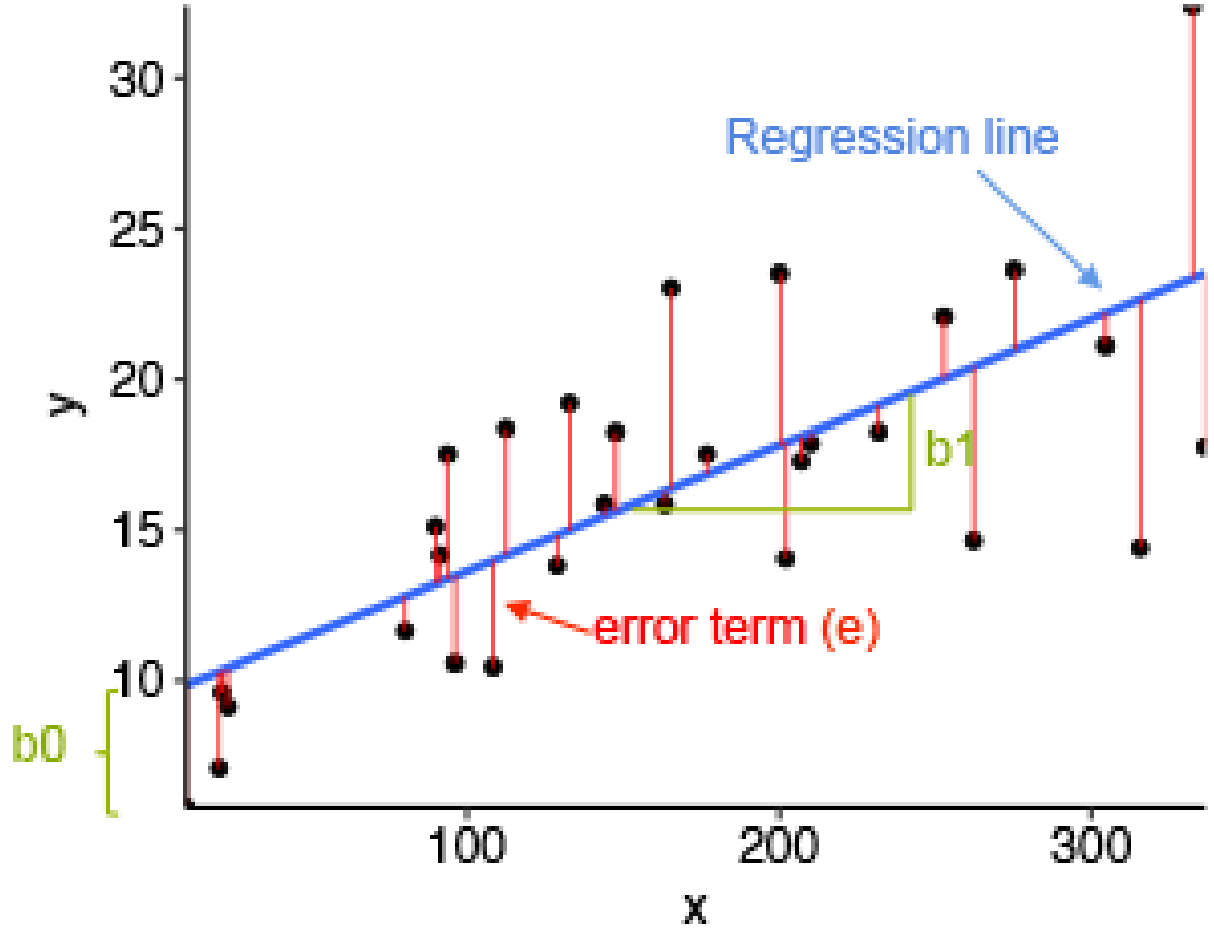
Module 2

- **Supervised Learning Regression and Classification Algorithms**
- **Regression:** Introduction to Regression- Regression Models- Linear Regression, Polynomial Regression
- **Decision Trees:** Introduction to Decision Trees-Tree Construction, Splitting Criteria, and Pruning-Handling Missing Values and Categorical Features- Gini Index-ID3-CART.

Regression

- Linear regression, is commonly used to estimate the ***linear relationship*** between independent variables - (x_1, x_2, …, x_n) and dependent variables- (y).
- It is a statistical method that is used for predictive analysis.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables
- We use linear regression when the dependent variable is a continuous variable (value ranging between [-∞,+∞]).
- For example, predicting prices of houses, cars and stocks.

Regression



Types of Linear Regression

- There are two main types of linear regression:
  Simple Linear Regression
- ***Multiple Linear Regression***

Simple Linear Regression

- This is the simplest form of linear regression, and it involves only one independent variable and one dependent variable. The equation for simple linear regression is:

$$y=\beta_0+\beta_1 x$$

- where:

Y is the dependent variable

X is the independent variable

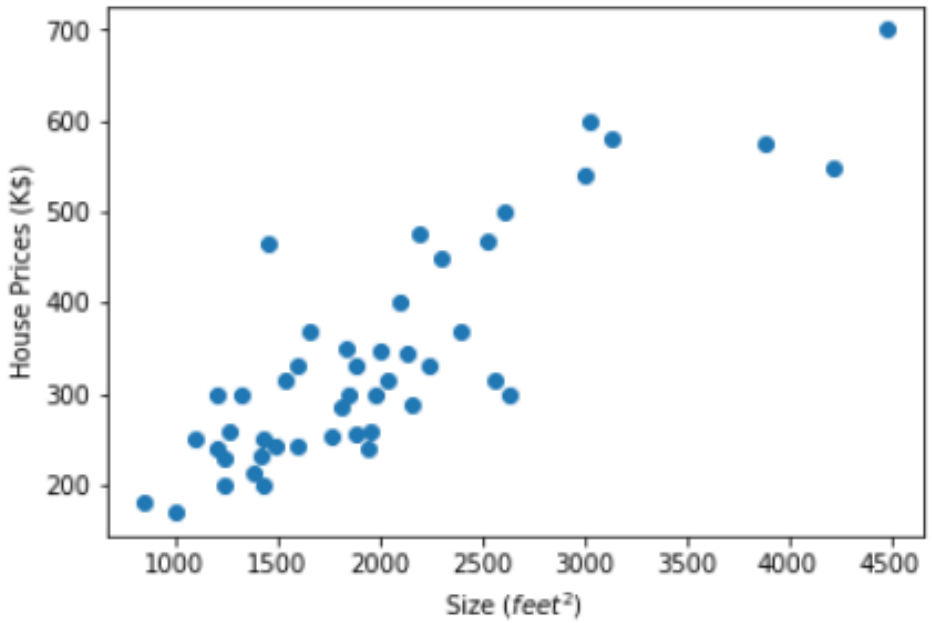
β0 is the intercept

β1 is the slope

Simple linear regression

- Example1:For instance, estimating a person's weight (y) given his/her height (x).
- **Independent variable**: Input feature. (x_1)
- **Dependent variable**: The variable you are trying to predict. Sometimes also known as target variable. (y).
- Intuitively, we would know that these two variables are positively correlated.

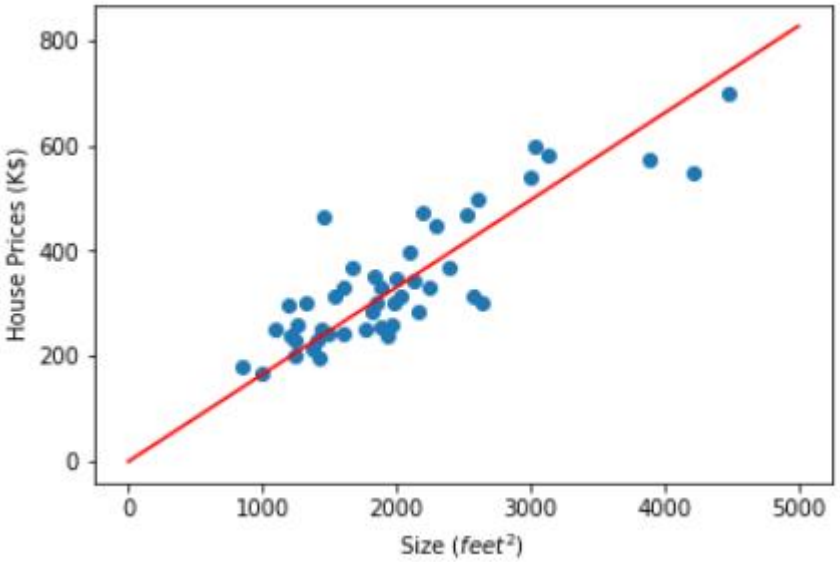
Simple linear regression-example



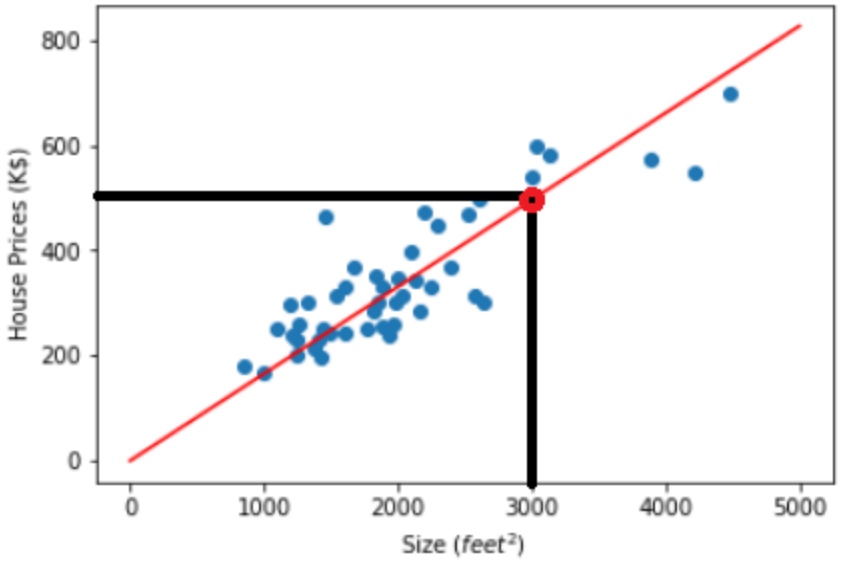
- In the graph above, the y-axis is the price (K$) of some houses, and the x-axis is the house's size ($feet^2$).

Regression

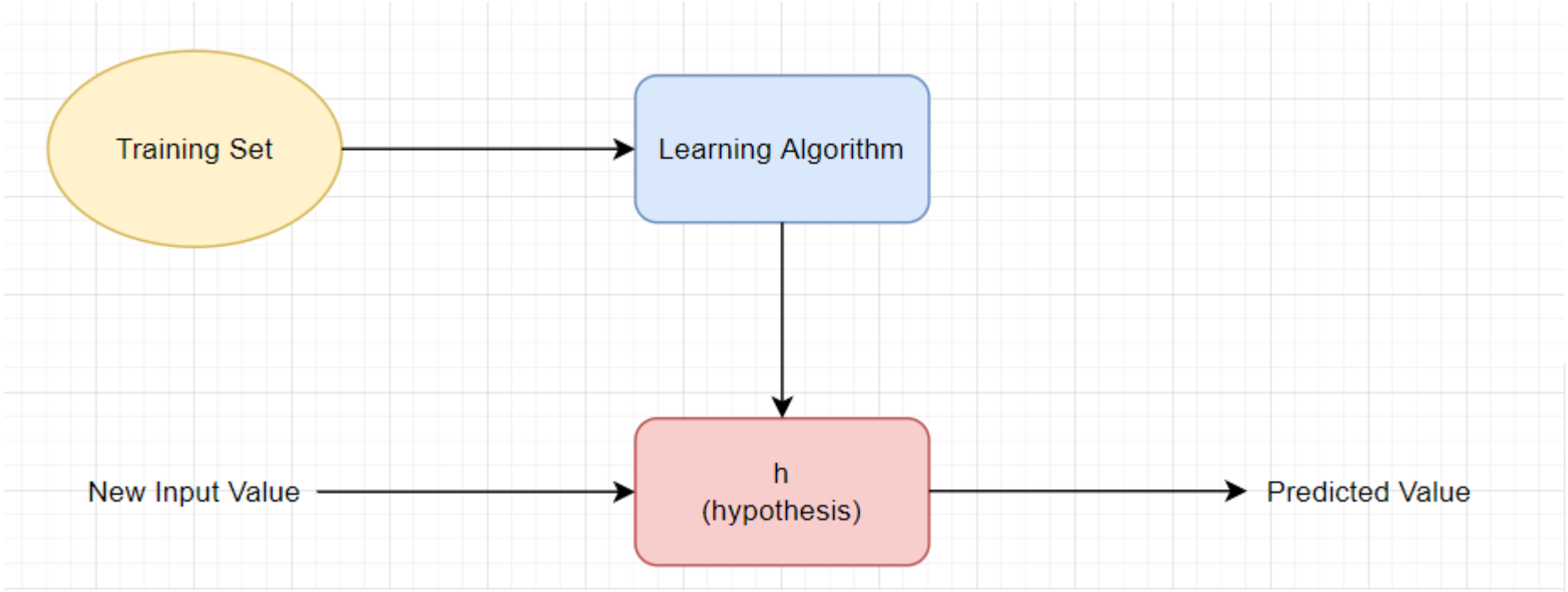
- We want to draw a straight line on this graph (such as the image below) so that when we know the size of a new home not in this dataset, we can predict its price by using that line.



- For example, we can predict that the price of a 3000 feet² house is approximately 500 K$, as seen in the graph below.



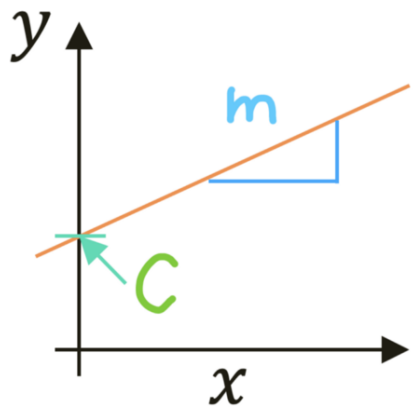
Regression Model Representation



Linear Regression

- It is also obvious that we can fit a straight linear line to describe the relationship between these two variables.

$$y = mx + c$$



where *m* is the slope and *c* is the y-intercept of the line.

