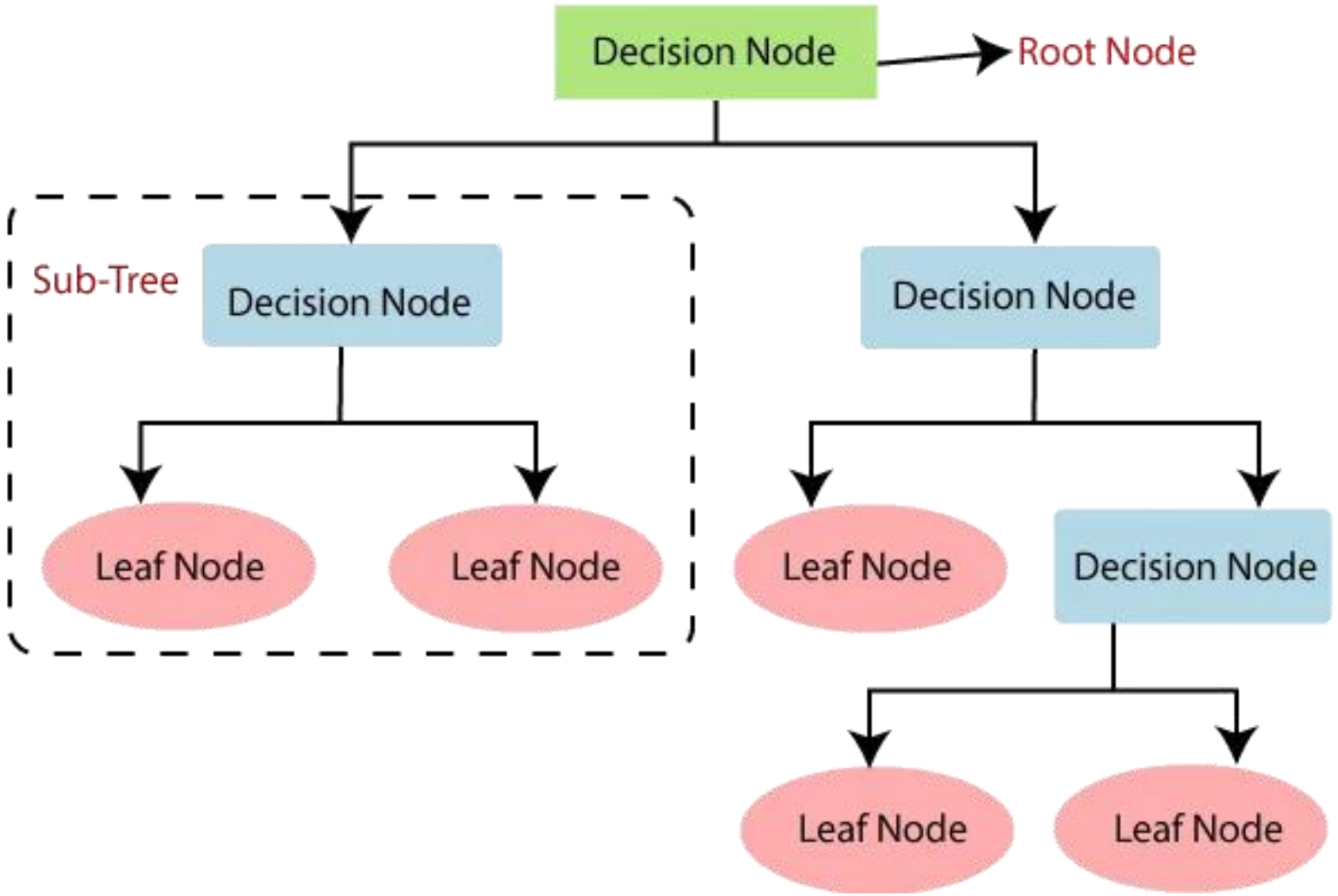
Decision Tree

A decision tree is a non-parametric supervised learning algorithm.



CART - Classification and Regression Trees

The logic of decision trees can also be applied to regression problems, hence the name CART

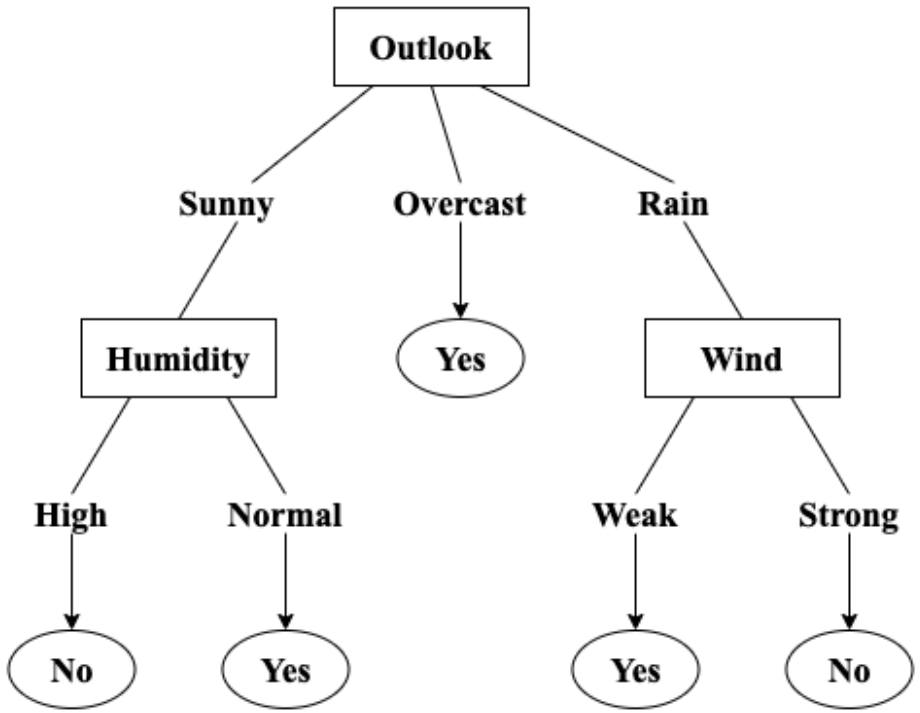
Scikit-learn uses an optimized version of the **CART algorithm**

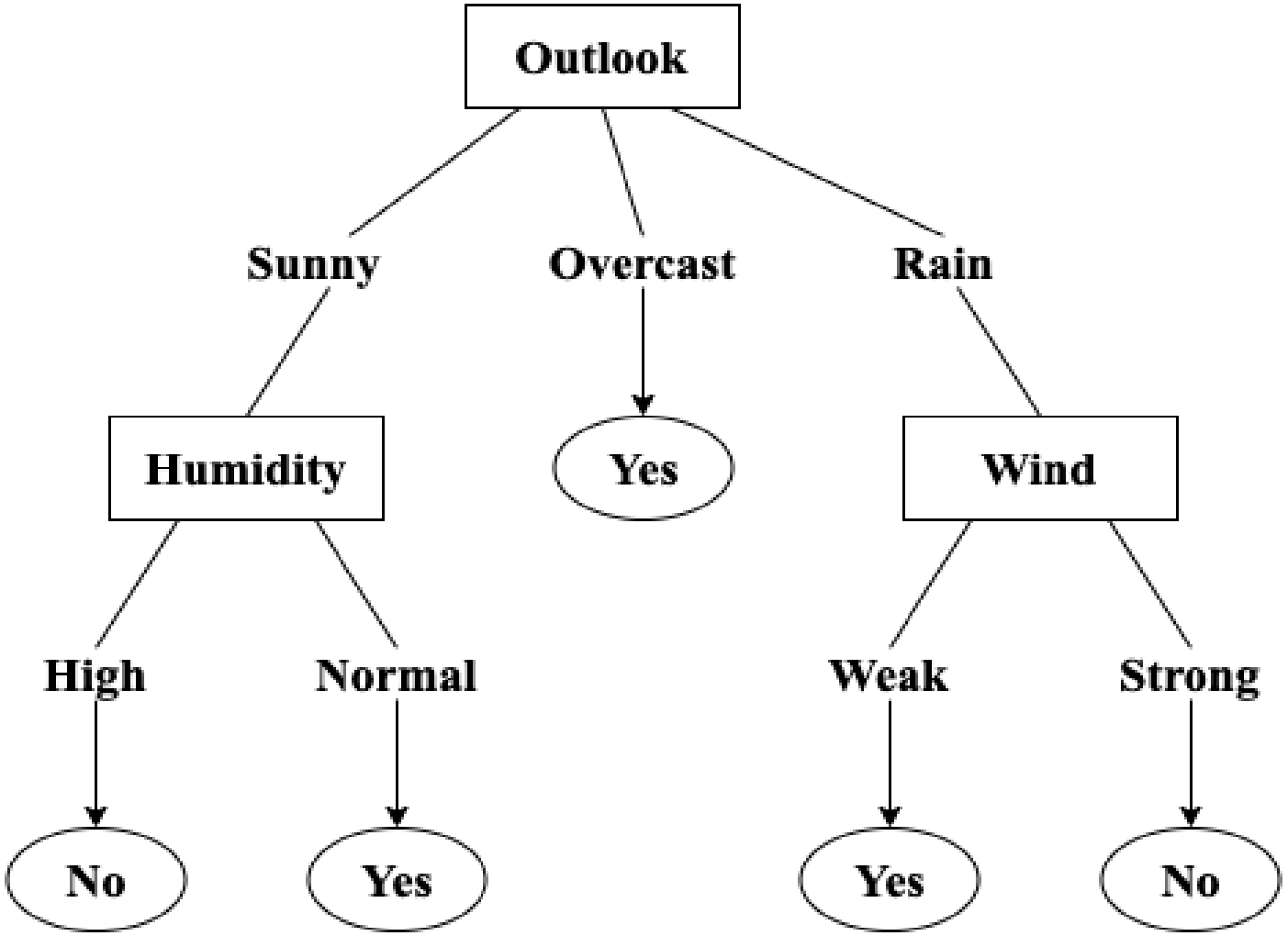
PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No





Input query point:
[Rainy, Mild, High, Strong]

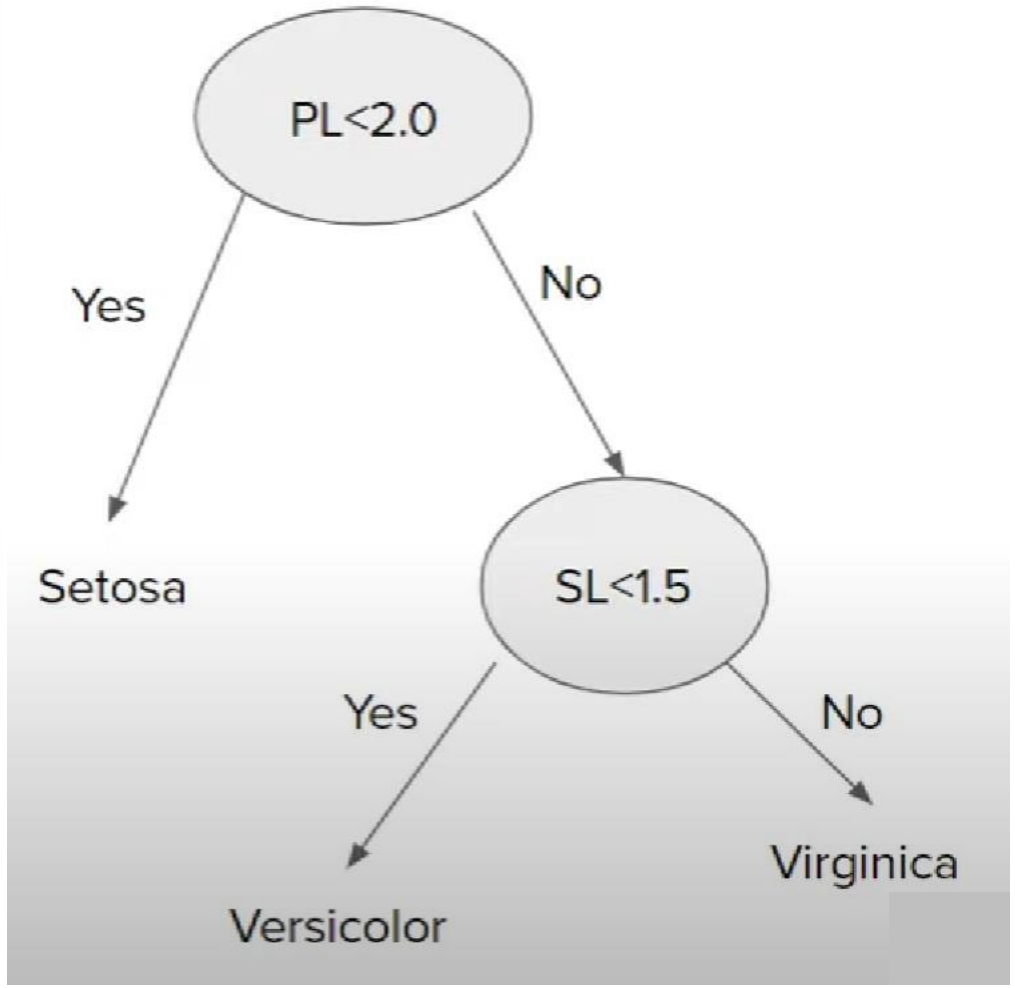
What if we have numerical data?

