AI VIETNAM
All-in-One Course
(TA Session)

Decision Tree and Random Forest

Exercise

AI VIET NAM
@aivietnam.edu.vn

Dinh-Thang Duong – TA

Year 2023

Outline

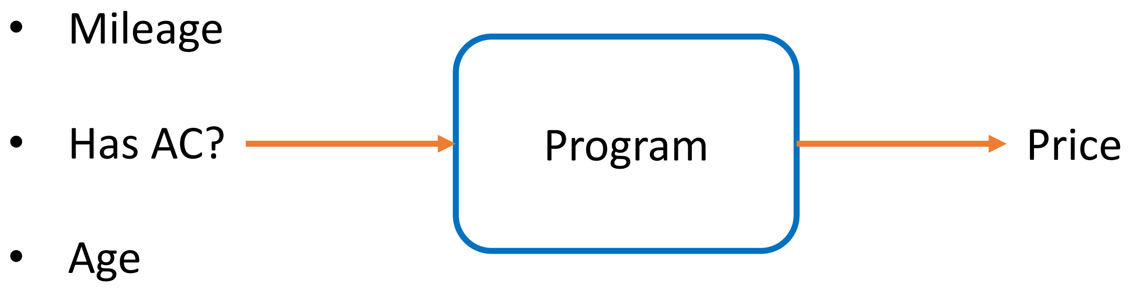
- Review
- Code Implementation
- Question

Review

❖ Getting Started

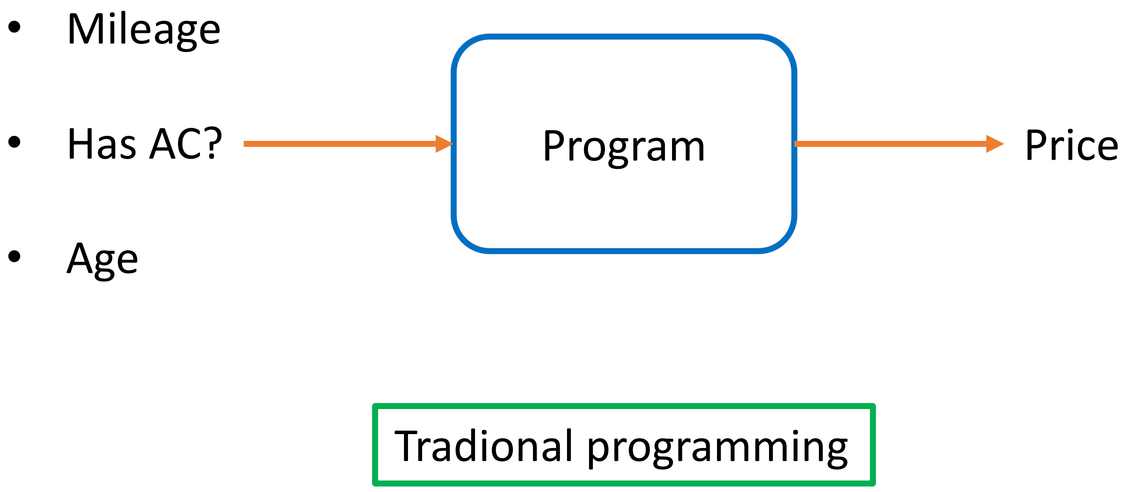
Mileage	Has AC?	Age	Price
7.5	yes	3	8.5
6.0	no	2	9.2
9.0	yes	4	7.8
4.5	yes	1	10.0
6.8	no	3	8.9
8.0	yes	2	8.3
5.5	no	2	9.5

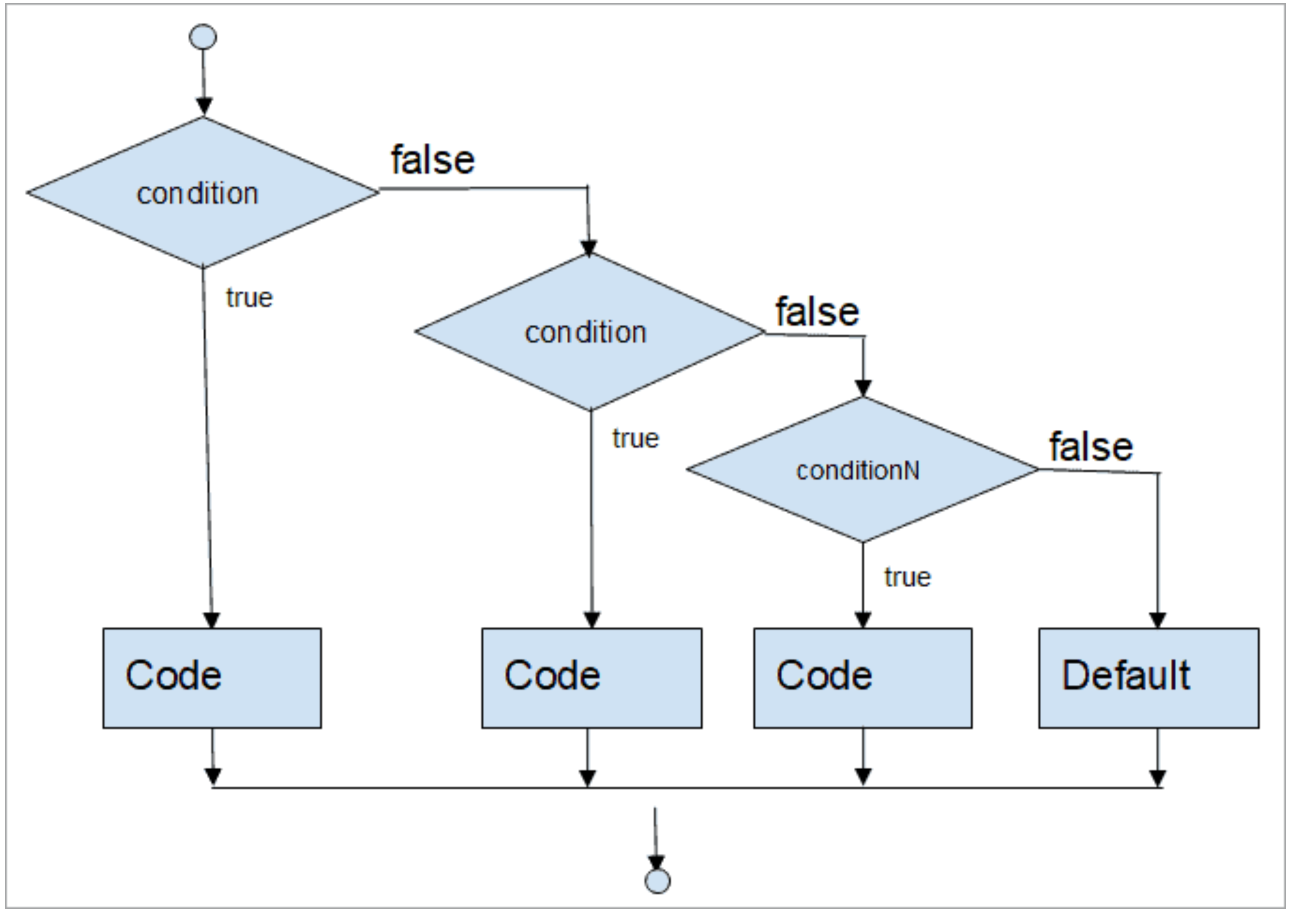
Car Price Prediction



Review

❖ Getting Started





Review

❖ Getting Started

To utilize exisiting dataset for this problem, we could use **Machine Learning**.

Mileage	Has AC?	Age	Price
7.5	yes	3	8.5
6.0	no	2	9.2
9.0	yes	4	7.8
4.5	yes	1	10.0
6.8	no	3	8.9
8.0	yes	2	8.3
5.5	no	2	9.5

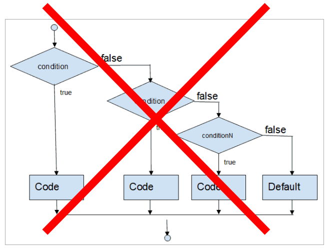
Features (Mileage, Has AC?, Age) — Target (Price)

Training

Machine Learning Model

- Mileage
- Has AC?
- Age

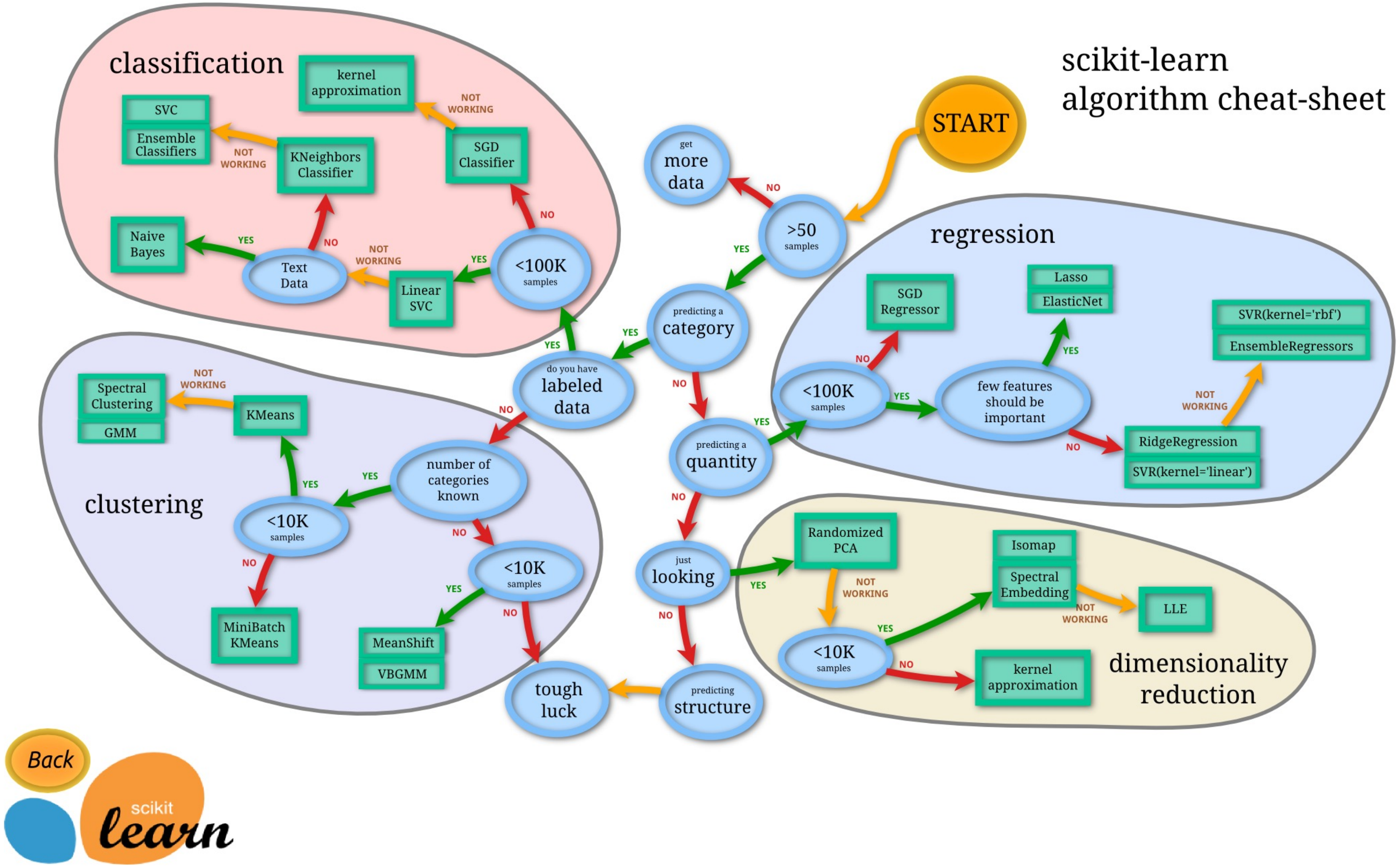
Price



Machine Learning **learns** to map a set of features X to target value y based on existing dataset.

Review

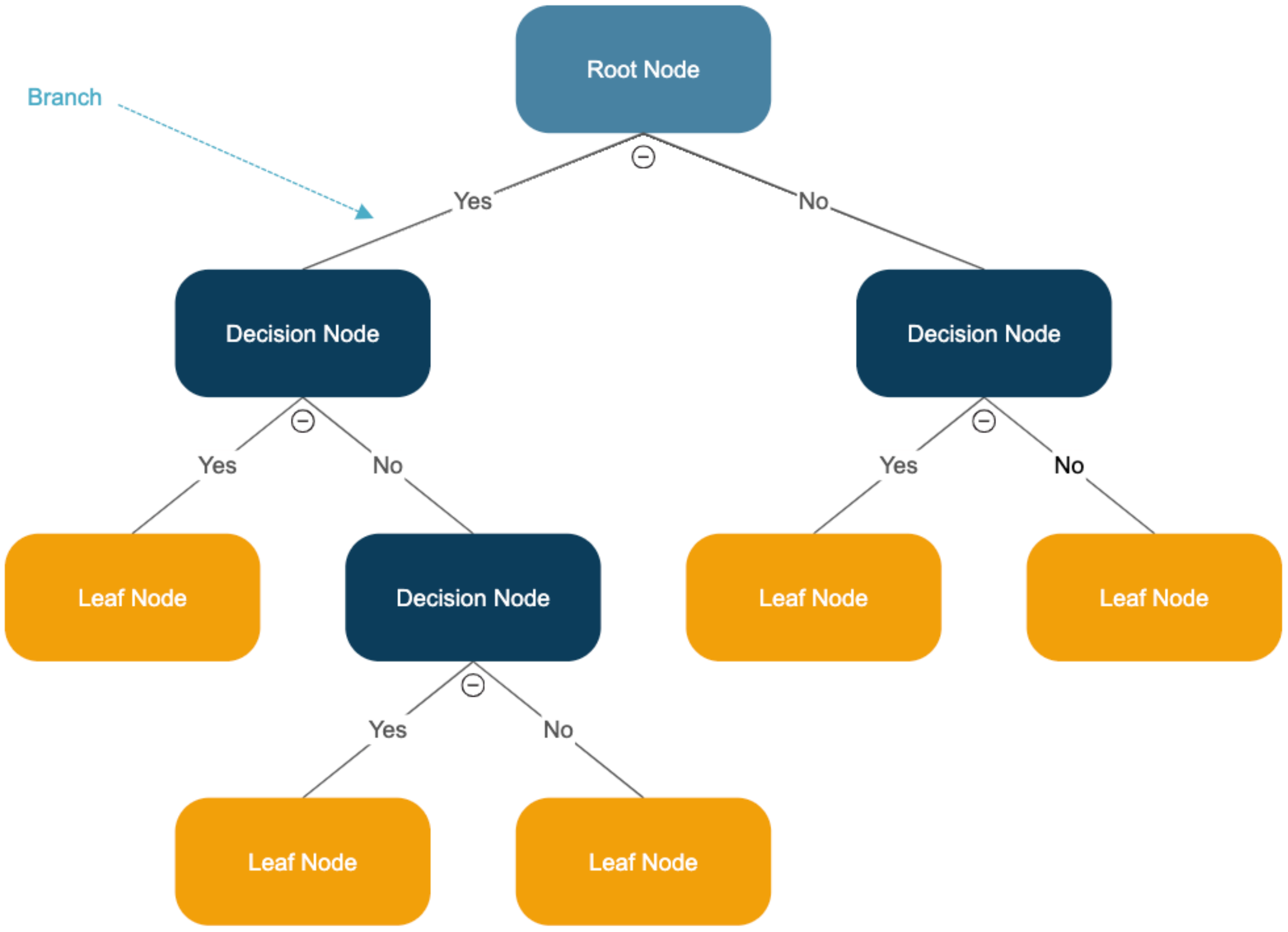
❖ Getting Started



https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Review

❖ Decision Tree

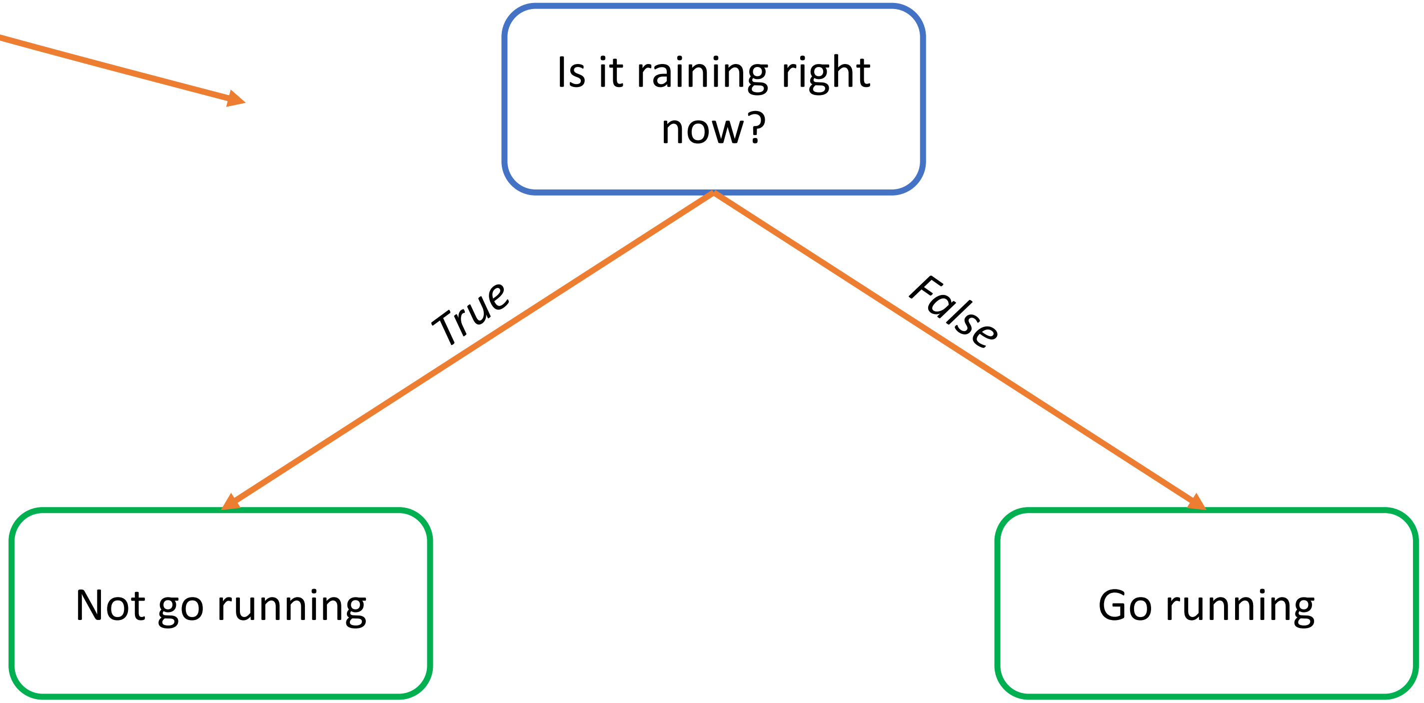


Decision Tree: A supervised-learning machine learning algorithm that build a tree-based structure. It can perform both classification and regression tasks.