It is common to see these two parameters to be denoted as $c=\theta_0$ and $m=\theta_1$.

Linear Regression

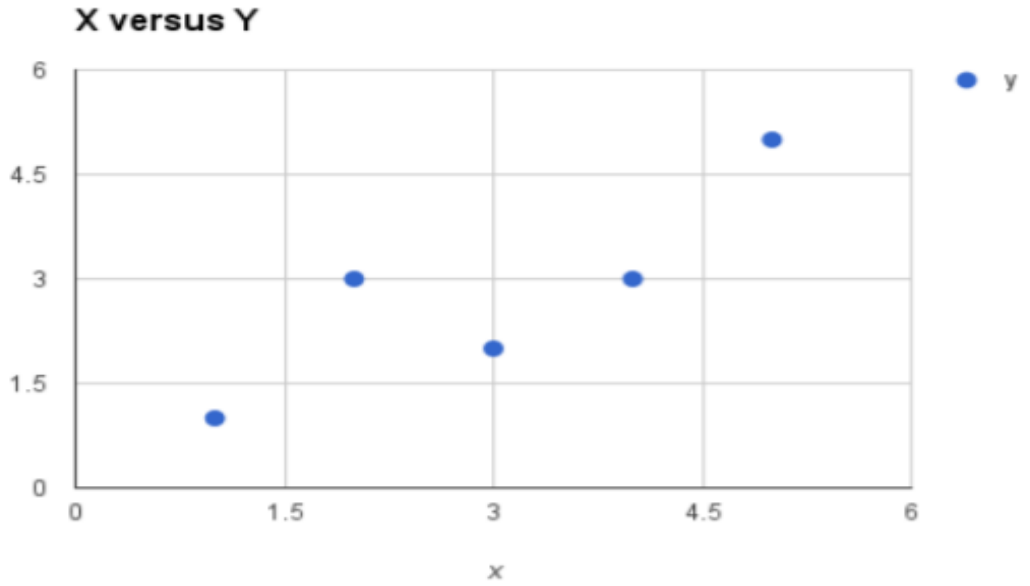
- The algorithm finds the values for θ_0 and θ_1 that best fit the inputs and outputs given to the algorithm. This is called **univariate linear regression** because the θ parameters only go up to 1.
- The goal is to find the best estimates for the coefficients to minimize the errors in predicting y from x.

$$B1 = \frac{\sum (Xi - \bar{X}) * (Yi - \bar{Y})}{\sum (Xi - \bar{X})^2}$$

- Where mean() is the average value for the variable in our dataset. The xi and yi refer to the fact that we need to repeat these calculations across all values in our dataset and i refers to the i'th value of x or y. We can calculate B0 using B1 and some statistics from our dataset, as follows:

$$B0 = \bar{Y} - (B1 * \bar{X})$$

Example



x	y
1	1
2	3
4	3
3	2
5	5

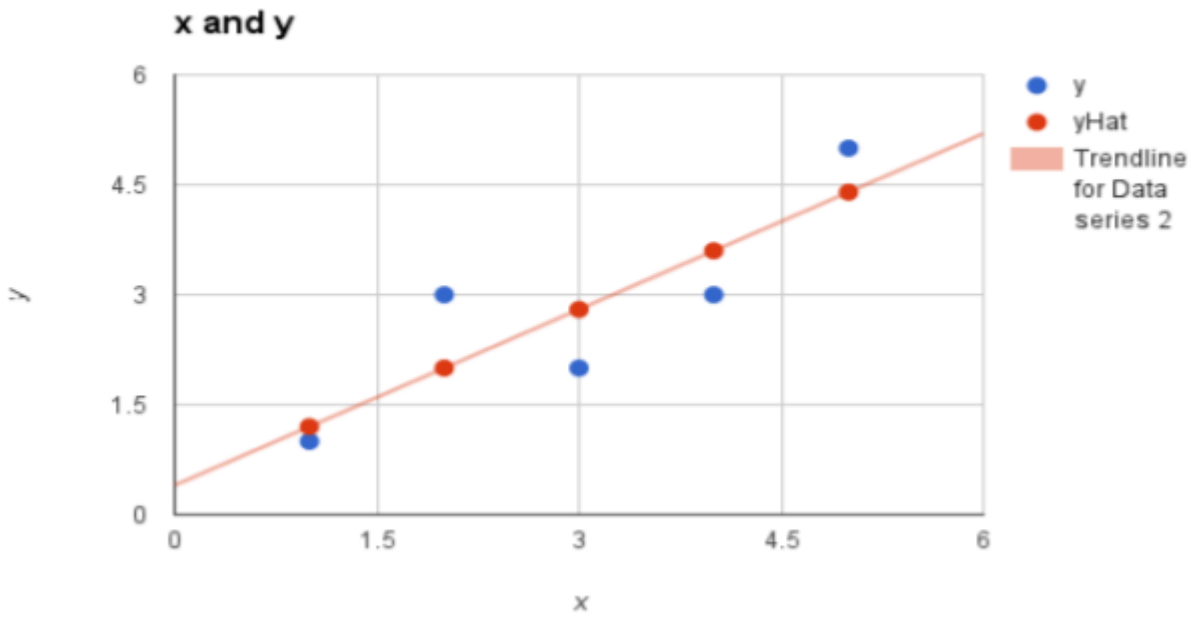
	x - mean(x)	y - mean(y)	Multiplication
2	-2	-1.8	3.6
3	-1	0.2	-0.2
4	1	0.2	0.2
5	0	-0.8	0
6	2	2.2	4.4

B1 = 8 / 10 so further B1 = 0.8

- **Estimating Intercept (B0)**
- This is much easier as we already know the values of all of the terms involved. $B0 = \bar{Y} - (B1 * \bar{X})$ or B0 = 2.8 – 0.8 * 3 , or further B0 = 0.4
- **Making Predictions**
- We now have the coefficients for our simple linear regression equation.
- y = B0 + B1 * x or y = 0.4 + 0.8 * x
- Let’s try out the model by making predictions for our training data.

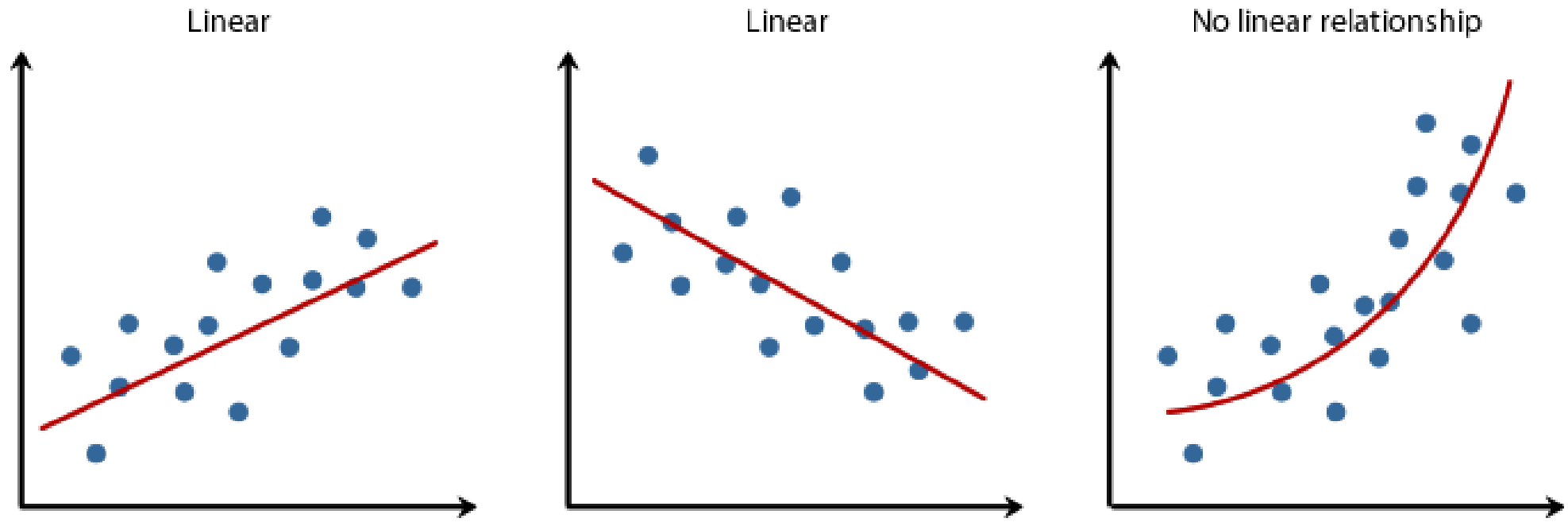
1	x	y	predicted y
2	1	1	1.2
3	2	3	2
4	4	3	3.6
5	3	2	2.8
6	5	5	4.4

Linear Regression



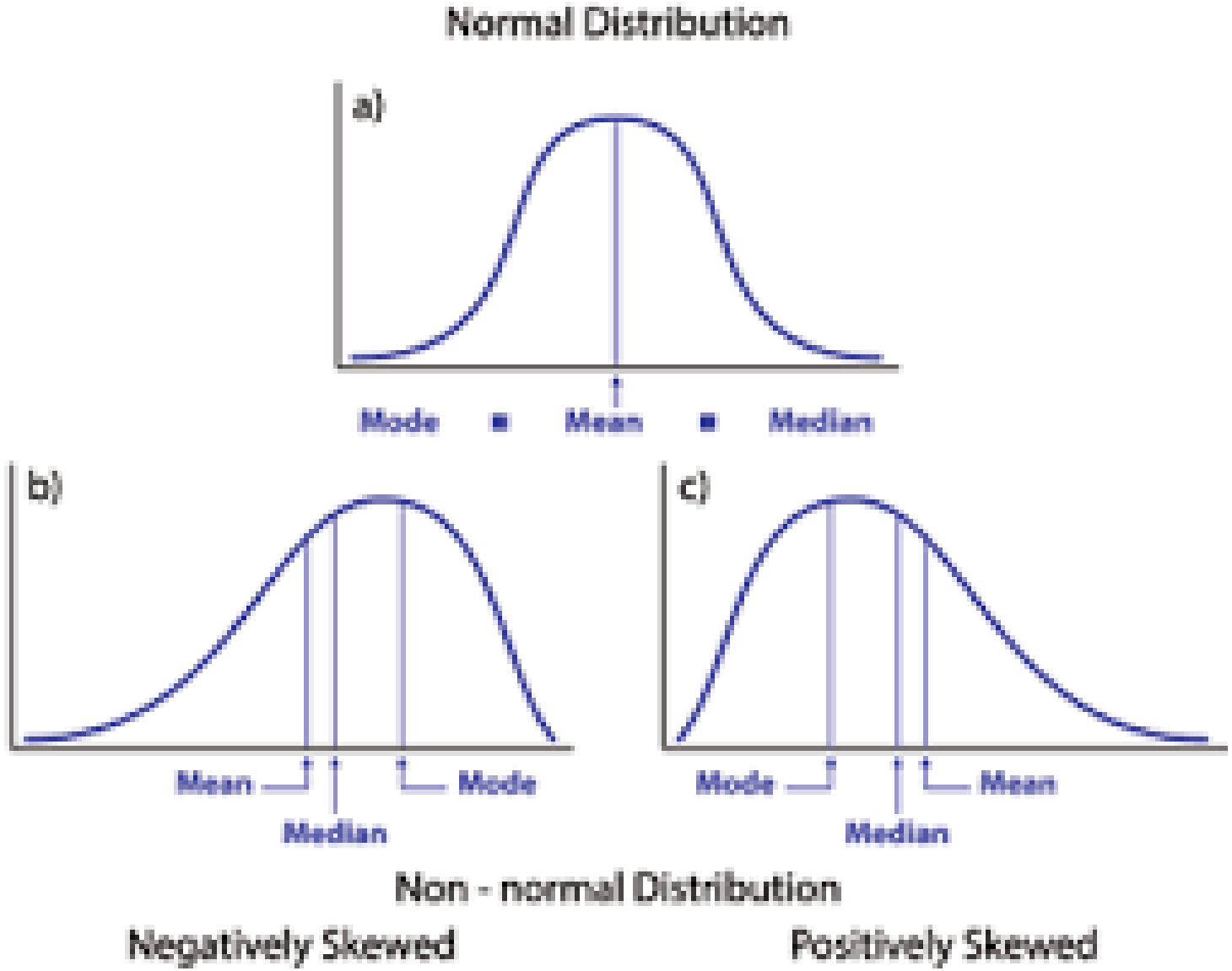
Assumptions of Linear Regression

Linearity: It states that the dependent variable Y should be linearly related to independent variables. This assumption can be checked by plotting a scatter plot between both variables.