Petal Length	Sepal Length	Type
1.34	0.34	Setosa
3.45	1.45	Versicolor
1.69	0.98	Setosa
2.56	1.79	Virginica
3.00	1.13	Versicolor
1.3	0.88	Setosa

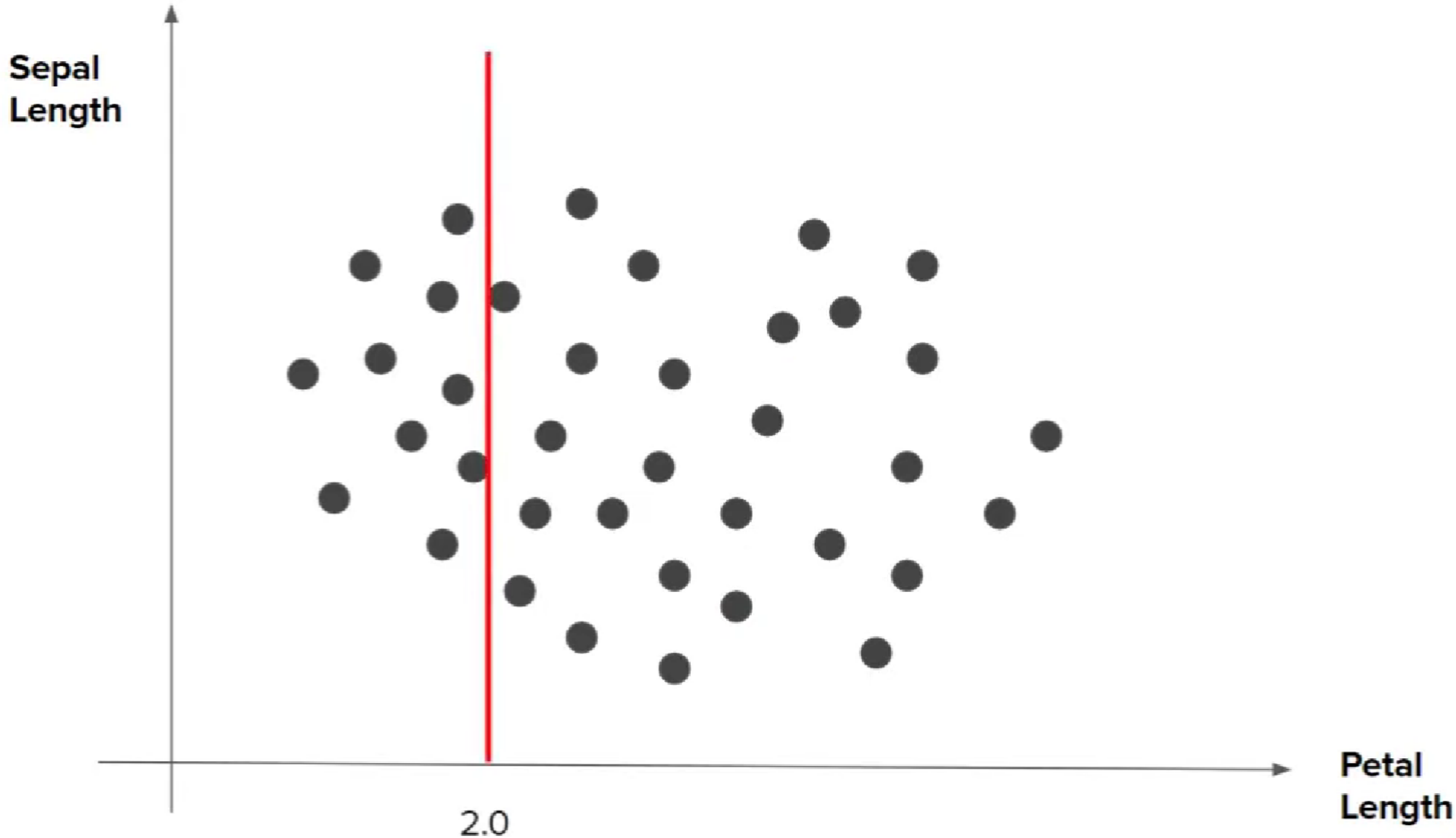
Iris Flower Classification

What if we have numerical data?

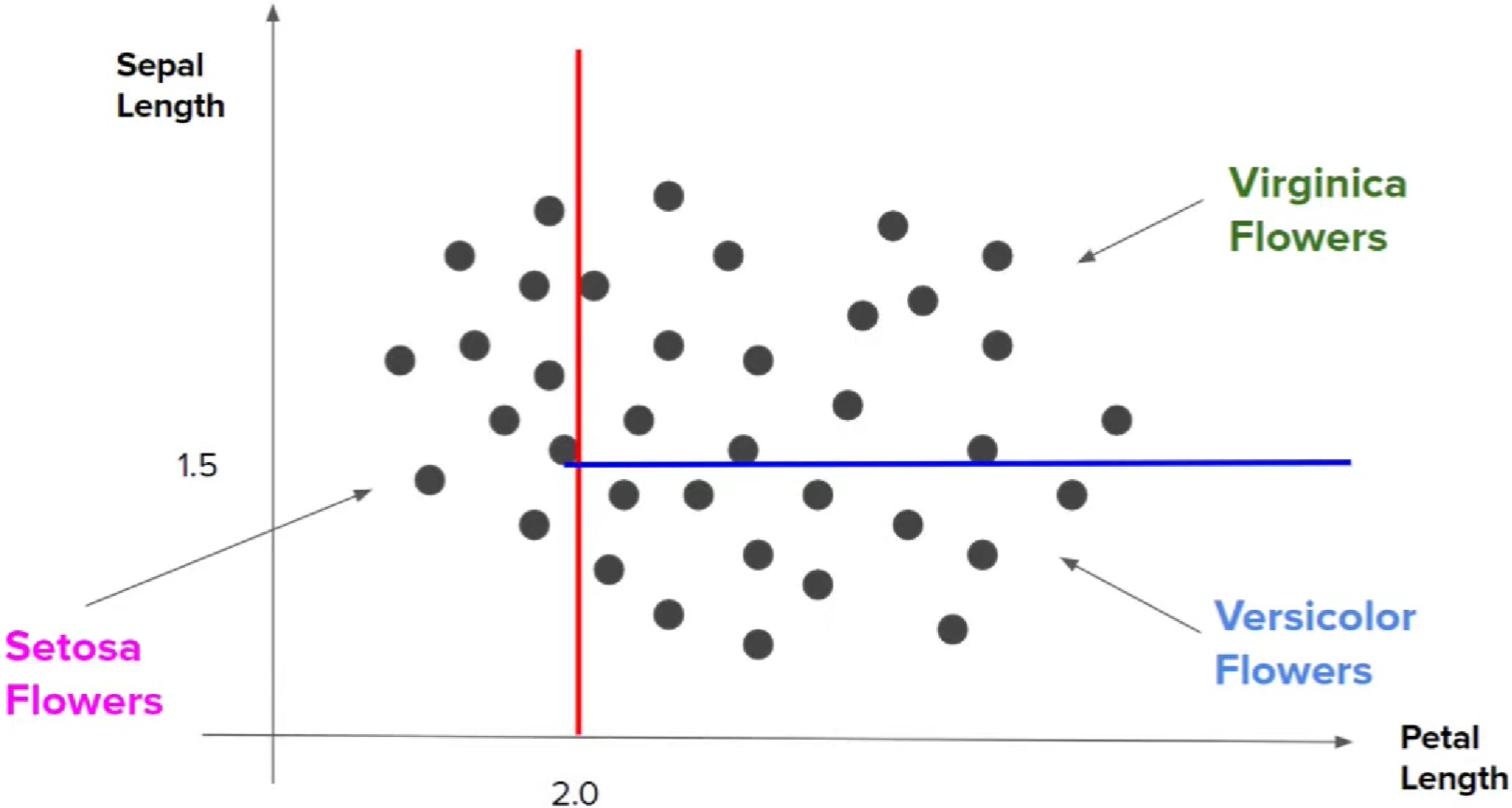
Petal Length	Sepal Length	Type
1.34	0.34	Setosa
3.45	1.45	Versicolor
1.69	0.98	Setosa
2.56	1.79	Virginica
3.00	1.13	Versicolor
1.3	0.88	Setosa



Geometric Intuition



Geometric Intuition



Pseudo code

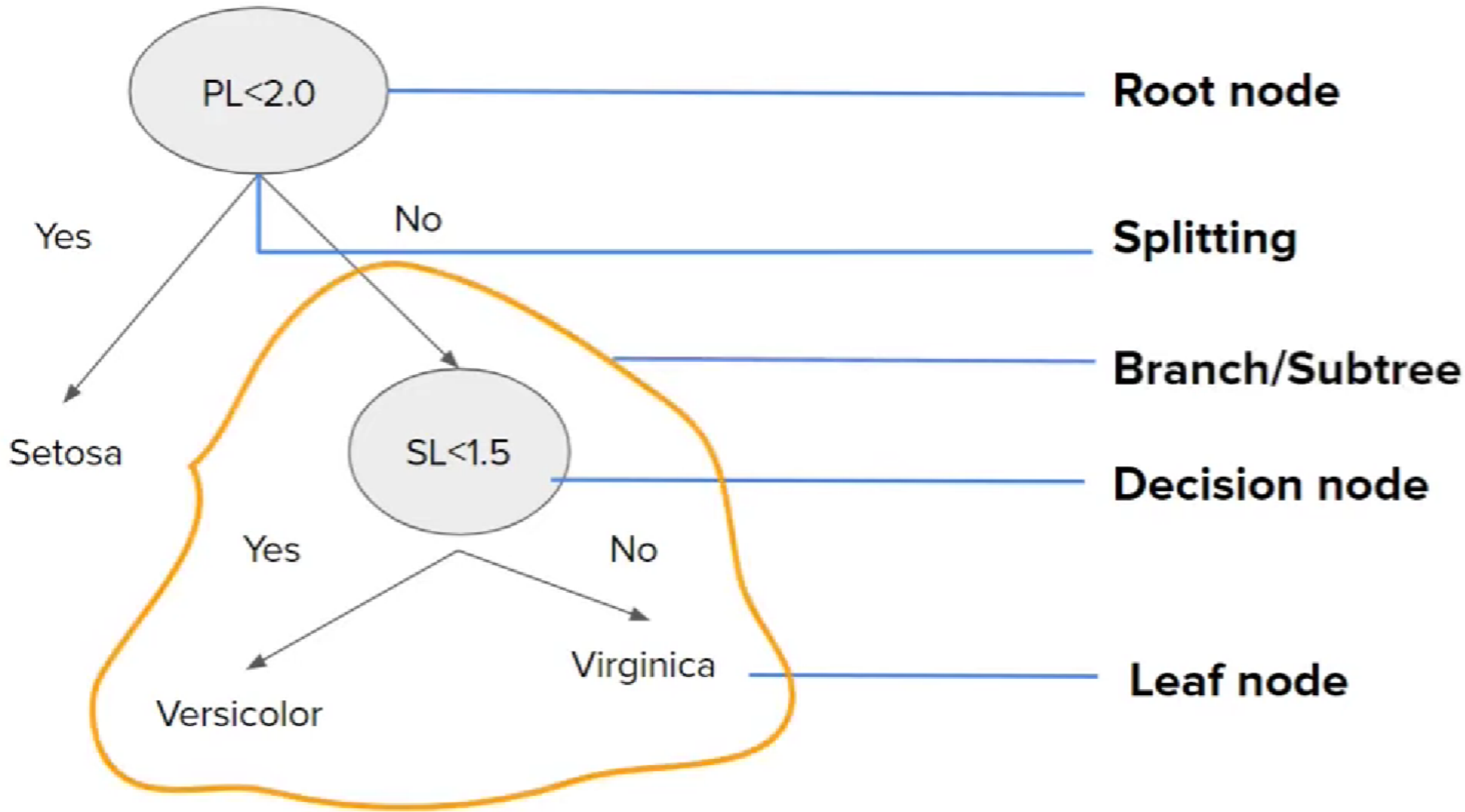
- Begin with your training dataset, which should have some feature variables and classification or regression output.
- Determine the “best feature” in the dataset to split the data on; more on how we define “best feature” later
- Split the data into subsets that contain the correct values for this best feature. This splitting basically defines a node on the tree i.e each node is a splitting point based on a certain feature from our data.
- Recursively generate new tree nodes by using the subset of data created from step 3.

