Python\AWS \ML Training:

Machine Learning
Day 3

SummitWorks™
GLOBAL SOLUTION ARCHITECTS

Authorized & published by Summitworks Technologies Inc

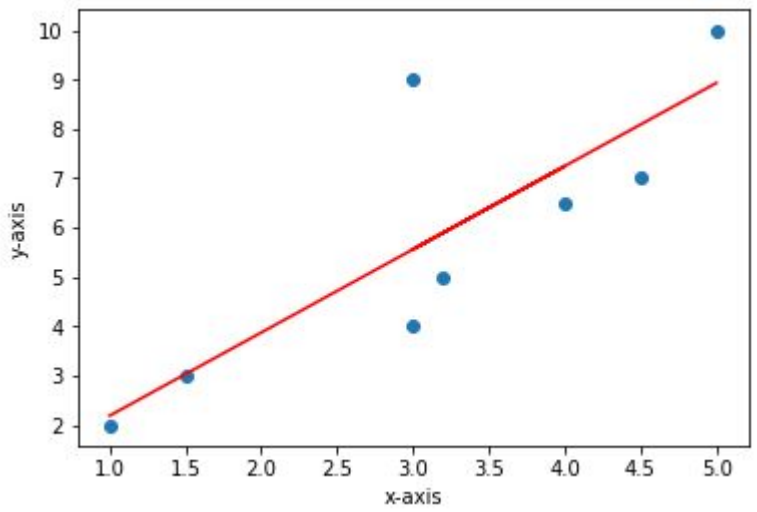
Supervised Learning 1

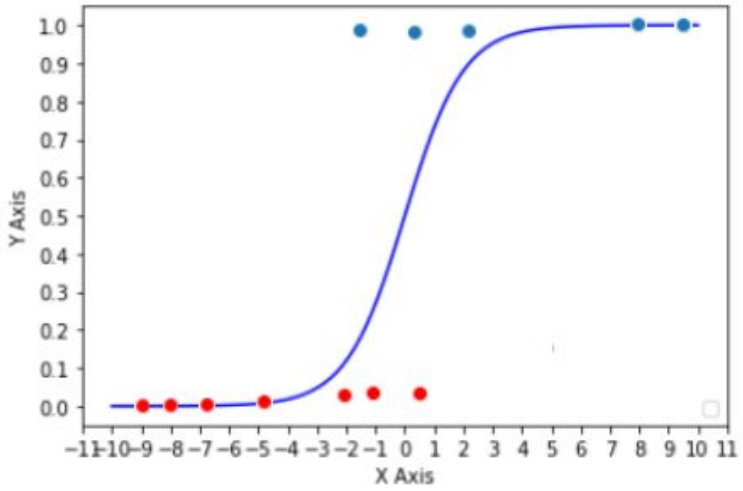
Supervised Learning 1

- Regression
- Types of regression
- Linear Progression
  - Disadvantage of Linear Regression
- Logistic Regression
  - Logistic Regression Curve
  - Logistic Regression Equation
- Polynomial Regression
- Decision Tree
- Building Decision Tree
- Random Forest
- Advantages of Random Forest
- Random Forest with Python
- Building the Random Forest Classifier

What is regression?

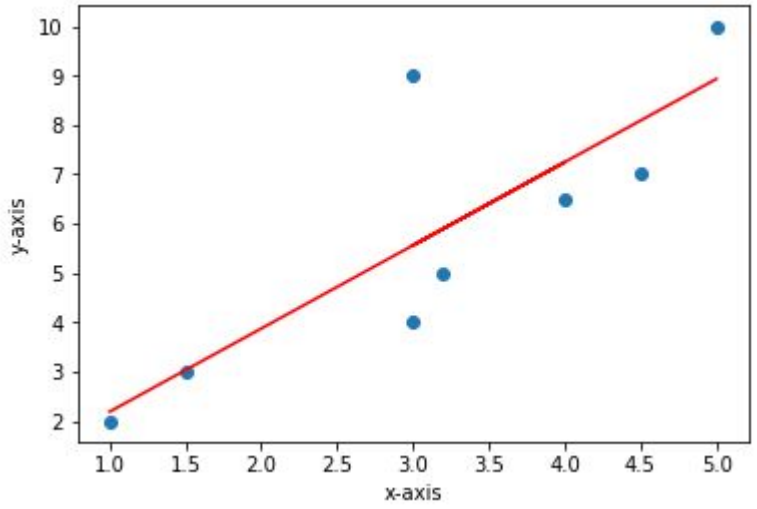
- A measure of the relation between the mean value of one variable (e.g. output) and corresponding values of other variables