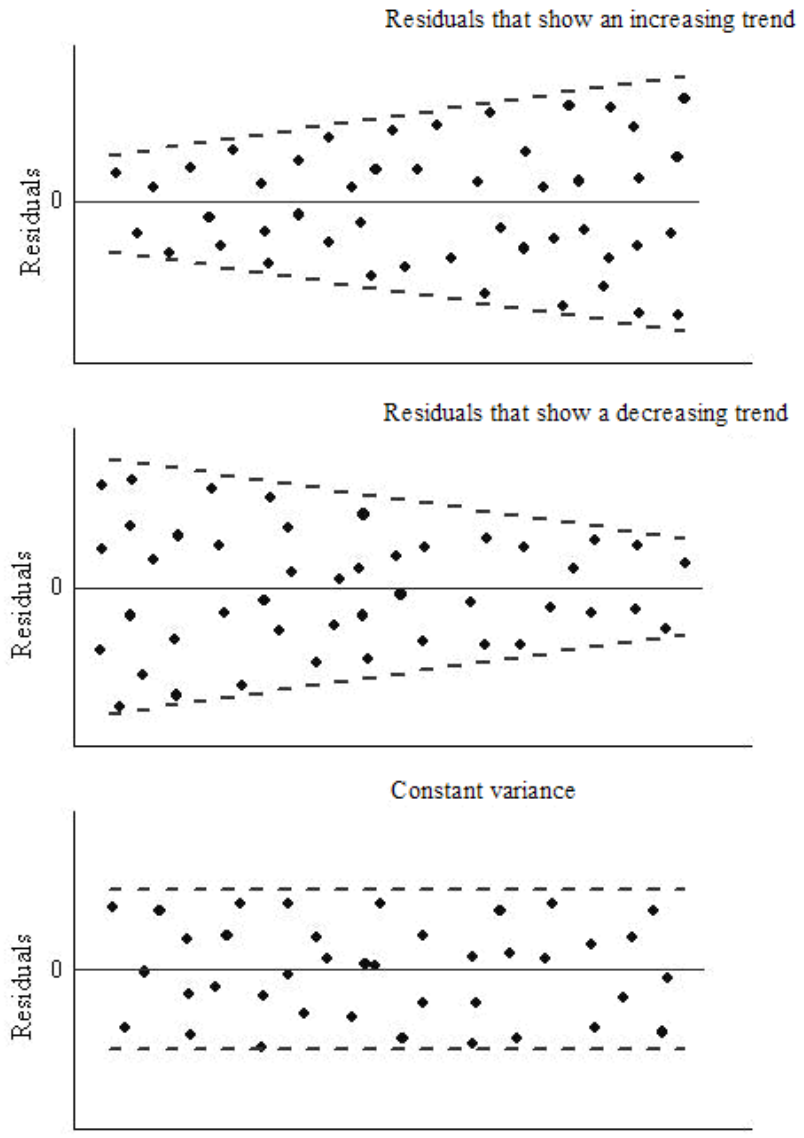
Normality

- **Normality**: The X and Y variables should be normally distributed. Histograms, KDE plots(Kernel Density Estimate) can be used to check the Normality assumption.



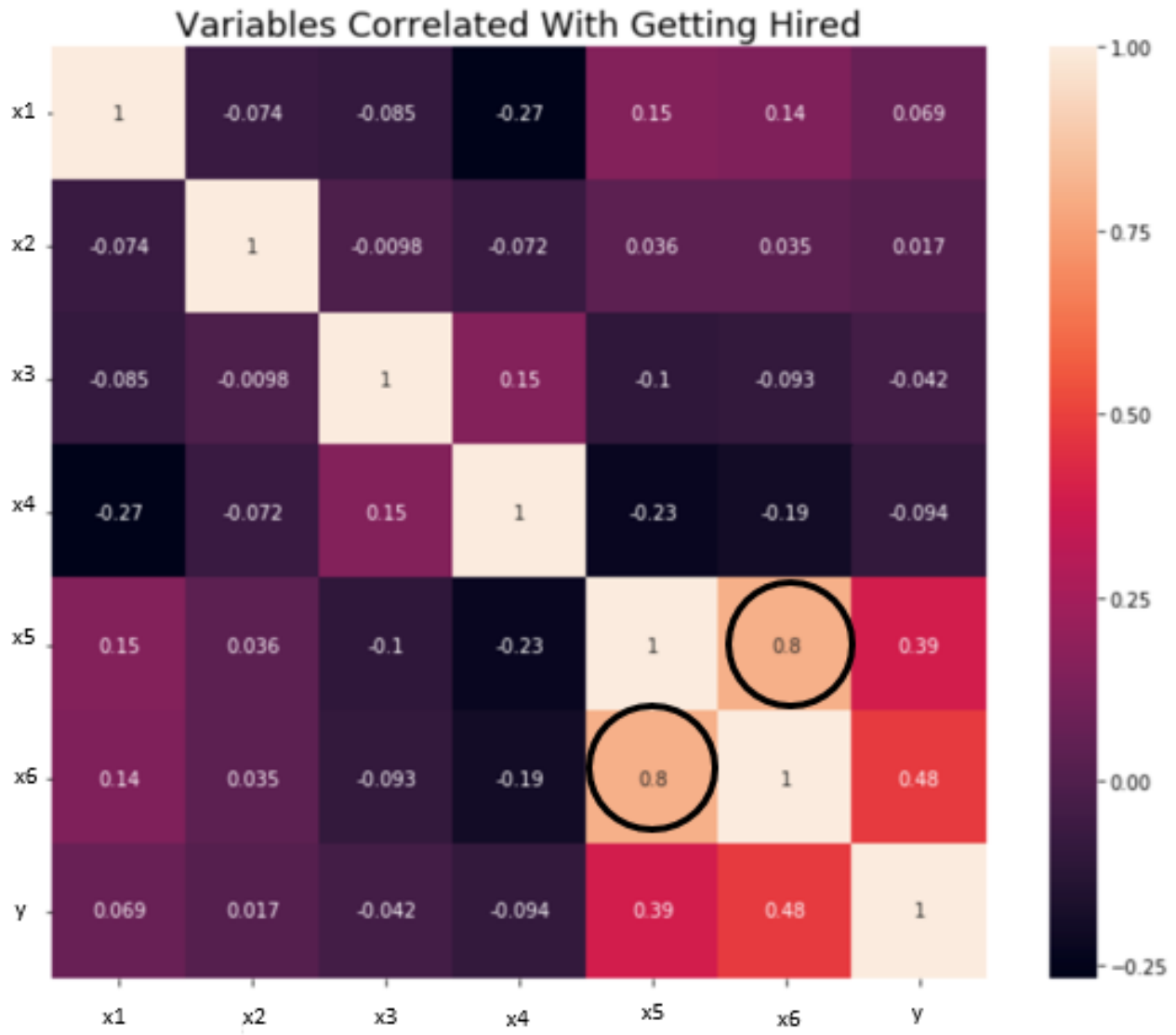
Homosce dasticity

- The variance of the error terms should be constant i.e the spread of residuals should be constant for all values of X. This assumption can be checked by plotting a residual plot. If the assumption is violated then the points will form a funnel shape otherwise they will be constant. Heteroscedasticity means a situation in which the variance of the dependent variable varies across the data



Independence/No Multicollinearity

- The variables should be independent of each other i.e no correlation should be there between the independent variables. To check the assumption, we can use a correlation matrix or VIF score(Variance Inflation Factor). If the VIF score is greater than 5 then the variables are highly correlated.



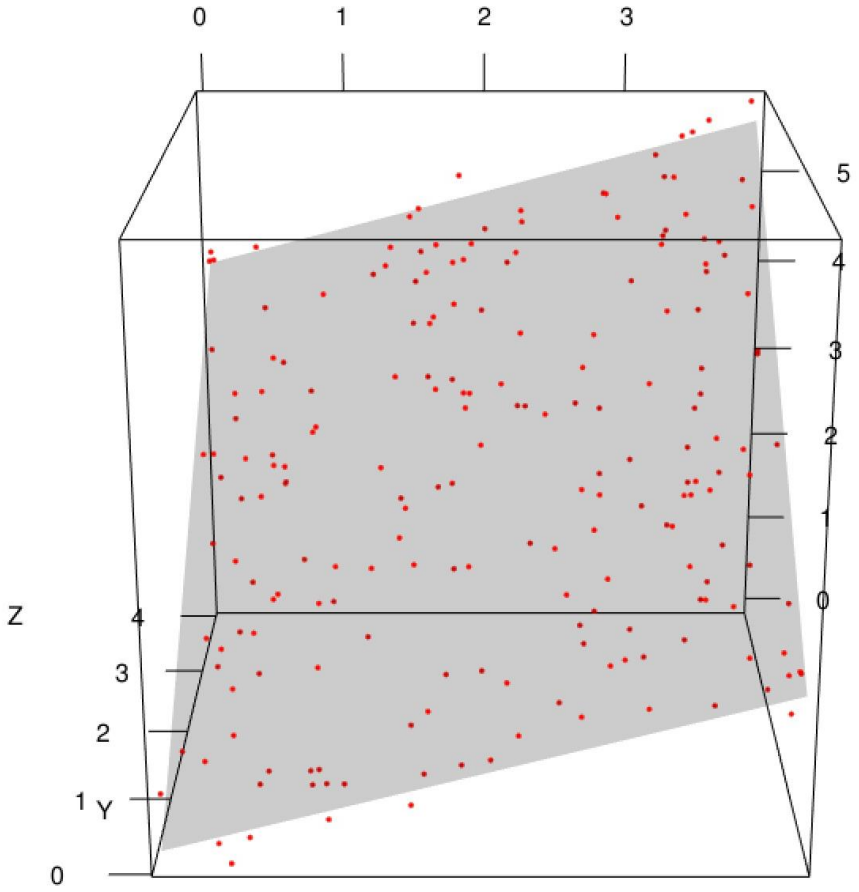
Linear Regression-Multiple

- **Multiple Linear regression**
- If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

$$h_\theta(x_1, x_2, x_3, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

Multiple Regression

- Now we have more than one dimension (x and z). We have an additional dimension. We want to express y as a combination of x and z.
- For a simple regression linear model a **straight line** expresses y as a function of x. Now we have an additional dimension (z).
- What will happen if an additional dimension is added to a line?
- **It becomes a plane.**

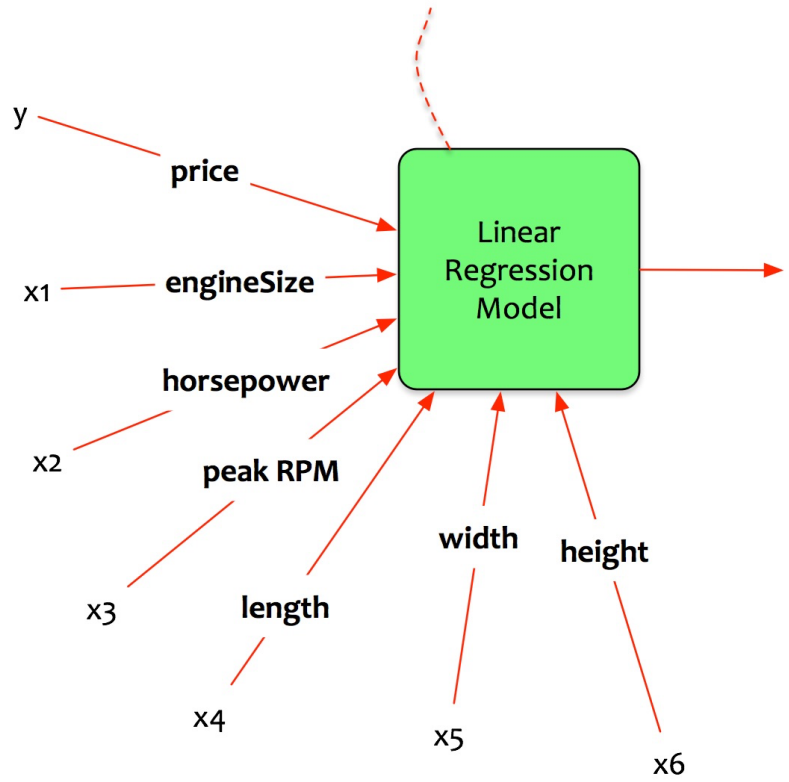


An Example with Car Dataset

- build a model that predicts the price based on the multiple data points.
- *Aim:Estimate* ***price*** *as a function of* ***engine size, horse power, peakRPM, length, width and height.***
- ***price = f(engine size, horse power, peak RPM, length, width, height)***

- ***price = β0 + β1. engine size + β2.horse power + β3. peak RPM + β4.length+ β5.width + β6.height***

- estimates β0, β1, β2, β3, β4, β5, β6
- creates model performance metrics



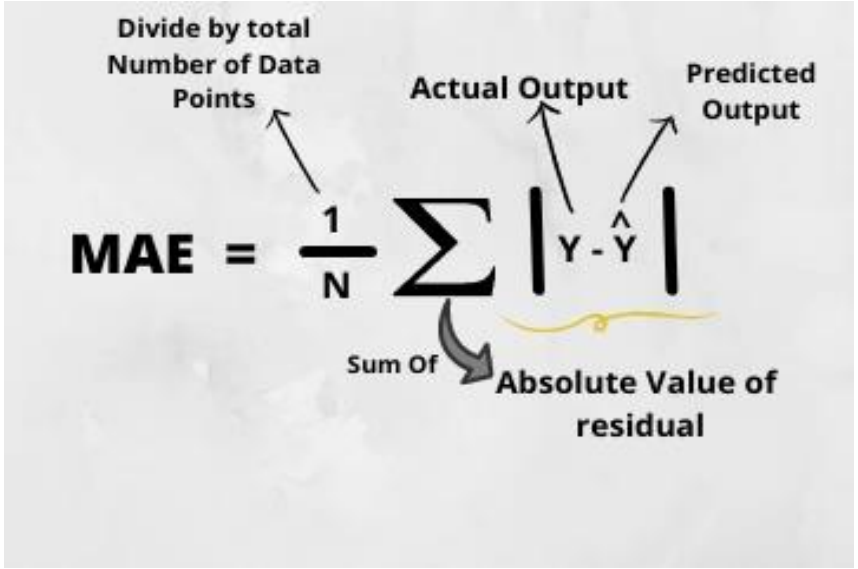
	Estimate
(Intercept)	-85086.39
engineSize	102.85
horsePower	43.79
peakRpm	1.52
length	-37.91
width	908.12
height	364.33

price = -85090 + 102.85* engineSize + 43.79* horse power + 1.52* peak RPM - 37.91* length + 908.12* width + 364.33* height

Evaluation Metrics

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- R-squared (R^2) Score
- Root Mean Squared Error (RMSE)

- **Mean Absolute Error (MAE)**
- The formula to calculate MAE for a data with “n” data points is:



- from sklearn.metrics import mean_absolute_error
  print("MAE",mean_absolute_error(y_test,y_pred))

Mean Squared Error(MSE) and Root Mean Squared Error (RMSE)

- Mean squared error states that finding the squared difference between actual and predicted value

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)}^{2}$$

The square of the difference between actual and predicted

- *from sklearn.metrics import mean_squared_error*
  print("MSE",mean_squared_error(y_test,y_pred))
- RMSE stands for Root Mean Squared Error.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$$

print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))

R-squared (R^2) Score

- also referred to as the coefficient of determination. It quantifies the percentage of the dependent variable's variation that the model's independent variables contribute to.