Review

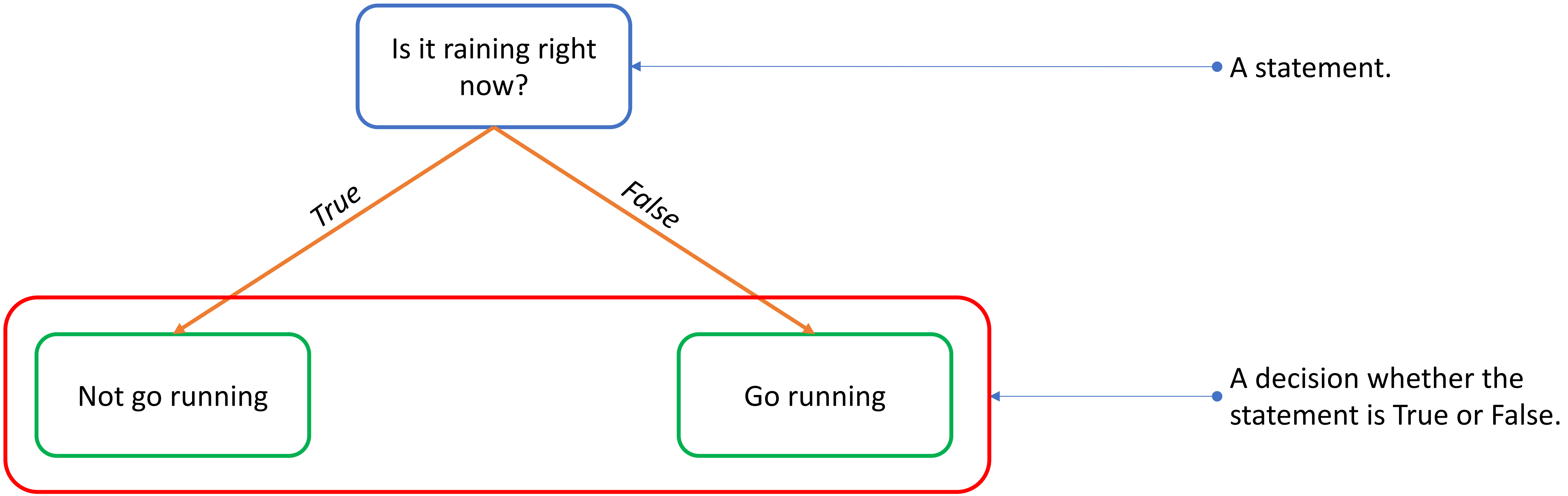
❖ Decision Tree Terminologies

Statement: I will go running if it is not raining outside.



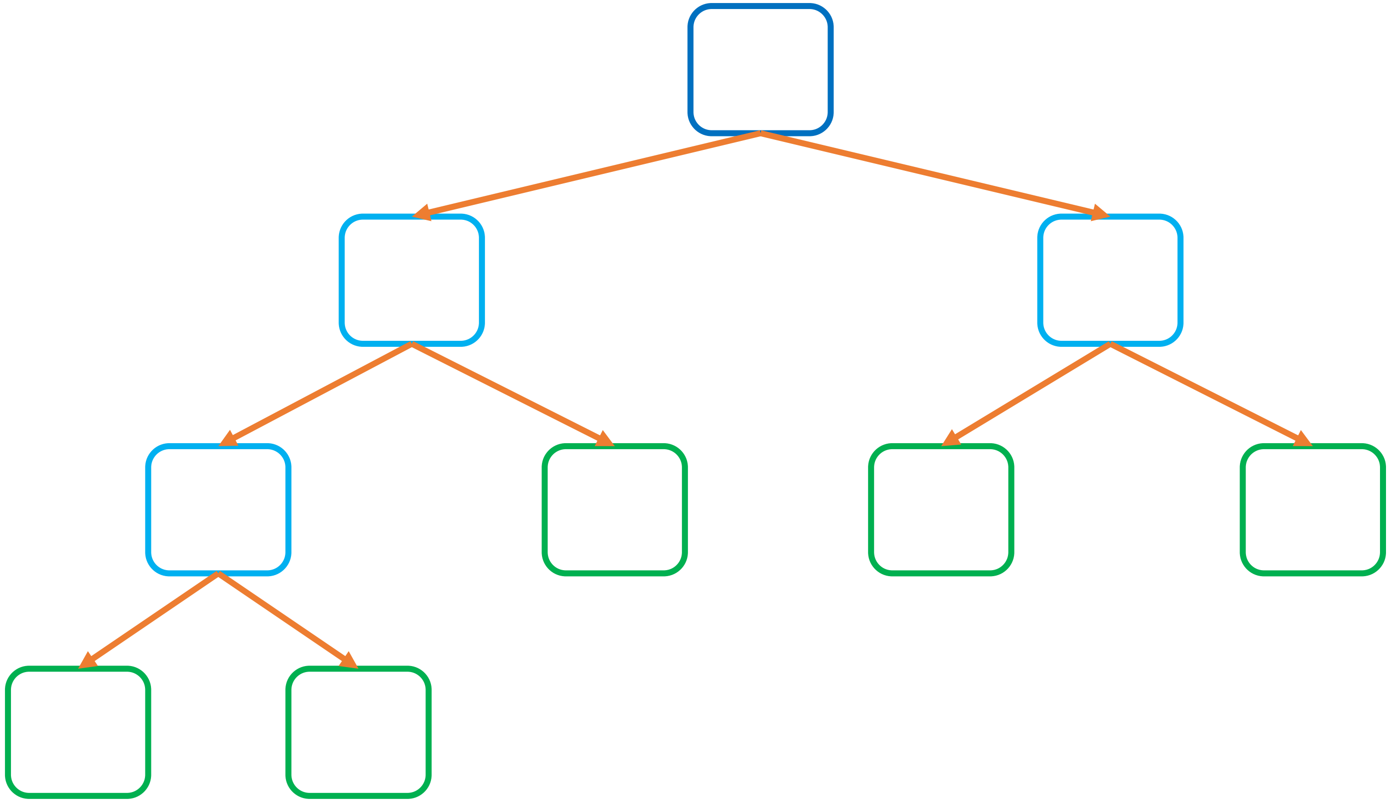
Review

❖ Decision Tree Terminologies



Review

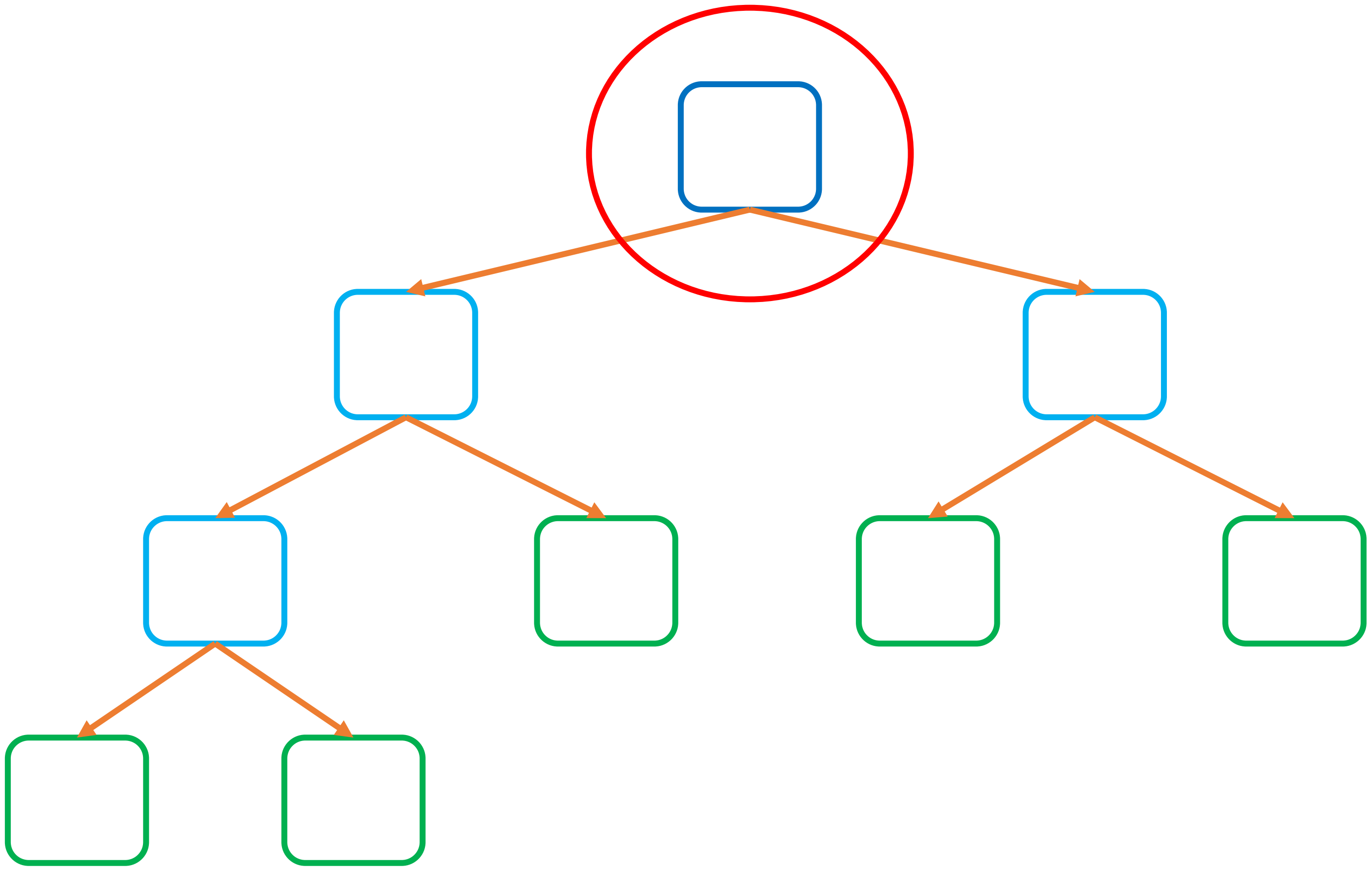
❖ Decision Tree Terminologies



A general Decsion Tree may contain many conditions and outcomes.

Review

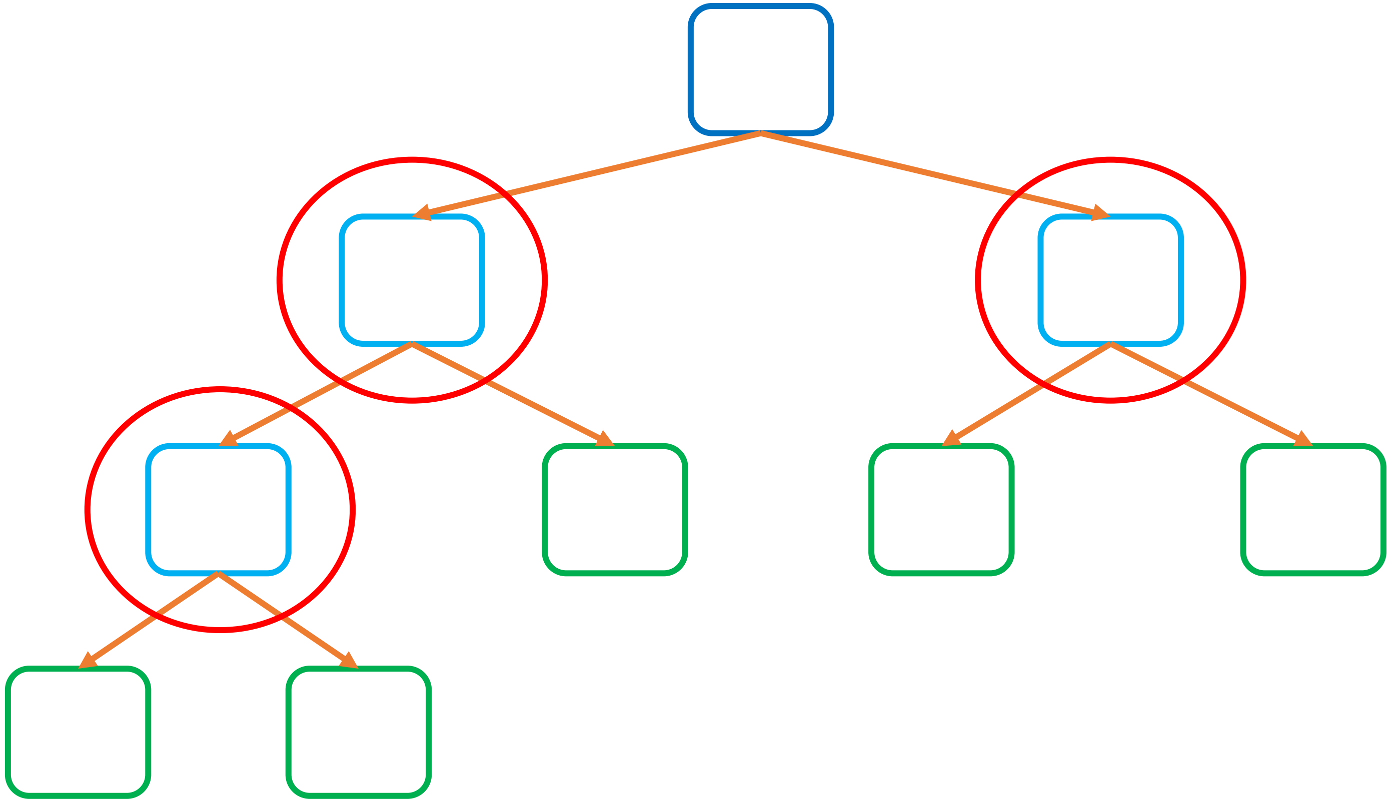
❖ Decision Tree Terminologies



Root Node: The initial condition (the first split) of the tree.

Review

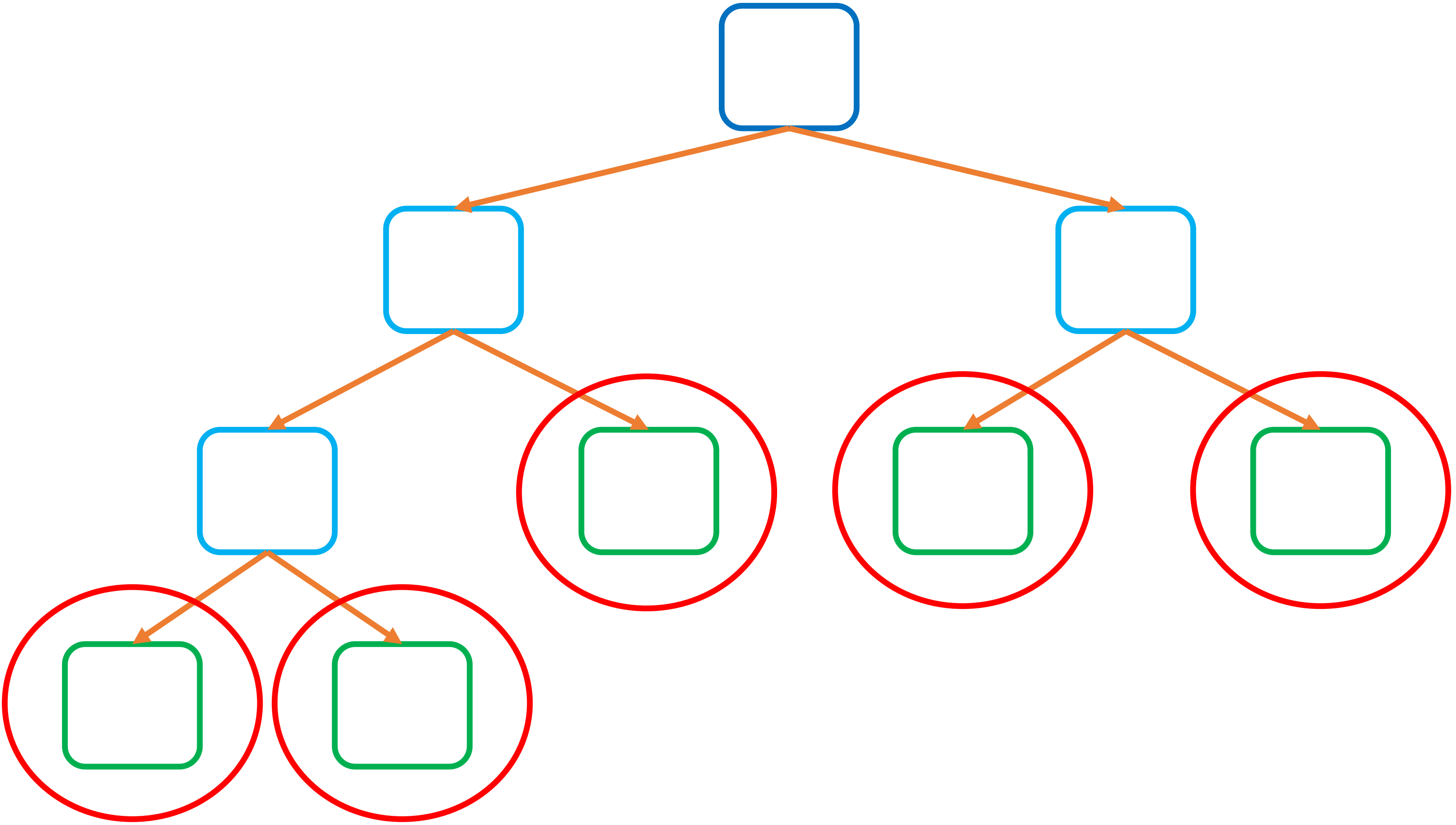
❖ Decision Tree Terminologies



Internal Nodes (Branch Nodes): The conditions within the tree that receive inputs from previous node and produce output to new nodes.