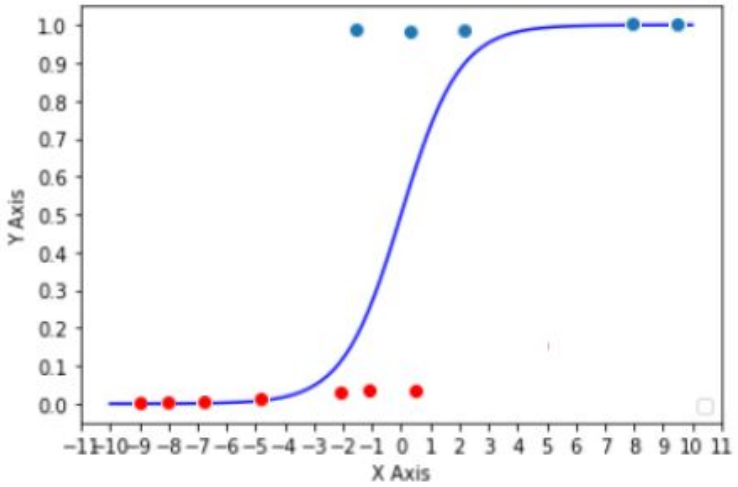


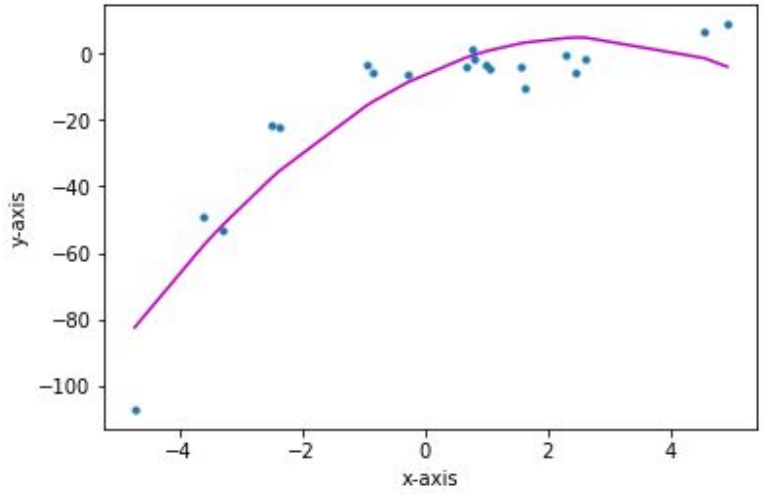


Types of regression

- Linear Regression
- Logistic Regression
- Polynomial Regression

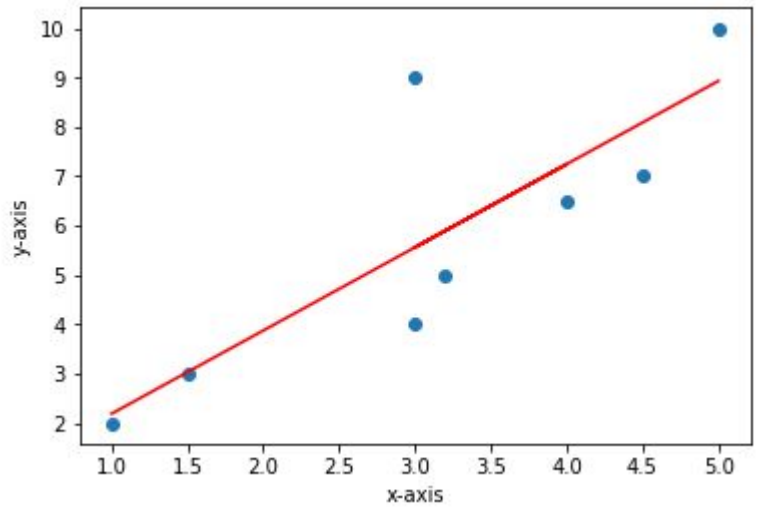






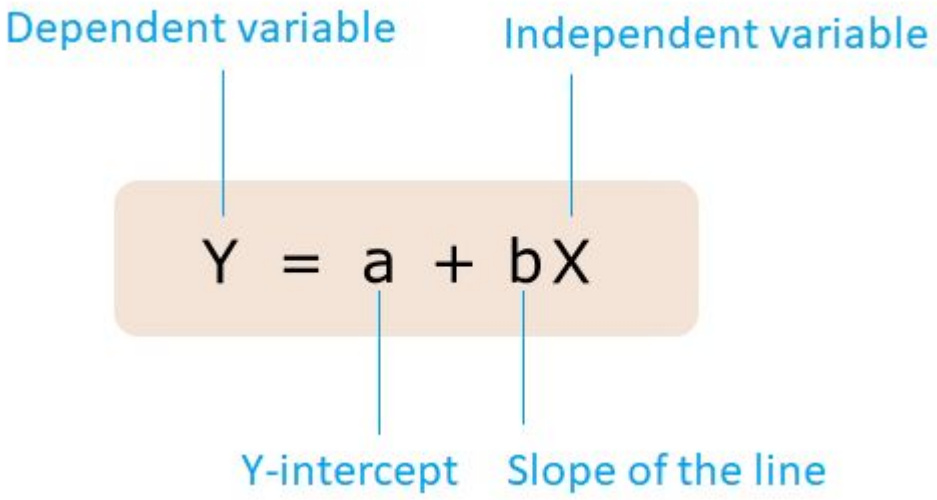
Linear Regression

- Linear Regression
  - Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data.
  - One variable is considered to be an independent variable, and the other is considered to be a dependent variable
  - A linear regression line has an equation of the form Y = a + bX, where X is the explanatory variable and Y is the dependent variable. The slope of the line is b, and a is the intercept (the value of y when x = 0).



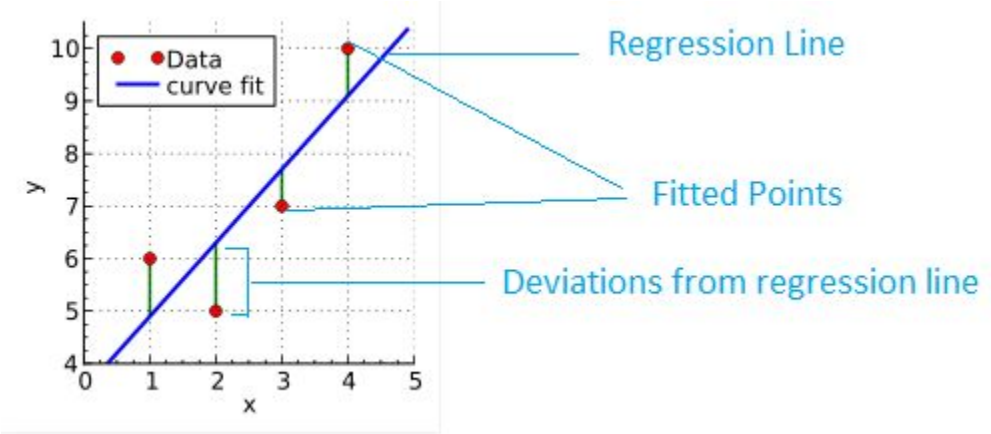
Simple Linear Regression

- **Y-intercept** is that value of the **Dependent Variable** when the value of **independent variable** is zero. It is the point the line cut y-axis
- **Slope** is the change in the **dependent variable** for a unit increase in the **independent variable**. It is the tangent of the angle made by the linen with the x-axis



The Regression Line

- The regression line is simply a single line that best fits the data (in terms of having the smallest overall distance from the line to the points)
- The technique is used for finding the regression line is “least square method”



The Regression Line

- The “least square method”:
- Step 1: For each (x,y) point calculate x^2 and xy
- Step 2: Sum all x, y, x^2 and xy, which gives us Σx, Σy, Σx^2 and Σxy
- Step 3: Calculate Slope b with N is number of points

$$b = \frac{N\,\Sigma(xy) - \Sigma x\,\Sigma y}{N\,\Sigma(x^2) - (\Sigma x)^2}$$

- Step 4: Calculate intercept a

$$a = \frac{\Sigma y - b\,\Sigma x}{N}$$

Gradient descent

- **Gradient descent** is an iterative optimization algorithm for finding the minimum of a function (by iteratively moving in the direction of steepest descent)
- In machine learning, we use gradient descent to update the parameters of our model. (Parameters refer to coefficients in Linear Regression and weights in neural networks)
- **Gradient Descent** is one of the important and difficult concepts. Practically, gradient descent is being used in complicated model such as there are many parameters.
- However, to make it easy to understand, we will use Linear Regression as an example this concept (Since Linear Regression is computionally simple)

Gradient descent

- **Hypothesis of Linear model**
- (Rewrite the linear equation with:
  - $h_\Theta(x)$ is hypothesis y
  - Θ_0 is a
  - Θ_1 is b

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- **Cost function**:

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} [h_\Theta(x_i) - y_i]^2$$

Predicted Value

True Value

- The purpose find the values of Θ_0 and Θ_1 that can minimize the cost (error)

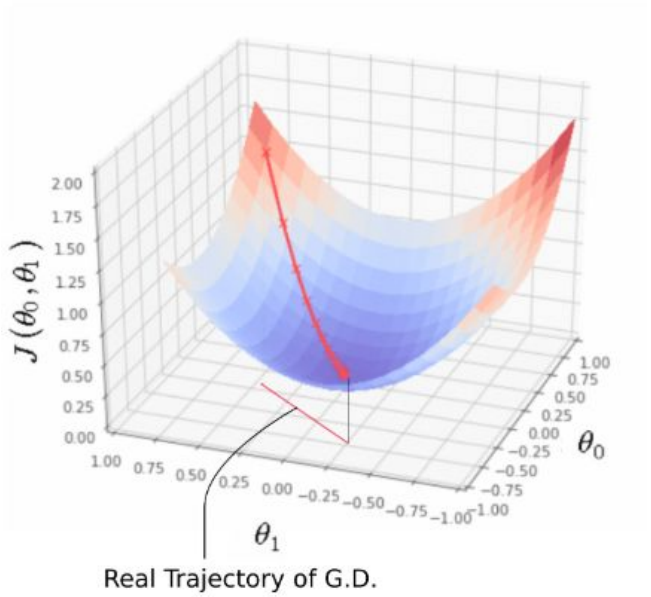
Gradient descent

- **Find the next step of descent**

$$\frac{\partial}{\partial \Theta} J_\Theta = \frac{\partial}{\partial \Theta} \frac{1}{2m} \sum_{i=1}^{m} [h_\Theta(x_i) - y]^2$$

$$= \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x_i) - y) \frac{\partial}{\partial \Theta_j} (\Theta x_i - y)$$

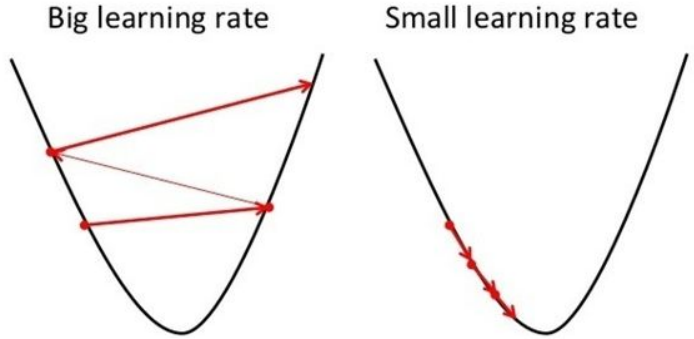
$$= \frac{1}{m} (h_\Theta(x_i) - y) x_i$$



- **Perform gradient descent (with learning rate alpha)**

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1)$$

Learning Rate



Gradient descent

Repeat until convergence

{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

- **Rewrite the formula for above updating Θ_0 and Θ_1 in details:**

repeat until convergence {

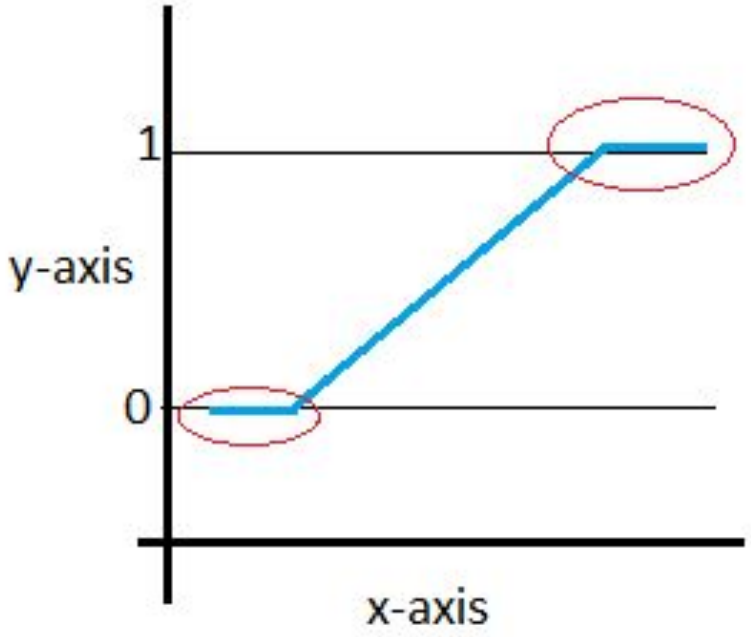
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