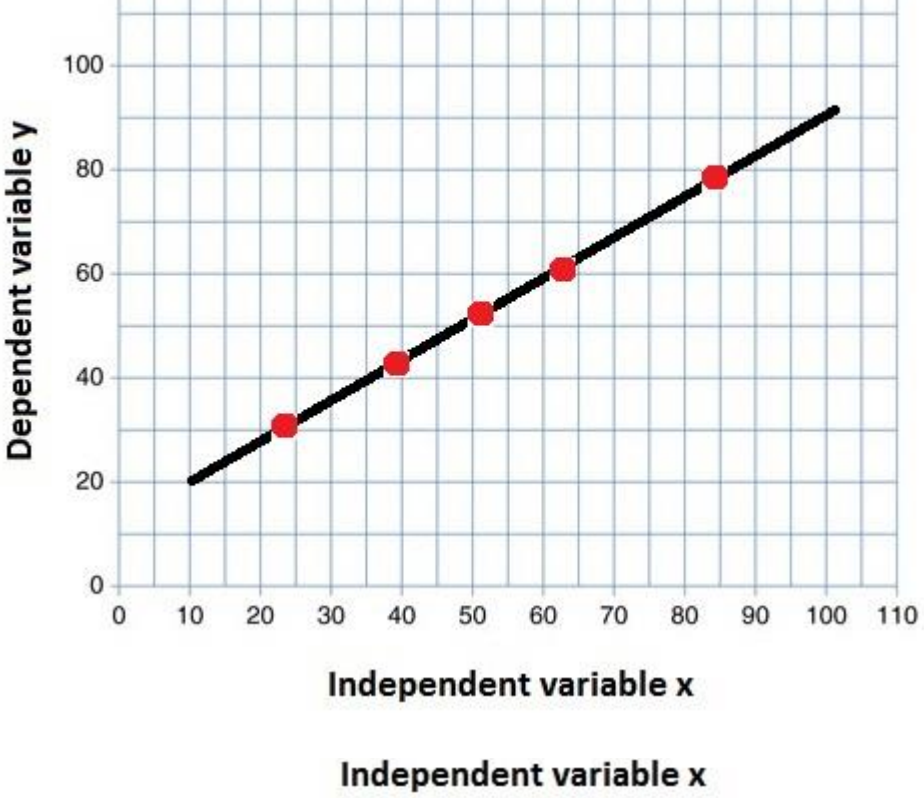
$$R^2 = 1 - \frac{\text{sum squared regression (SSR)}}{\text{total sum of squares (SST)}},$$
$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}.$$

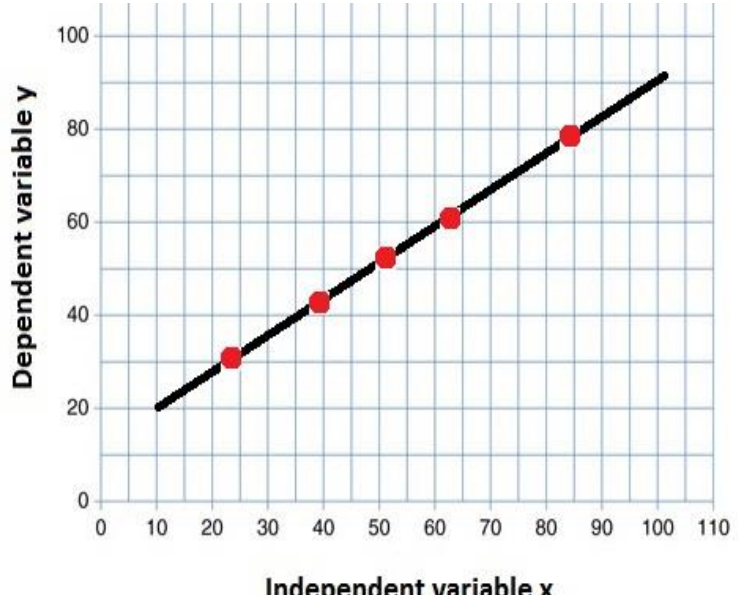
- SSR represents the sum of squared residuals between the predicted values and actual values.
- SST represents the total sum of squares, which measures the total variance in the dependent variable.
- The value of R-square lies between 0 to 1.

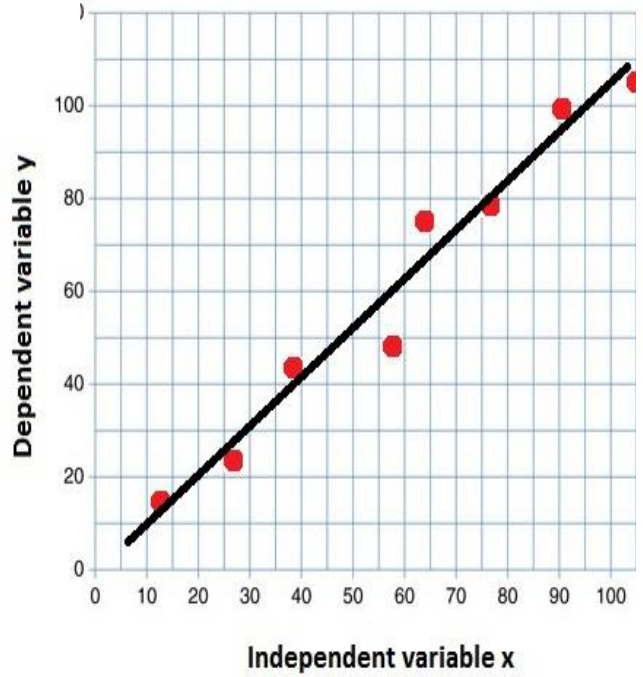
Score

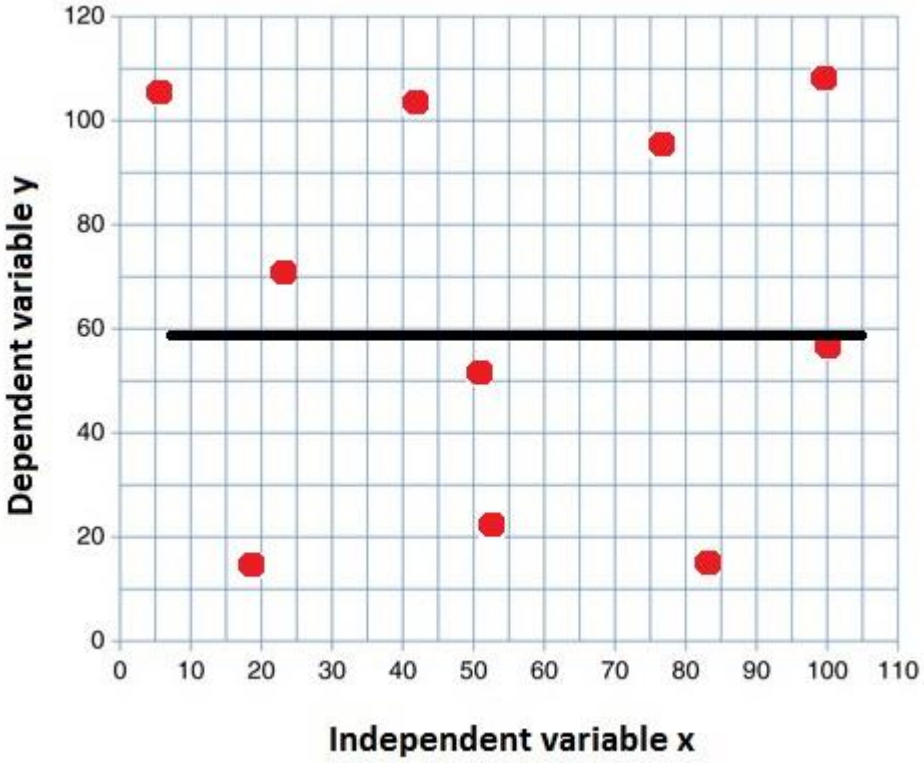
8

R2=0

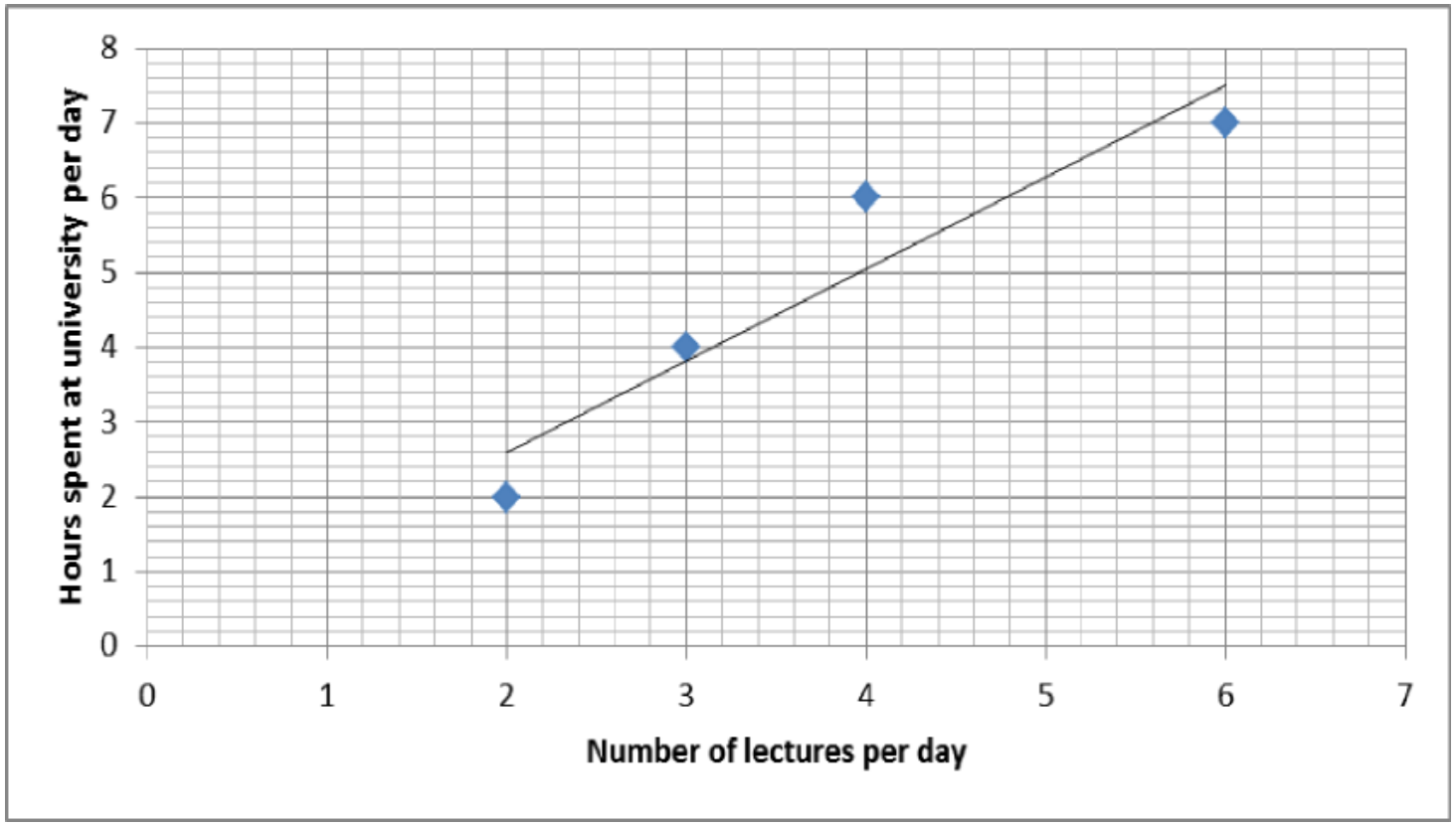

Dependent variable y
Independent variable x


Dependent variable y
Independent variable x


Dependent variable y
Independent variable x


Dependent variable y
Independent variable x

- Below is a graph showing how the number lectures per day affects the number of hours spent at university per day. The equation of the regression line is drawn on the graph and it has equation
- $\hat{y}=0.143+1.229x$. **Calculate $R2$**
-



- To calculate $R2$ we need to find the sum of the residuals squared and the total sum of squares.
- Start off by finding the residuals, which is the distance from regression line to each data point. Find the predicted y value by plugging in the corresponding x value into the regression line equation.
- For the point (2,2)
- y=0.143+1.229x=0.143+(1.229×2)=0.143+2.458=2.601
- The actual value for y is 2.
- Residual=actual y value−predicted y value
- r1=yi− ^yi=2−2.601=−0.601

- For the point (3,4)
- ^y=0.143+1.229x=0.143+(1.229×3)=0.143+3.687=3.83
  The actual value for y is 4.
- Residual=actual y value−predicted y value r2=yi−^yi=4−0.3.83=0.17

As you can see from the graph the actual point is above the regression line, so it makes sense that the residual is positive.

- For the point (4,6),
  ^y=0.143+1.229x=0.143+(1.229×4)=0.143+4.916=5.059

- The actual value for y is 66. Residual=actual y value−predicted y value
- r3=yi−^yi=6−5.059=0.941

- For the point (6,7)
- $y=0.143+1.229x=0.143+(1.229\times6)=0.143+7.374=7.517$

The actual value for y is 7.