Conclusion

Programatically speaking, Decision trees are nothing but a giant structure of nested if-else condition

Mathematically speaking, Decision trees use hyperplanes which run parallel to any one of the axes to cut your coordinate system into hyper cuboids

Terminology



Some unanswered questions

How to decide which column should be considered as root node?

How to select subsequent decision nodes?

How to decide splitting criteria in case of numerical columns?

What is Entropy?

In the most layman terms, Entropy is nothing but the measure of disorder. Or you can also call it the measure of purity/impurity. Let's see an example...

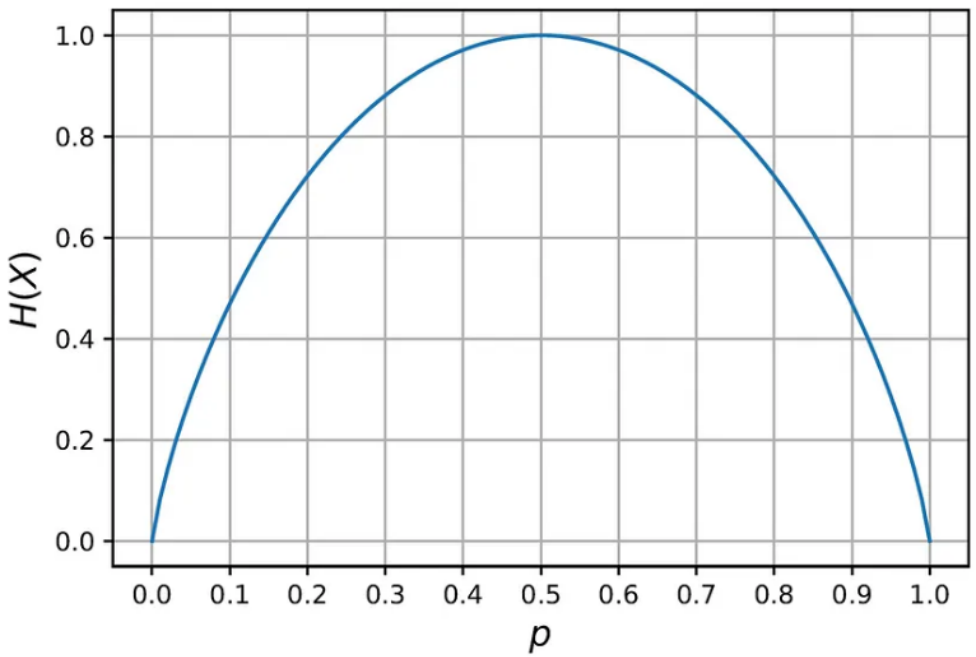
Entropy

Entropy of a random variable is a measure of the uncertainty in the variable's possible outcomes. The more uncertain we are about the value of the variable, the higher its entropy.

Formally, let X be a random variable with values x_1, x_2, ... , x_n, and let's denote the probability that X gets the value x_i by $p(x_i)$. Then the entropy of X is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Definition of entropy



Entropy of a Bernoulli variable

Entropy in Decision Trees

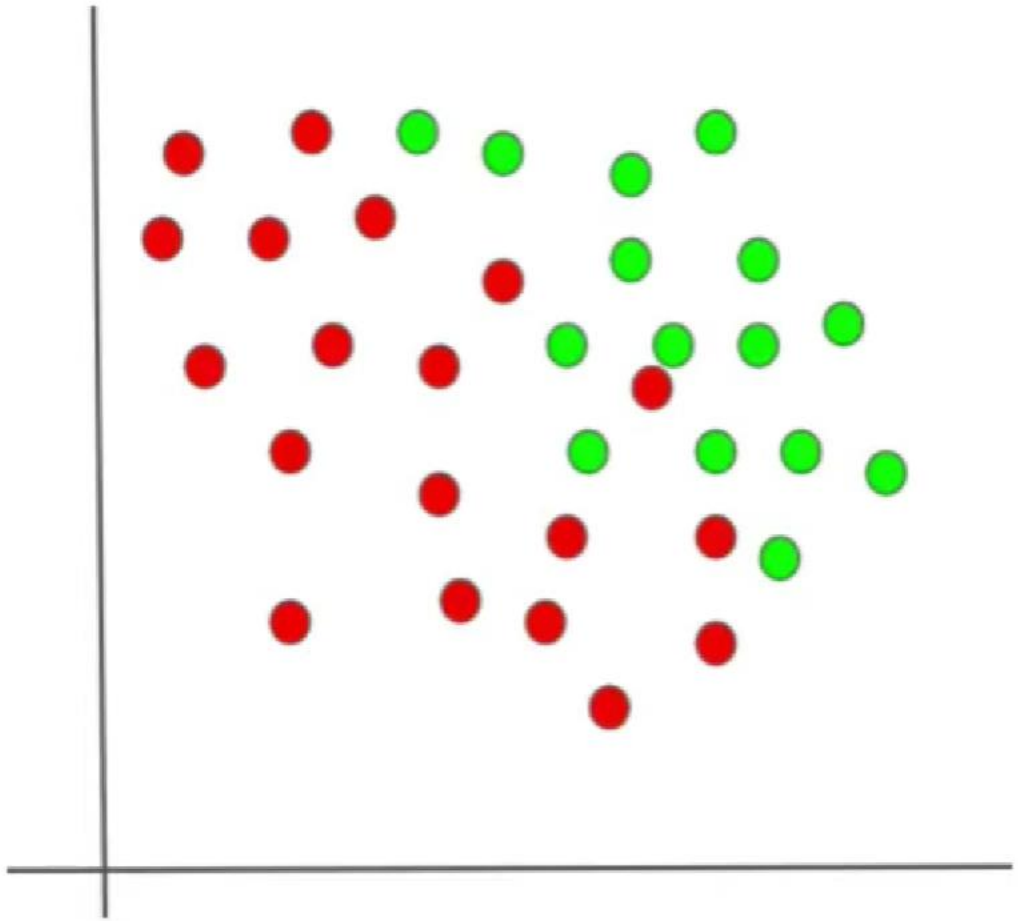
Now that we understand the basic definition of entropy, we can define the **entropy of a given node** in the decision tree.

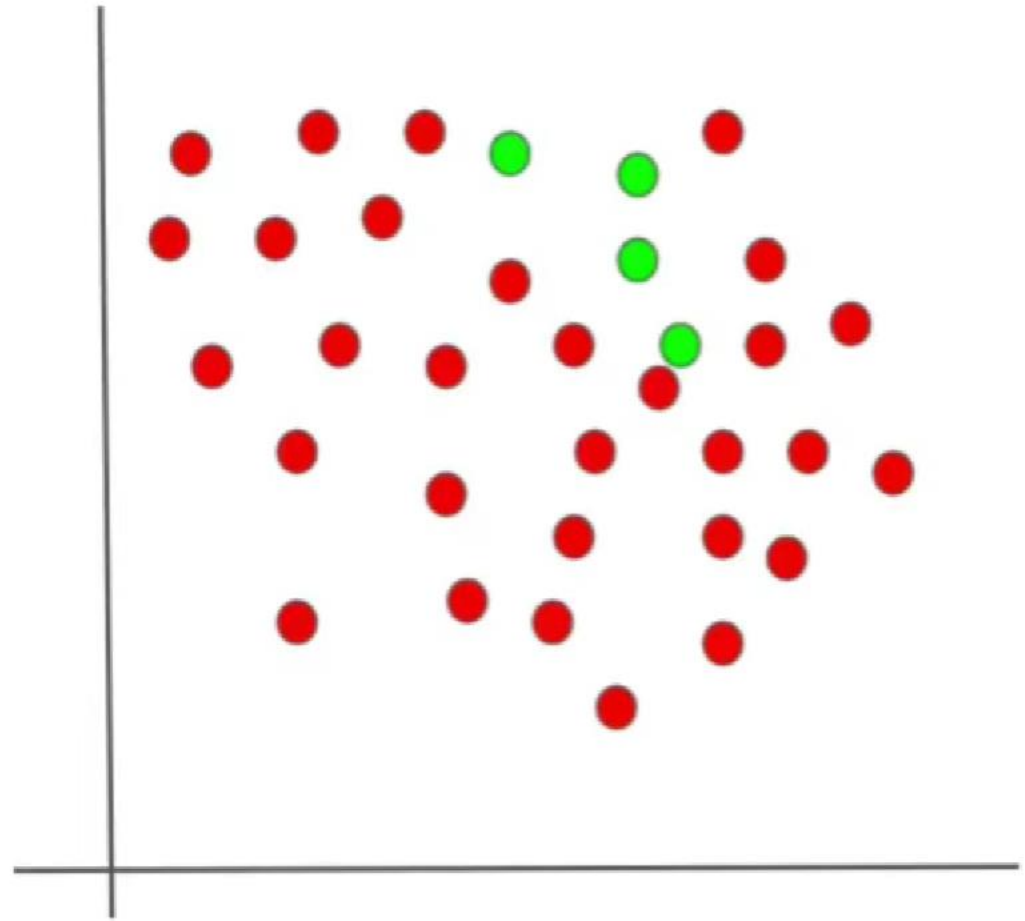
Suppose that we have a classification problem with k classes: C_1, C_2, ..., C_k. Let the fraction of training samples in node v that belong to class C_i be $P(C_i|v)$. This quantity represents the probability that a sample that reaches node v belongs to class C_i.

Then, the entropy of node v is defined as:

$$H(v) = -\sum_{i=1}^{k} P(C_i|v) \log_2 P(C_i|v)$$

The entropy of node v in the tree





How to calculate Entropy?

The mathematical formula for entropy is:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Where 'Pi' is simply the frequentist probability of an element/class 'i' in our data.

For e.g if our data has only 2 class labels **Yes** and **No.**

$$\mathbf{E(D) = -p_{yes} log_2(p_{yes}) - p_{no} log_2(p_{no})}$$

Example 3 - Dataset

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$$H(d)= -P_y \log_2(P_y) -P_n \log_2(P_n)$$

$$H(d)= -2/5 \log_2(2/5) -3/5 \log_2(3/5)$$

$$H(d)= 0.97$$

$$H(d)= -P_y \log_2(P_y) -P_n \log_2(P_n)$$

$$H(d)= -1/5 \log_2(1/5) -4/5 \log_2(4/5)$$

$$H(d)= 0.72$$

Salary	Age	Purchase
20000	21	No
10000	45	No
60000	27	No
15000	31	No
12000	18	No

$$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$$

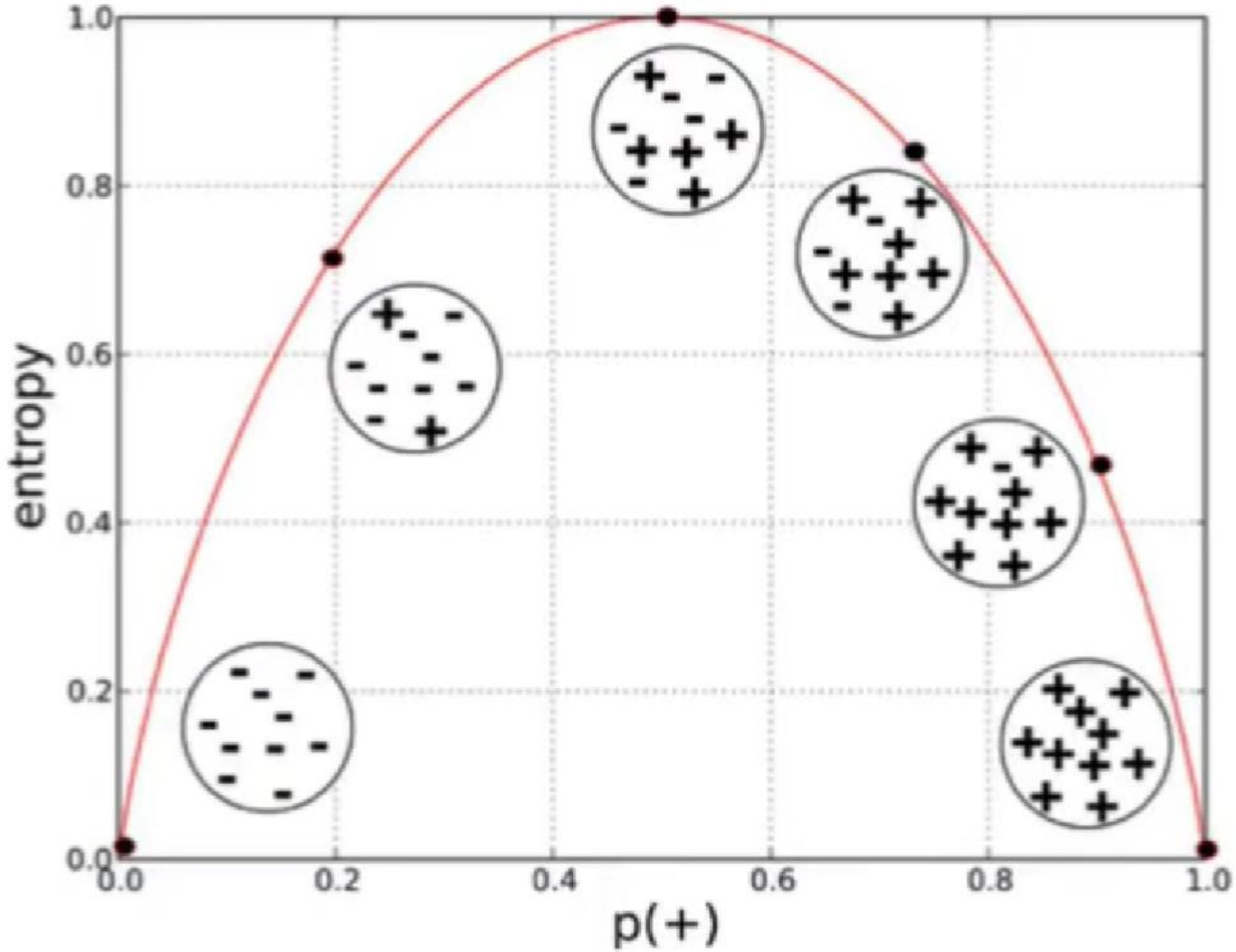
$$H(d) = -0/5 \log_2(0/5) - 5/5 \log_2(5/5)$$

$$H(d) = 0$$

Observation

- More the uncertainty more is entropy
- For a 2 class problem the min entropy is 0 and the max is 1
- For more than 2 classes the min entropy is 0 but the max can be greater than 1
- Both $\log_2$ or $\log_e$ can be used to calculate entropy