}

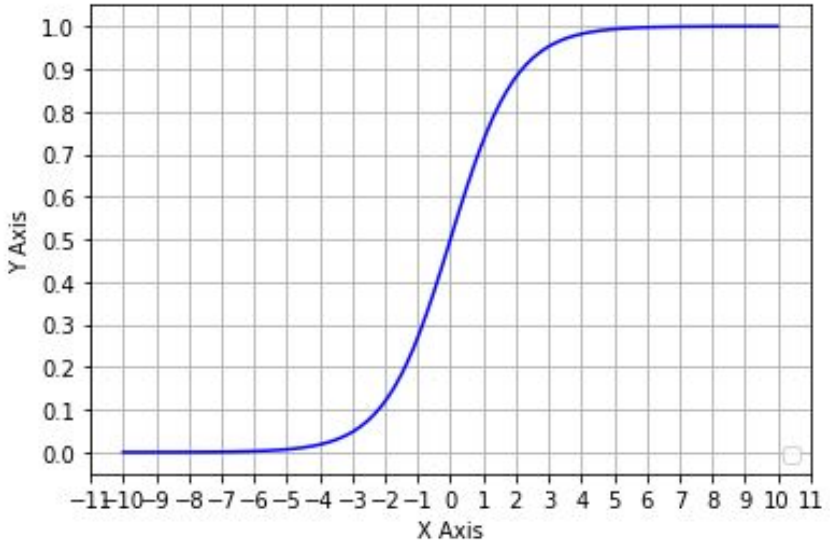
Disadvantage of Linear Regression

- In case of the value of Y is discrete. It means that the values of Y will be between 0 and 1, the linear line has be clipped at 0 and 1
- We can not formulate a single formula for this curve using linear regression
- To solve this type of problem, we need to use logistic regression



Logistic Regression

- Logistic Regression
  - Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables
  - The logistic Regression Curve is called Sigmoid Curve, also known as S-Curve



Logistic Regression

- Logistic Regression Use case
  - Logistic regression is used to determine an outcome which is a binary class type
  - We can apply logistic regression on patient (medical record) data to determine their illness. For example, the outcome of an tumor could be 0 if malignant, or 1 if benign

Logistic Regression Equation

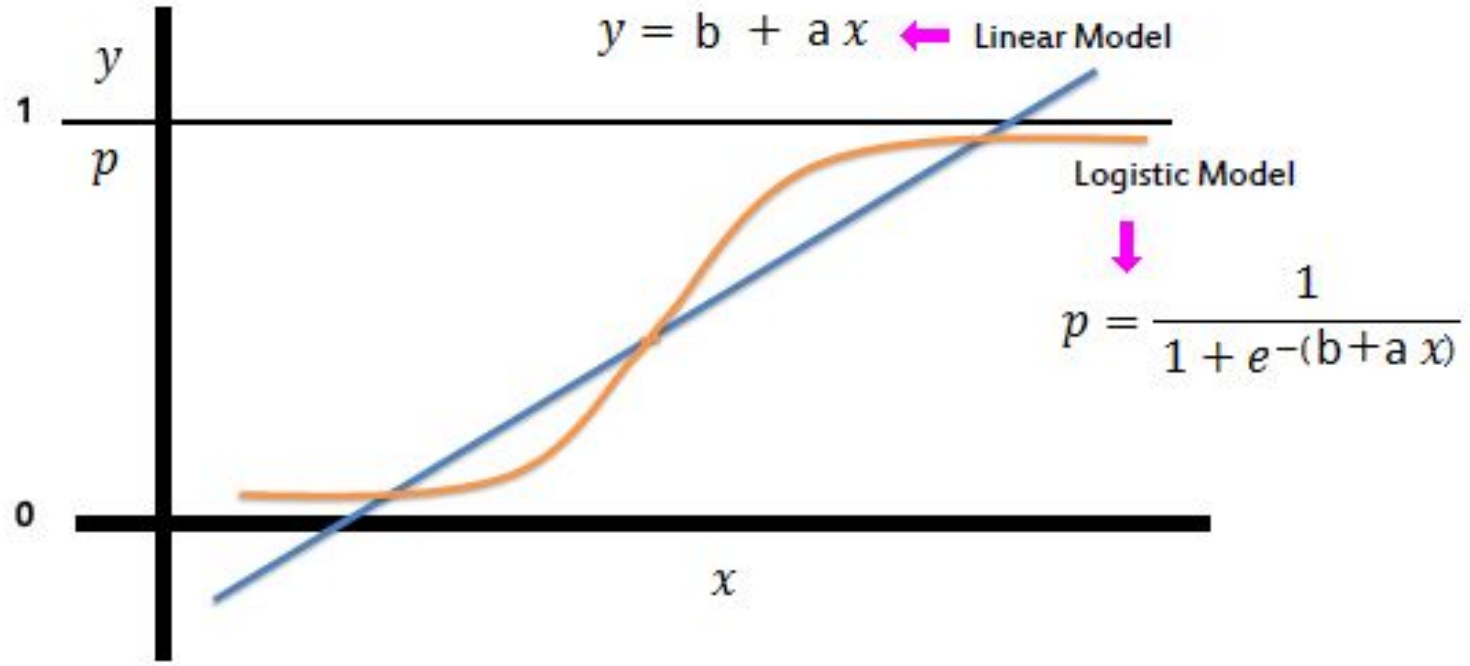
- Logistic Regression equation is derived from linear equations
  - Equation of a line which outcome is from -infinity to infinity
  - We can limits the range of outcome between 0 and 1 by using sigmoid function

$$S(z) = \frac{1}{1 + e^{-z}}$$

  - As z goes minus infinity, S(z) goes to 0 (since e^{-z} is really big)
  - As z goes infinity, S(z) goes to 1 (since e^{-z} is very small)
  - If x =0, y = 1/2

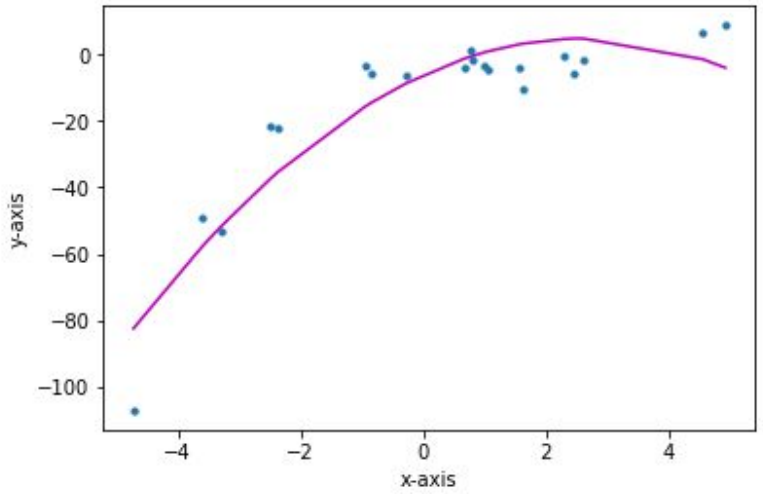
Logistic Regression Equation

- Then, we can have logistic regression which is statistical method to predict binary outcome. The target variable is binary and the independent variables are continuous.



Polynomial Regression

- When there is non linear data which can’t be predicted with a linear model. We use polynomial regression.