Review

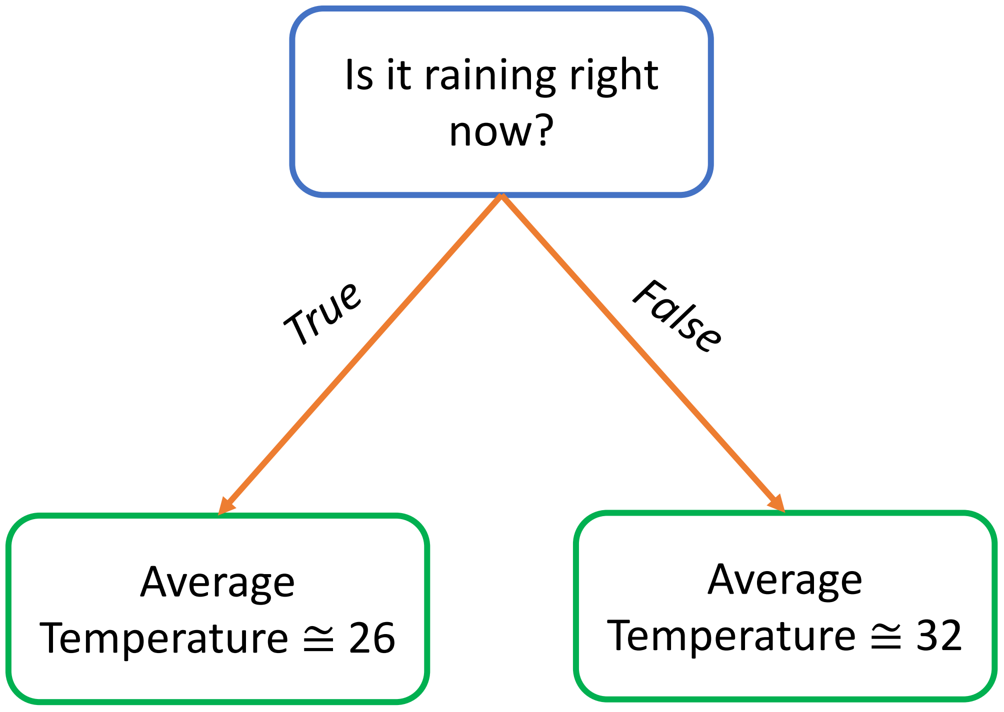
❖ Decision Tree Terminologies



Leaf Nodes (Terminal Nodes): The final decision of the tree. It does not make any further splits.

Review

❖ Decision Tree Terminologies



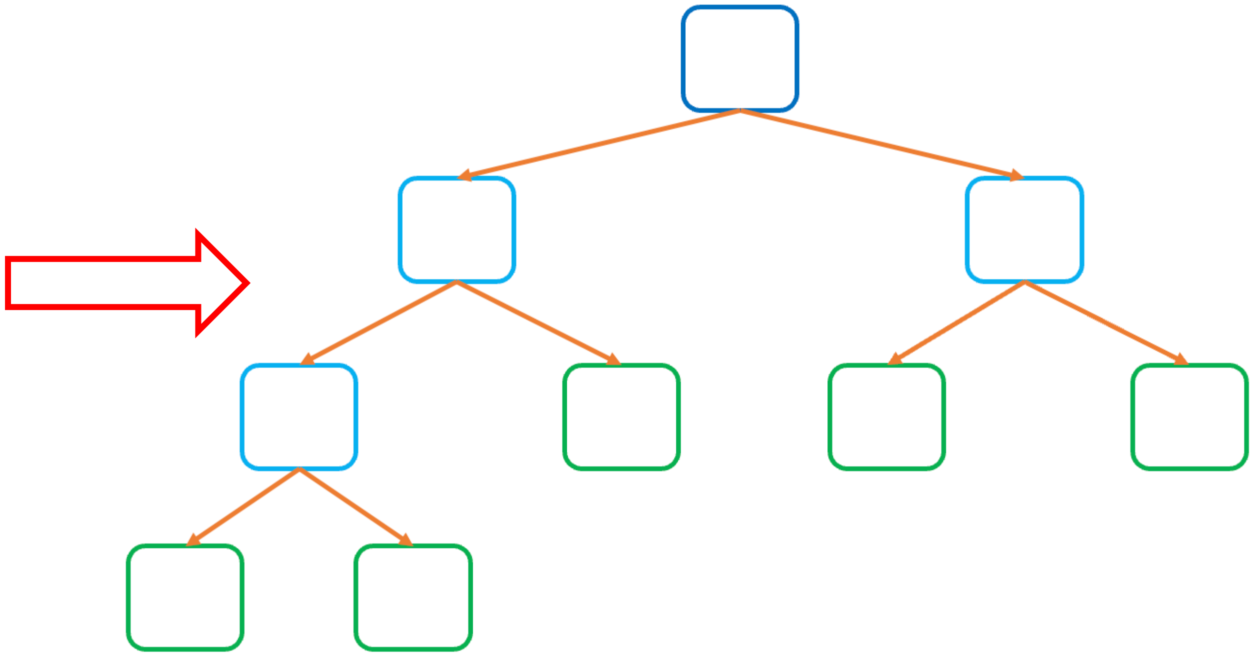
Classification Tree: Output of leaf nodes are categorical.

Regression Tree: Output of leaf nodes are numerical.

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
7.5	yes	3	8.5
6.0	no	2	9.2
9.0	yes	4	7.8
4.5	yes	1	10.0
6.8	no	3	8.9
8.0	yes	2	8.3
5.5	no	2	9.5



How to build a tree from a dataset?

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
7.5	yes	3	8.5
6.0	no	2	9.2
9.0	yes	4	7.8
4.5	yes	1	10.0
6.8	no	3	8.9
8.0	yes	2	8.3
5.5	no	2	9.5

Consider Mileage feature.

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

Sort the dataset by Mileage in ascending order

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

Calculate the average of each adjacent pair of Mileage:

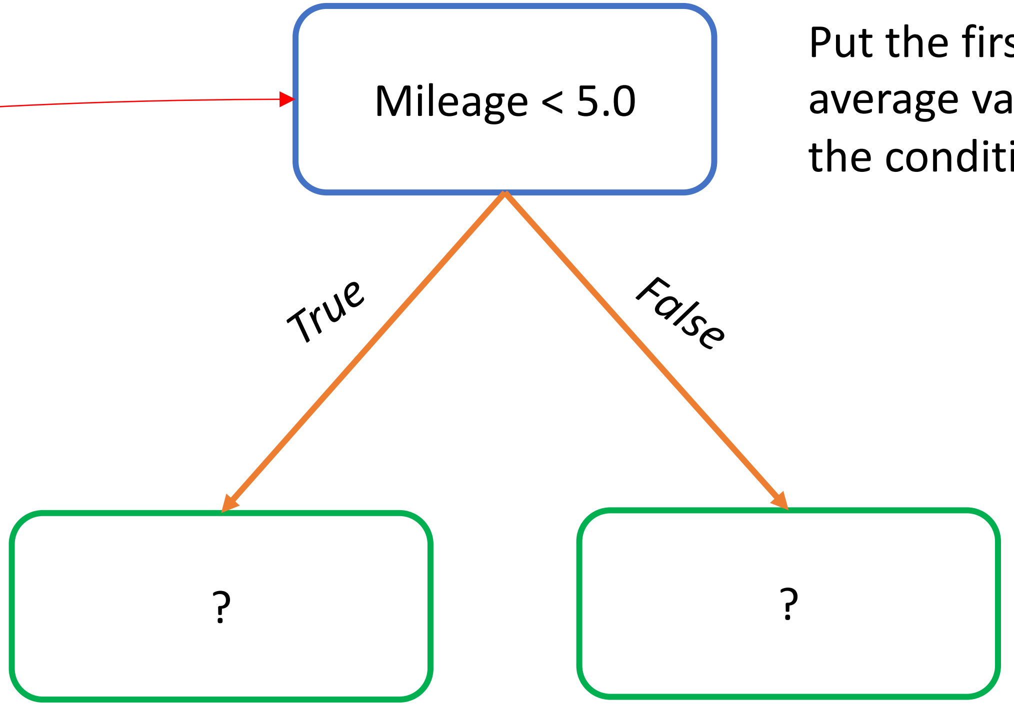
- Pair 1: (4.5 + 5.5) / 2 = 5.0
- Pair 2: (5.5 + 6.0) / 2 = 5.75
- Pair 3: (6.0 + 6.8) / 2 = 6.4
- Pair 4: (6.8 + 7.5) / 2 = 7.15
- Pair 5: (7.5 + 8.0) / 2 = 7.75
- Pair 6: (8.0 + 9.0) / 2 = 8.5

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

5.0



Put the first average value as the condition

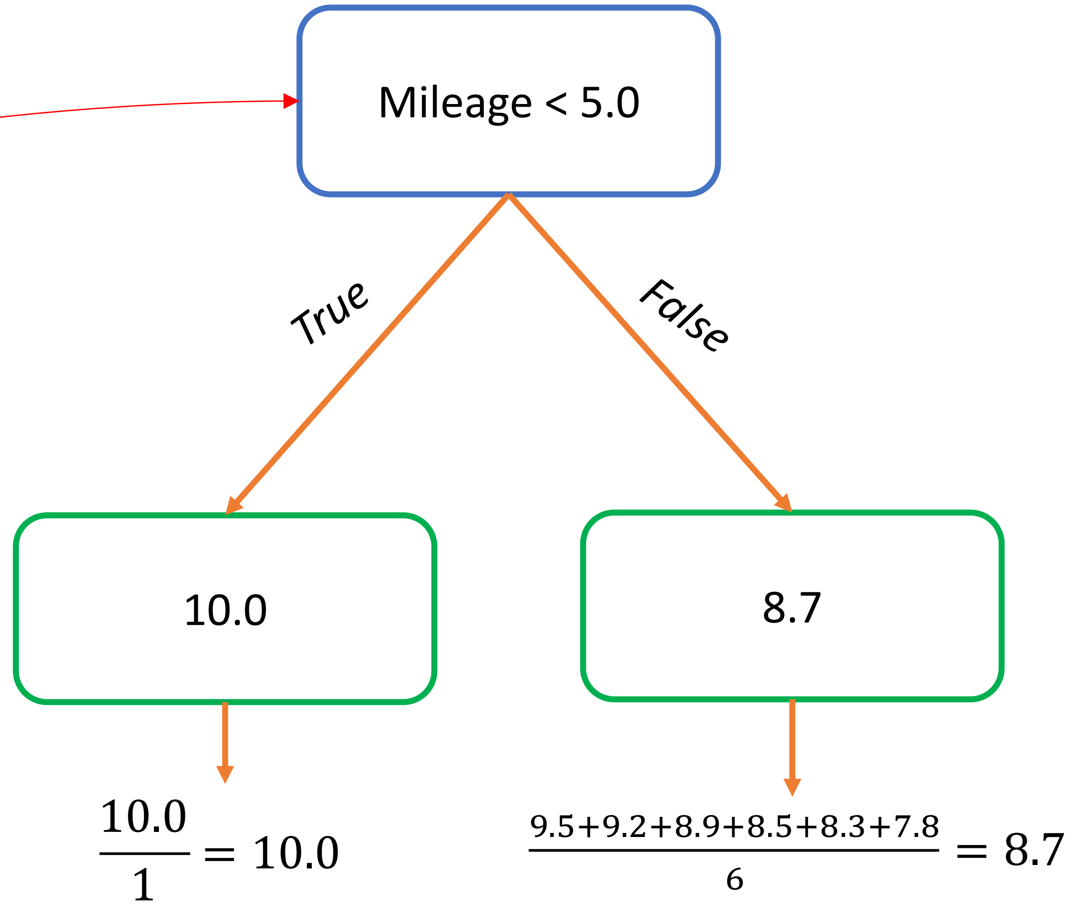
How to determine the leaf node for this condition?

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

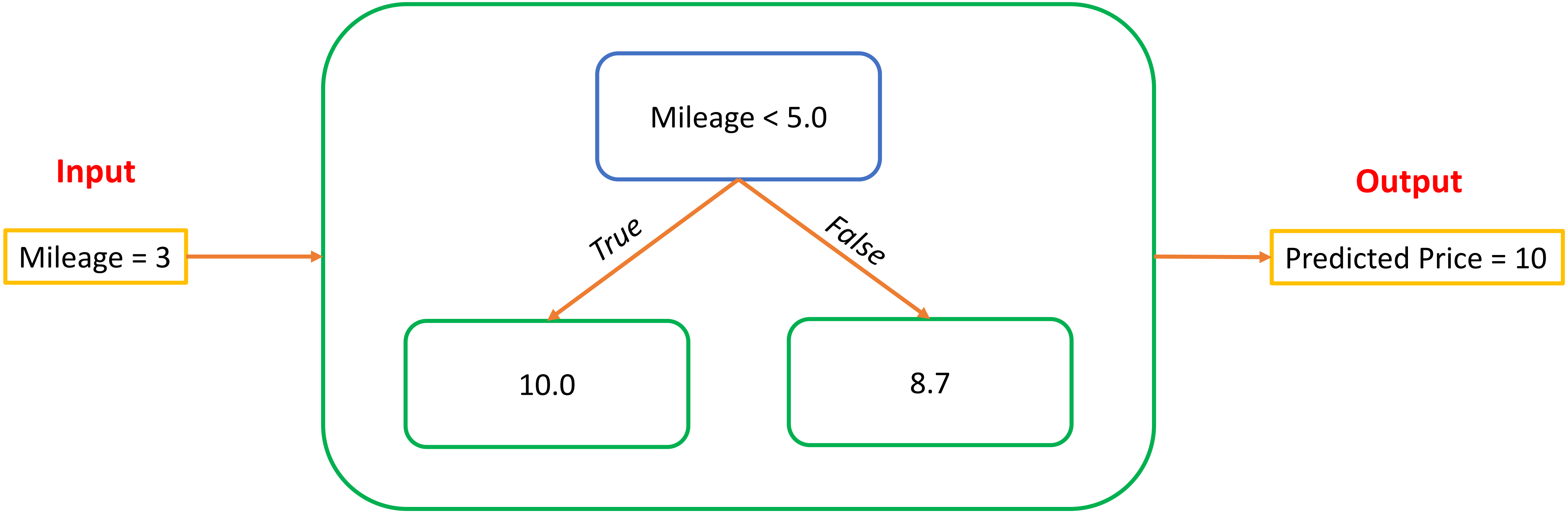
5.0



Use the average value of Price that satisfying the condition in the dataset.