- Residual=actual y value−predicted y value
- $r4=yi-\hat{}yi=7-7.517=-0.517$
  To find the residuals squared we need to square each of $r1$ to $r4$ and sum them.
- $\sum(yi-\hat{}yi)^2=\sum ri=r1^2+r2^2+r3^2+r4^2$
- $=(-0.601)^2+(0.17)^2+(0.941)^2-(-0.517)^2=1.542871$

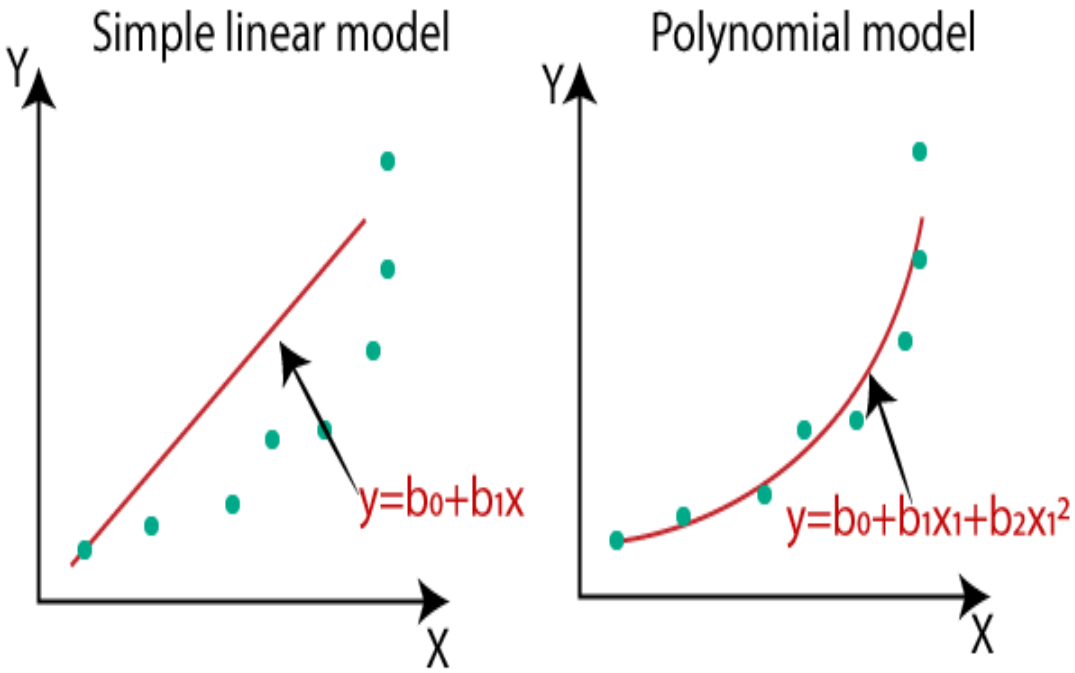
- To find $\sum(yi-\bar{y})2$ you first need to find the mean of the *y* values. $\bar{y}=\sum y/n=4.75$
- Now we can calculate $\sum(yi-\bar{y})^2$.
- $\sum(yi-\bar{y})2=(2-4.75)^2+(4-4.75)^2+(6-4.75)^2+(7-4.75)^2$
- $=(-2.75)^2+(-0.75)^2+(1.25)^2+(2.25)^2=14.75$

- Therefore;
- R2=1−sum squared regression (SSR) /total sum of squares (SST)
- $=1-\sum(yi-\hat{y}i)^2 / \sum(yi-\bar{y})2=0.895$
  This means that the number of lectures per day account for 89.589% of the variation in the hours people spend at university per day.

The problem of Non Linear Data

- If we apply a linear model on a **linear dataset**, then it provides us a good result as we have seen in Simple Linear Regression, but if we apply the same model without any modification on a **non-linear dataset**, then it will produce a drastic output. Due to which loss function will increase, the error rate will be high, and accuracy will be decreased.

- So for such cases, **where data points are arranged in a non-linear fashion, we need the Polynomial Regression model**. We can understand it in a better way using the below comparison diagram of the linear dataset and non-linear dataset.



The problem of Non Linear Data

- The problem of non-linear regression can be solved by
- A)Transform the non linear data into linear data and then use linear regression
- B)Use Polynomial Regression
- **A)Transform the non linear data into linear data,use the following equation $y= ae^{bx}$**

$\ln(y)= \ln(\mathbf{ae^{bx}})$

- $\ln(y)=\ln(\mathbf{a}) + \mathbf{bx} * \ln(\mathbf{e})$ **[ln(e)=1]**
- $\ln(y)=\ln(\mathbf{a}) + \mathbf{bx}$

Polynomial Regression

- Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as n^{th} degree polynomial. The Polynomial Regression equation is given below:

$$y= b_0+b_1x_1+ b_2x_1^2+ b3x_1^3+..... b_nx_1^n$$

- The dataset used in Polynomial regression for training is of non-linear nature.
- It makes use of a linear regression model to fit the complicated and non-linear functions and datasets.
- ***In Polynomial regression, the original features are converted into Polynomial features of required degree (2,3,..,n) and then modeled using a linear model."***

Polynomial Regression

- Consider $y= a_0+ a_1x+ a_2x^2$
- The coefficients a_0, a_1, a_2 is calculated using $a= X^{-1}b$

$$X = \begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \quad B = \begin{bmatrix} \sum y_i \\ \sum (x_i, y_i) \\ \sum (x_i^2, y_i) \end{bmatrix}$$

Polynomial Regression

X	Y
1	1
2	4
3	9
4	15

X_i	Y_i	Xi Yi	Xi^2	Xi^2 Y	Xi^3	Xi^4
1	1	1	1	1	1	1
2	4	8	4	16	8	16
3	9	27	9	81	27	81
4	15	60	16	240	64	256
Sum =10	29	96	30	338	100	354

- Apply in $a = X^{-1}b$

$$\begin{bmatrix} a0 \\ a1 \\ a2 \end{bmatrix} = \text{inverse} \begin{bmatrix} 4 & 10 & 30 \\ 10 & 30 & 100 \\ 30 & 100 & 354 \end{bmatrix} * \begin{bmatrix} 29 \\ 96 \\ 338 \end{bmatrix} = \begin{bmatrix} -0.75 \\ 0.95 \\ 0.75 \end{bmatrix}$$

$Y=-0.75+0.95x+0.75x^2$

Example

- There is a Human Resource company, which is going to hire a new candidate. The candidate has told his previous salary 160K per annum, and the HR have to check whether he is telling the truth or bluff.
- So to identify this, they only have a dataset of his previous company in which the salaries of the top 10 positions are mentioned with their levels.
- By checking the dataset available, we have found that there is a **non-linear relationship between the Position levels and the salaries**. Our goal is to build a **Bluffing detector regression** model, so HR can hire an honest candidate.

Position	Level(X-variable)	Salary(Y-Variable)
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

Steps for Polynomial Regression:

- Data Pre-processing
- Build a Linear Regression model and fit it to the dataset
- Build a Polynomial Regression model and fit it to the dataset
- Visualize the result for Linear Regression and Polynomial Regression model.
- Predicting the output.

Example Code

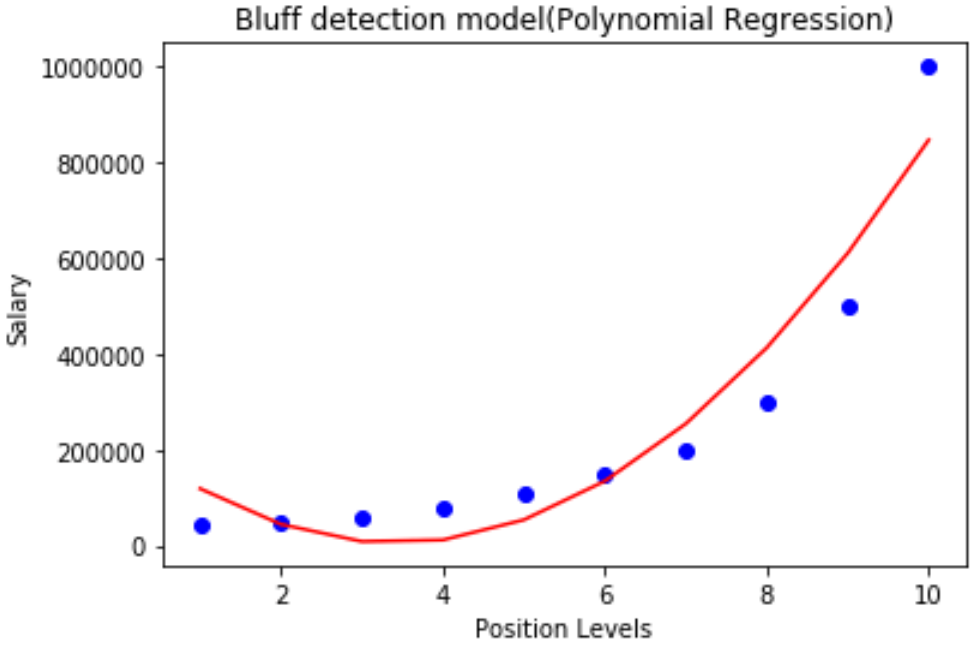
```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd


#importing datasets
data_set= pd.read_csv('Position_Salaries.csv')
```

```
#Extracting Independent and dependent Variable
x= data_set.iloc[:, 1:2].values
y= data_set.iloc[:, 2].values
#Fitting the Linear Regression to the dataset
from sklearn.linear_model
import LinearRegression
lin_regs= LinearRegression()  lin_regs.fit(x,y)
```

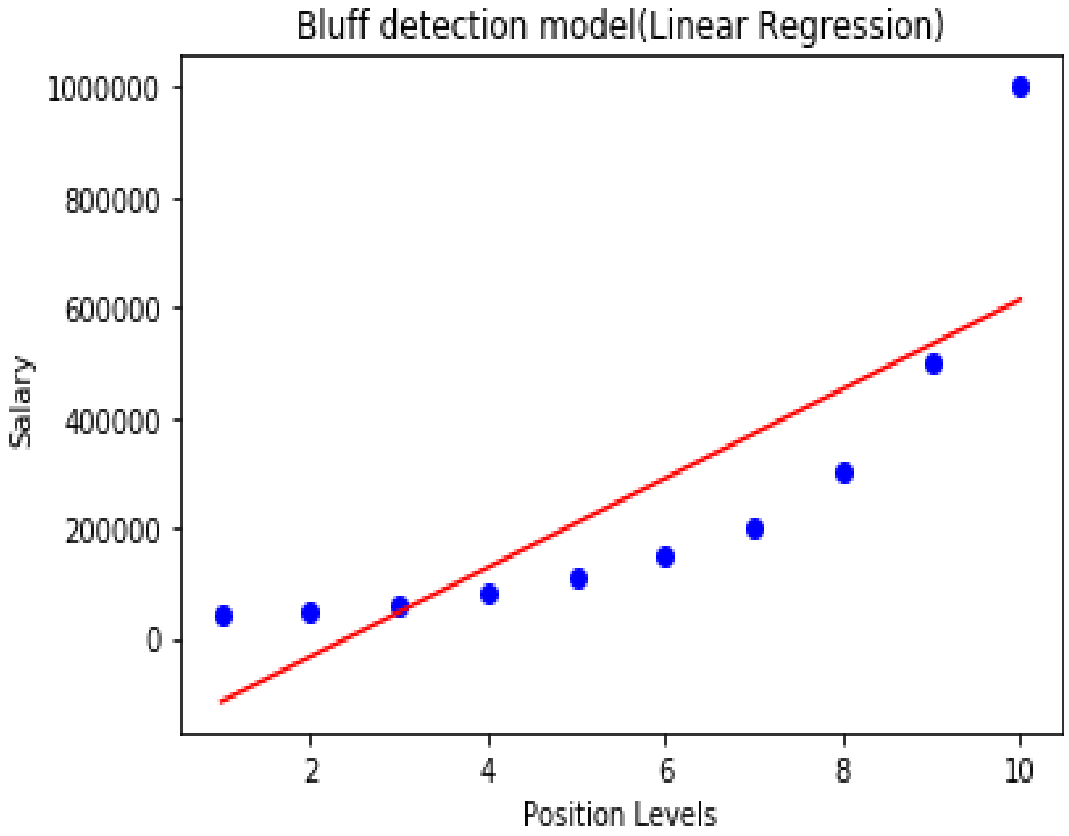
```
#Fitting the Polynomial regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_regs= PolynomialFeatures(degree= 2)
x_poly= poly_regs.fit_transform(x)
lin_reg_2 =LinearRegression()
lin_reg_2.fit(x_poly, y)
```

Polynomial Regression

```
#Visulaizing the result for Polynomial Regression
mtp.scatter(x,y,color="blue")
mtp.plot(x, lin_reg_2.predict(poly_regs.fit_transform(x)), color="red")
mtp.title("Bluff detection model(Polynomial Regression)")
mtp.xlabel("Position Levels")
mtp.ylabel("Salary")
mtp.show()
```



```
#Visulaizing the result for Linear Regression model
mtp.scatter(x,y,color="blue")
mtp.plot(x,lin_regs.predict(x), color="red")
mtp.title("Bluff detection model(Linear Regression)")
mtp.xlabel("Position Levels")
mtp.ylabel("Salary")
mtp.show()
```



- Sum = Minimum Quantity
- Suppose when we have to determine the equation of line of best fit for the given data, then we first use the following formula.
- The equation of least square line is given by $Y = a + bX$
- Normal equation for ‘a’:
- $\sum Y = na + b\sum X$
- Normal equation for ‘b’:
- $\sum XY = a\sum X + b\sum X^2$
- Solving these two normal equations we can get the required trend line equation.
- Thus, we can get the line of best fit with formula $y = a + bX$

Example

- The Least Squares Model for a set of data (x_1, y_1), (x_2, y_2), (x_3, y_3), …, (x_n, y_n) passes through the point (x_a, y_a), where x_a is the average of the x_i's and y_a is the average of the y_i's.
- The below example explains how to find the equation of a straight line or a least square line using the least square method.
- Consider the following data points