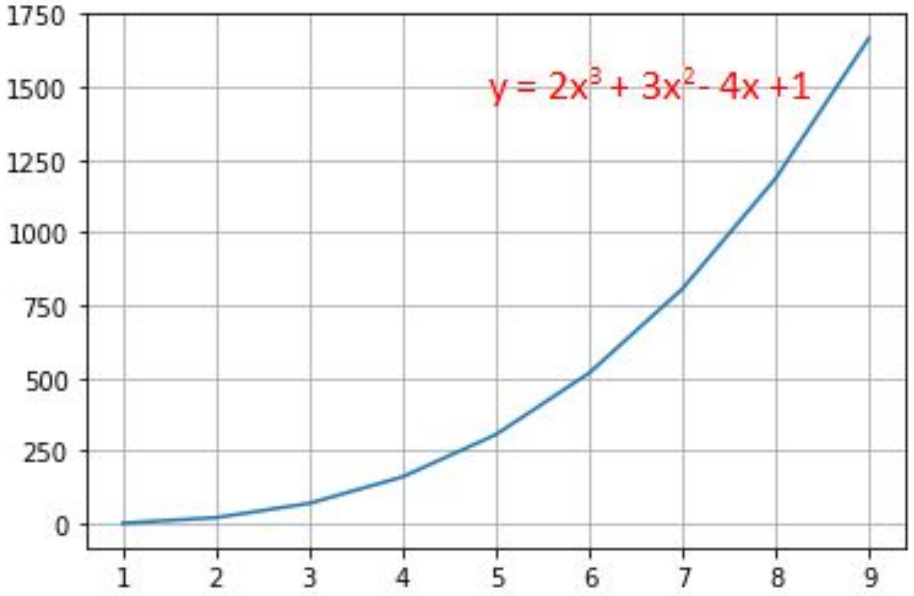


Polynomial Regression

- Polynomial equation will create the curved line to classify data
- Polynomial equation has the form as following:

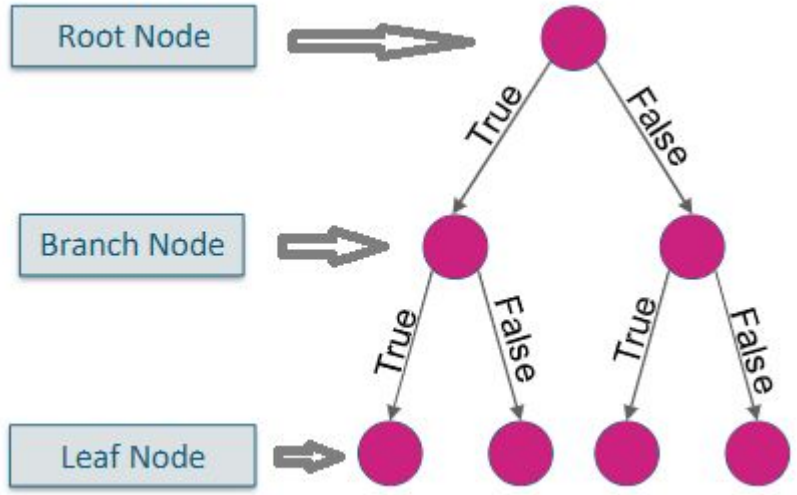
$$y = a_1x + a_2x^2 + a_3x^3$$

- Visualizing an example equation:



Decision Tree

- A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
- A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label.



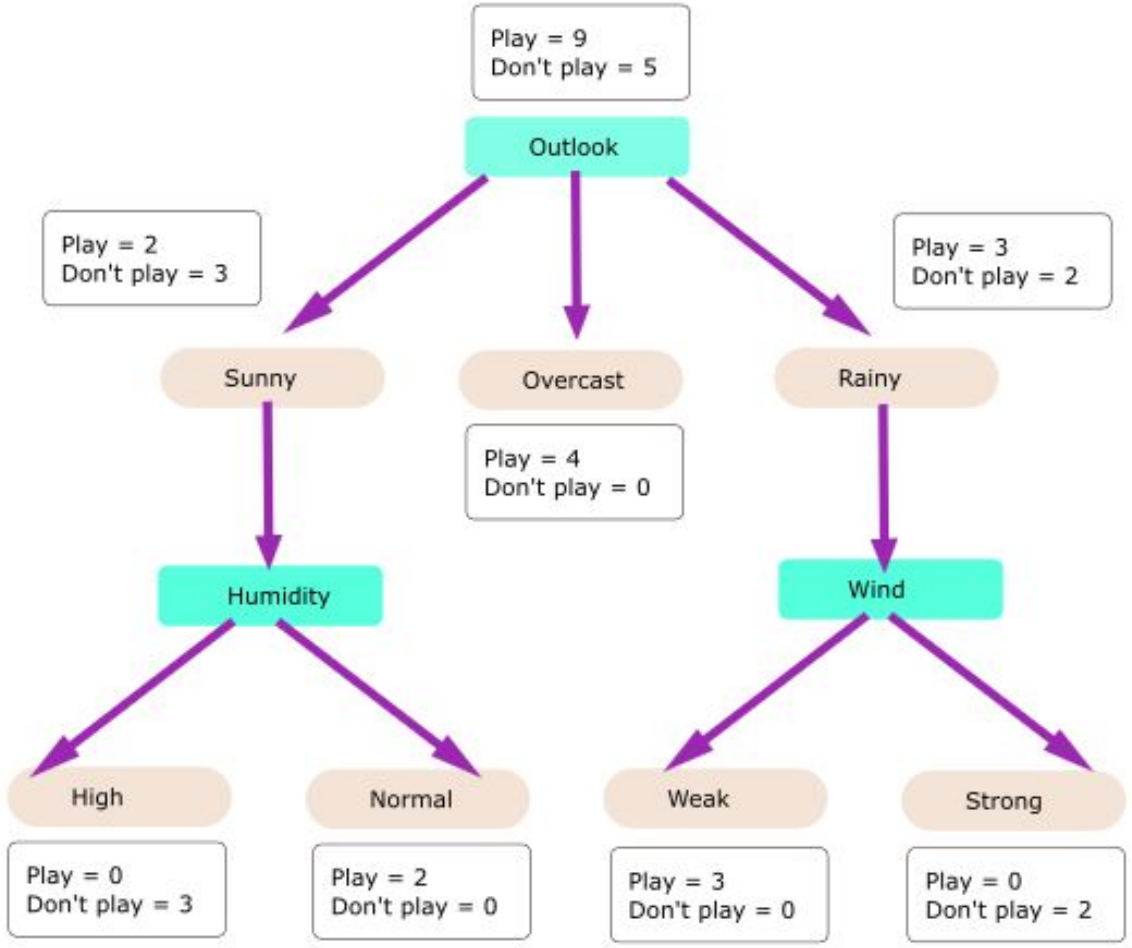
Decision Tree

- Let’s take a look at an example data set:
- We will use the decision tree to predict whether we should go outside if the condition is given as follow:
  - Outlook = Rain
  - Humidity = High
  - Wind = Weak
  - Play outside = ?

	Day	Outlook	Temperature	Humidity	Wind	Play Outside
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

Decision Tree

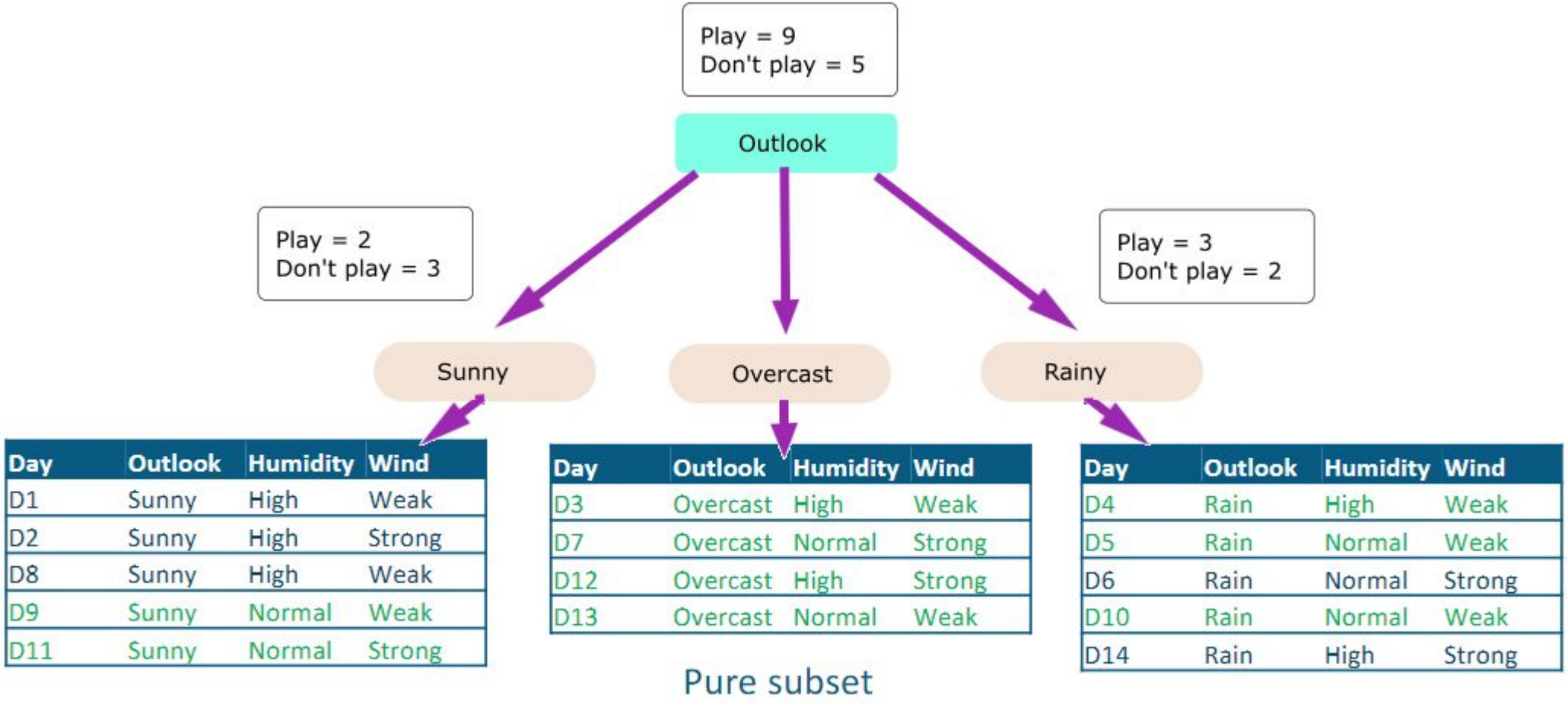
- From the decision tree you can find the result