Review

❖ Build a Regression Tree

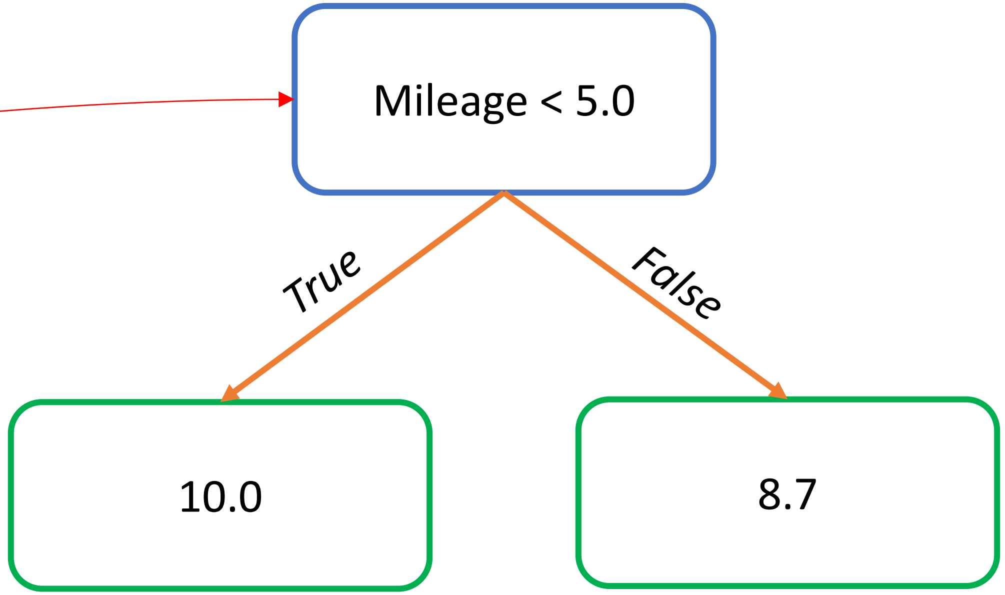


Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

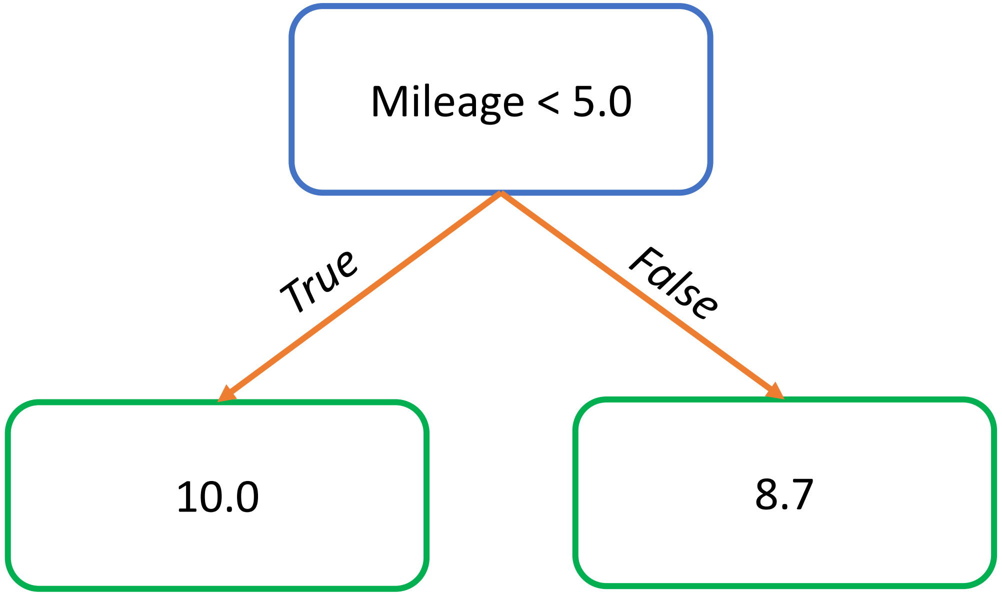
5.0

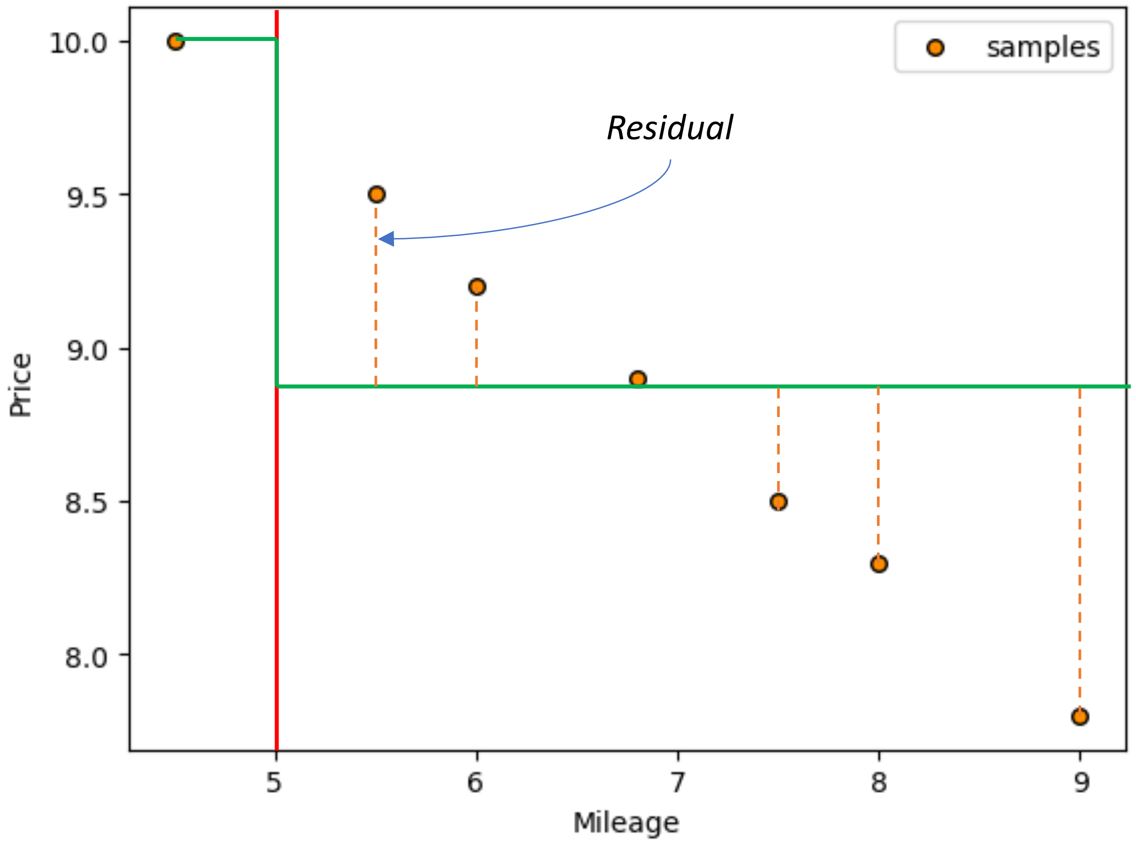


How to determine whether this tree is good enogh or not?

Review

❖ Build a Regression Tree





How to determine whether this tree is good enogh or not? (Impurity Measurement)

=> Using the prediction of the tree to evaluate on training dataset.

Compare the predicted value and the true value.

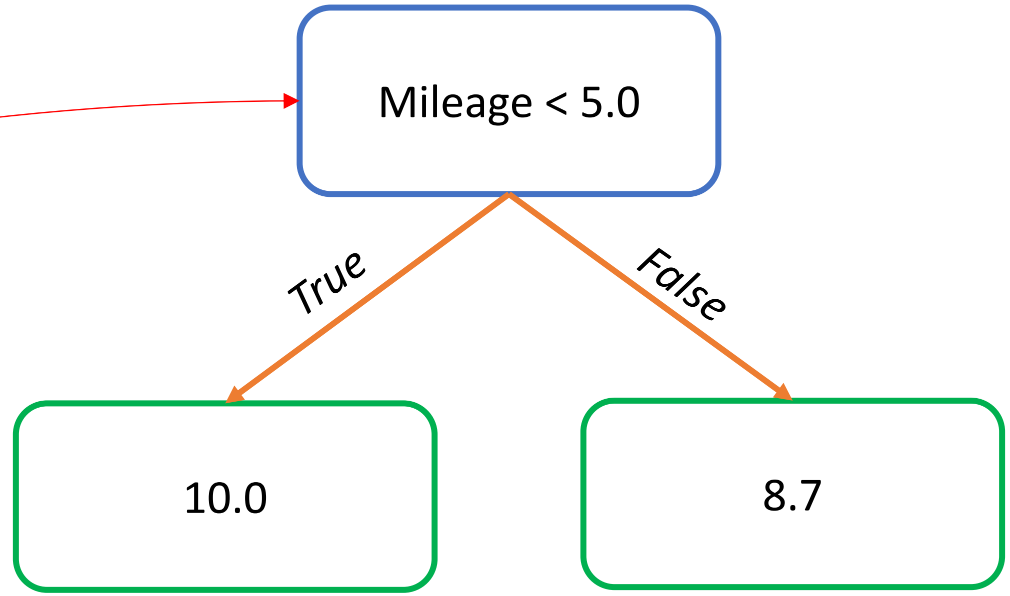
=> Residual Sum of Squares $RSS = \sum_{i=1}^{n}\left(y_i - f(x_i)\right)^2$

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

5.0

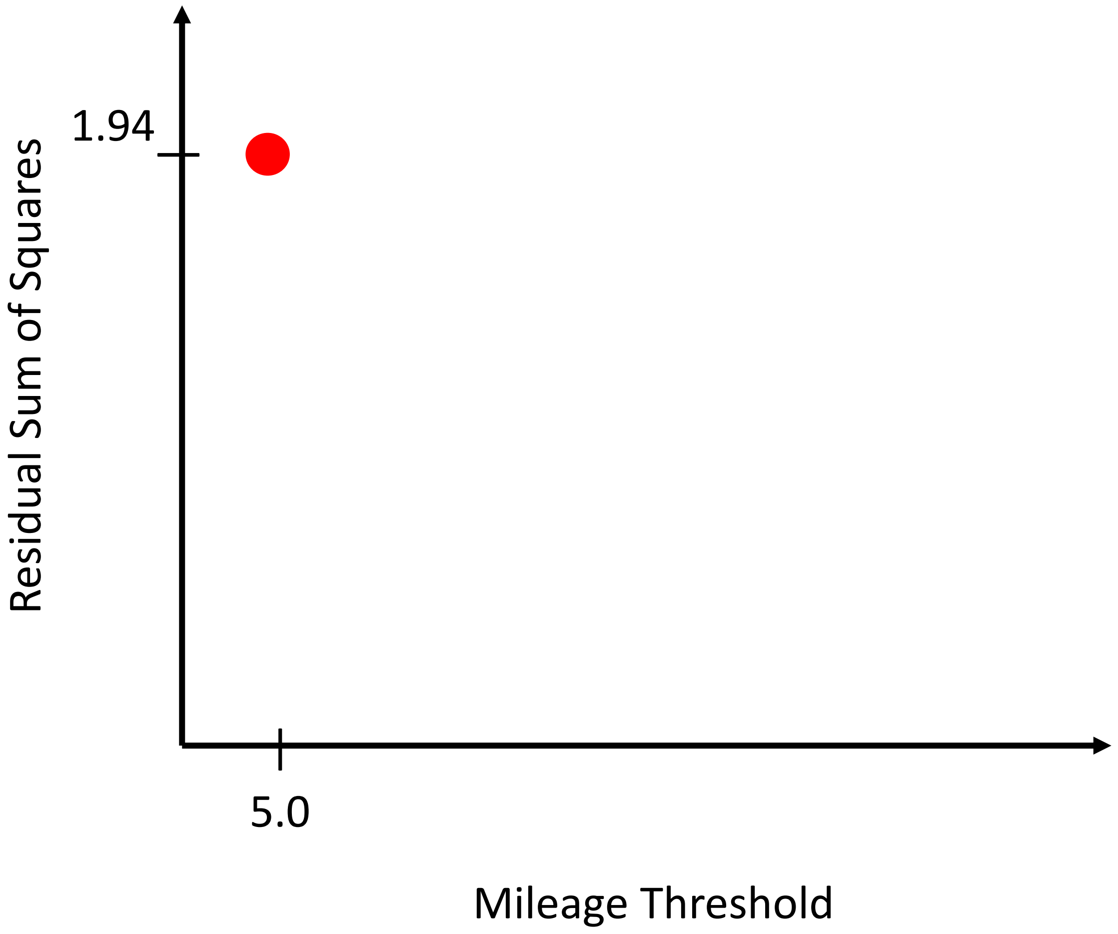


Calculate the Residual Sum of Squares of the tree:

$$(10-10)^2+(9.5-8.7)^2+(9.2-8.7)^2+(8.9-8.7)^2 + (8.5-8.7)^2+(8.3-8.7)^2+(7.8-8.7)^2=1.94$$

Review

❖ Build a Regression Tree: Consider each pair



We can plot the Residual Sum of Squares of each pair to 2D chart.

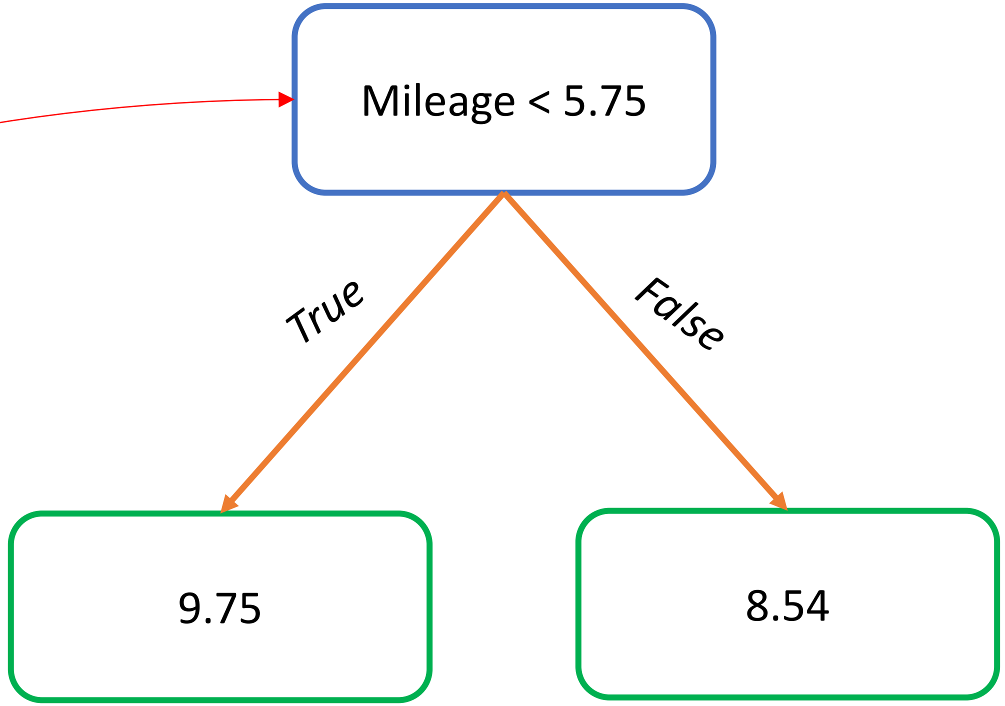
The objective is to find the threshold that have the minimum Residual Sum of Squares.

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

5.75



$$(10-9.75)^2+(9.5-9.75)^2+(9.2-8.54)^2+(8.9-8.54)^2+(8.5-8.54)^2+(8.3-8.54)^2+(7.8-8.54)^2=1.297$$

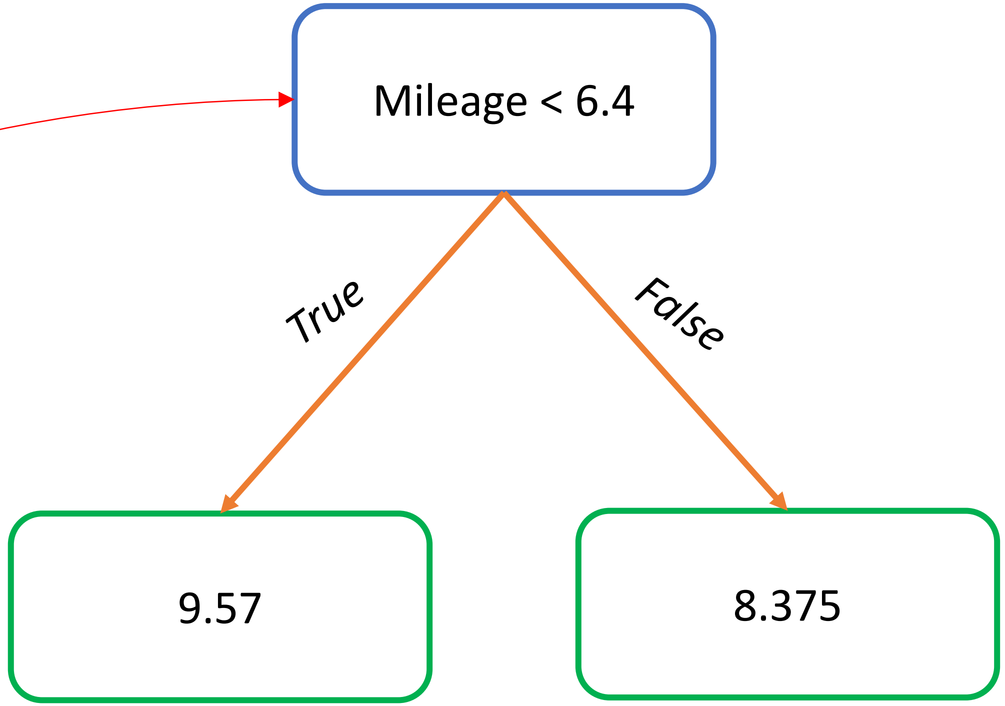
We will do the same for other pairs

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

6.4



$$(10-9.57)^2+(9.5-9.57)^2+(9.2-9.57)^2 \\ +(8.9-8.375)^2+(8.5-8.375)^2+(8.3-8.375)^2 \\ +(7.8-8.375)^2=0.9542$$

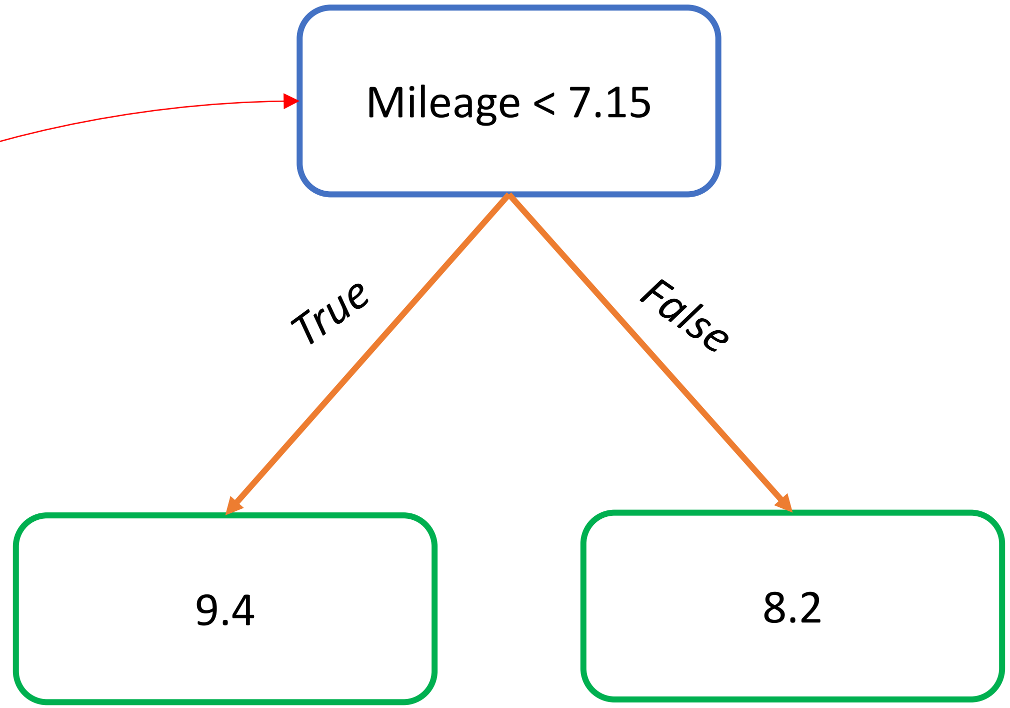
We will do the same for other pairs

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

7.15



$$(10-9.4)^2+(9.5-9.4)^2+(9.2-9.4)^2+(8.9-9.4)^2 + (8.5-8.2)^2+(8.3-8.2)^2+(7.8-8.2)^2 = 0.92$$