Entropy Vs Probability



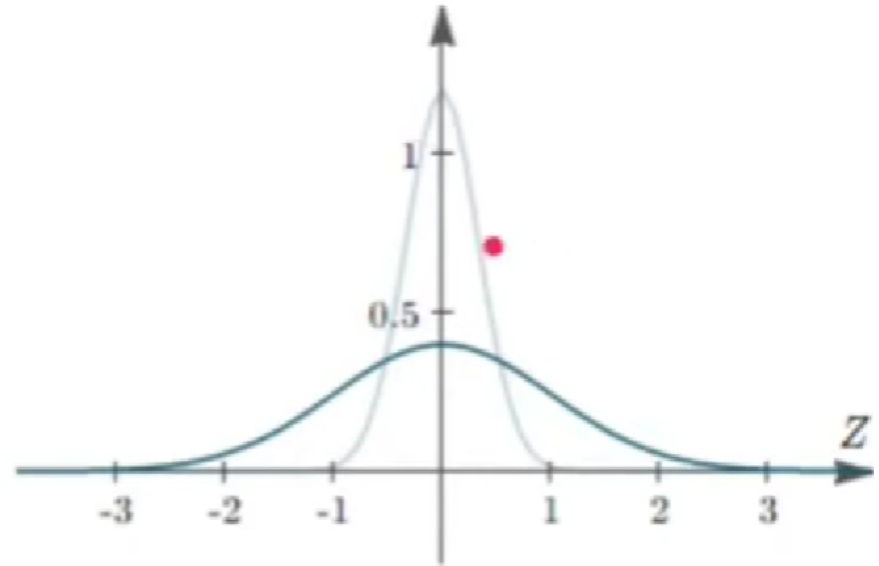
Entropy for continuous variables

Area	Built in	Price
1200	1999	3.5
1800	2011	5.6
1400	2000	7.3
...	...	...

Dataset 1

Area	Built in	Price
2200	1989	4.6
800	2018	6.5
1100	2005	12.8
...	...	...

Dataset 2



Quiz: Which of the above datasets have higher entropy?

Information Gain

Information Gain, is a metric used to train Decision Trees. Specifically, this metric measures the quality of a split.

The information gain is based on the decrease in entropy after a data-set is split on an attribute.Constructing a decision tree is all about finding attribute that returns the highest information gain

Information Gain = E(Parent) - {Weighted Average} * E{Children}

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Step 1:

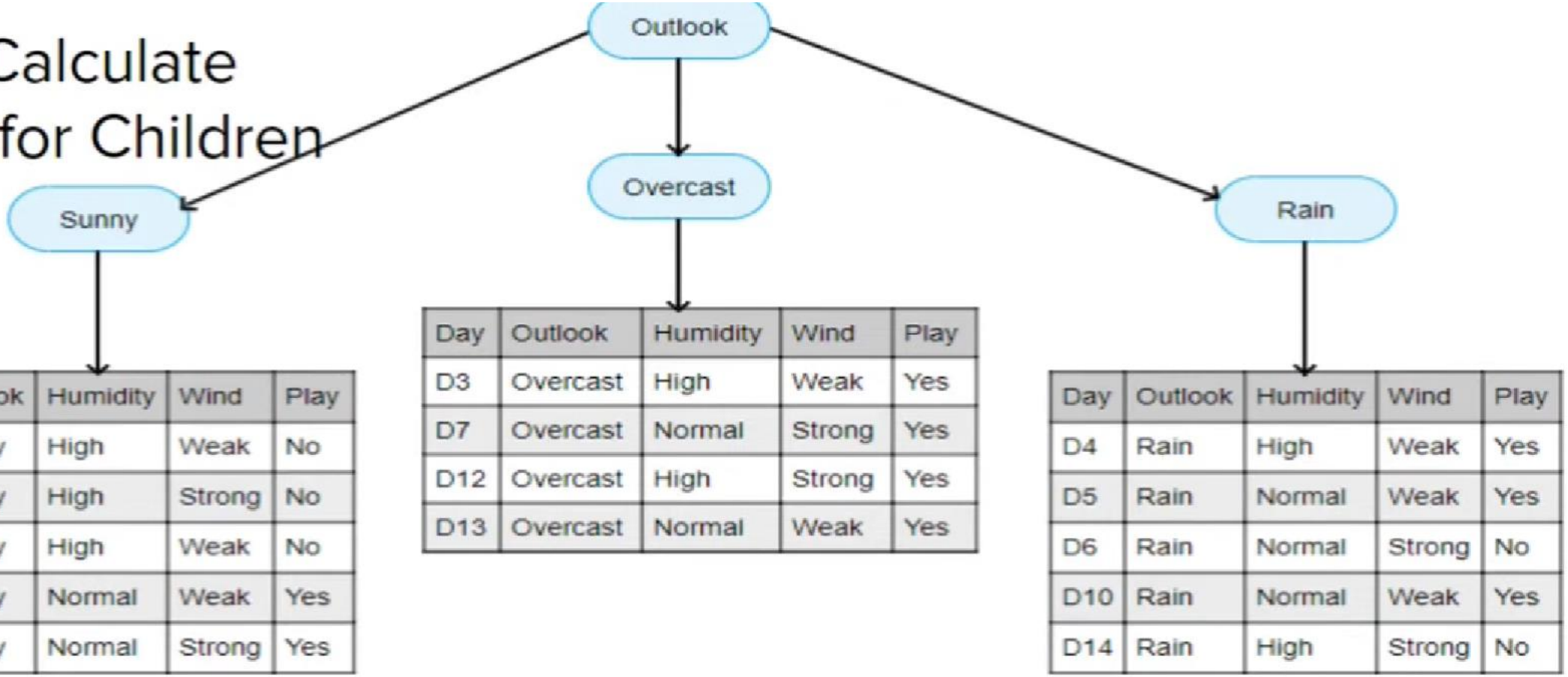
Entropy of Parent

$E(P) = -p_y \log_2(p_y) - p_n \log_2(p_n)$

$= 9/14 \log_2(9/14) - 5/14 / \log_2(5/14)$

$E(P) = \mathbf{0.94}$

Step 2: Calculate Entropy for Children



Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes

Day	Outlook	Humidity	Wind	Play
D3	Overcast	High	Weak	Yes
D7	Overcast	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes

Day	Outlook	Humidity	Wind	Play
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D10	Rain	Normal	Weak	Yes
D14	Rain	High	Strong	No

E(S) = -2/5log(2/5) - 3/5log(3/5) E(S)= 0.97	E(O) = -5/5log(5/5) - 0/5log(0/5) E(O)= 0	E(R) = -3/5log(3/5) - 2/5log(2/5) E(S)= 0.97

Step 3 : Calculate weighted Entropy of Children

Weighted Entropy = 5/14 * 0.97 + 4/14 * 0 + 5/14 * 0.97

W.E(Children) = **0.69**

P(Overcast) is a leaf node as it's entropy is 0

Step 4 : Calculate Information Gain

Information Gain = E(Parent) - {Weighted Average} * E(Children)

IG = **0.97 - 0.69 = 0.28**

So the information gain(or the decrease in entropy/impurity) when you split this data on the basis of **Outlook** condition/column is **0.28**

Step 5 : Calculate Information Gain for all the columns

Whichever column has the highest Information Gain(maximum decrease in entropy) the algorithm will select that column to split the data.

Step 6 : Find Information Gain recursively

Decision tree then applies a recursive greedy search algorithm in top bottom fashion to find Information Gain at every level of the tree.

Once a leaf node is reached (Entropy = 0), no more splitting is done.

Decision Trees

Gini Impurity

$$\text{Entropy} = -\sum_{i=1}^{n} p_i * \text{Log}_2(p_i)$$

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$G = 1 - (Py^2 + Pn^2)$

$G = 1 - (4/25 + 9/25)$

$G = 0.48$

$G = 1 - (Py^2 + Pn^2)$

$G = 1 - (1/25 + 16/25)$

$G = 0.32$

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassifier**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation.

splitter : *{"best", "random"}, default="best"*

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

Advantages

Intuitive and easy to understand

Minimal data preparation is required

The cost of using the tree for inference is **logarithmic** in the number of data points used to train the tree