	Day	Outlook	Temperature	Humidity	Wind	Play Outside
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

Building the Decision Tree

- Divide data using outlook column



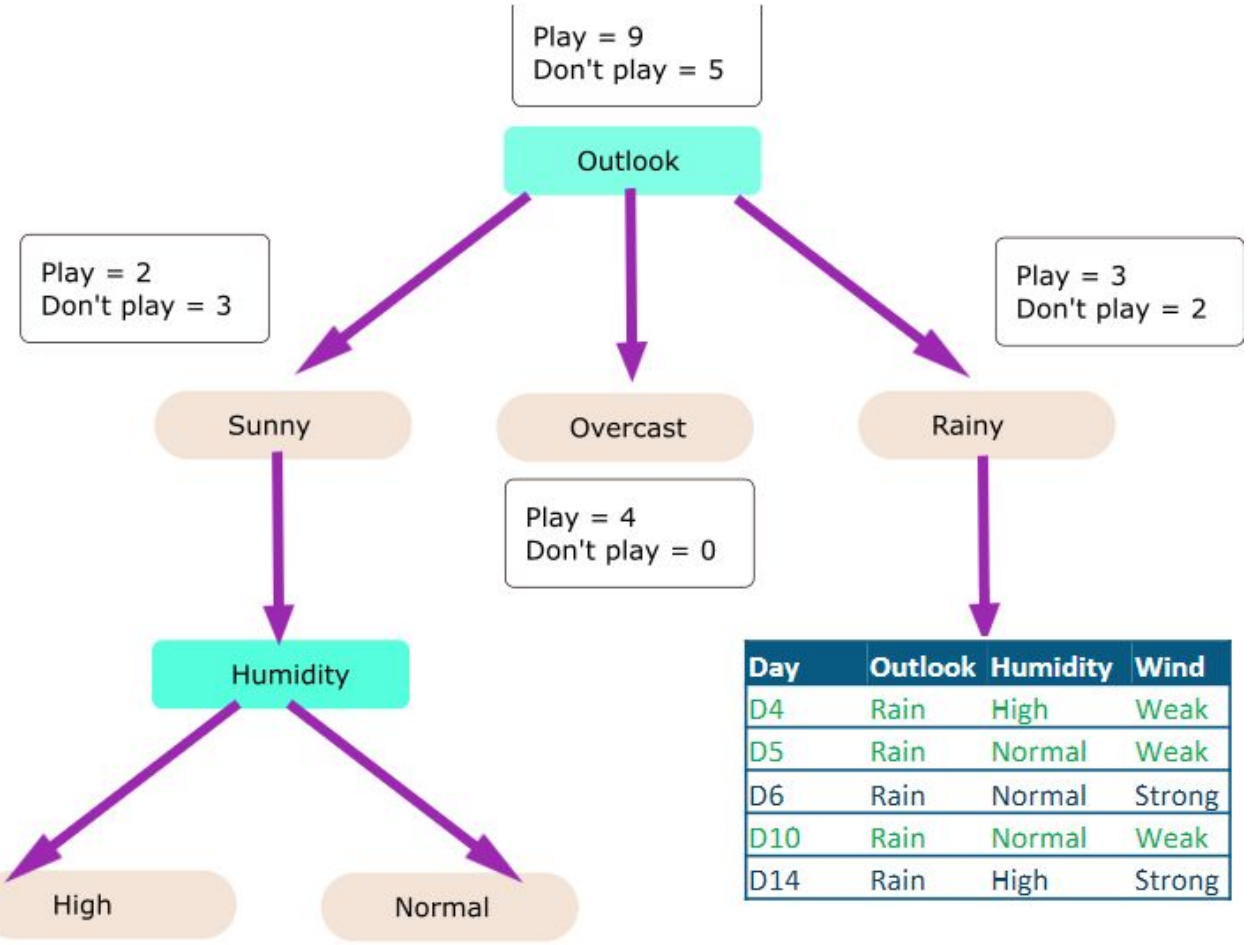
Day	Outlook	Humidity	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

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

Day	Outlook	Humidity	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

Building the Decision Tree

Divide data on Humidity



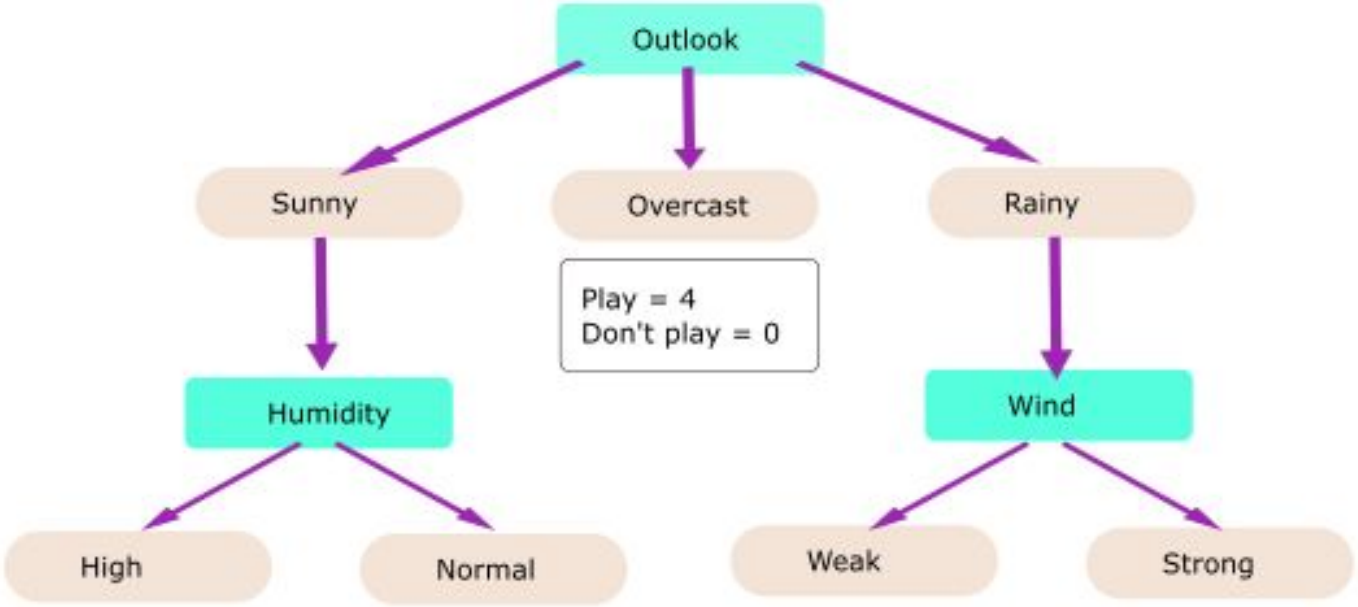
Day	Outlook	Humidity	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