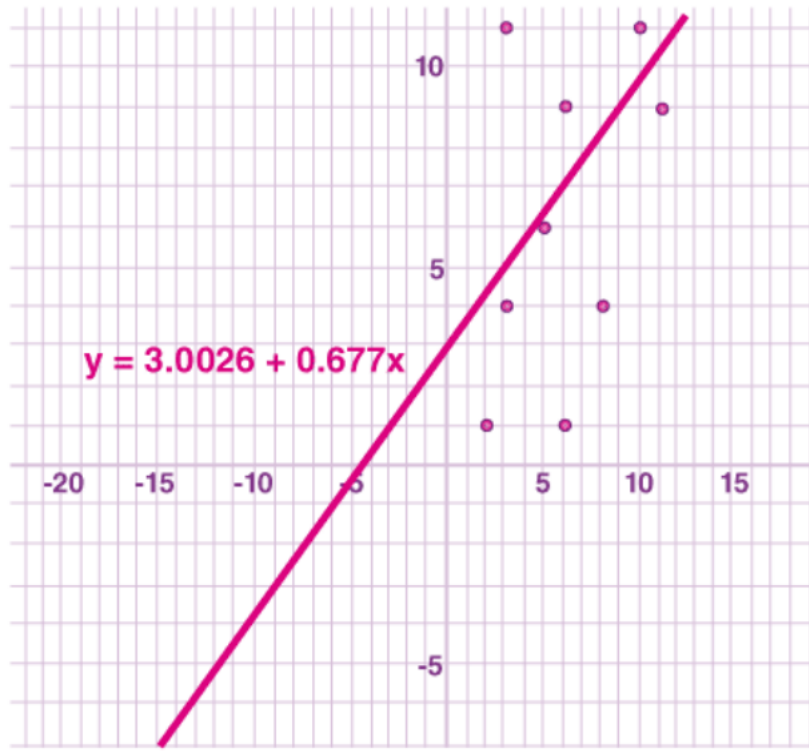
x_i	8	3	2	10	11	3	6	5	6	8
y_i	4	12	1	12	9	4	9	6	1	14

- Mean of x_i values = (8 + 3 + 2 + 10 + 11 + 3 + 6 + 5 + 6 + 8)/10 = 62/10 = 6.2
- Mean of y_i values = (4 + 12 + 1 + 12 + 9 + 4 + 9 + 6 + 1 + 14)/10 = 72/10 = 7.2
- Straight line equation is $y = a + bx$.
- The normal equations are
- $\sum y = an + b\sum x$
- $\sum xy = a\sum x + b\sum x^2$

x	y	x^2	xy
8	4	64	32
3	12	9	36
2	1	4	2
10	12	100	120
11	9	121	99
3	4	9	12
6	9	36	54
5	6	25	30
6	1	36	6
8	14	64	112
$\sum x = 62$	$\sum y = 72$	$\sum x^2 = 468$	$\sum xy = 503$

- Substituting these values in the normal equations,
- $10a + 62b = 72$….(1)
- $62a + 468b = 503$….(2)
- (1) × 62 – (2) × 10,
- $620a + 3844b – (620a + 4680b) = 4464 – 5030$
- $-836b = -566$
- $b = 566/836$
- $b = 283/418$
- $b = 0.677$

- Substituting b = 0.677 in equation (1),
- 10a + 62(0.677) = 72
- 10a + 41.974 = 72
- 10a = 72 – 41.974
- 10a = 30.026
- a = 30.026/10
- a = 3.0026
- Therefore, the equation becomes,
- y = a + bx
- y = 3.0026 + 0.677x
-


y = 3.0026 + 0.677x
10
5
-5
-20
-15
-10
5
10
15

- Now, we can find the sum of squares of deviations from the obtained values as:
- $d_1 = [4 - (3.0026 + 0.677*8)] = (-4.4186)$
- $d_2 = [12 - (3.0026 + 0.677*3)] = (6.9664)$
- $d_3 = [1 - (3.0026 + 0.677*2)] = (-3.3566)$
- $d_4 = [12 - (3.0026 + 0.677*10)] = (2.2274)$
- $d_5 = [9 - (3.0026 + 0.677*11)] = (-1.4496)$
- $d_6 = [4 - (3.0026 + 0.677*3)] = (-1.0336)$
- $d_7 = [9 - (3.0026 + 0.677*6)] = (1.9354)$
- $d_8 = [6 - (3.0026 + 0.677*5)] = (-0.3876)$
- $d_9 = [1 - (3.0026 + 0.677*6)] = (-6.0646)$
- $d_{10} = [14 - (3.0026 + 0.677*8)] = (5.5814)$
- $\sum d^2 = (-4.4186)^2 + (6.9664)^2 + (-3.3566)^2 + (2.2274)^2 + (-1.4496)^2 + (-1.0336)^2 + (1.9354)^2 + (-0.3876)^2 + (-6.0646)^2 + (5.5814)^2 = 159.27990$

Multivariate Linear Regression

- Multivariate linear regression resembles simple linear regression except that in multivariate linear regression, *multiple independent variables* contribute to the dependent variables and so multiple coefficients are used in the computation.
- It is used to derive a *mathematical relationship* amongst *multiple random variables*. It explains how many multiple independent variables are associated with one dependent variable.

- The details of the *multiple independent variables* are used to make an accurate prediction of the influence they have on the outcome variable.
- Multivariate linear regression model generates a relationship in a linear form (a form of a straight line) with the best approximation of each data point.
- The equation of the Multivariate linear regression model is:

$$y = \beta_0 + \beta_1.x_1 + \ldots + \beta_n.x_n$$

Regression cost Function:

- Regression models deal with predicting a continuous value for example salary of an employee, price of a car, loan prediction, etc.
- A cost function used in the regression problem is called “Regression Cost Function”. They are calculated on the distance-based error as follows:

- Error = y-y’

- Where,

- Y – Actual Input
- Y’ – Predicted output

Decision Tree-Introduction

- Classification is a two-step process, ***learning step*** and ***prediction step***, in machine learning.
- In the learning step, the model is developed based on given training data. In the prediction step, the model is used to predict the response for given data.
- Decision Tree is one of the easiest and popular classification algorithms to understand and interpret.
- The decision tree algorithm can be used for solving **regression and classification problems**

Decision Tree

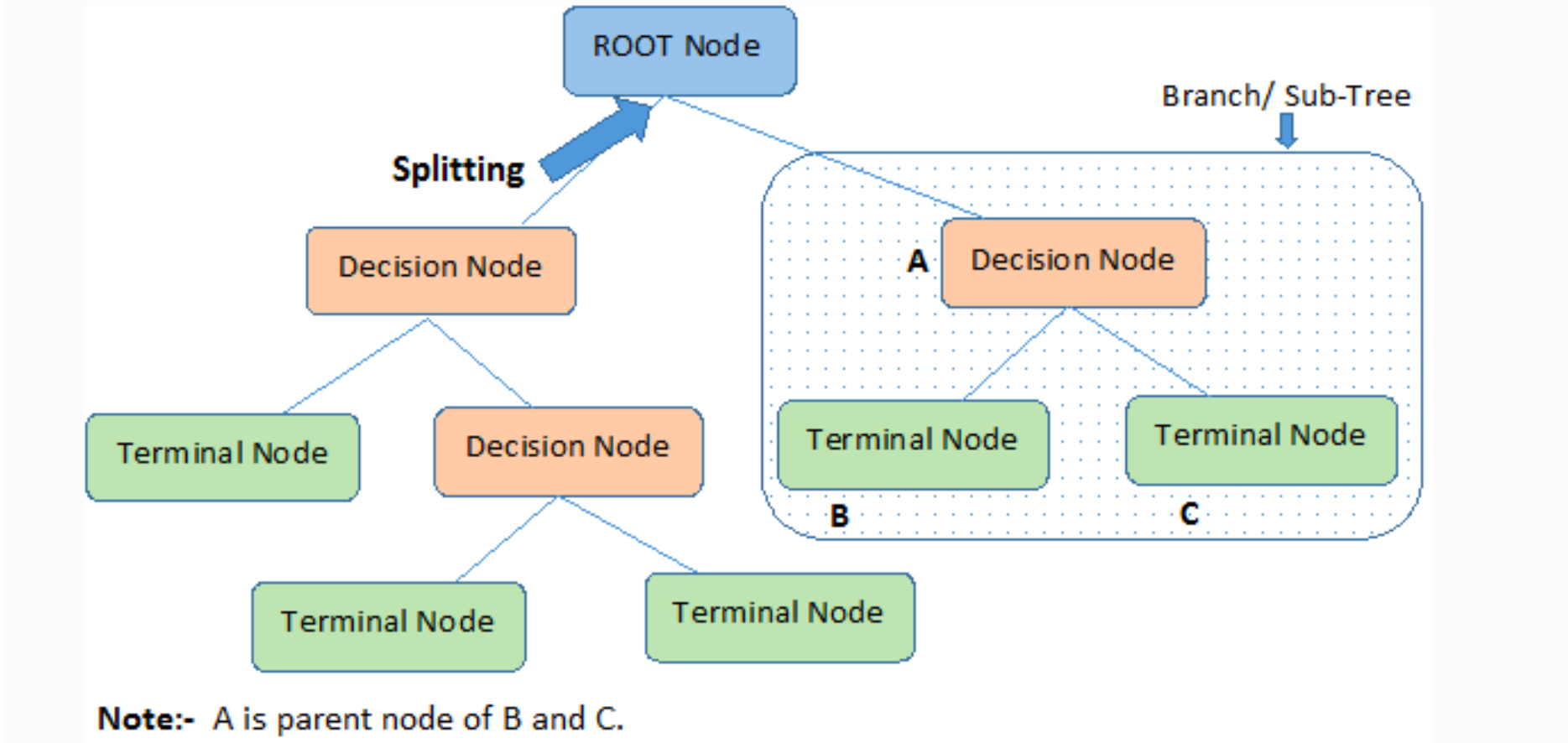
- The ***goal*** of using a Decision Tree is to *create a training model* that can use to *predict the class* or value of the target variable by **learning simple decision rules** inferred from training data.
- In Decision Trees, for predicting a class label for a record we start from the **root** of the tree.
- We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Decision Tree

- Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.
- **Each node in the tree acts as a test case** for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.

Terminologies

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning.By removing error-prone components, the classifier's performance may be improved
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.



Note:- A is parent node of B and C.

- The primary challenge in the decision tree implementation is to identify *which attributes do we need to consider as the root node and each level*.
- Handling this is to know as the **attributes selection**. We have different attributes selection measures to identify the attribute which can be considered as the root node at each level.
- Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes.
- The creation of *sub-nodes increases* the homogeneity of resultant sub-nodes. In other words, we can say that the *purity of the node increases with respect to the target variable*. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

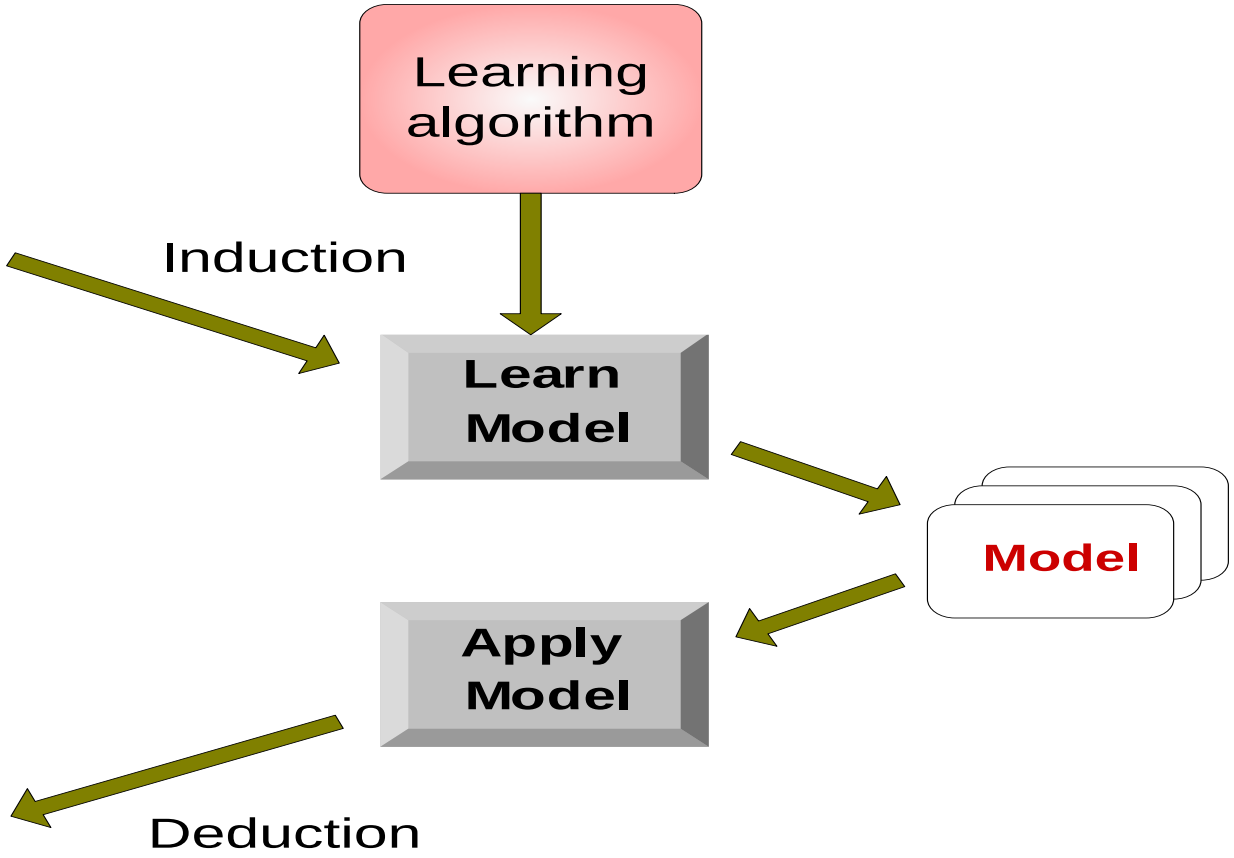
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