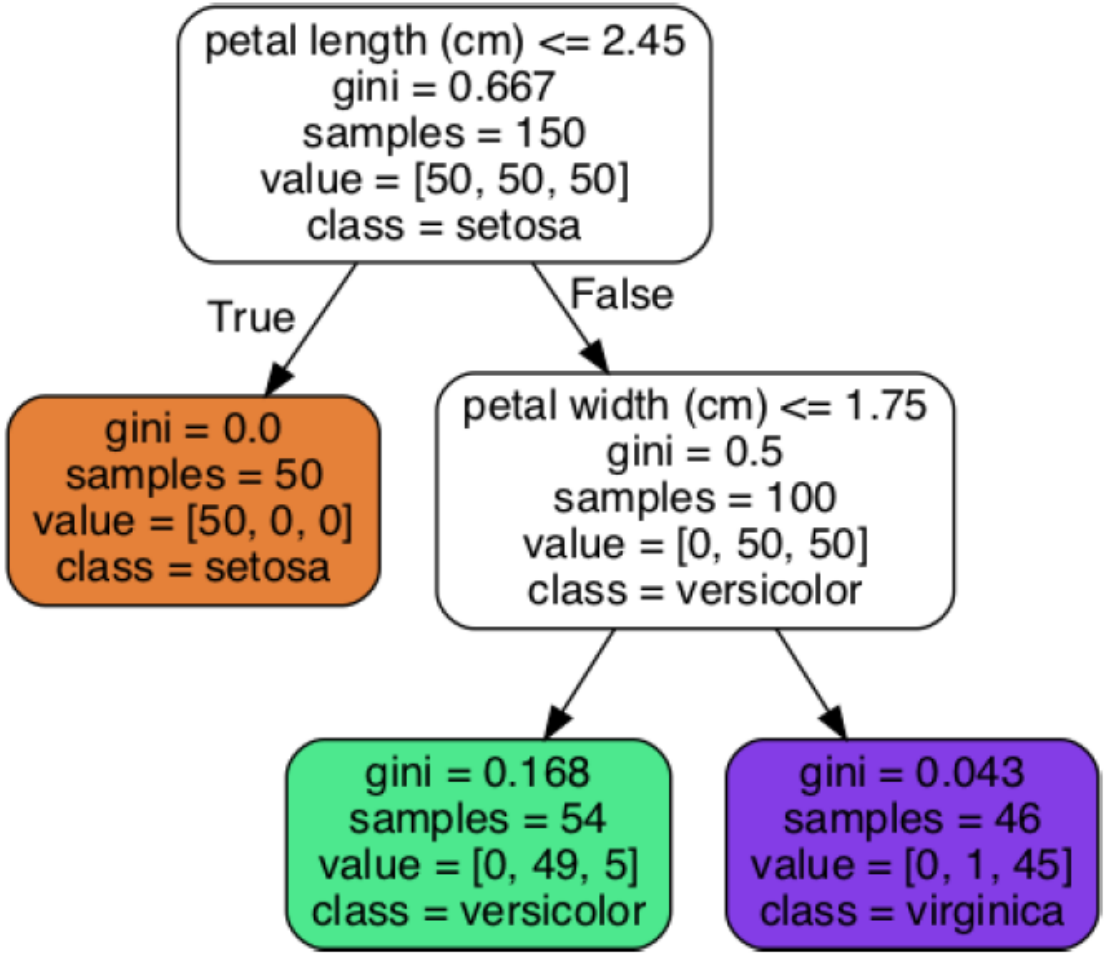
Disadvantages

Overfitting

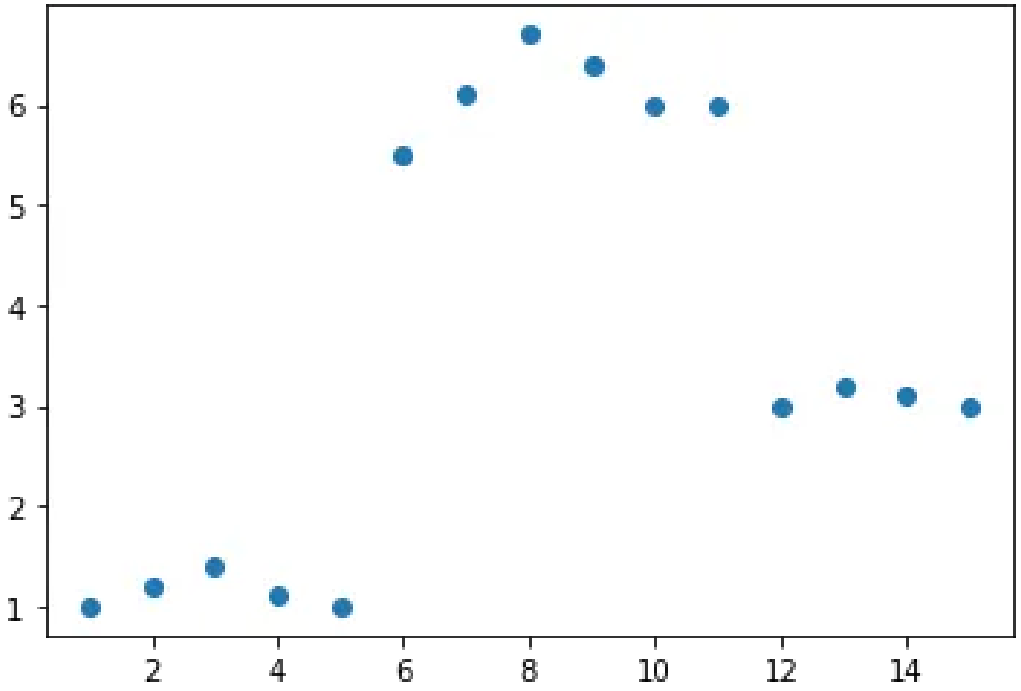
Prone to errors for imbalanced datasets

Iris dataset

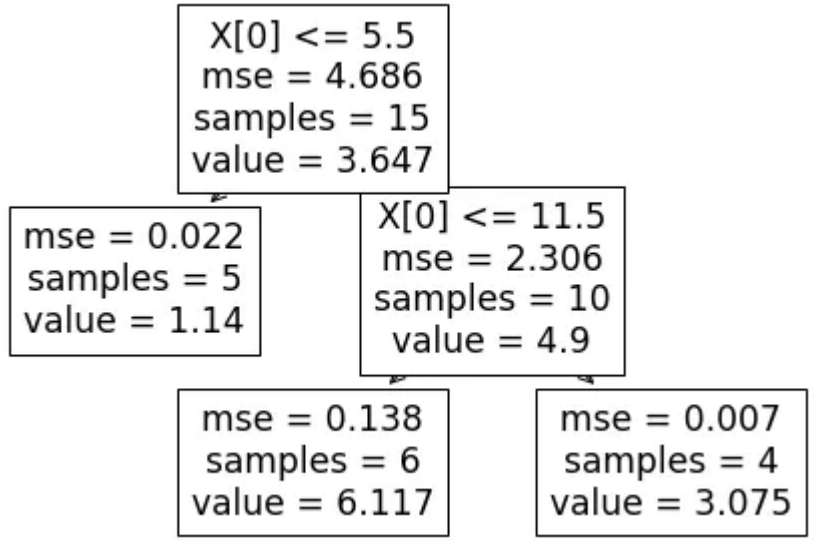
Petal Length	Sepal Length	Type
1.34	0.34	Setosa
3.45	1.45	Versicolor
1.69	0.98	Setosa
2.56	1.79	Virginica
3.00	1.13	Versicolor
1.3	0.88	Setosa

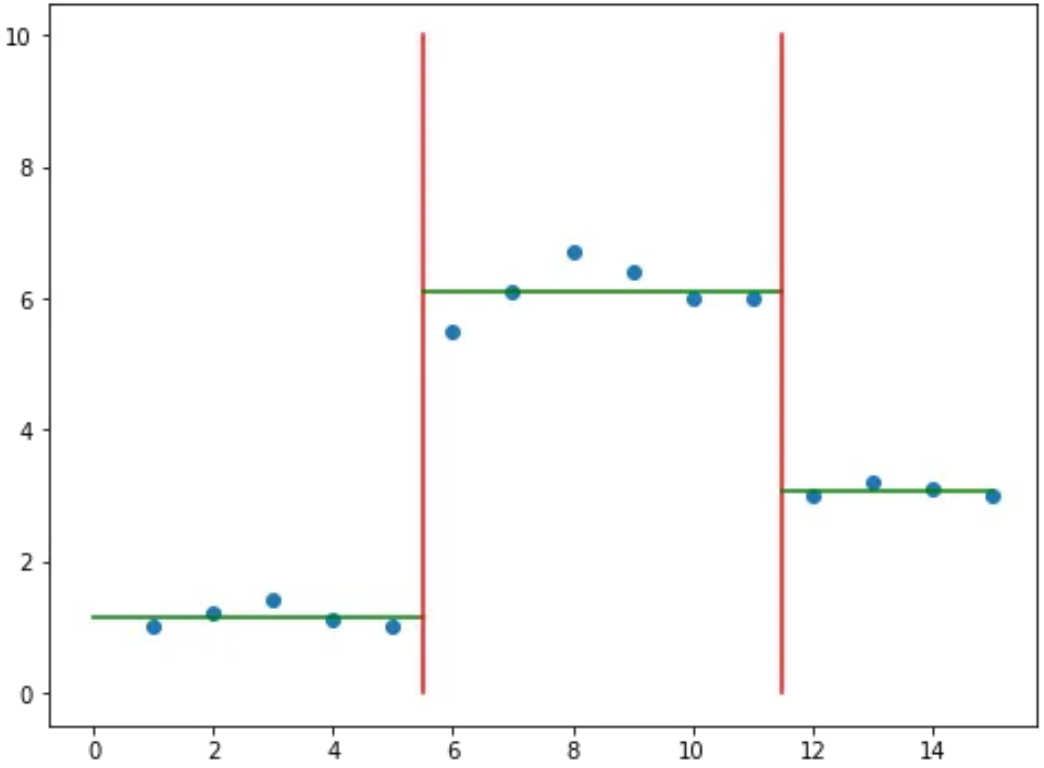


Building a Regression Tree



X	Y
1	1
2	1.2
3	1.4
4	1.1
5	1
6	5.5
7	6.1
8	6.7
9	6.4
10	6
11	6
12	3
13	3.2
14	3.1





How Does CART Work in Regression with one predictor?

CART in classification cases uses Gini Impurity in the process of splitting the dataset into a decision tree. On the other hand CART in regression cases uses least squares, intuitively splits are chosen to minimize the **residual sum of squares** between the observation and the mean in each node.
Mathematically, we can write residual as follow

$$\varepsilon_i = y_i - \hat{y}_i$$

Mathematically, we can write **RSS (residual sum of squares)** as follow

A metric to build a regression tree

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$RSS = \varepsilon_1^2 + \varepsilon_2^2 + .. + \varepsilon_n^2$$

In order to find out the "best" split, we must minimize the RSS

Building a Regression Tree

Step 1

The first step is to sort the data based on X (In this case, it is already sorted). Then, take the average of the first 2 rows in variable X (which is (1+2)/2 = 1.5 according to the given dataset). Divide the dataset into 2 parts (Part A and Part B) , separated by $x < 1.5$ and $X \geq 1.5$.