Day	Humidity	Wind
D1	High	Weak
D2	High	Strong
D8	High	Weak

Day	Humidity	Wind
D9	Normal	Weak
D11	Normal	Strong

Building the Decision Tree

Divide data on wind



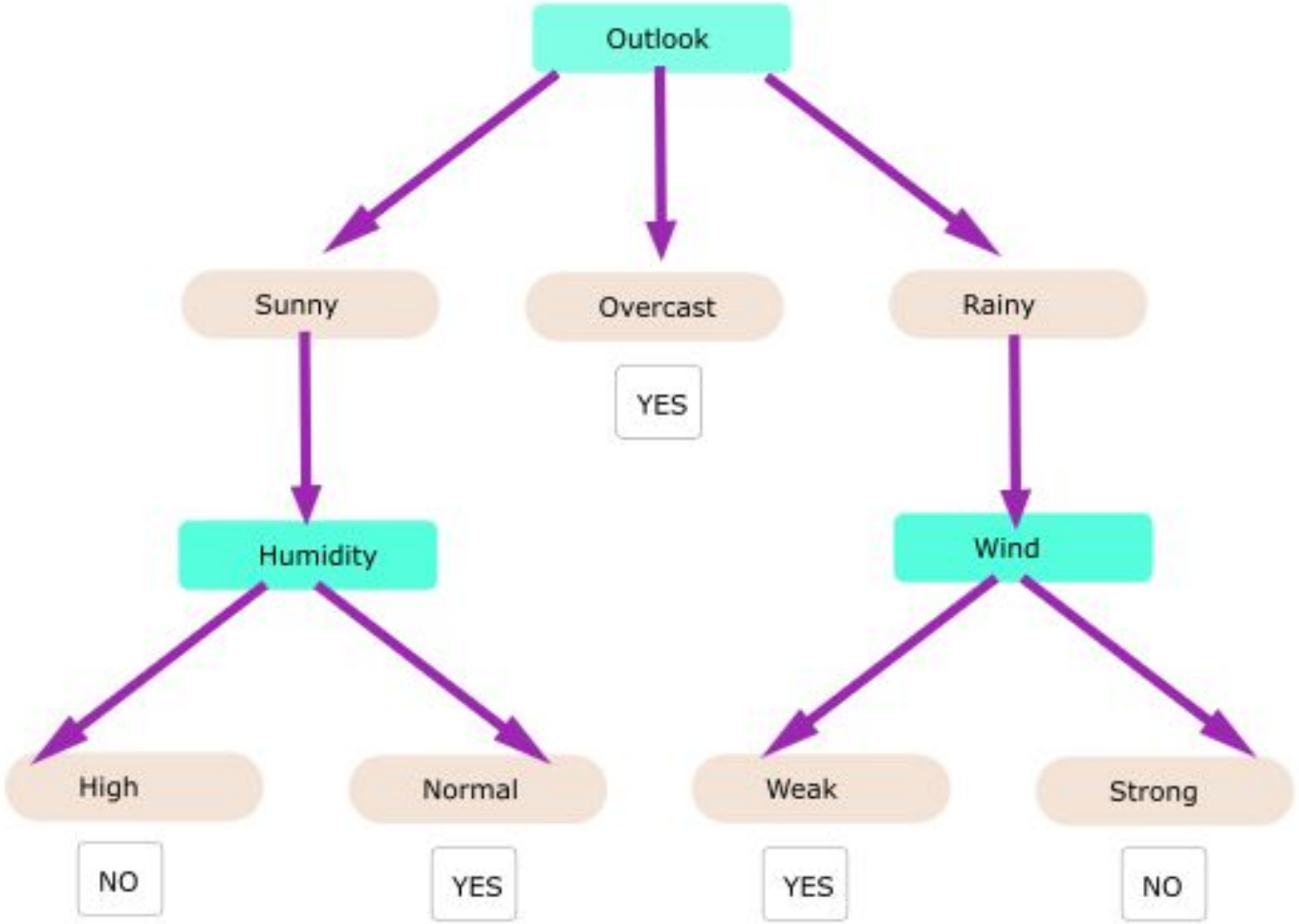
Day	Humidity	Wind
D1	High	Weak
D2	High	Strong
D8	High	Weak

Day	Humidity	Wind
D9	Normal	Weak
D11	Normal	Strong

Day	Humidity	Wind
D4	High	Weak
D5	Normal	Weak
D10	Normal	Weak

Day	Humidity	Wind
D6	Normal	Strong
D14	High	Strong

Building the Decision Tree



How to select the root of the tree

- We need to compute entropy and information gain (**ID3** algorithm - induction of decision trees based on information gain)
- **Entropy**:
  - Entropy (Shannon Entropy) is a measure of uncertainty. It tells you how much uncertainty is in the system
  - The measure of information entropy associated with each possible data value is the negative logarithm of the probability mass function for the value:
    - Entropy using the frequency table of one attribute:

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

    - Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

- **Information gain:**

Information Gain = entropy(parent) - [weights average] * entropy(children)

How to select the root of the tree

- Compute the entropy of data set on target (entropy on one attribute)

Out of 14 instances, 9 are classified as yes, and 5 as no

pyes = −(9/14)*log2(9/14) = 0.41
pno = −(5/14)*log2(5/14) = 0.53

E(playGolf) = pyes + pno = 0.94

Play Golf	
Yes	No
9	5

How to select the root of the tree

- Compute the information gain each attribute (Entropy on 2 attributes)

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(play,Outlook) = P(Sunny)*E(Sunny) + P(Overcast)*E(Overcast) + P(Rainy)*E(Rainy)

= (5/14)*(0.971) + 4/1*0.0 + (5/14)*0971

= 0.693

How to select the root of the tree

- Compute the information gain each attribute
- Then choose the attribute that has largest information gain
    - IG(play,Oulook) = E(play) - E(Play,Outlook)
      = 0.940 - 0.693 = 0.247

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

How to select the root of the tree

- The branch with entropy of zero is a leaf node
- The branch has entropy larger than zero is a non-leaf branch and need further splitting
- Recursively run algorithm on on-leaf branches until all of data is classified

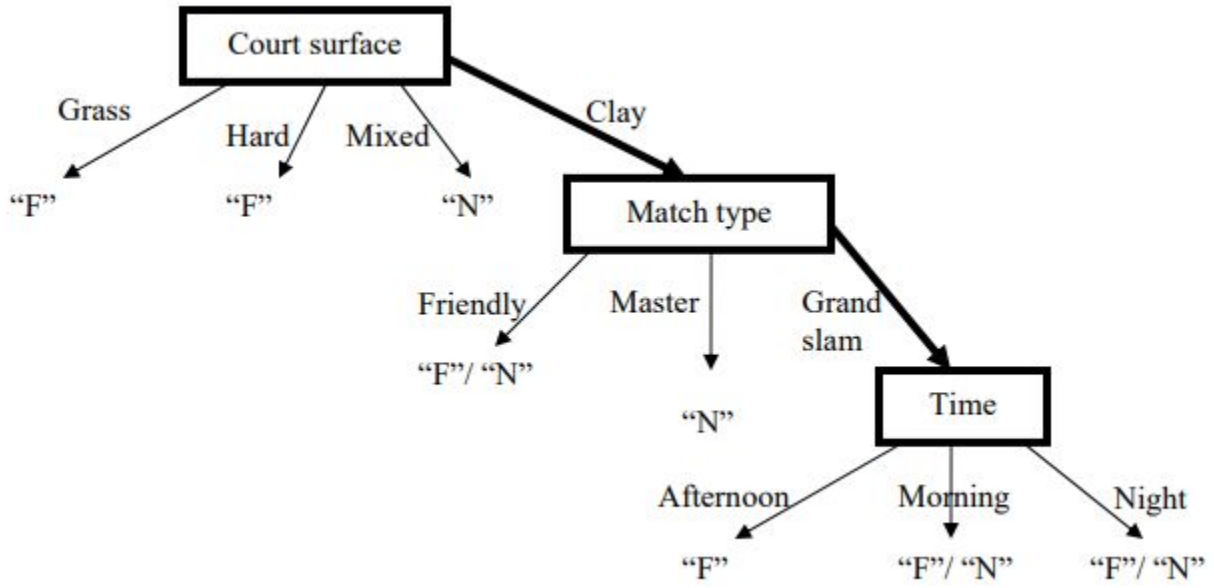
Practice on Decision Tree

- Build a decision tree with this data: will Federera win the next match
- (Court surface=”Clay”, Match type=”Grand slam”, Time=”Afternoon”,Best effort=”1”).