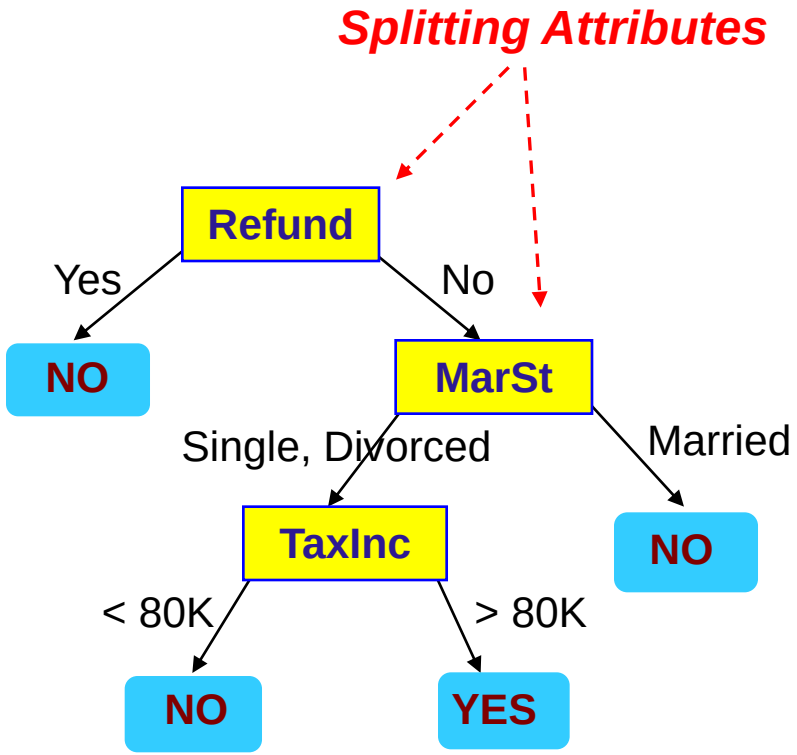
Test Set



Example of a Decision Tree

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

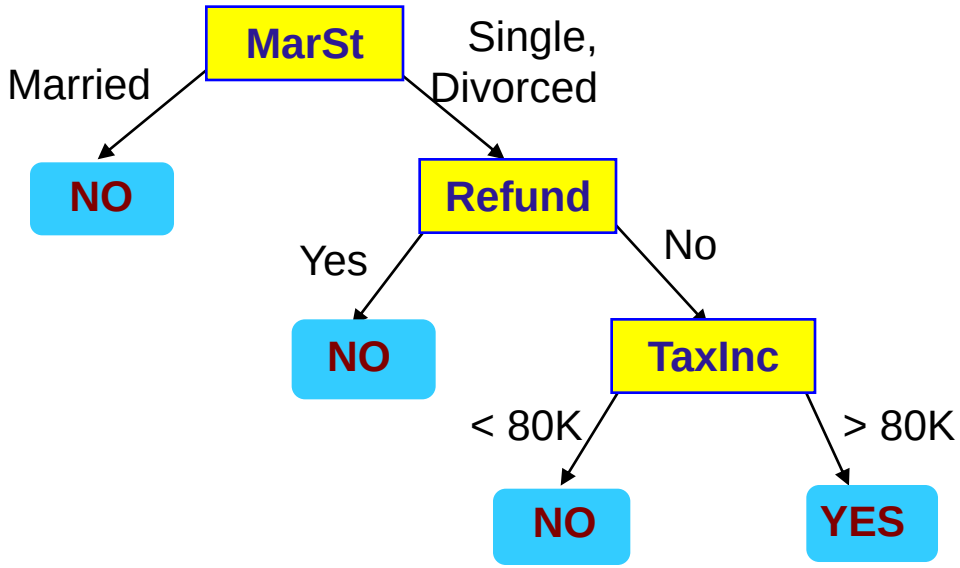
Training Data



Model: Decision Tree

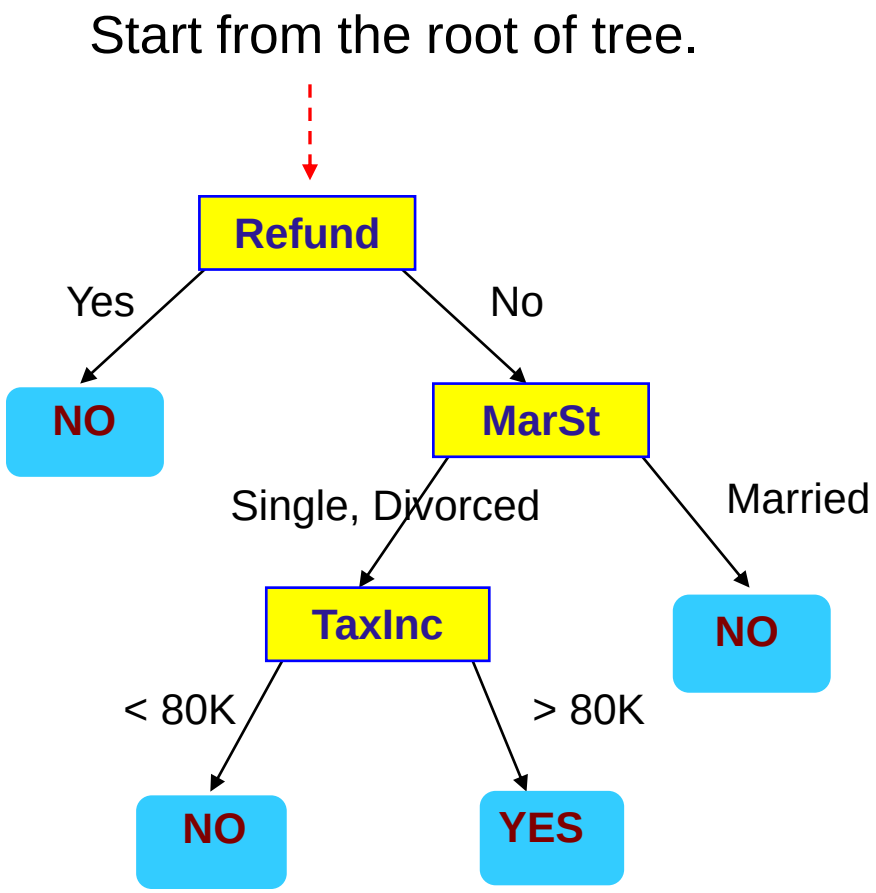
Another Example of Decision Tree

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

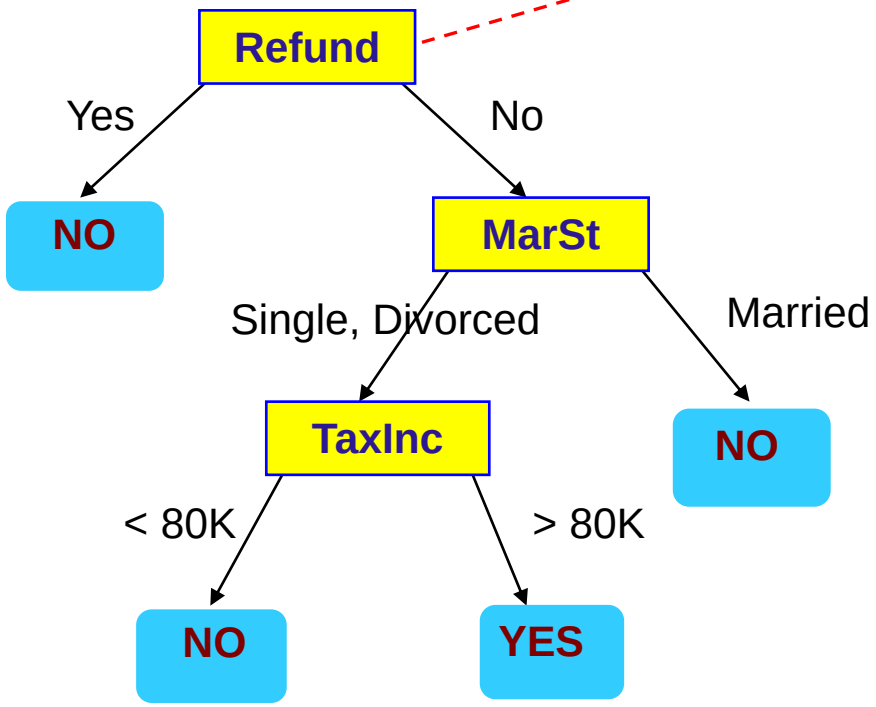
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

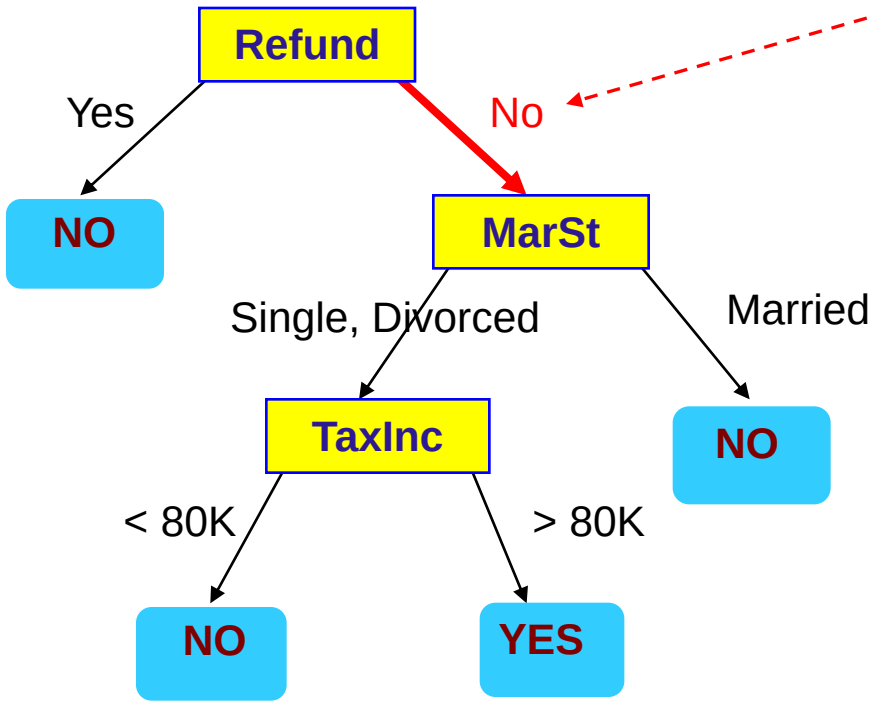
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