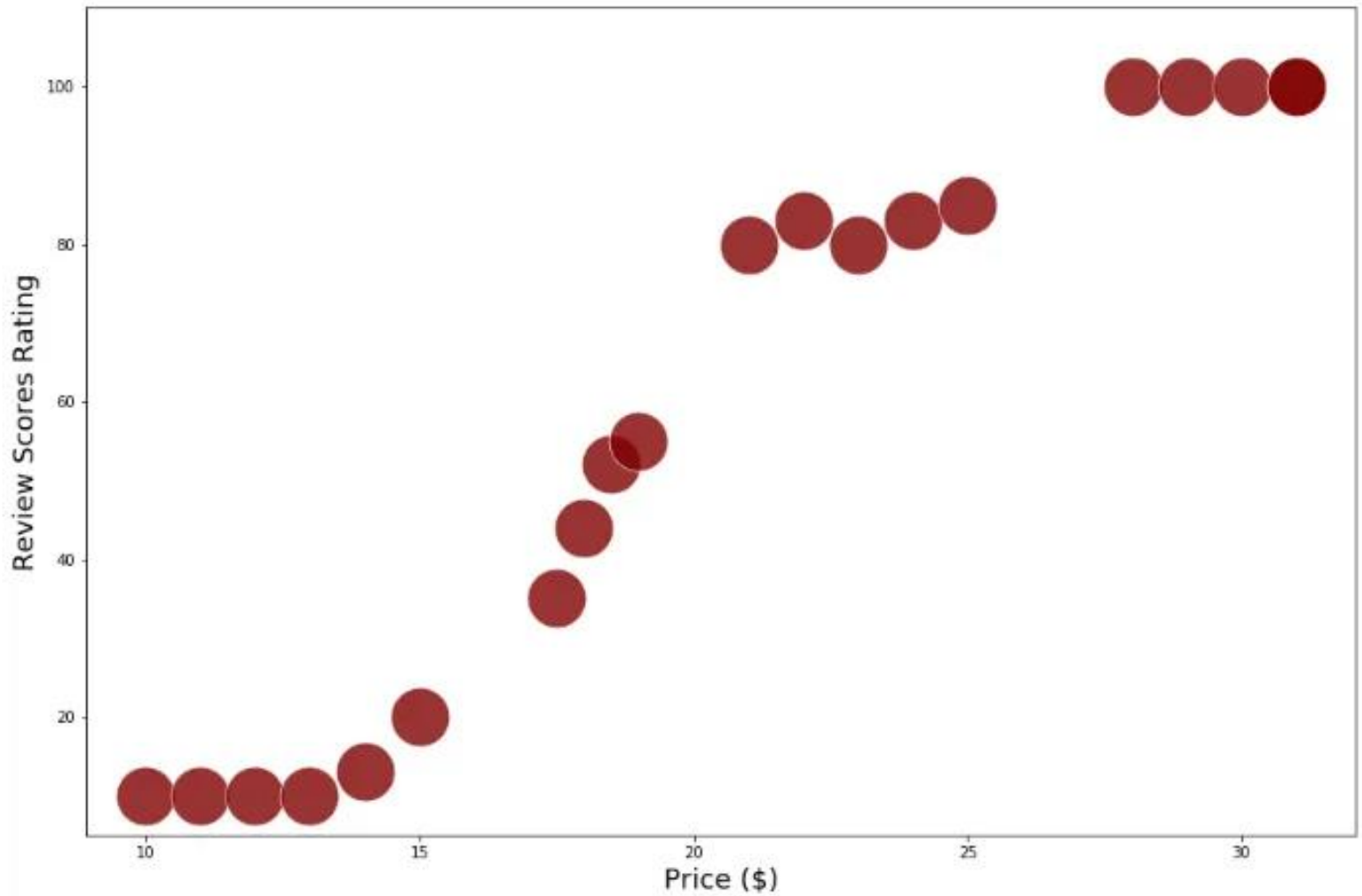
Step 2

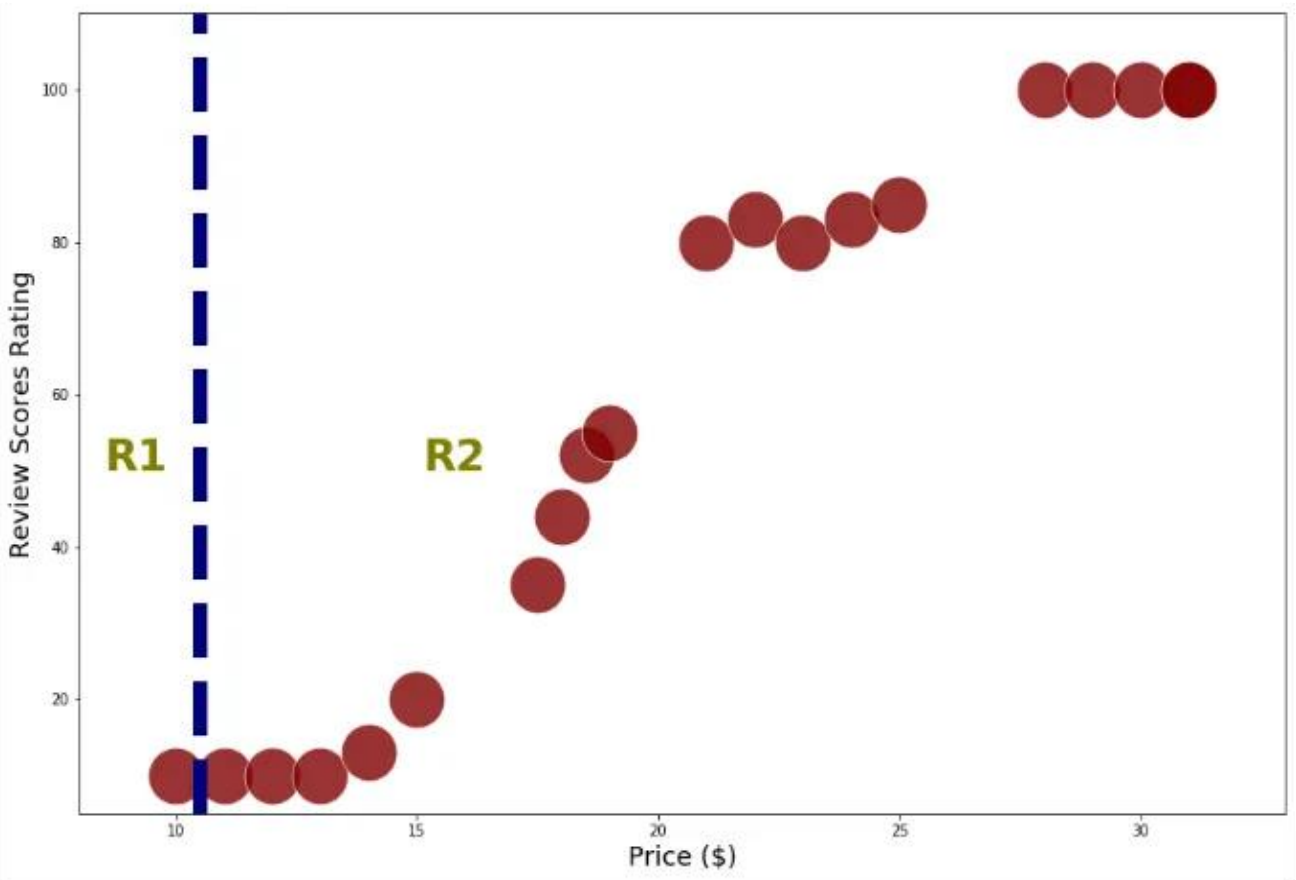
In step 1, we calculated the average for the first 2 numbers of sorted X and split the dataset based on that and calculated the predictions. Then, we do the same process again but this time, we calculate the average for the second 2 numbers of sorted X ((2+3)/2 = 2.5).

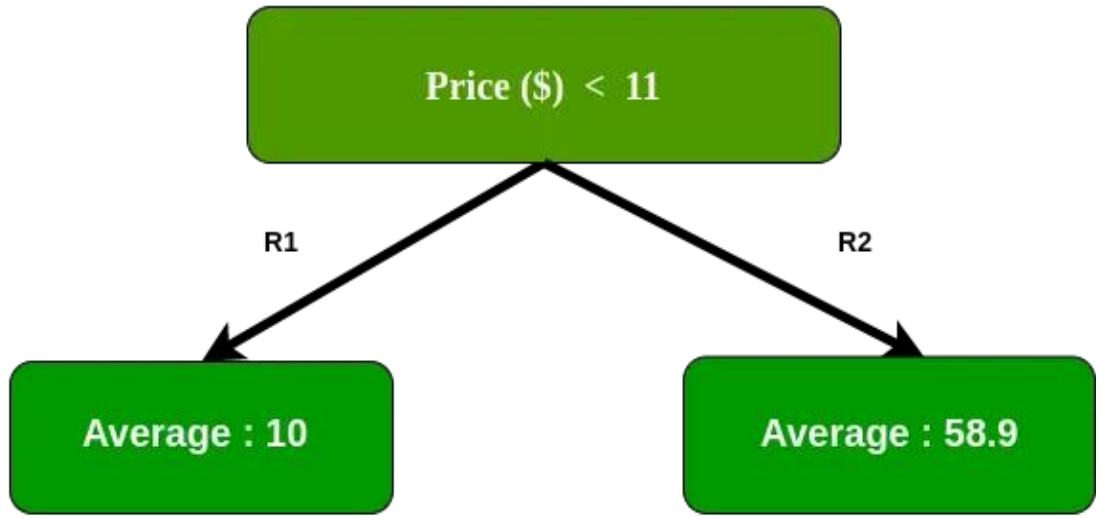
Step 3

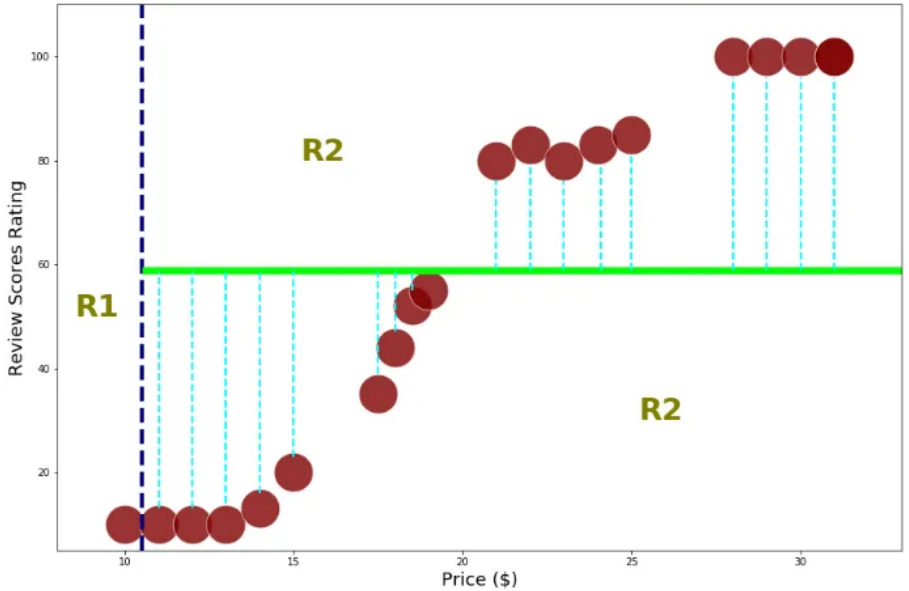
Now that we have n-1 mean squared errors calculated , we need to choose the point at which we are going to split the dataset. and that point is the point, which resulted in the lowest mean squared error on splitting at it.

	Price ($)	Review Scores Rating
0	10.0	10
1	11.0	10
2	12.0	10
3	13.0	10
4	14.0	13
5	15.0	20
6	17.5	35
7	18.0	44
8	18.5	52
9	19.0	55
10	21.0	80
11	22.0	83
12	23.0	80
13	24.0	83
14	25.0	85
15	28.0	100
16	29.0	100
17	30.0	100
18	31.0	100
19	31.0	100




R1
R2
Review Scores Rating
Price ($)
100
80
60
40
20
10
15
20
25
30


Price ($) < 11
R1
R2
Average : 10
Average : 58.9

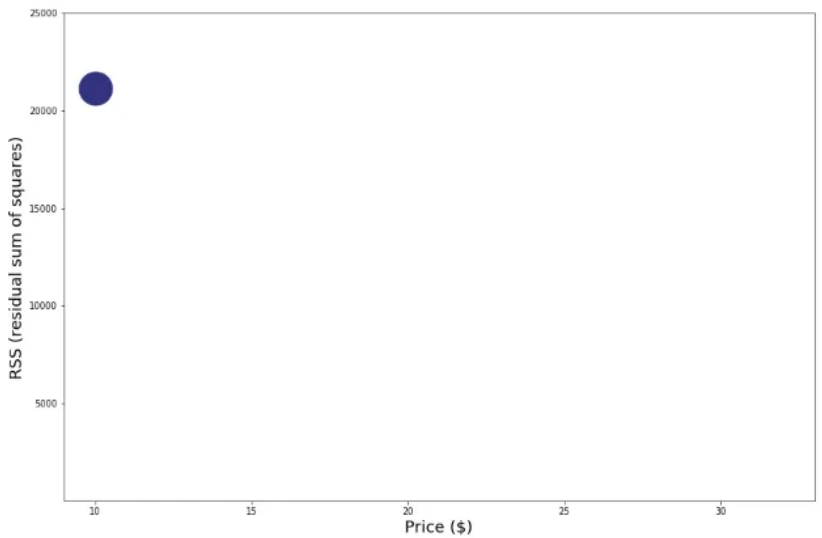


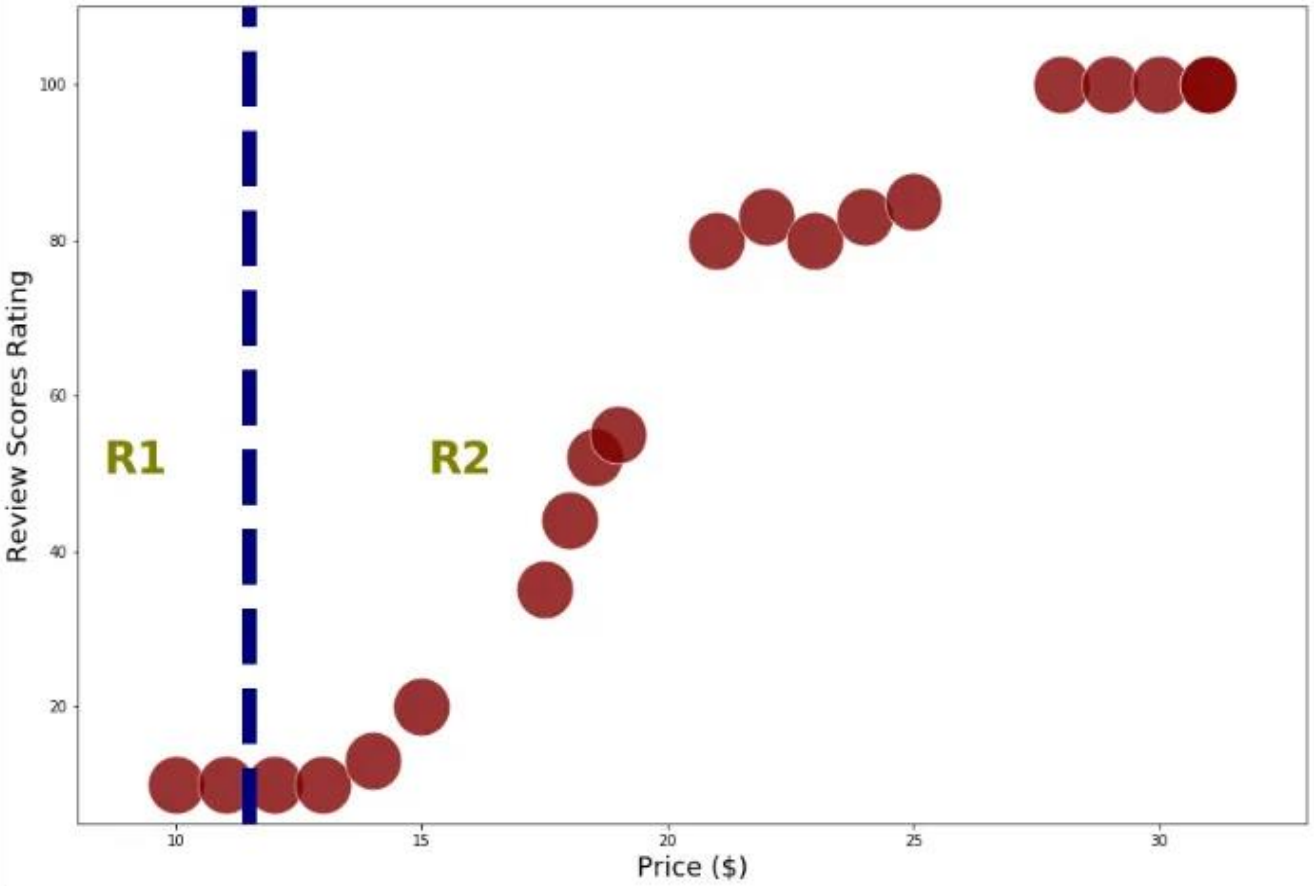
$$\varepsilon = \sum_{i=1}^{n} (actual\ value - average\ value\ in\ each\ region)^2$$

$$RSS = \varepsilon_1^2 + \varepsilon_2^2 + .. + \varepsilon_n^2$$

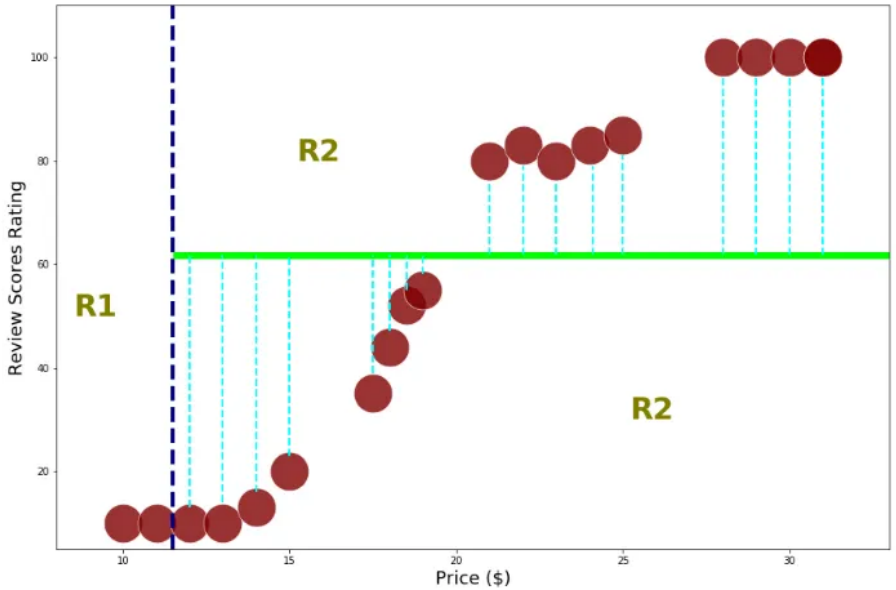
$$RSS = (10 - 10)^2 + (10 - 58.9)^2 + (10 - 58.9)^2 \ldots (10 - 58.9)^2 = 21139.78$$