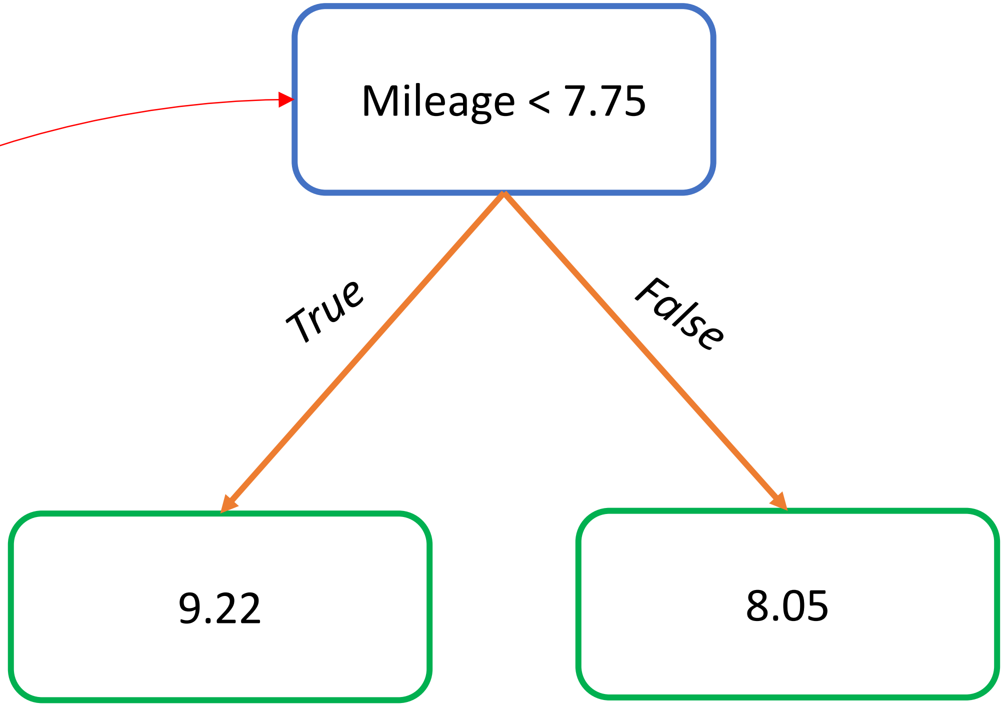
We will do the same for other pairs

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

7.75



$$(10-9.22)^2+(9.5-9.22)^2+(9.2-9.22)^2+(8.9-9.22)^2 + (8.5-9.22)^2+(8.3-8.05)^2+(7.8-8.05)^2 = 1.433$$

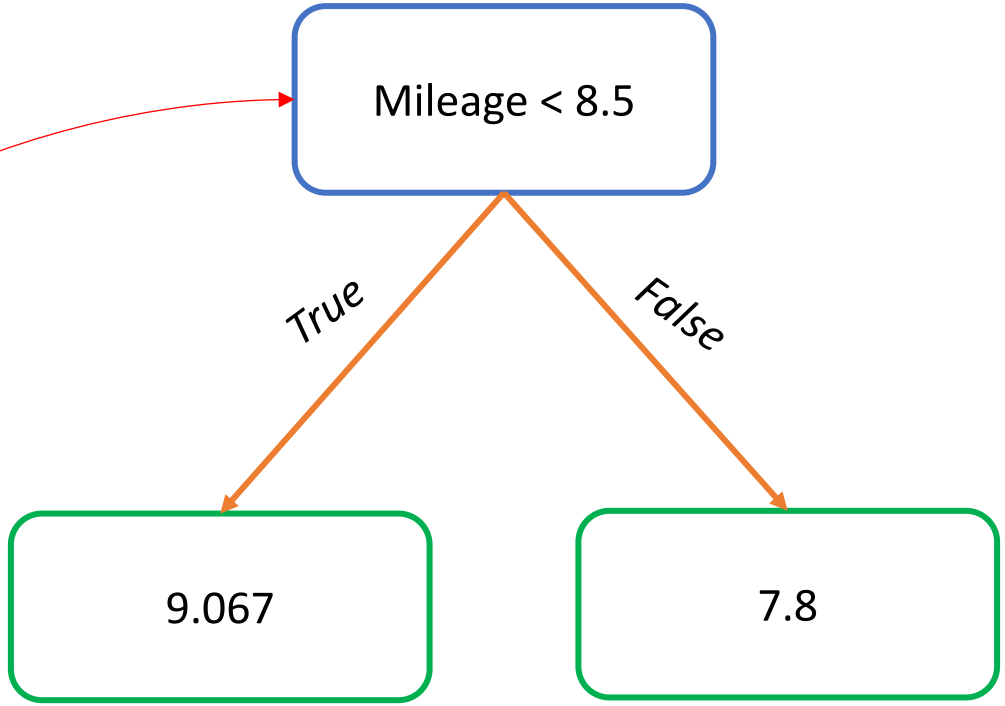
We will do the same for other pairs

Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

8.5

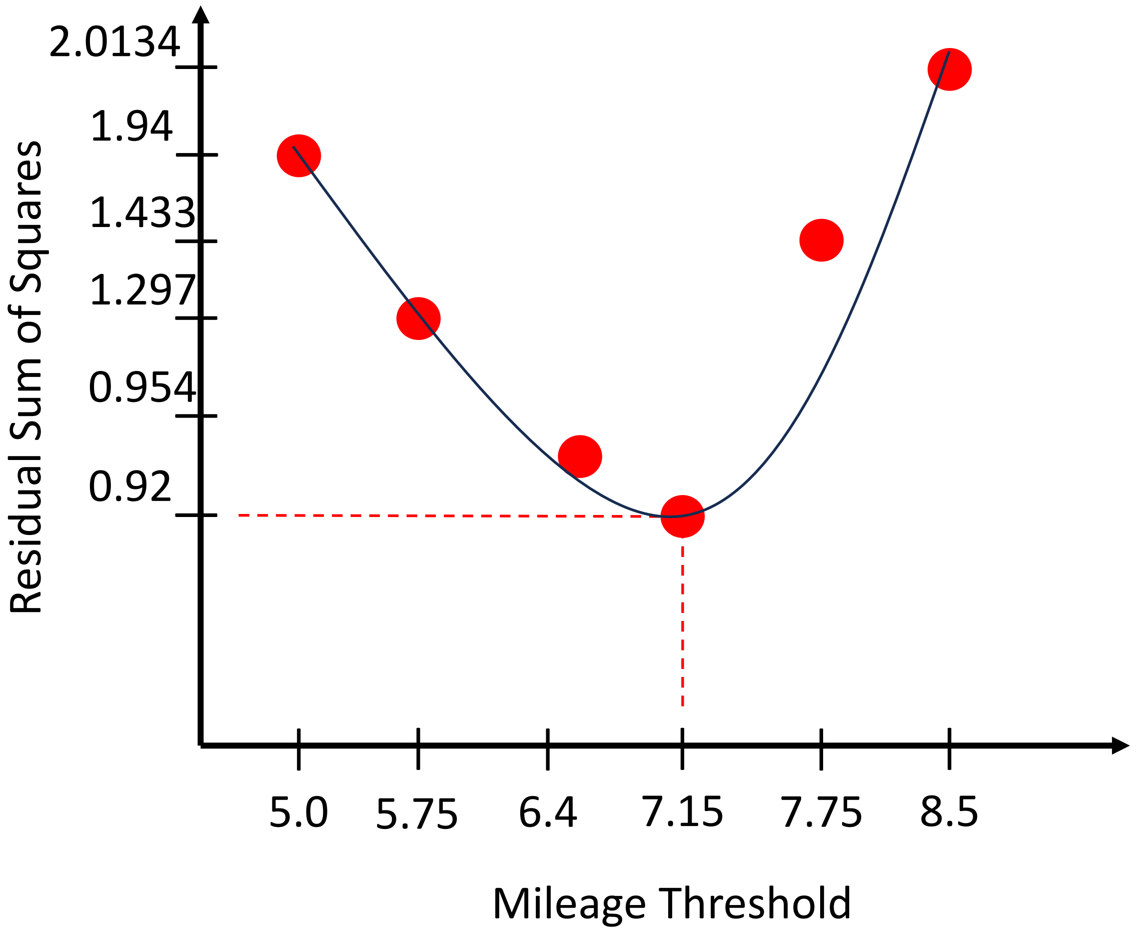


$$(10-9.067)^2+(9.5-9.067)^2+(9.2-9.067)^2$$
$$+(8.9-9.067)^2+(8.5-9.067)^2+(8.3-9.067)^2$$
$$+(7.8-7.8)^2=2.0134$$

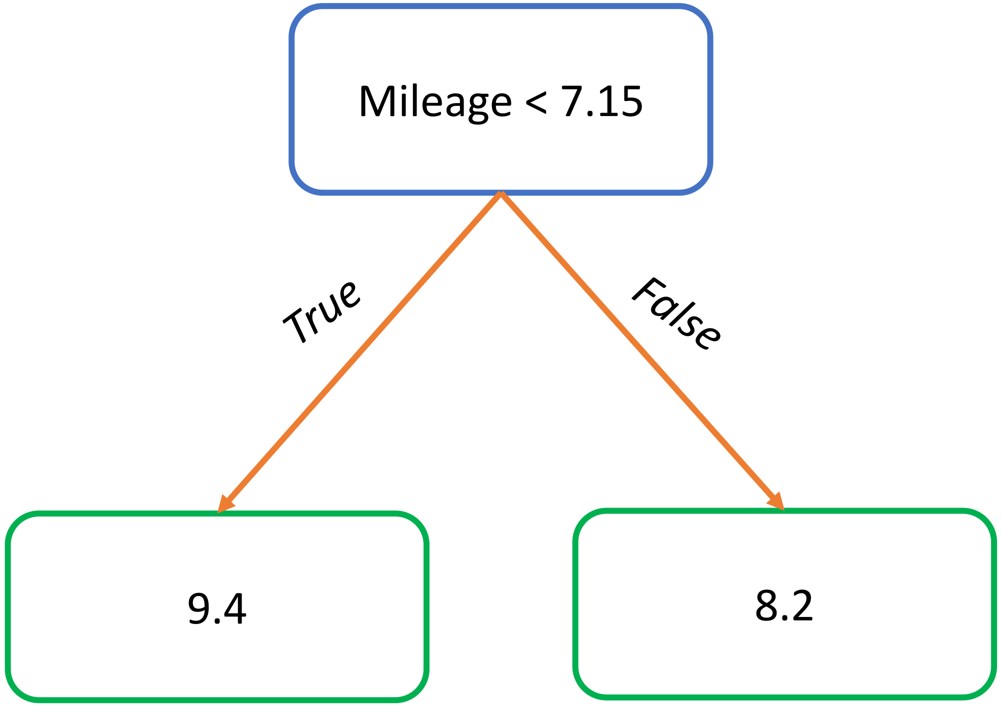
We will do the same for other pairs

Review

❖ Build a Regression Tree: RSS Visualization



With the chart, we now know that Mileage < 7.15 gives the smallest Residual Sum of Squares.



Review

❖ Build a Regression Tree

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

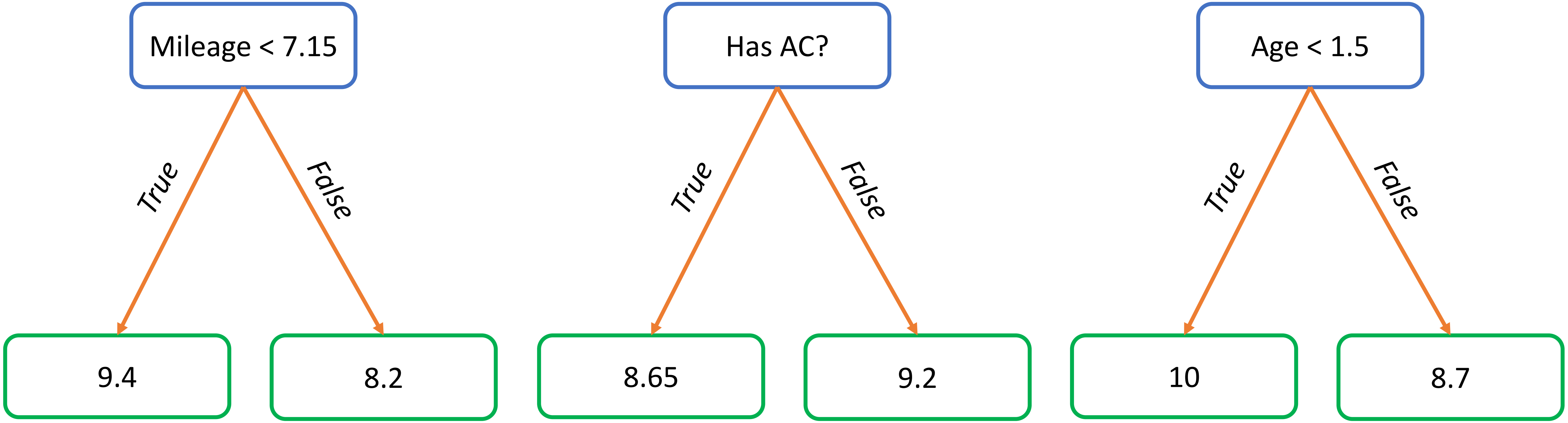
How about other features?

We also need to consider other features to find which one produce the least RSS.

To do that, apply the same step as for Mileage.

Review

❖ Build a Regression Tree



$(10-9.4)^2 + (9.5-9.4)^2 + (9.2-9.4)^2 + (8.9-9.4)^2 + (8.5-8.2)^2 + (8.3-8.2)^2 + (7.8-8.2)^2 = 0.92$

$(8.5-8.65)^2 + (7.8-8.65)^2 + (10-8.65)^2 + (8.3-8.65)^2 + (9.2-9.2)^2 + (8.9-9.2)^2 + (9.5-9.2)^2 = 2.87$

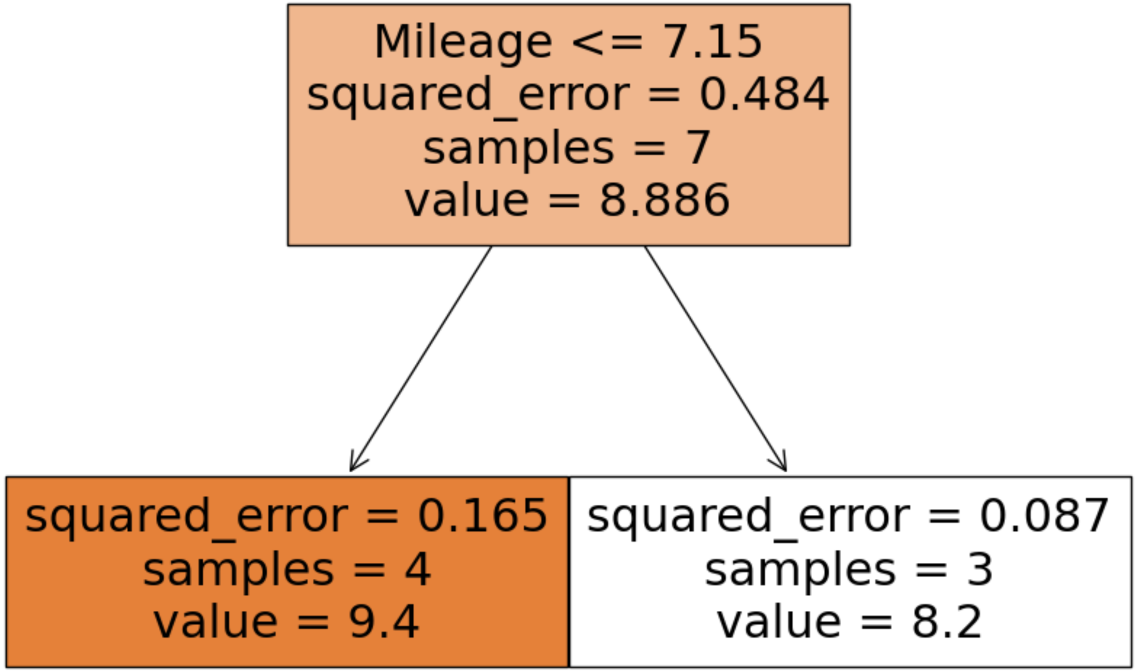
$(10-10)^2 + (9.2-8.7)^2 + (8.3-8.7)^2 + (9.5-8.7)^2 + (8.9-8.7)^2 + (8.5-8.7)^2 + (7.8-8.7)^2 = 1.94$

Mileage < 7.15 is the most appropriate

Review

❖ Build a Regression Tree

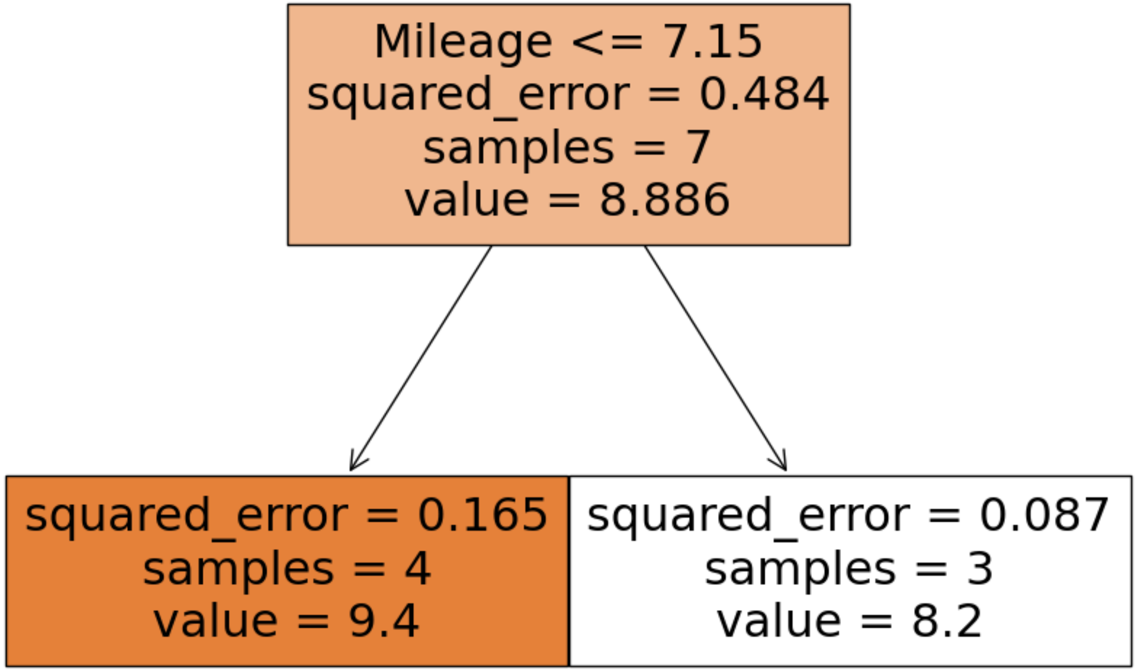
```python
import numpy as np
import pandas as pd

from sklearn.tree import DecisionTreeRegressor

data = {
    'Mileage': [4.5, 5.5, 6.0, 6.8, 7.5, 8.0, 9.0],
    'Has AC': [1, 0, 0, 0, 1, 1, 1],
    'Age': [1, 2, 2, 3, 3, 2, 4],
    'Price': [10.0, 9.5, 9.2, 8.9, 8.5, 8.3, 7.8]
}

df = pd.DataFrame(data)
dataset_arr = df.to_numpy()
X, y = dataset_arr[:, :-1], dataset_arr[:, -1]

regressor = DecisionTreeRegressor(
    random_state=1,
    max_depth=1
).fit(X, y)
```

