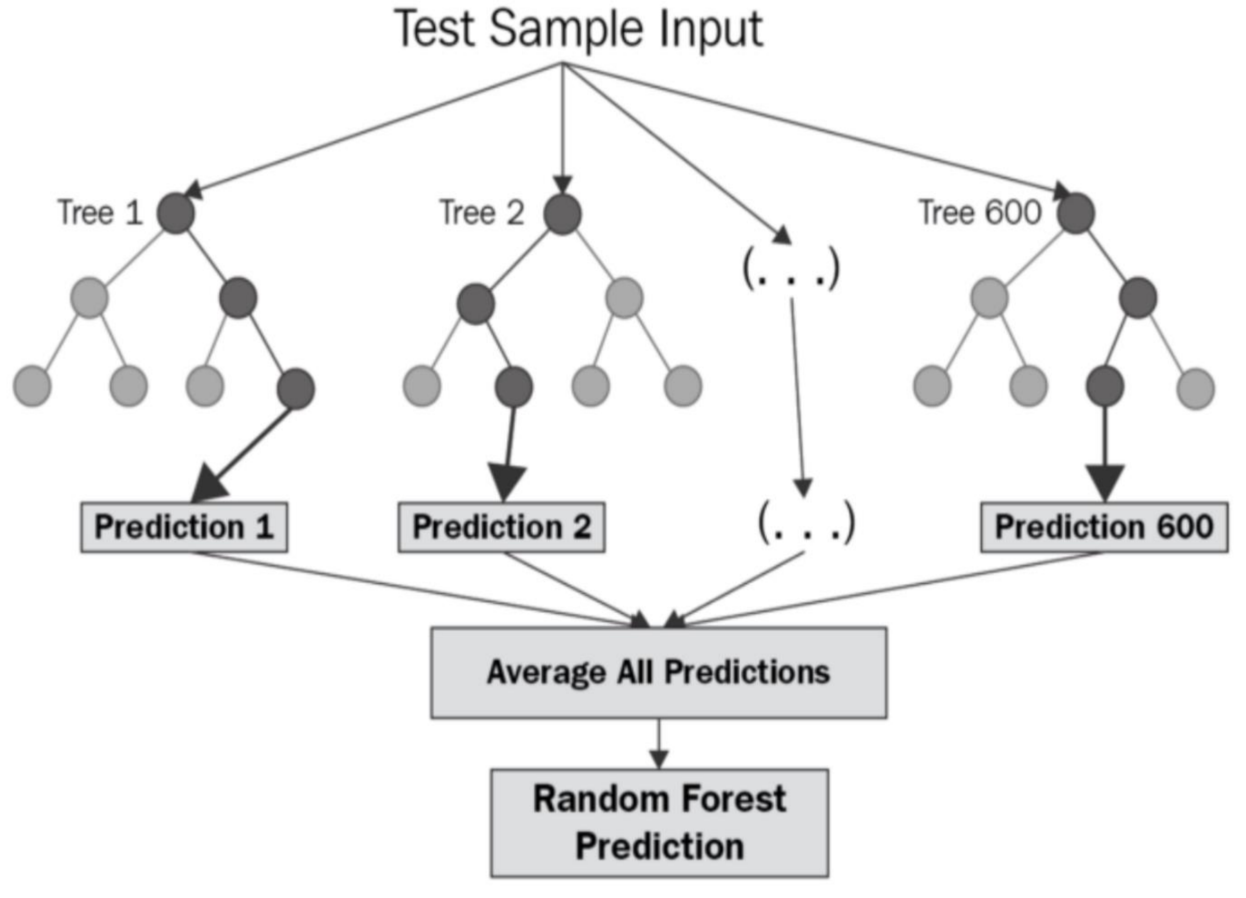
insize <dbl>	save <dbl>
8	8511.453
8	7957.647
8	7846.225
8	8777.368
10	9784.506
10	8763.865
10	8511.453
10	7963.344
10	7780.320
10	8525.603

Random Forest

The basic idea behind a random forest is to combine multiple decision trees into a single model that can make more accurate predictions than any individual tree.

Each decision tree in the random forest is trained on **a random subset of the data** and **a random subset of the input variables**. This randomness helps to reduce overfitting and makes the model more robust to noise and outliers in the data.

The final prediction of the random forest is based on the average prediction of all the trees in the forest, weighted by their individual performances.