The next step is RSS analysis in graphics as following




Review Scores Rating
Price ($)
R1
R2
100
80
60
40
20
10
15
20
25
30


Price ($) < 12
R1
R2
Average : 10
Average : 61.8

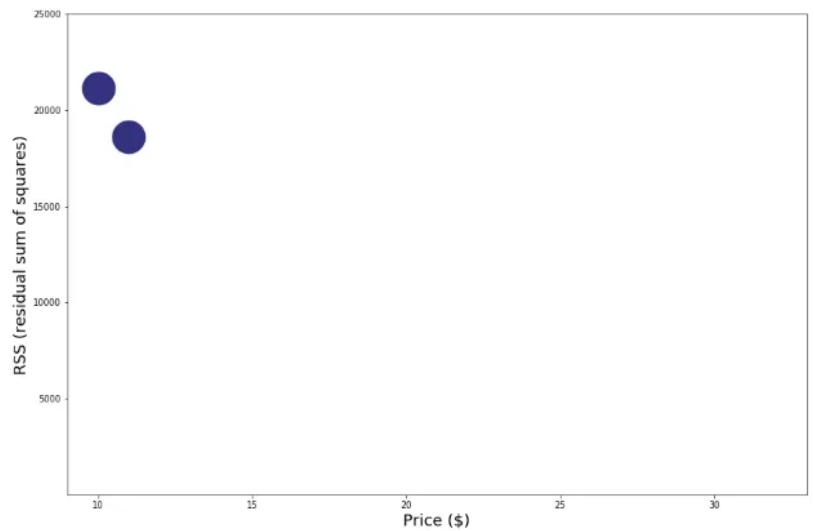


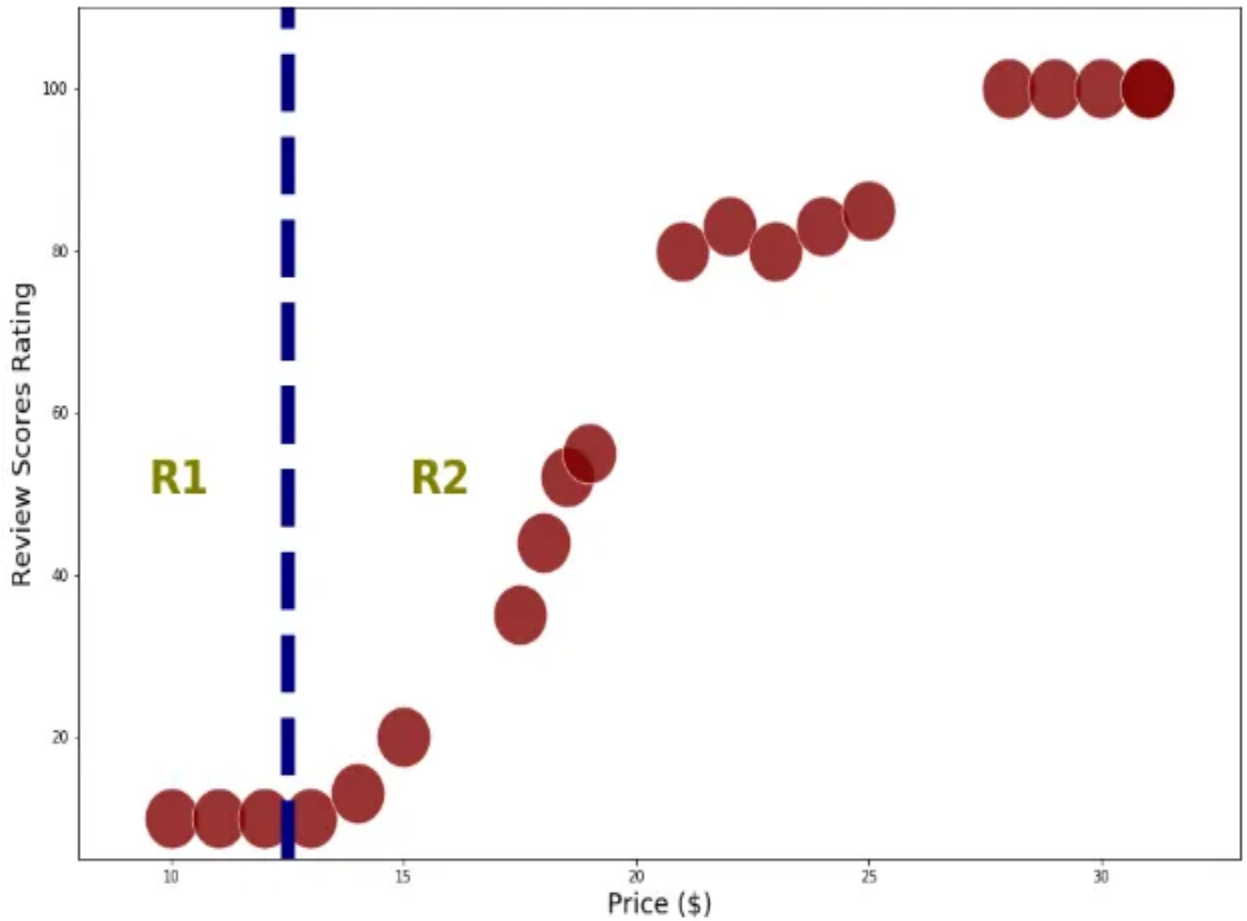
$$\varepsilon = \sum_{i=1}^{n} (\textit{actual value} - \textit{average value in each region})^2$$

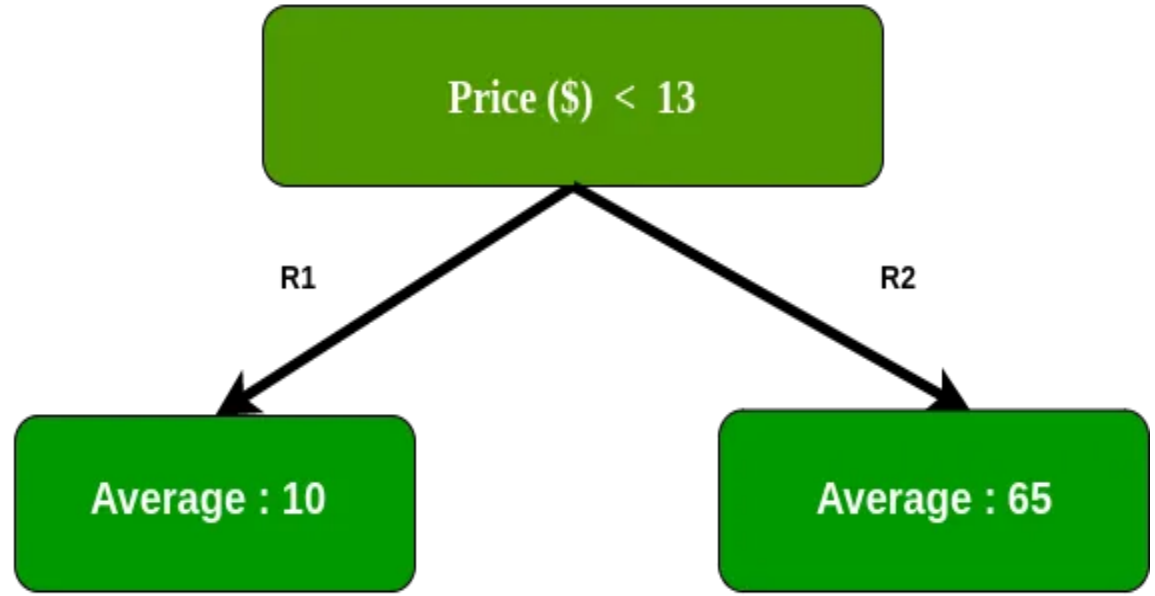
$$RSS = \varepsilon_1^2 + \varepsilon_2^2 + .. + \varepsilon_n^2$$

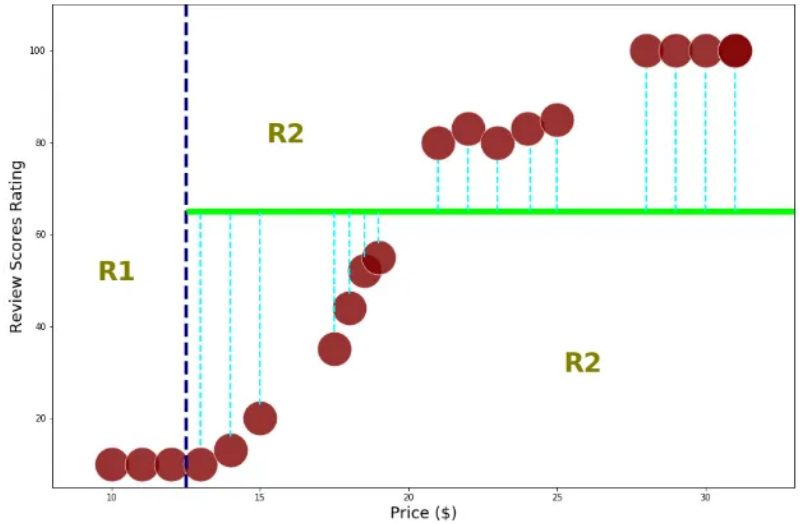
$$RSS = (10 - 10)^2 + (10 - 10)^2 + (10 - 58.9)^2 \ldots (10 - 58.9)^2 = 18609.08$$

The next step is RSS analysis in graphics as following




Review Scores Rating
100
80
60
40
20
R1
R2
10
15
20
25
30
Price ($)