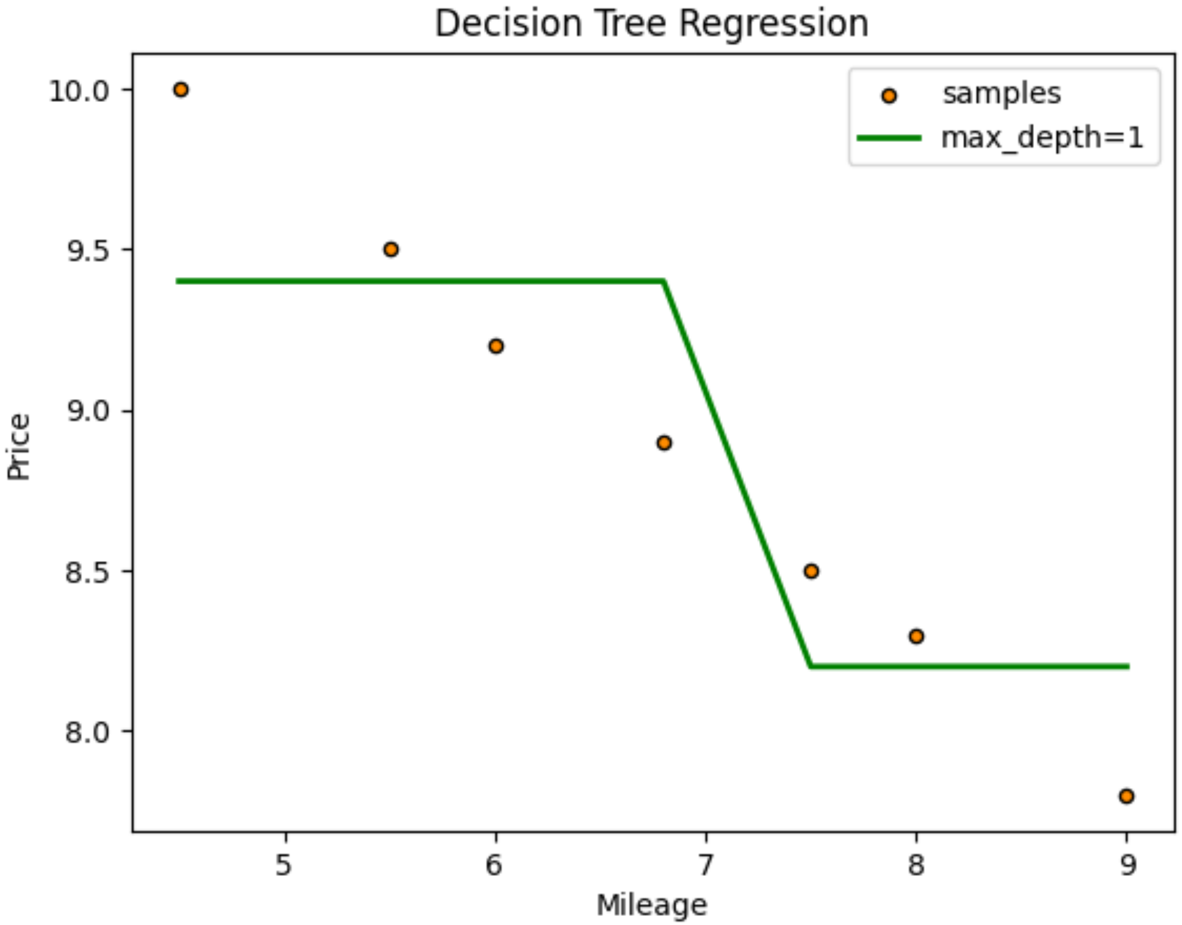


Tree built from sklearn with only root node

Review

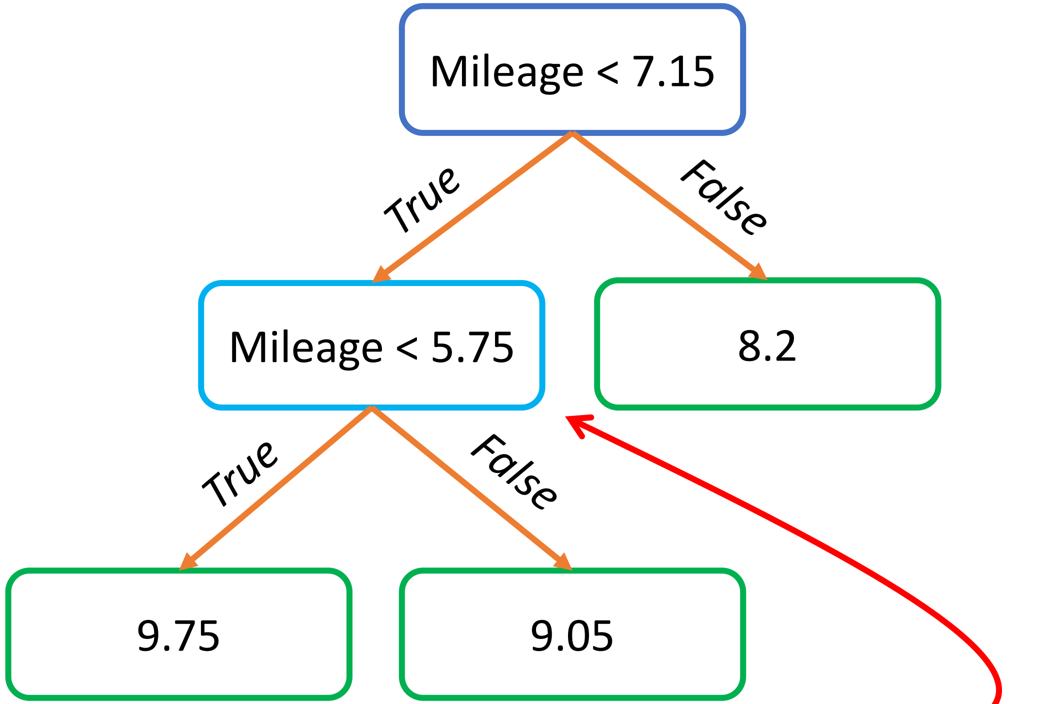
❖ Build a Regression Tree





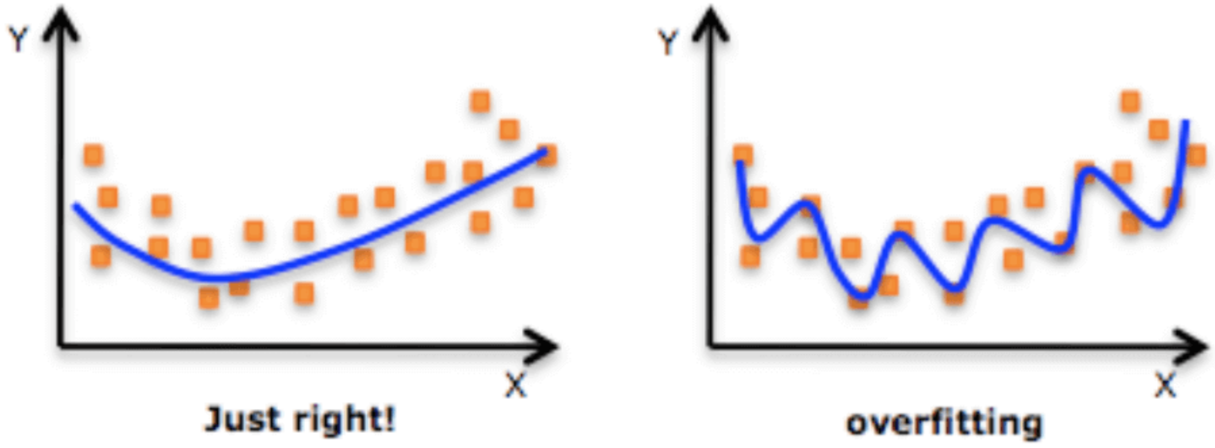
Review

❖ Build a Regression Tree: Further expand the tree



In theory, we can further expand the tree by adding more internal node (conditions) to the tree.

A bigger tree might get better performance. But it might also subject to **overfitting** problem.



Therefore, it is crucial to appropriately choose optimal hyperparameters of the decision tree.

Review

❖ Ensemble Learning

Mileage	Has AC?	Age	Price
4.5	yes	1	10.0
5.5	no	2	9.5
6.0	no	2	9.2
6.8	no	3	8.9
7.5	yes	3	8.5
8.0	yes	2	8.3
9.0	yes	4	7.8

Feaures: [7.8, ‘no’, 5].

True Label: 6.5

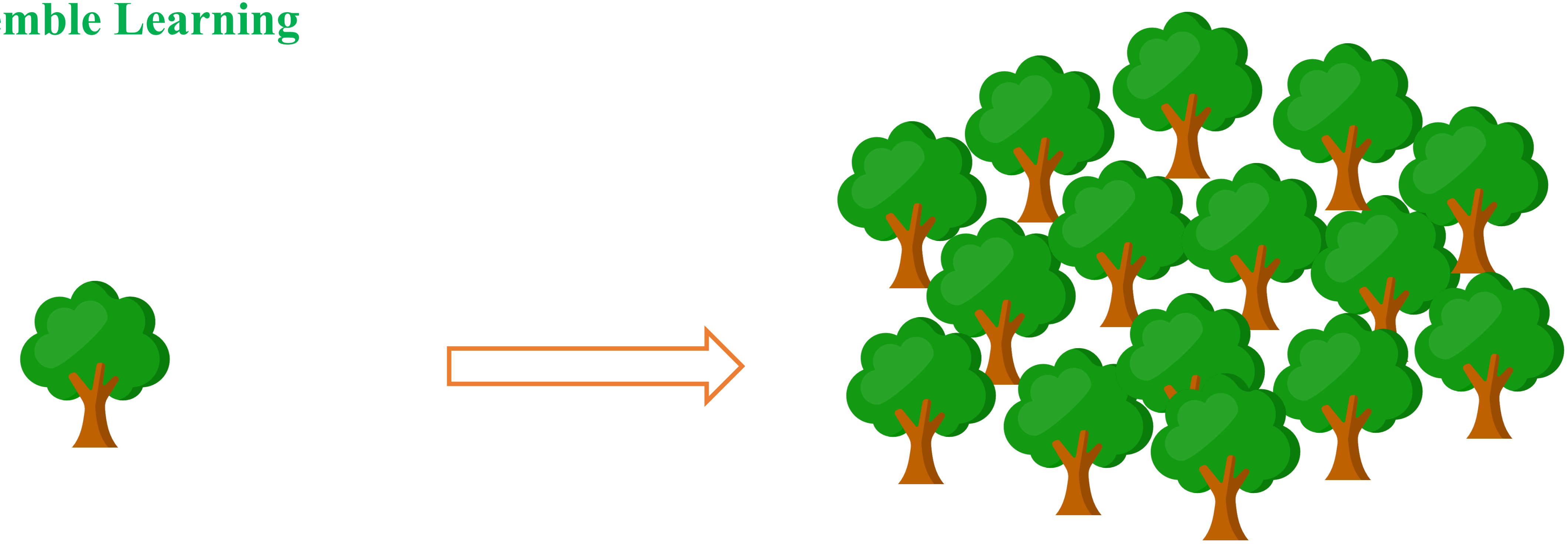
8.2

This prediction is unrealiable, how do we make sure that we receive a more stable result?

Consider the Car Price Prediction problem again

Review

❖ Ensemble Learning

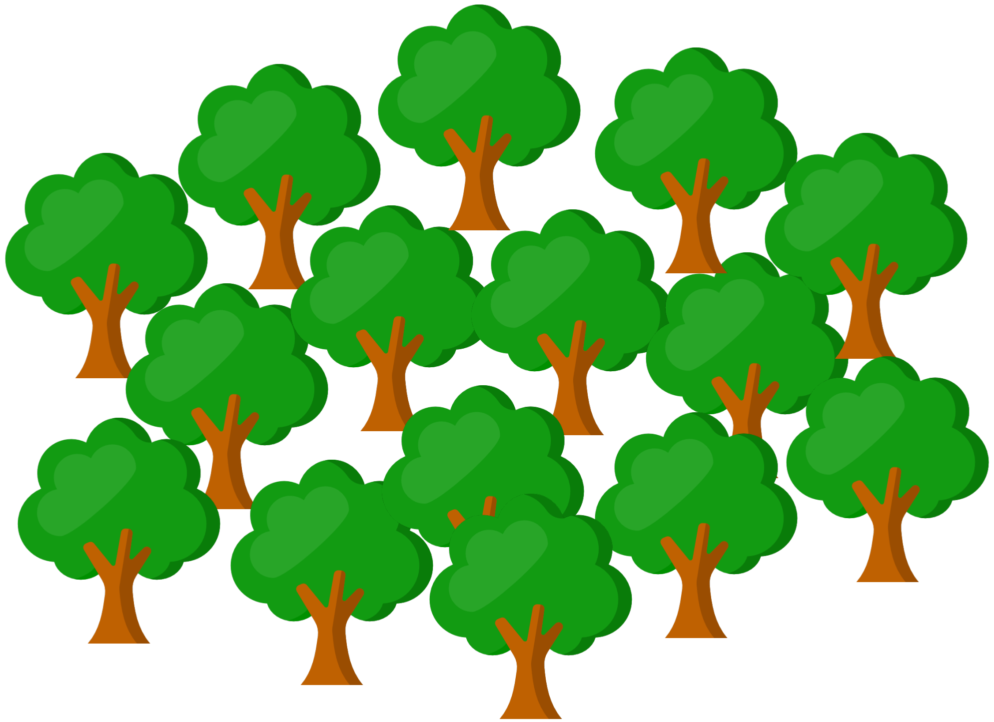


If the result from 1 tree is not good…

Why don't we just use more trees?

Review

❖ Ensemble Learning

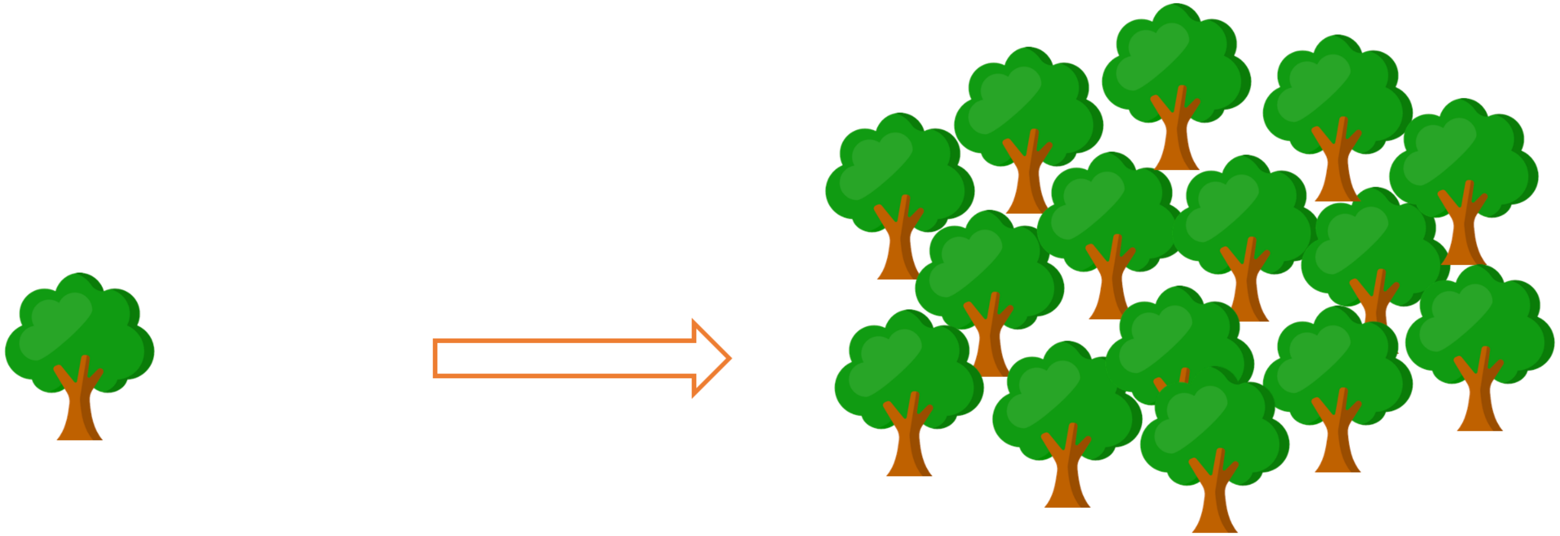


These are Decision Trees

Ensemble Learning: A machine learning technique that combines the predictions from multiple individual models to produce a more accurate and robust prediction that any single model.