Here, we don't care about overfitting in the trees, but we do care about two things – the number of trees (Num.trees) and the number of predictors that we randomly select for each tree (mtry)

Random Forest

Default
Num.trees = 500
mtry = 4
Test MSE: around 6687.04

After out-of-bag
Num.trees = 2500
mtry = 6
Test MSE: around 6452.34

Description: df [25 x 3]

ntree <dbl>	mtry <dbl>	save <dbl>
500	5	6964.842
1000	5	6947.934
1500	5	6947.823
2000	5	6944.713
2500	5	6940.144
500	6	6966.729
1000	6	6956.601
1500	6	6946.466
2000	6	6946.601
2500	6	6943.243

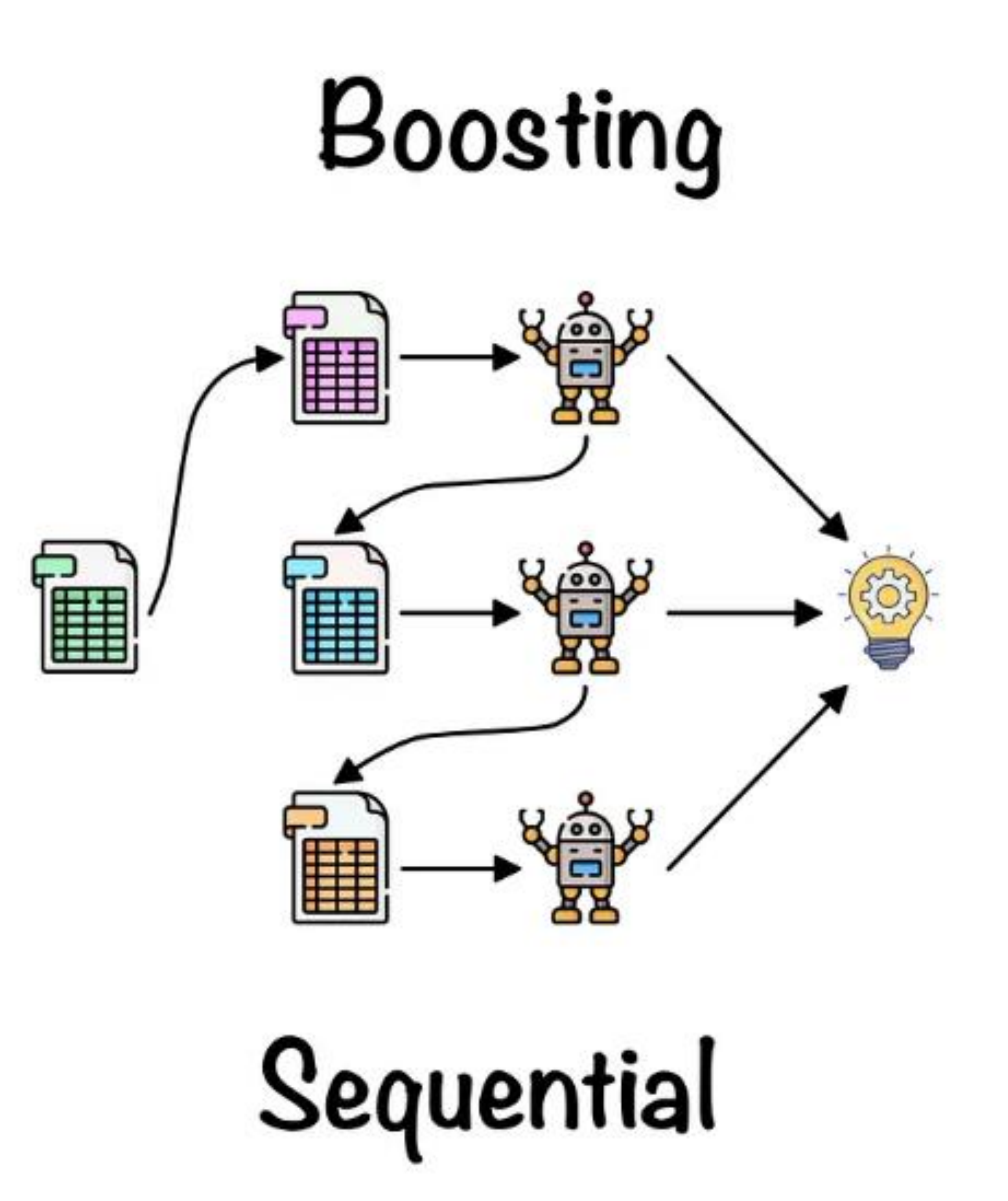
11-20 of 25 rows

Out-of-bag error: a biproduct of bagging. Because every time we use a subset of data to train the model, we can use the rest of the data as validation and test the model accuracy.