Time	Match type	Court surface	Best Effort	Outcome
Morning	Master	Grass	1	F
Afternoon	Grand slam	Clay	1	F
Night	Friendly	Hard	0	F
Afternoon	Friendly	Mixed	0	N
Afternoon	Master	Clay	1	N
Afternoon	Grand slam	Grass	1	F
Afternoon	Grand slam	Hard	1	F
Afternoon	Grand slam	Hard	1	F
Morning	Master	Grass	1	F
Afternoon	Grand slam	Clay	1	N
Night	Friendly	Hard	0	F
Night	Master	Mixed	1	N
Afternoon	Master	Clay	1	N
Afternoon	Master	Grass	1	F
Afternoon	Grand slam	Hard	1	F
Afternoon	Grand slam	Clay	1	F

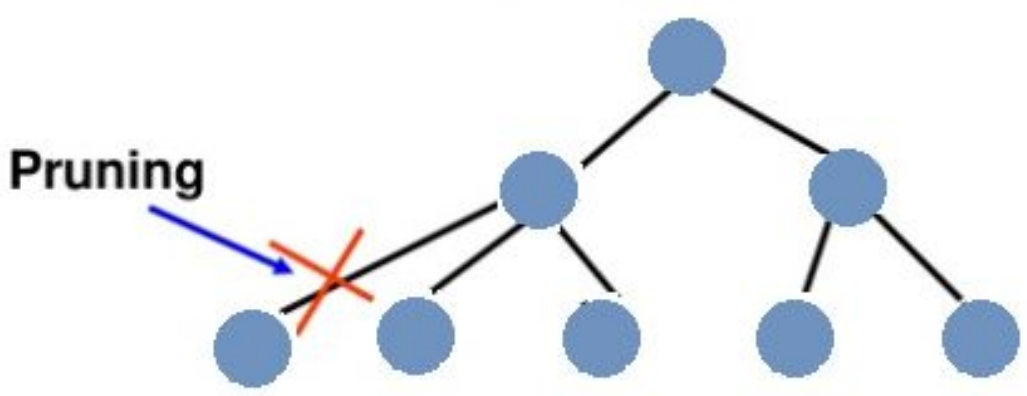
Practice on Decision Tree

- Federera is more likely to win
- (Court surface=”Clay”, Match type=”Grand slam”, Time=”Afternoon”,Best effort=”1”).



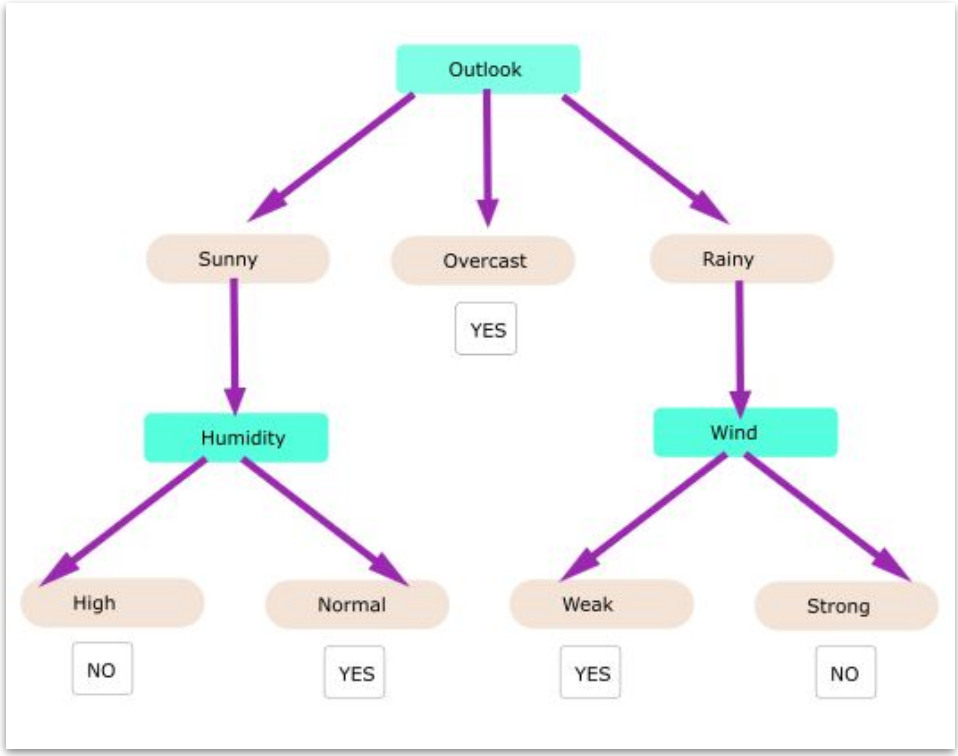
Avoiding overfitting

- Two approaches to avoiding overfitting (each approach has various methods)
  - **Pre-pruning**: Stop growing the tree before it perfectly classifies the training data. (Problem: to know when to stop.)
  - **Post-pruning**: Allow the tree to overfit the data, and then post-prune it



Post-pruning

- Understand post-pruning using an example of rule post-pruning method
- Rule post pruning
  - The post-pruning approach requires to split the data into a training set and a validation set, which is used for pruning.
  - Algorithm:
    - Convert tree to rules (one for each path from root to a leaf)
    - For each antecedent in a rule, remove it if error rate on validation set does not decrease
    - Sort final rule set by accuracy
      - Outlook= sunny ^ humidity=high -> No
      - Outlook= sunny ^ humidity=normal -> Yes
      - Outlook= overcast -> Yes
      - Outlook= rain ^ wind=strong -> No
      - Outlook= rain ^ wind=weak -> Yes



An example of post-pruning

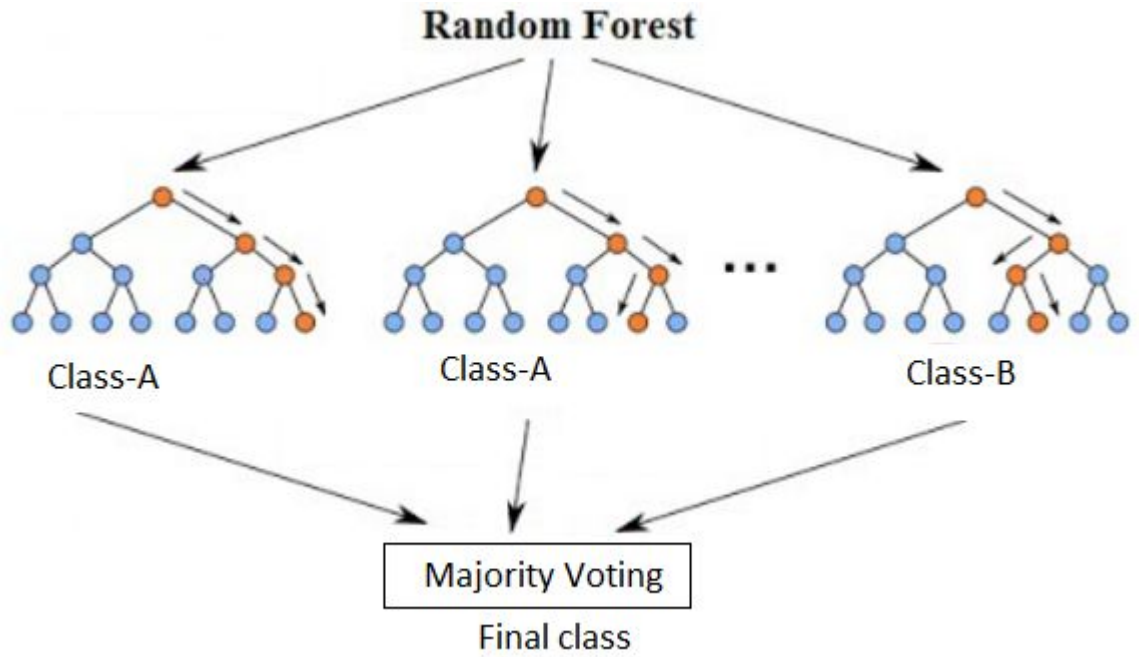
- Start with first rule:
- Compare it with: Outlook= sunny ^ humidity=high -> No
  - Outlook=sunny->No
  - Humidity=high->No
- Calculate accuracy of 3 rules based on validation set and pick best version.
- Then remove one of two rules if the error rate on data set If the error does not decrease significantly enough then we remove it

Pre-pruning

- Pre-pruning prevent the generation of non-significant branches.
- Pre-pruning a decision tree involves using a “termination condition” to decide when it is desirable to terminate some of the branches prematurely as the tree is generated.
- One of method we can use:
  - At each stage of splitting the tree, we check the cross-validation error.
  - If the error does not decrease significantly enough then we stop.
  - Early stopping may underfit by stopping too early. Since, the current split may be of little benefit, but having made it, subsequent splits more significantly reduce the error.