Review

❖ Random Forest



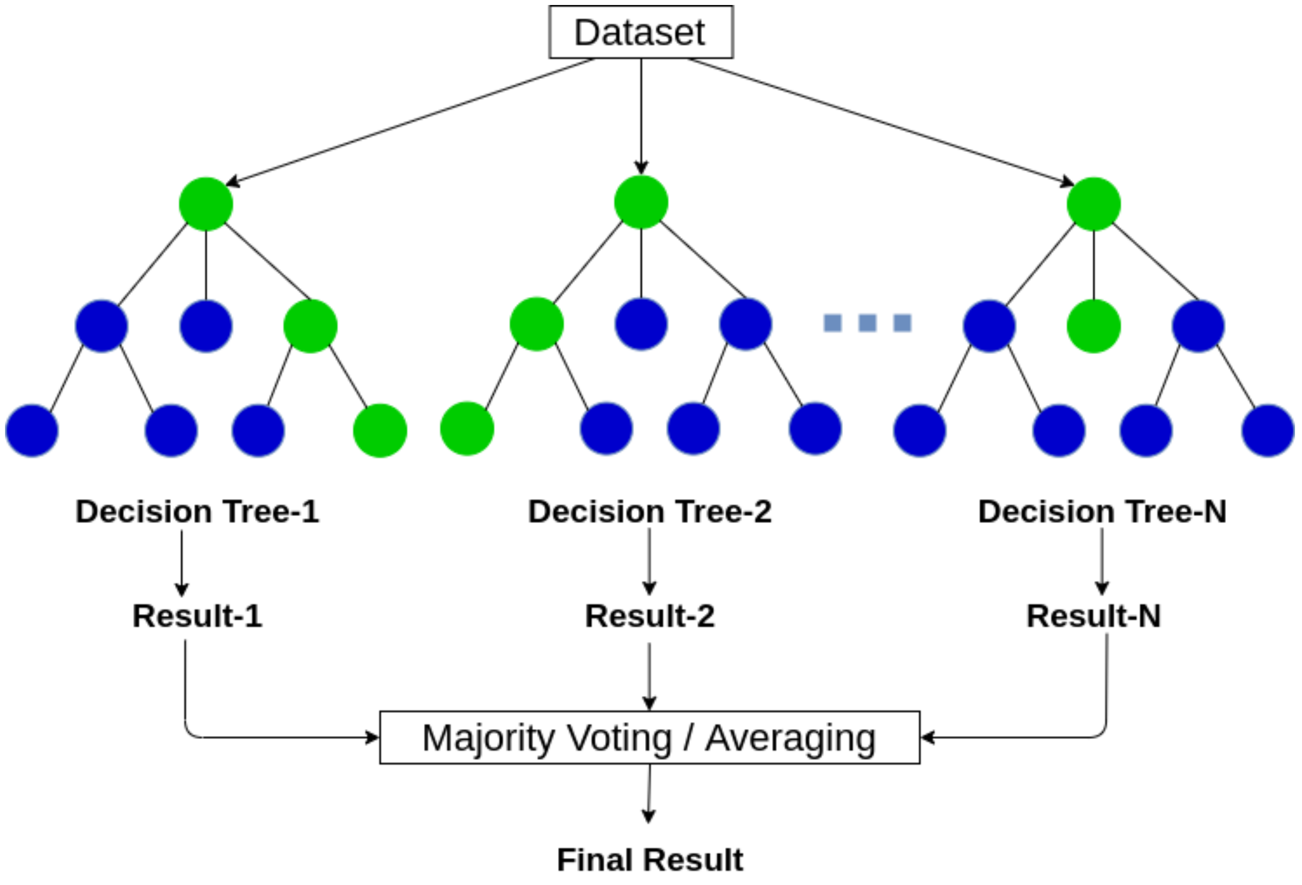
If the result from 1 tree is not good…

Why don’t we just use more trees?

In previous example, there is an algorithm that uses multiple Decision Trees to produce a new single output called **Random Forest**.

Review

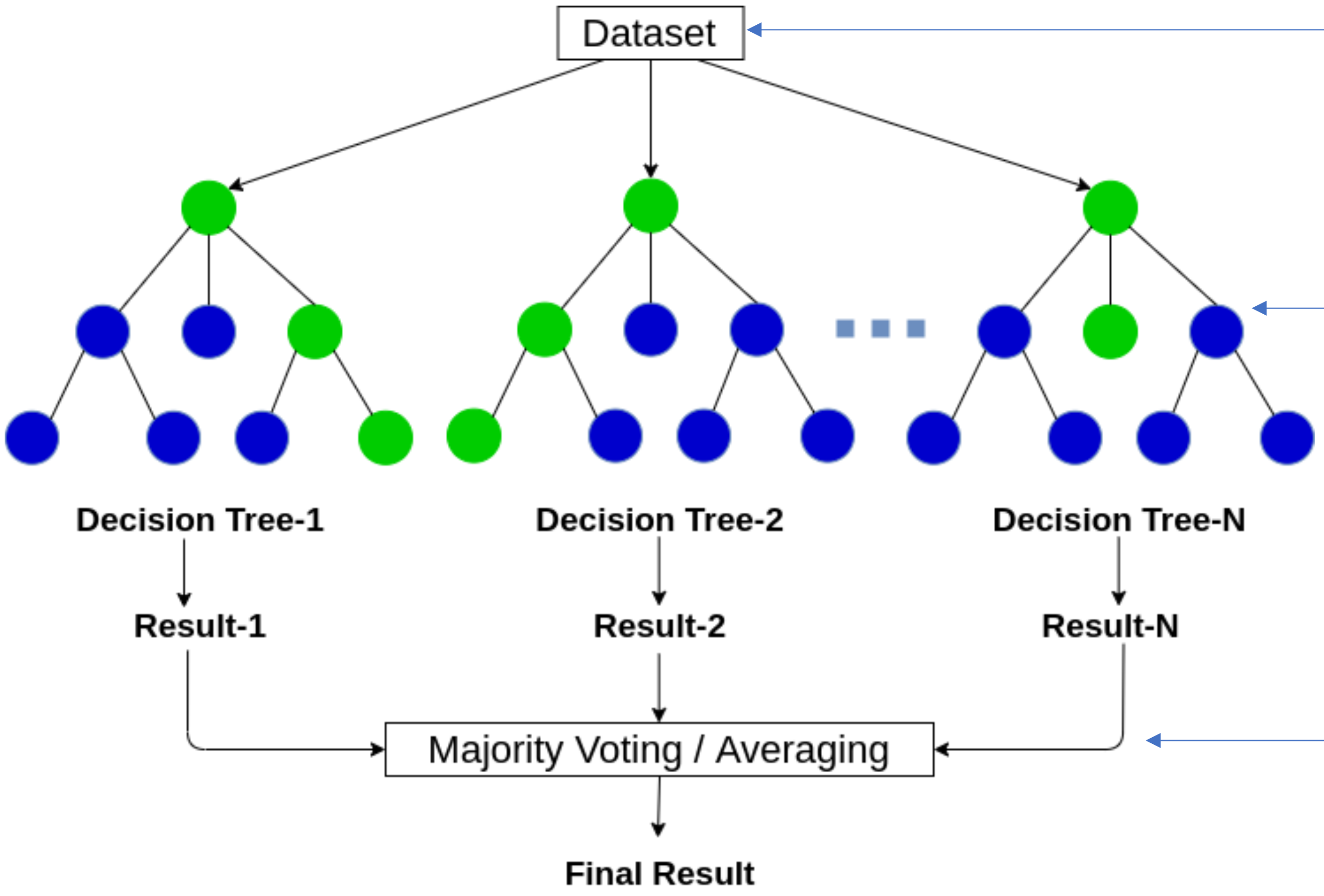
❖ Random Forest



Random Forest: A supervised-learning machine learning algorithm that combines the output of multiple Decision Trees to reach a single outcome. It can perform both classification and regression tasks.

Review

❖ Random Forest: Key idea



From original dataset, create multiple Bootstrap Dataset using Bootstrapping technique.

With Bootstrap Datasets, train *n* Decision Trees (estimators).

In inference phrase, with the predictions of *n* trees, we do voting/averaging them to get final result (Aggregating).

Review

❖ Random Forest: Bootstrapping

Original Dataset	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Bootstrap Dataset 1	S5	S4	S7	S3	S2	S2	S8	S1	S10	S1
Bootstrap Dataset 2	S7	S6	S9	S9	S9	S2	S1	S10	S4	S8

Bootstrapping = Random sampling with replacement

We create new dataset by taking samples from original dataset (sampling) which can be **duplicated**.

Review

❖ Random Forest: Bootstrapping

Index	X1	X2	X3	Y
0				
1				
2				
3				
4				

Original Dataset

Index	X1	Y
1		
1		
4		
3		
4		

Bootstrap Dataset 1

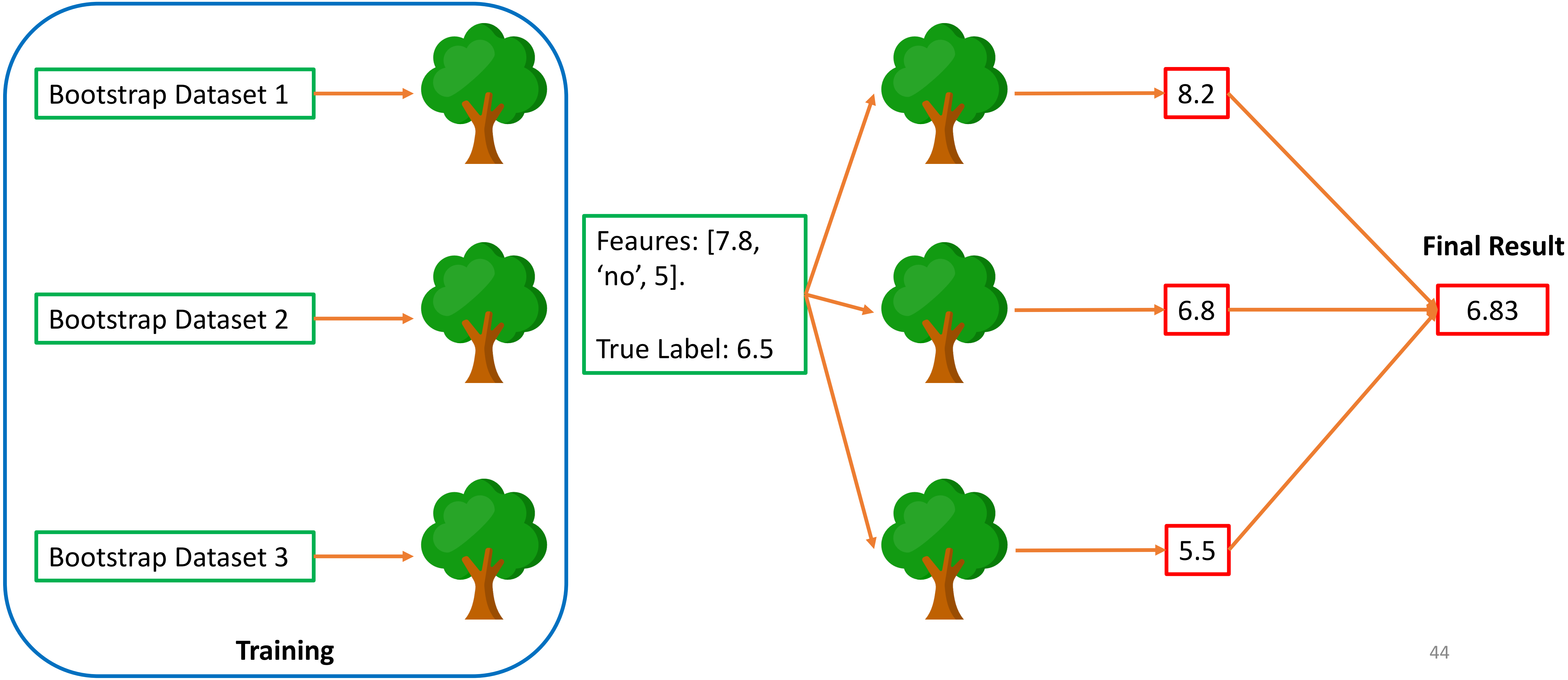
Index	X3	Y
0		
3		
1		
4		
4		

Bootstrap Dataset 2

In Random Forest, we also **randomly select features** for Bootstrap Datasets.

Review

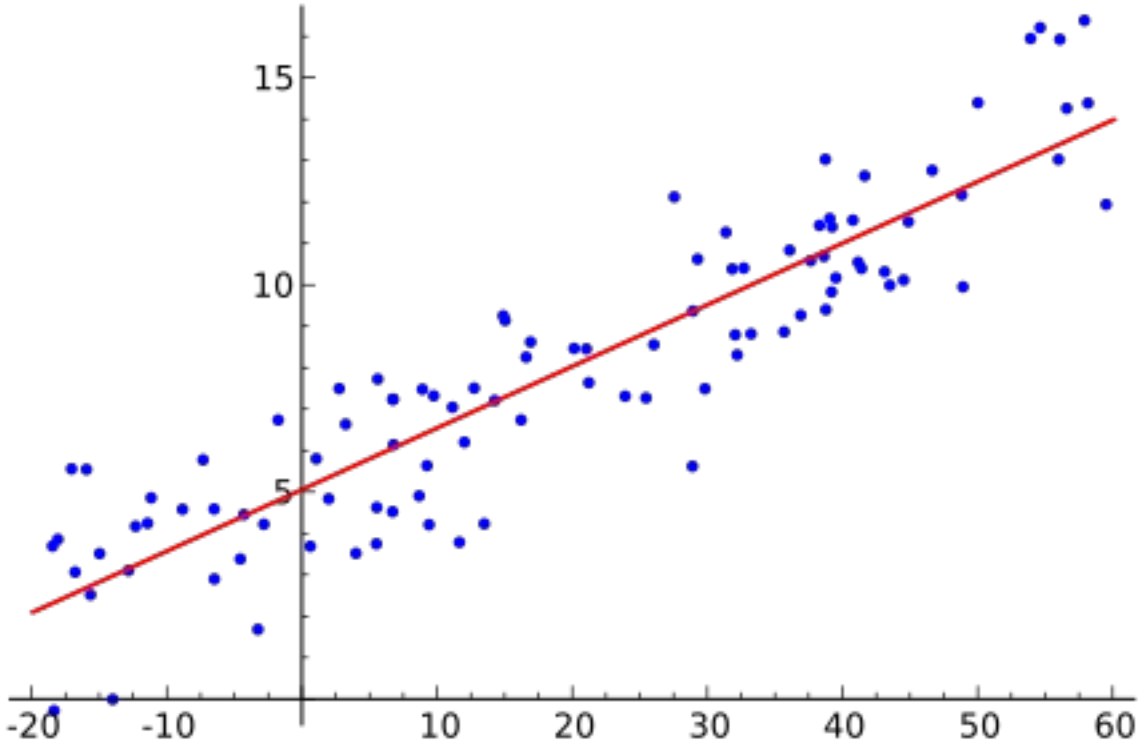
❖ Random Forest: Aggregating



Code Implementation

❖ Introduction

Code exercise description: Given Housing.csv dataset, train a Decision Tree and a Random Forest models to predict house price based on some input features about the house.



Code Implementation

❖ Step 1: Import necessary libraries

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    mean_absolute_error,
    mean_squared_error
)
```

scikit-learn (sklearn): An open-source library for Python language that features various classification, regression and clustering algorithms.

Code Implementation

❖ Step 1: Import necessary libraries

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    mean_absolute_error,
    mean_squared_error
)
```

sklearn.tree: The module includes decision tree-based models for classification and regression. (In this case we will use regression).

sklearn.ensemble: The module includes ensemble-based methods for classification, regression and anomaly detection. (In this case we will use regression).

Code Implementation

❖ Step 2: Load dataset

To read .csv file, we use pandas.read_csv():

```
dataset_path = './Housing.csv'
df = pd.read_csv(dataset_path)
df
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
...	...	...	...	...	...	...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	no	unfurnished
541	1767150	2400	3	1	1	no	no	no	no	no	0	no	semi-furnished
542	1750000	3620	2	1	1	yes	no	no	no	no	0	no	unfurnished
543	1750000	2910	3	1	1	no	no	no	no	no	0	no	furnished
544	1750000	3850	3	1	2	yes	no	no	no	no	0	no	unfurnished

545 rows × 13 columns

Code Implementation

❖ Step 3: Check missing values and get numerical features statistic

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

```
1 df.describe()
```

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

Using pandas.DataFrame.info() and pandas.DataFrame.describe() to check missing values and get statistic of numerical features.

Code Implementation

❖ Step 4: Deal with categorical variables

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
...	...	...	...	...	...	...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	no	unfurnished
541	1767150	2400	3	1	1	no	no	no	no	no	0	no	semi-furnished
542	1750000	3620	2	1	1	yes	no	no	no	no	0	no	unfurnished
543	1750000	2910	3	1	1	no	no	no	no	no	0	no	furnished
544	1750000	3850	3	1	2	yes	no	no	no	no	0	no	unfurnished

545 rows × 13 columns

Categorical variable: A type of variable that represents distinct categories or groups. These variables often in the form of string.

Code Implementation

❖ Step 4: Deal with categorical variables

X1	X2	X3	Y
12	5.5	yes	9.0
4	1.0	no	6.8
9	3.2	no	8.0
10	4.4	yes	8.5

X3 has unique values of [‘yes’, ‘no’]

Idea: Convert strings using integer number starting from 0.
=>
- ‘yes’: 1
- ‘no’: 0

X1	X2	X3	Y
12	5.5	yes	9.0
4	1.0	no	6.8
9	3.2	no	8.0
10	4.4	yes	8.5

X1	X2	X3	Y
12	5.5	1	9.0
4	1.0	0	6.8
9	3.2	0	8.0
10	4.4	1	8.5

Code Implementation

❖ Step 4: Deal with categorical variables

1. Check all features that are in form of string (object).

```
categorical_cols = df.select_dtypes(
    include=['object']
).columns.to_list()
categorical_cols
```

```
['mainroad',
 'guestroom',
 'basement',
 'hotwaterheating',
 'airconditioning',
 'prefarea',
 'furnishingstatus']
```

2. Check number of unique values for each feature that are in form of string (object).

```
for col_name in categorical_cols:
    n_categories = df[col_name].nunique()
    print(f'Number of categories in {col_name}: {n_categories}')
```

```
Number of categories in mainroad: 2
Number of categories in guestroom: 2
Number of categories in basement: 2
Number of categories in hotwaterheating: 2
Number of categories in airconditioning: 2
Number of categories in prefarea: 2
Number of categories in furnishingstatus: 3
```

Code Implementation

❖ Step 4: Deal with categorical variables

3. Apply OrdinalEncoder() for all categorical features.