Random Fo

Default
Num.trees = 500
mtry = 4
Test MSE: around 668

After out-of-bag
Num.trees = 2500
mtry = 6
Test MSE: around 645



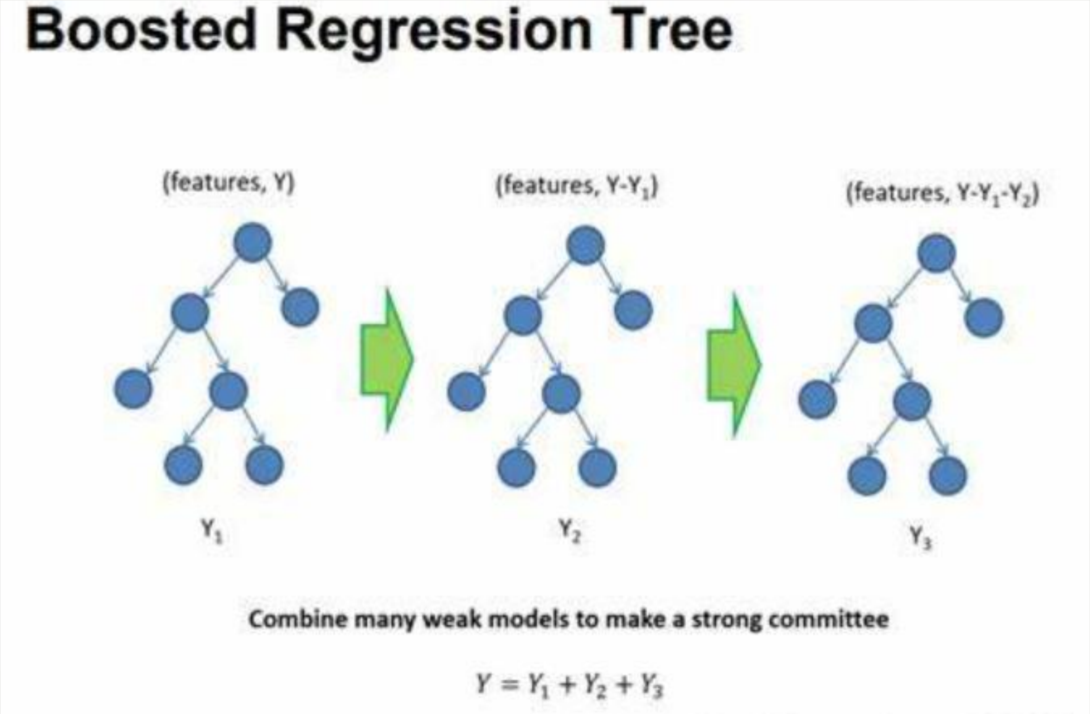
save <dbl>
6964.842
6947.934
6947.823
6944.713
6940.144
6966.729
6956.601
6946.466
6946.601
6943.243

GBoost

In boosting, each tree is grown using information from pervious tree.

The algorithm works by **iteratively** adding new trees to the model, with each tree attempting to correct the errors of the previous trees. Specifically, GBM works by minimizing the loss function of the model through gradient descent.

Here, we also want to **tune the parameters** for boosting. The three of the most important parameters in gradient boost model are the number of trees (n.trees), the shrinkage parameter (shrinkage), and the interaction depth (interaction.depth).



Shrinkage parameter: the learning rate

Interaction depth: controls the interaction order of the boosted model

GBoost

Default
n.trees = 5000
interaction.depth = 4
shrinkage = 0.01
Test MSE: around 6850.88

After 5-fold cross-validation
n.trees = 3000
interaction.depth = 4
shrinkage = 0.05
Test MSE: around 6823.86

Description: df [54 x 4]

n_trees <dbl>	interaction <dbl>	shrinkage <dbl>	save <dbl>
5000	1	0.005	7784.900
1000	2	0.005	7901.848
3000	2	0.005	7513.295
5000	2	0.005	7440.054
1000	4	0.005	7459.886
3000	4	0.005	7222.695
5000	4	0.005	7227.873
1000	8	0.005	7352.441
3000	8	0.005	7301.823
5000	8	0.005	7467.749

21-30 of 54 rows Previous 1 2 3 4 5

Model Comparison

Best Linear Regression Model:
Test MSE: around 8179.18

Best Decision Tree Model:
Test MSE: around 7222.99

Best Random Forest Model:
Test MSE: around 6452.34

Best Linear Regression Model:
Test MSE: around 6823.86

price <int>	random_forest_pred <dbl>
65	71.54094
89	139.10448
80	139.47164
199	249.05285
140	68.84643
99	154.58445
135	136.00069
65	94.24215
135	134.43797
195	195.40612

Thank you for watching!

Mentee: Liuyixin Shao

Mentor: Charlie Wolock

SPA DRP winter 2023, University of Washington

References

Book:
https://link.springer.com/book/10.1007/978-1-4614-7138-7?view=modern

The NYC Airbnb dataset that I used:
The dataset on: https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data
The original source can be found on: http://insideairbnb.com/

Images:
https://towardsai.net/p/machine-learning/why-choose-random-forest-and-not-decision-trees

https://blog.csdn.net/w576233728/article/details/80231601

https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422