Random Forest

- **Ensemble Model**
  - Ensemble models combine the results from different models
- Random forests or random decision forests are an ensemble learning method that operates by using many decision trees
- Every tree votes for one class, the final decision is based upon the majority of votes

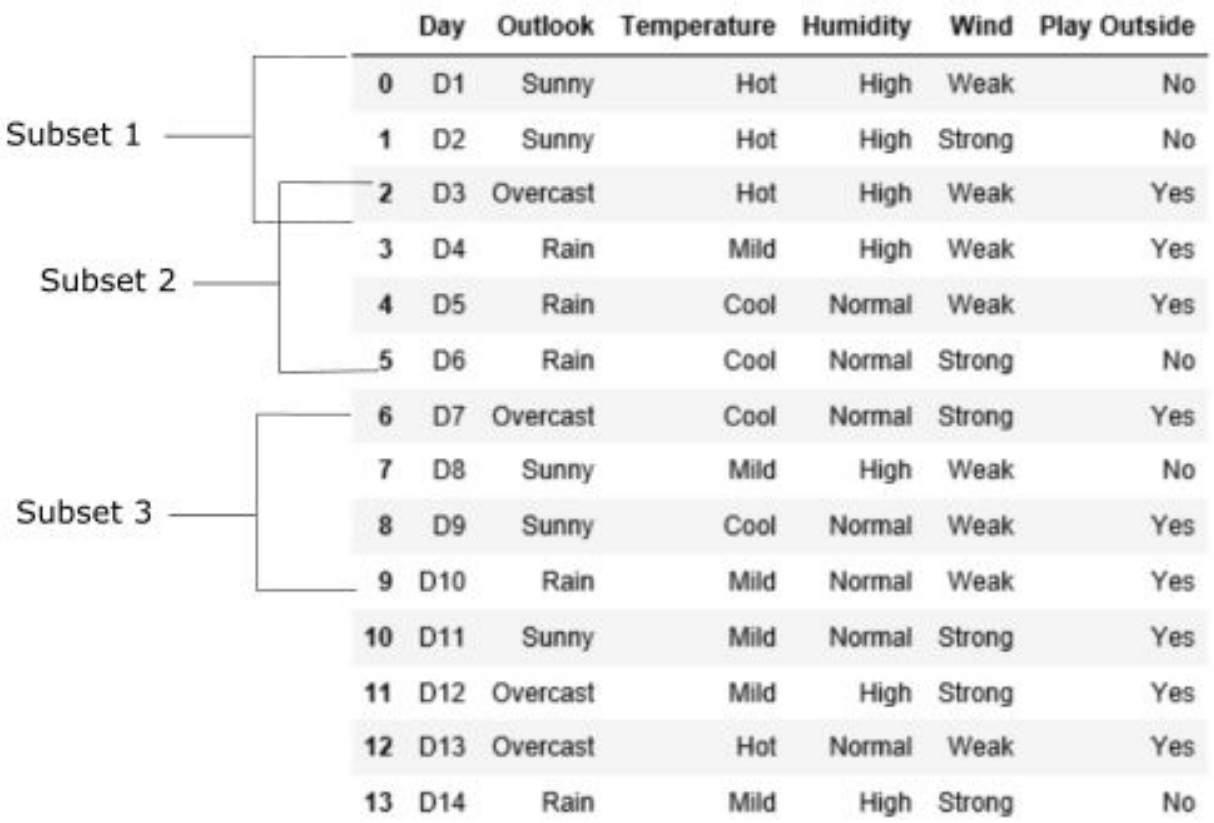


Advantage of Random Forest

- It can be much more accurate compare to decision tree
  - Tend not to overfit. It receives results from the many trees means that no single tree sees all the data. This helps to focus on the general patterns within the training data and reduce sensitivity to noise.
- It performs efficiently on large databases
- It is useful to identify variables most helpful in prediction. It splits based on the attribute significance.

Random Forest

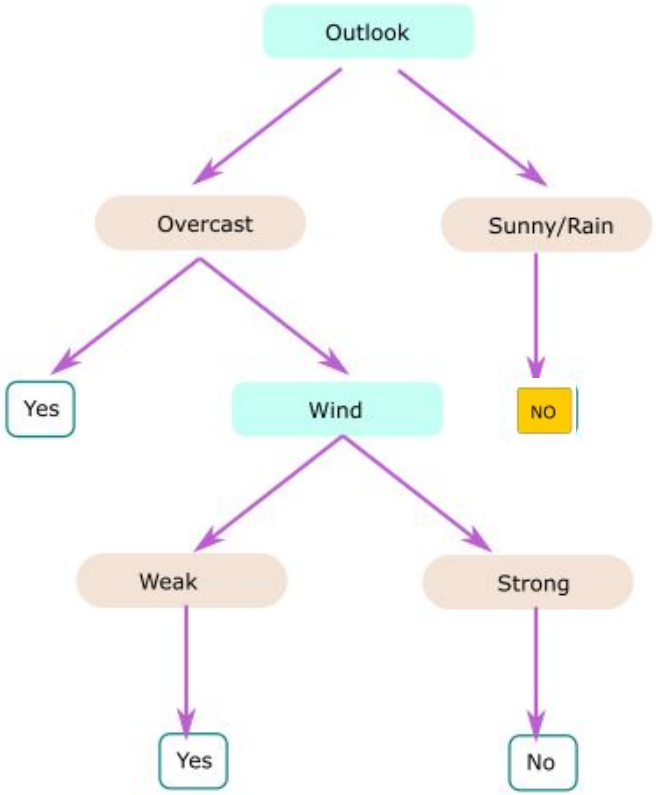
- Example of random forest
- Applying random forest on the previous problem
- Some subsets may have overlap data

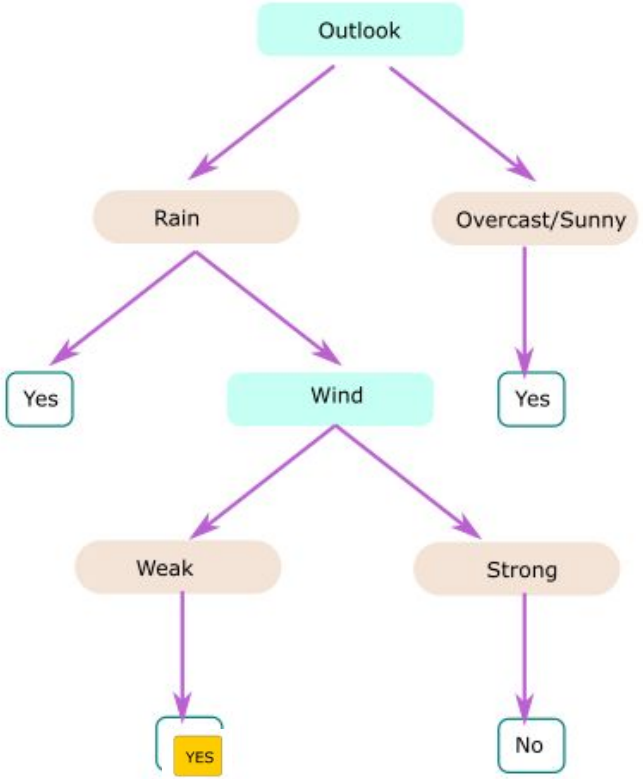


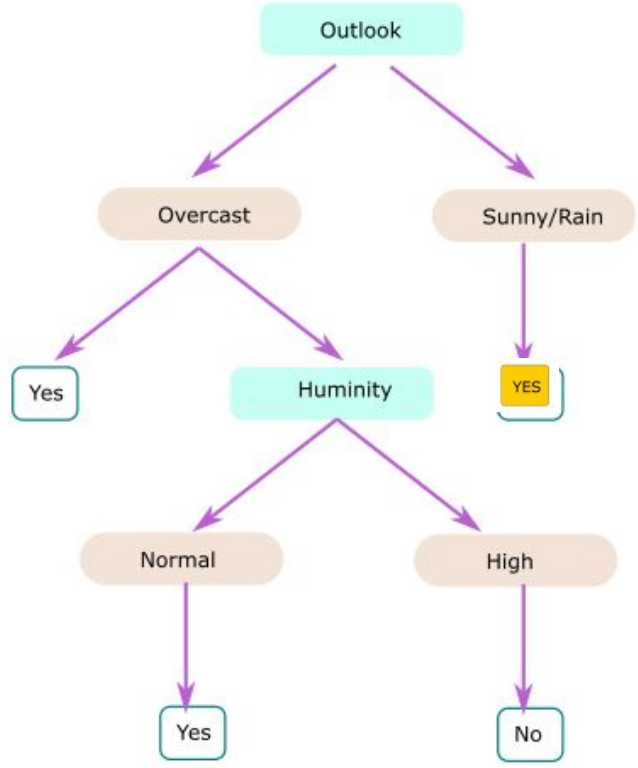
	Day	Outlook	Temperature	Humidity	Wind	Play Outside
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

Random Forest

- Create a decision tree for each subset
- Outlook = Rain, Humidity = High, Wind = Weak
- Based on number of votes of each decision tree. In this case, play outside is yes







Random Forest in Python

- We can use sklearn to implement Random Forest in Python

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators = 3)
```

Number of tree in the forest

- Then using the training data for building the model

```
model.fit(X_train,Y_Train)
predictions = model.predict(X_test)
```

- predictions are the outcome that we can use to compare with outcome of testing data (Y_test) to get the accuracy

Exercise

- Using provided data set to write program to compare the accuracy between linear regression and random forest

Q & A