```
ordinal_encoder = OrdinalEncoder()
encoded_categorical_cols = ordinal_encoder.fit_transform(
    df[categorical_cols]
)
encoded_categorical_df = pd.DataFrame(
    encoded_categorical_cols,
    columns=categorical_cols
)
numerical_df = df.drop(categorical_cols, axis=1)
encoded_df = pd.concat(
    [numerical_df, encoded_categorical_df], axis=1
)
```

Create an instance of OrdinalEncoder().

Apply OrdinalEncoder() to all categorical columns using fit_transform().

Create a new DataFrame that only contains encoded categorical data.

Drop all categorical data in original dataframe.

Concatenate both DataFrames.

Code Implementation

❖ Step 4: Deal with categorical variables

```
encoded_df
```

	price	area	bedrooms	bathrooms	stories	parking	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	furnishingstatus
0	13300000	7420	4	2	3	2	1.0	0.0	0.0	0.0	1.0	1.0	0.0
1	12250000	8960	4	4	4	3	1.0	0.0	0.0	0.0	1.0	0.0	0.0
2	12250000	9960	3	2	2	2	1.0	0.0	1.0	0.0	0.0	1.0	1.0
3	12215000	7500	4	2	2	3	1.0	0.0	1.0	0.0	1.0	1.0	0.0
4	11410000	7420	4	1	2	2	1.0	1.0	1.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	2	1.0	0.0	1.0	0.0	0.0	0.0	2.0
541	1767150	2400	3	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
542	1750000	3620	2	1	1	0	1.0	0.0	0.0	0.0	0.0	0.0	2.0
543	1750000	2910	3	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
544	1750000	3850	3	1	2	0	1.0	0.0	0.0	0.0	0.0	0.0	2.0

545 rows × 13 columns

Code Implementation

❖ Step 5: Normalization

Using sklearn.preprocessing.StandardScaler() to scale all values in dataset.

```
normalizer = StandardScaler()
dataset_arr = normalizer.fit_transform(
    encoded_df
)
```

$$z = \frac{x_i - \mu}{\sigma}$$

```
dataset_arr
```

```
array([[ 4.56636513,  1.04672629,  1.40341936, ...,  1.4726183 ,
         1.80494113, -1.40628573],
       [ 4.00448405,  1.75700953,  1.40341936, ...,  1.4726183 ,
        -0.55403469, -1.40628573],
       [ 4.00448405,  2.21823241,  0.04727831, ..., -0.67906259,
         1.80494113, -0.09166185],
       ...,
       [-1.61432675, -0.70592066, -1.30886273, ..., -0.67906259,
        -0.55403469,  1.22296203],
       [-1.61432675, -1.03338891,  0.04727831, ..., -0.67906259,
        -0.55403469, -1.40628573],
       [-1.61432675, -0.5998394 ,  0.04727831, ..., -0.67906259,
        -0.55403469,  1.22296203]])
```

Code Implementation

❖ Step 6: Split X, y

	price	area	bedrooms	bathrooms	stories	parking	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	furnishingstatus
0	13300000	7420	4	2	3	2	1.0	0.0	0.0	0.0	1.0	1.0	0.0
1	12250000	8960	4	4	4	3	1.0	0.0	0.0	0.0	1.0	0.0	0.0
2	12250000	9960	3	2	2	2	1.0	0.0	1.0	0.0	0.0	1.0	1.0
3	12215000	7500	4	2	2	3	1.0	0.0	1.0	0.0	1.0	1.0	0.0
4	11410000	7420	4	1	2	2	1.0	1.0	1.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...

- **Dependent Variable:** Price.

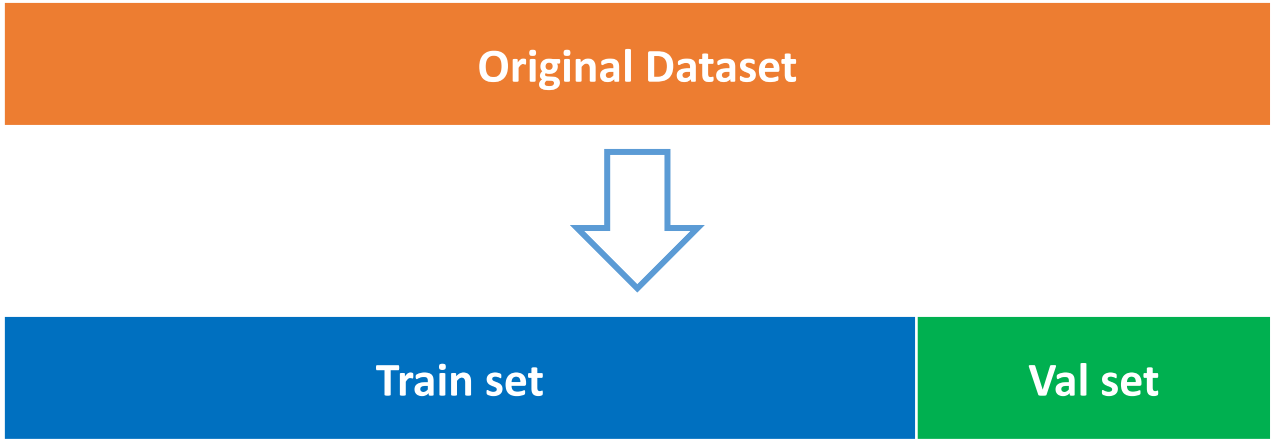
- **Independent Variables**: area, bedrooms, bathrooms, stories, parking, mainroad, guestroom, basement, hotwaterheating, airconditioning, prefarea, furnishingstatus.

```
X = dataset_arr[:, 1:]
y = dataset_arr[:, 0]

print(f'Independent Variables shape: {X.shape}')
print(f'Dependent Variable shape: {y.shape}')
```

```
Independent Variables shape: (545, 12)
Dependent Variable shape: (545,)
```

Code Implementation

❖ Step 7: Split train, val set




```python
test_size = 0.3
random_state = 1
is_shuffle = True
X_train, X_val, y_train, y_val = train_test_split(
    X, y,
    test_size=test_size,
    random_state=random_state,
    shuffle=is_shuffle
)
```

Code Implementation

❖ Step 8: Train models

For Random Forest:

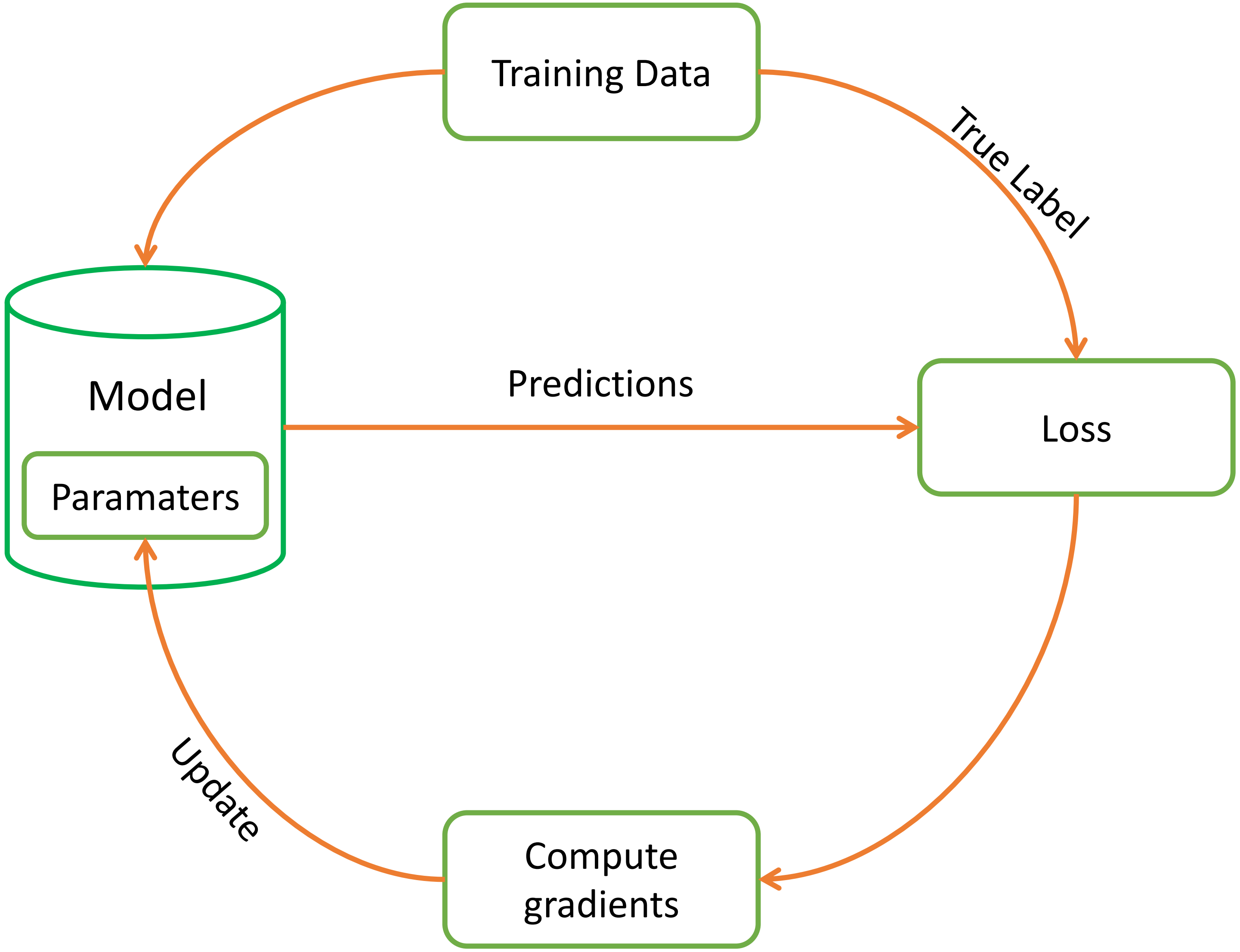
```
regressor = RandomForestRegressor(
    random_state=random_state
)
regressor.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=1)
```

For Decision Tree:

```
regressor = DecisionTreeRegressor(
    random_state=random_state
)
regressor.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=1)
```

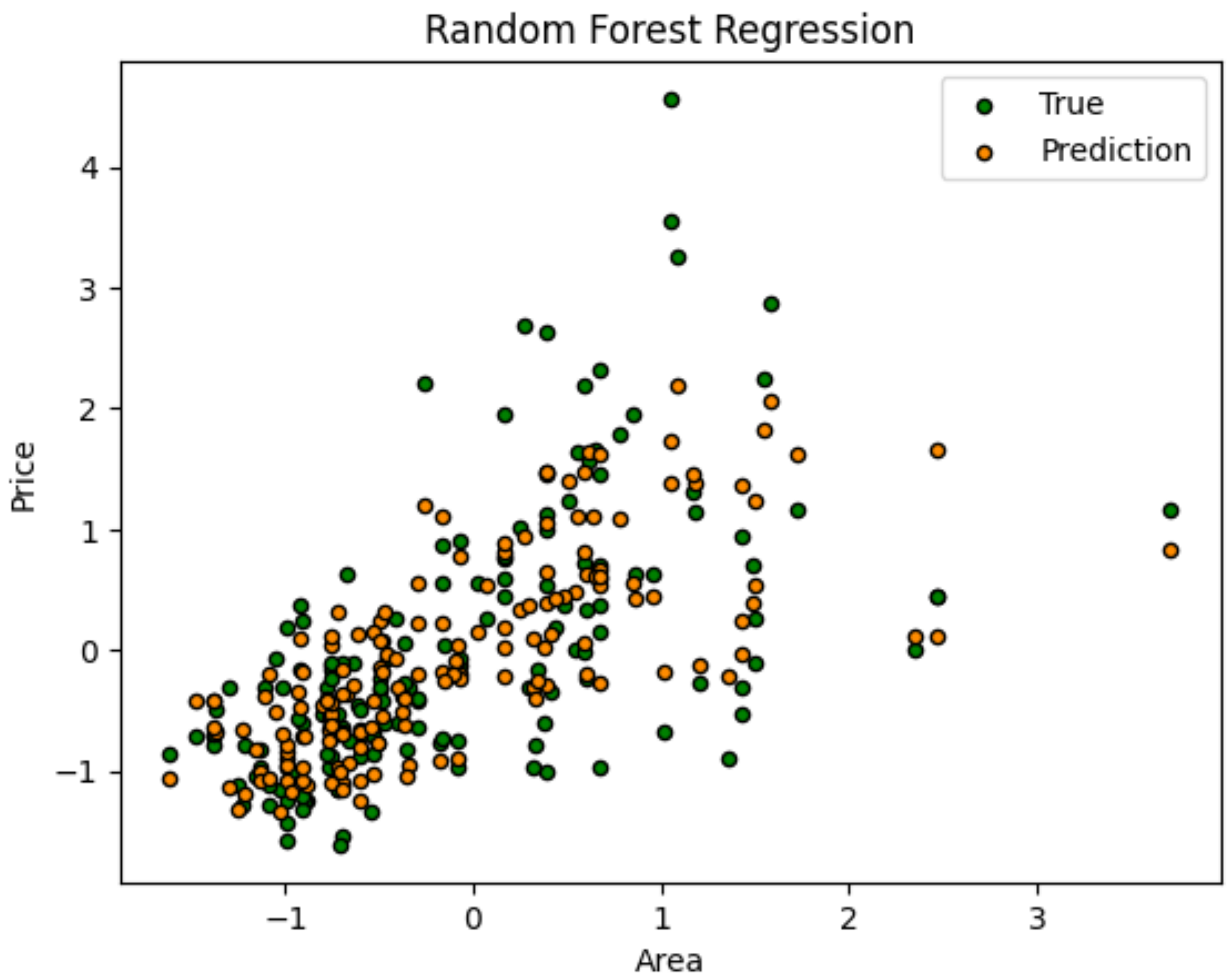


Code Implementation

❖ Step 9: Evaluation

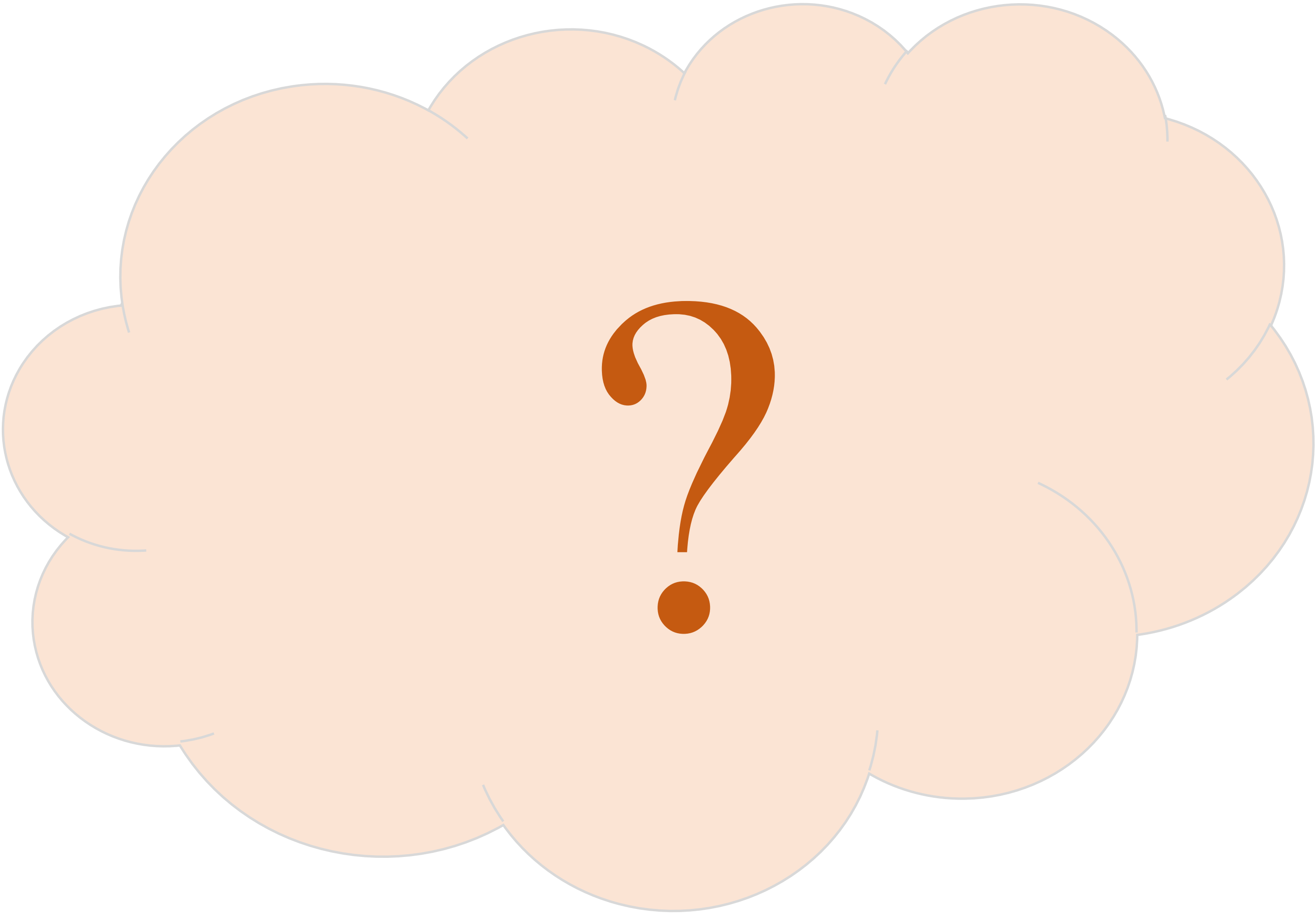
Let trained model predict X of val, then calculating MAE and MSE:

```python
y_pred = regressor.predict(X_val)

mae = mean_absolute_error(y_val, y_pred)
mse = mean_squared_error(y_val, y_pred)

print('Evaluation results on validation set:')
print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
```



Performance of Random Forest on Validation set.

Question




Thank you!