Price ($) < 13
R1
R2
Average : 10
Average : 65

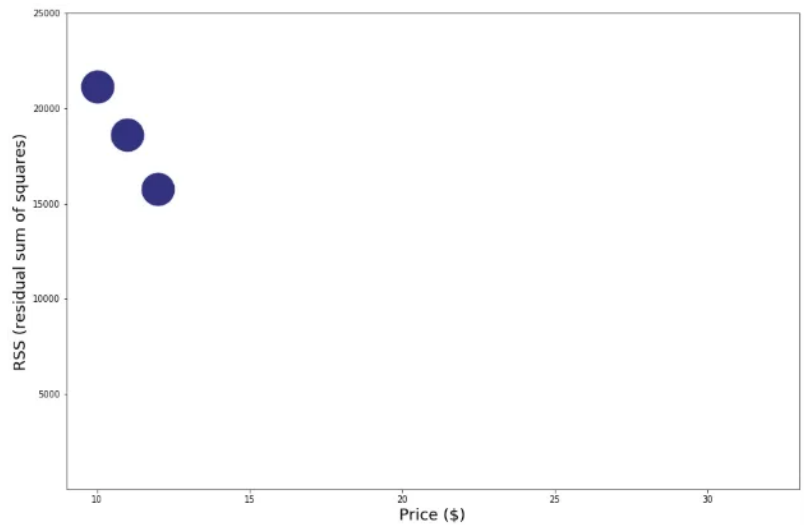


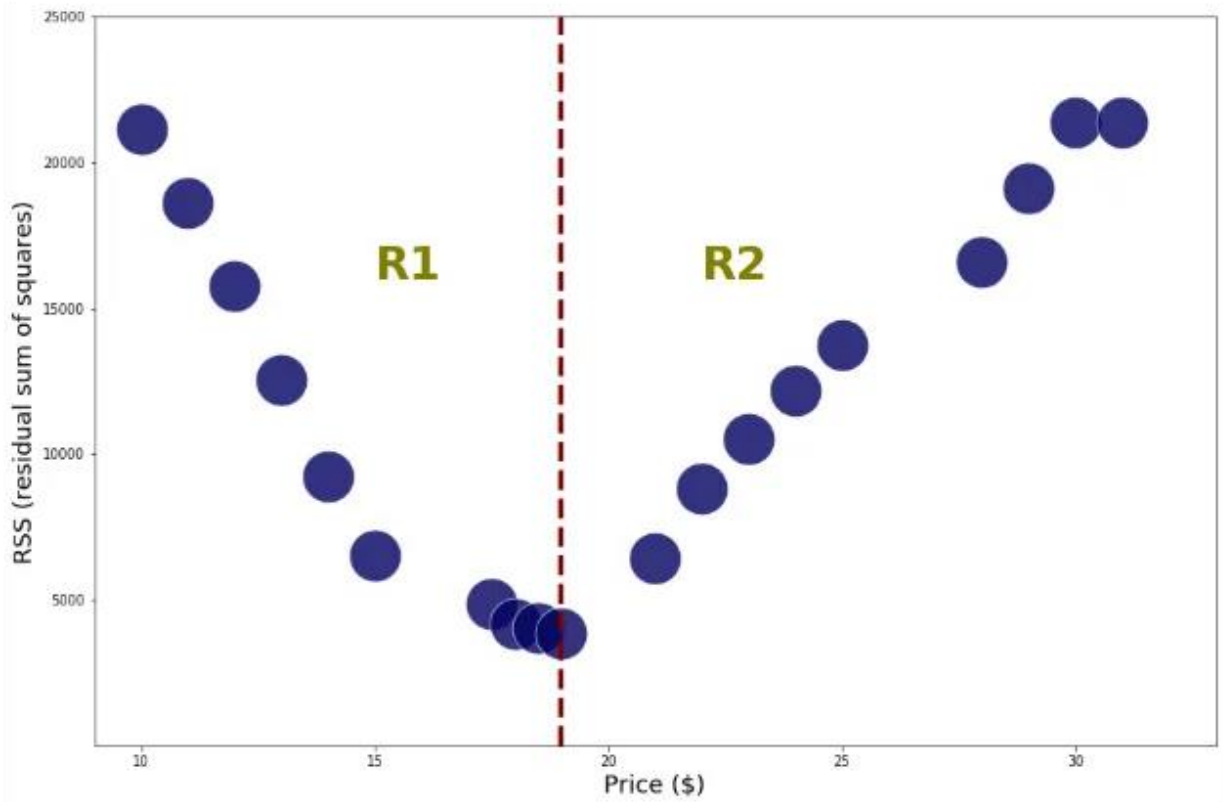
$$\varepsilon = \sum_{i=1}^{n} (actual\ value - average\ value\ in\ each\ region)^2$$

$$RSS = \varepsilon_1^2 + \varepsilon_2^2 + .. + \varepsilon_n^2$$

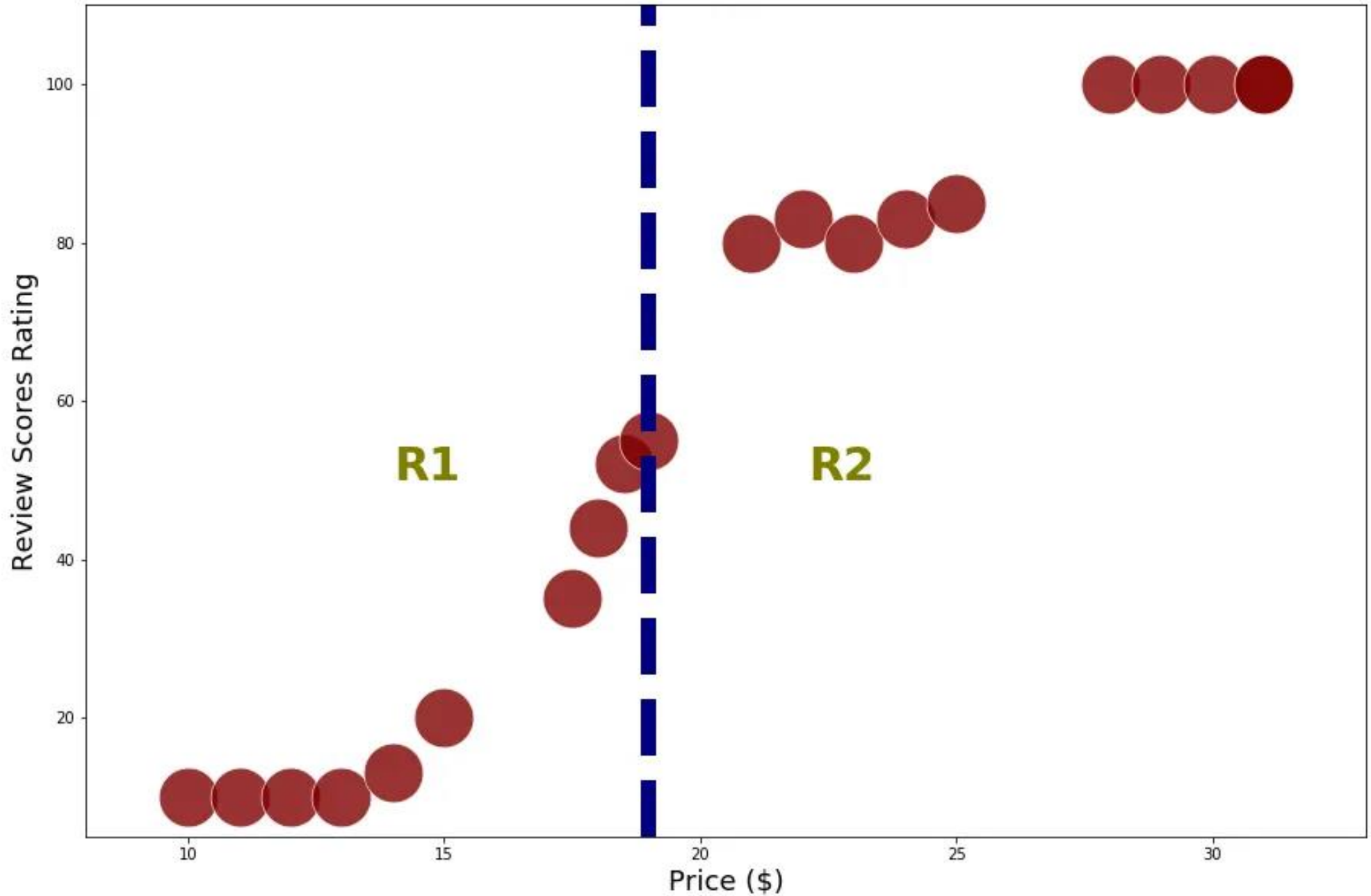
$$RSS = (10 - 10)^2 + (10 - 10)^2 + (10 - 10)^2 \ldots (10 - 58.9)^2 = 15762.0$$

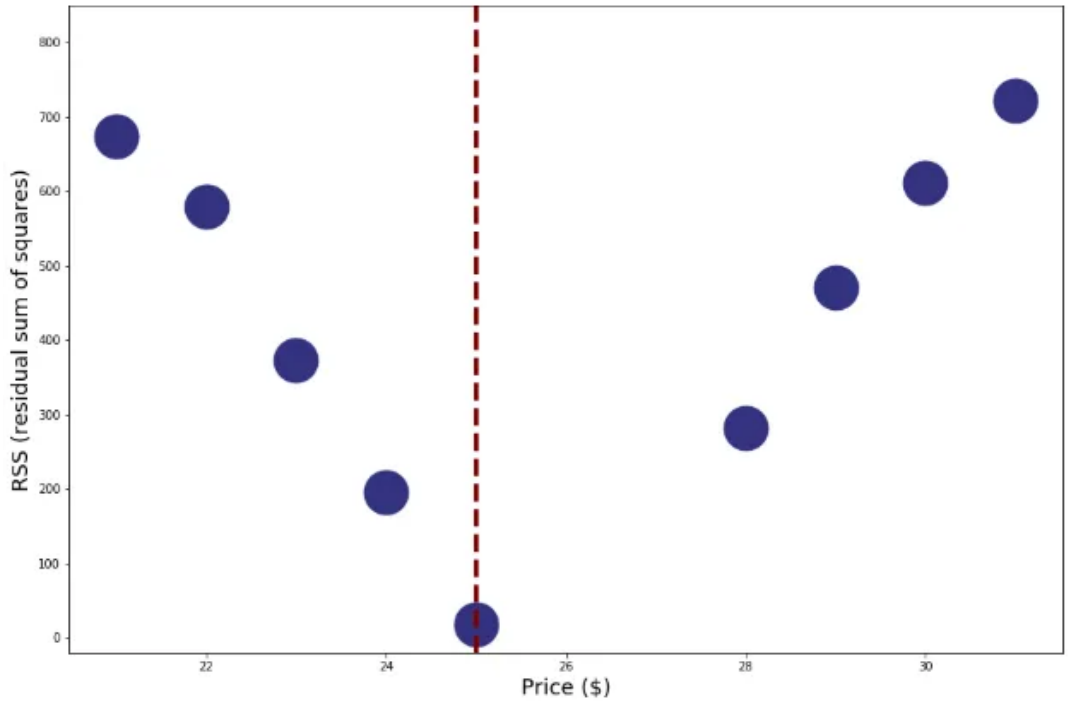
The next step is RSS analysis in graphics as following

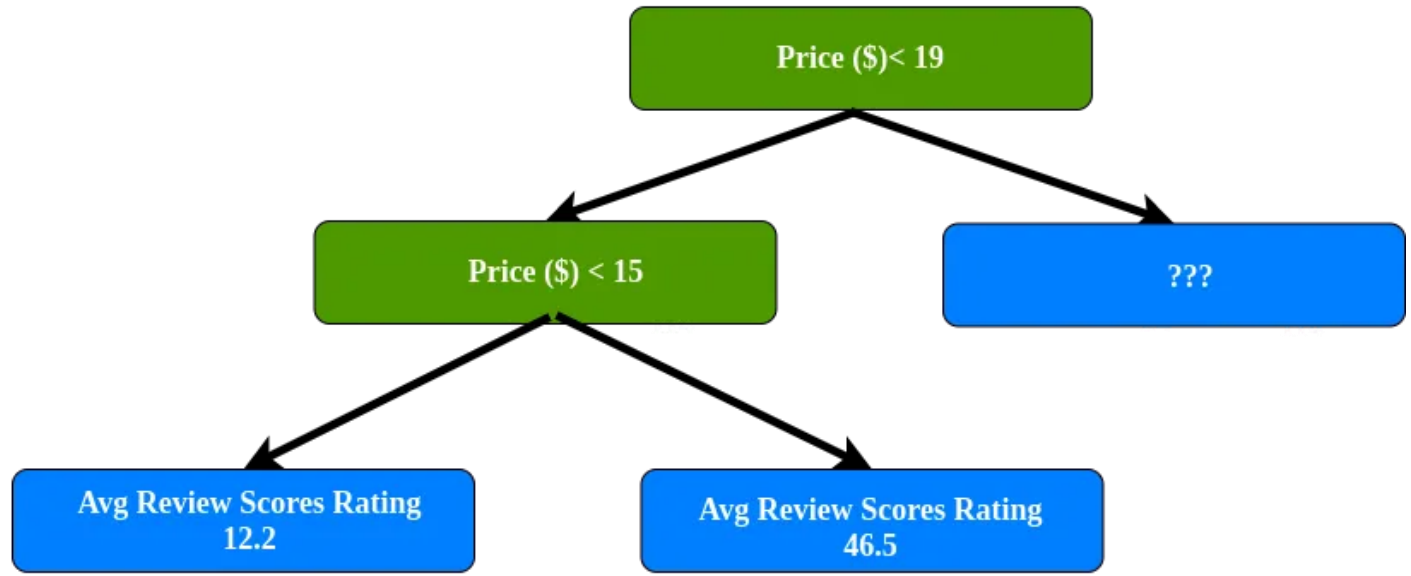



25000
20000
15000
10000
5000
RSS (residual sum of squares)
R1
R2
10
15
20
25
30
Price ($)


Price ($) < 19
Avg Review Scores Rating
25.9
Avg Review Scores Rating
91.1


Review Scores Rating
100
80
60
40
20
10
15
20
25
30
Price ($)
R1
R2


RSS (residual sum of squares)
800
700
600
500
400
300
200
100
0
22
24
26
28
30
Price ($)


Price ($)< 19
Price ($) < 15
???
Avg Review Scores Rating
12.2
Avg Review Scores Rating
46.5


Price ($) < 19
Price ($) < 15
Price ($) < 25
Avg Review Scores Rating
12.2
Avg Review Scores Rating
46.5
Avg Review Scores Rating
82.2
Avg Review Scores Rating
100