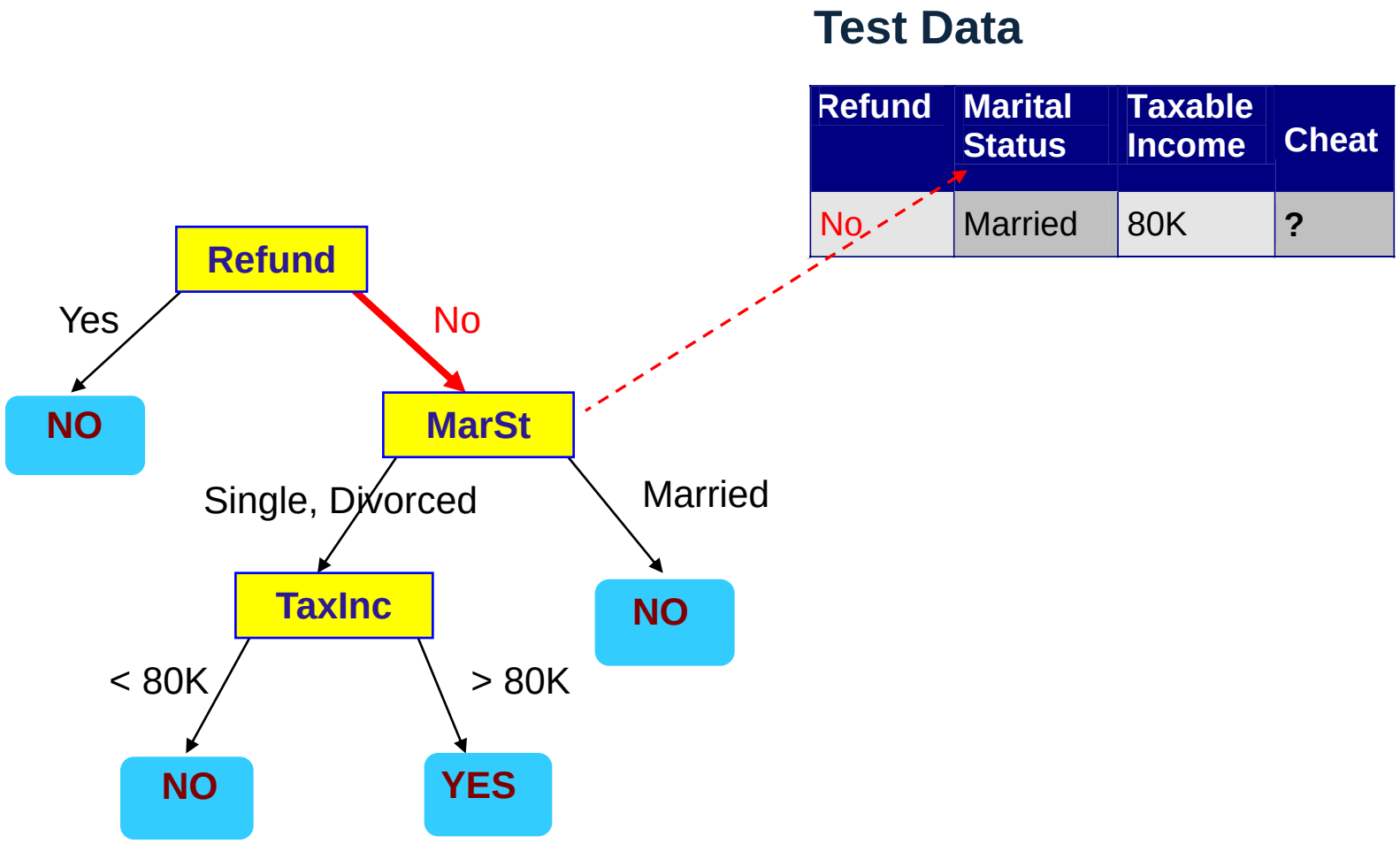
Apply Model to Test Data



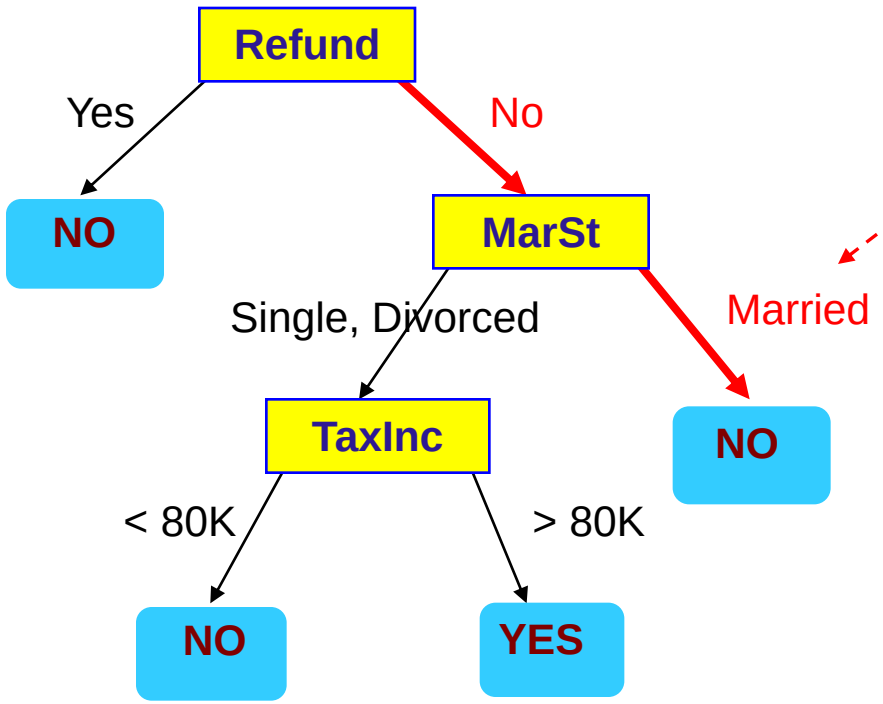
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



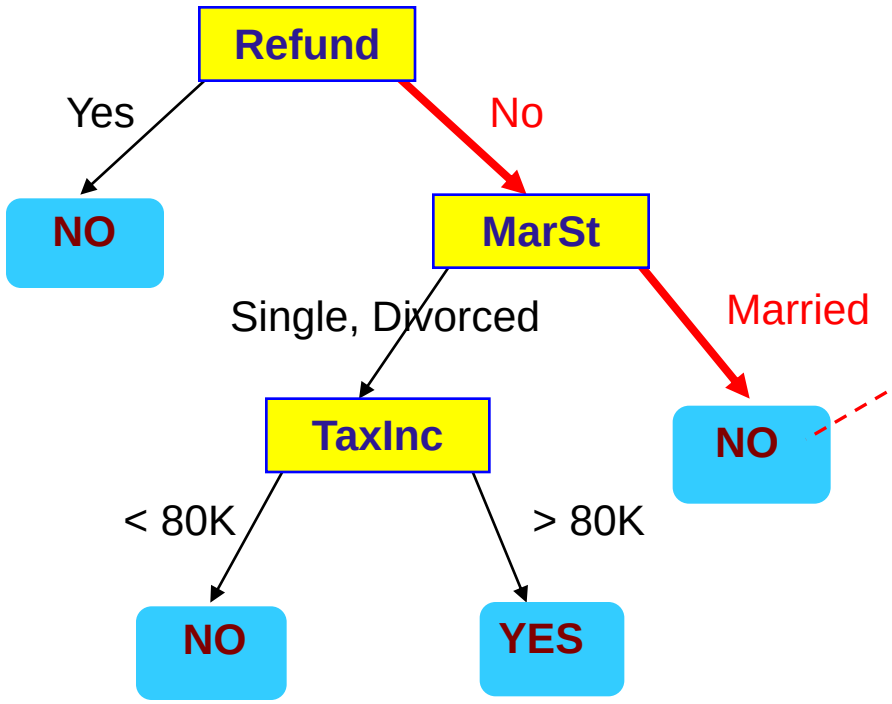
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

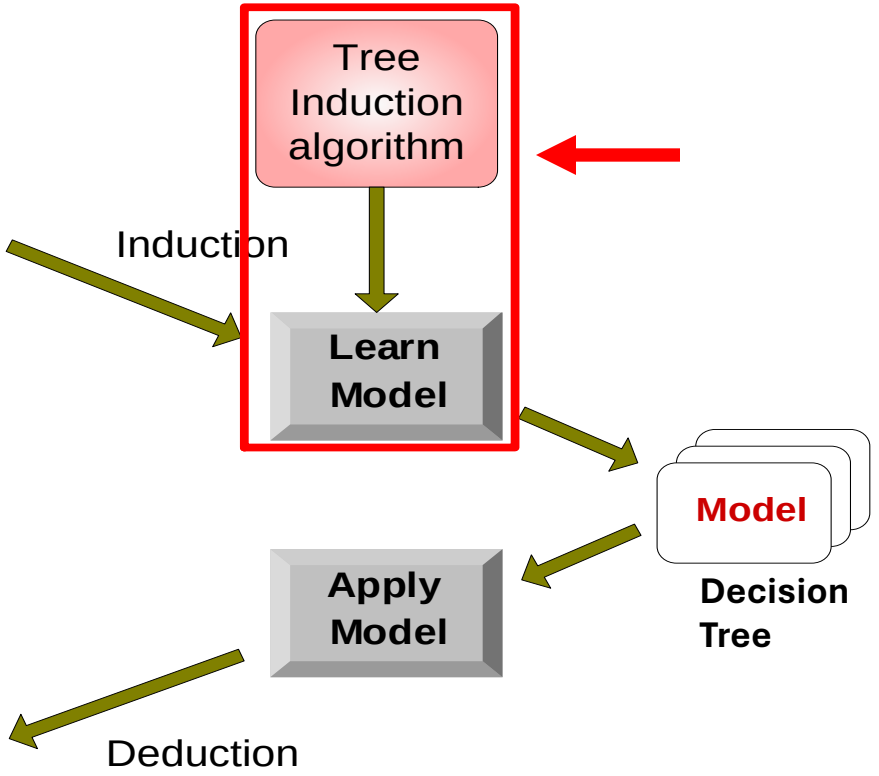
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

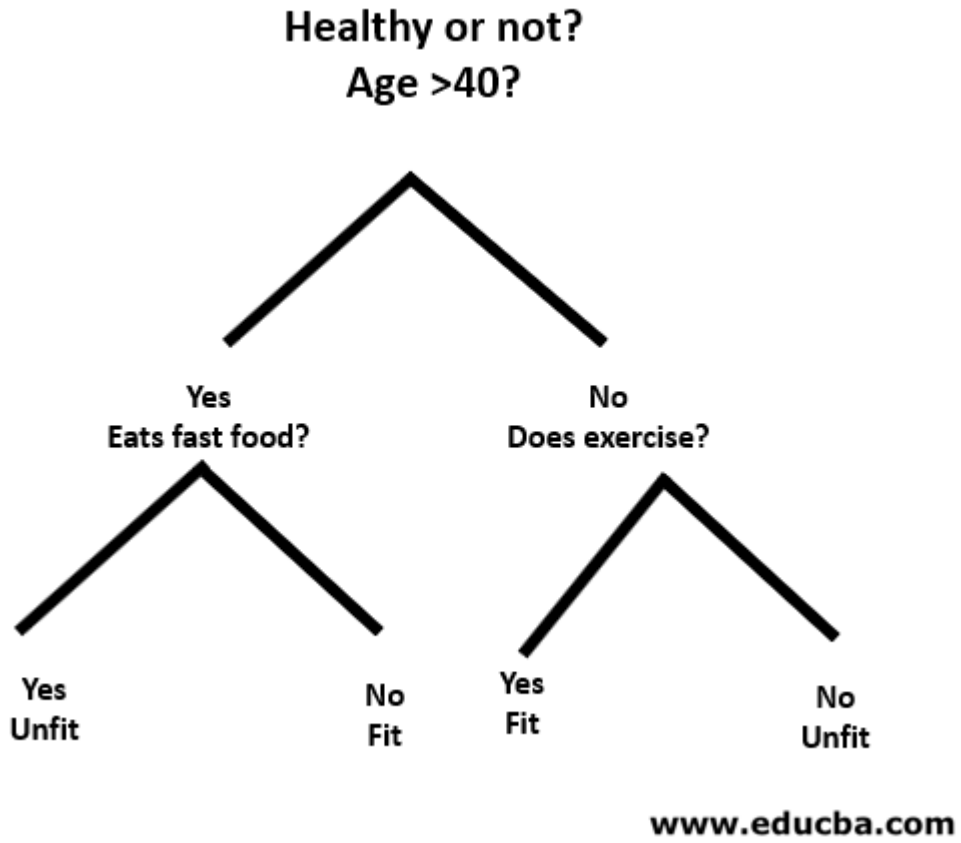


Exercise –Construct a Decision tree for Fruit Classification

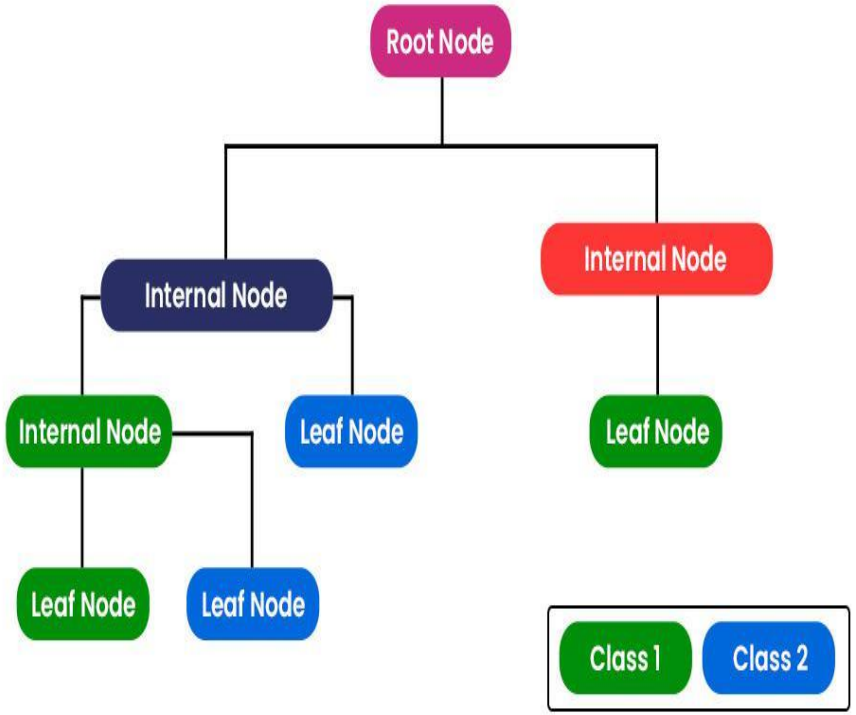
Color	Size	Fruit
Red	Small	Cherry
Red	Large	Apple
Yellow	Large	Banana
Yellow	Small	Lemon
Green	Small	Grape
Green	Large	Watermelon

Example-Decision Tree

- Example



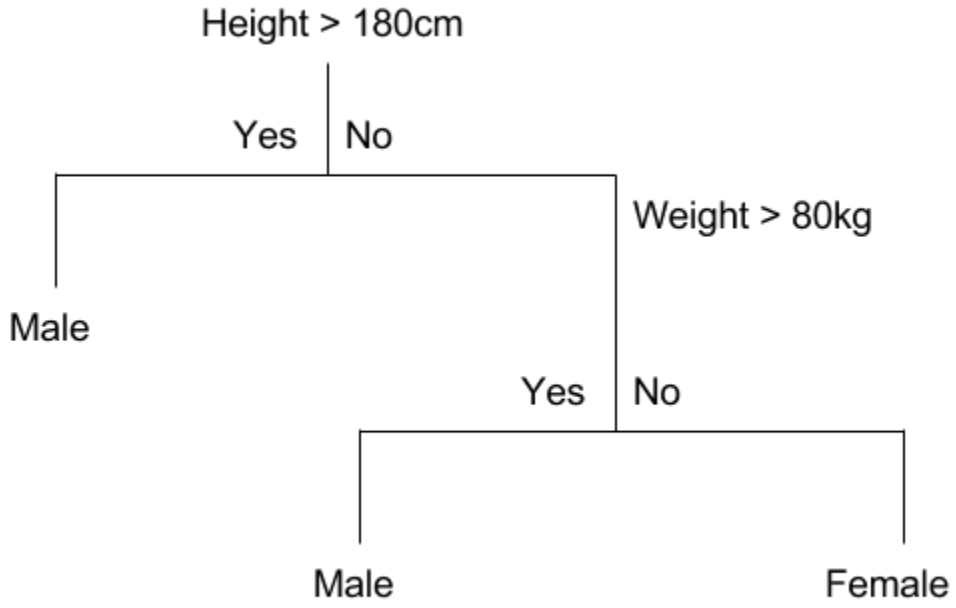
CART Model Representation



- **CART(Classification And Regression Trees)** is a variation of the decision tree algorithm. It can handle both classification and regression tasks.
- **Classification Trees:** The tree is used to determine which "class" the target variable is most likely to fall into when it is continuous.
- **Regression trees:** These are used to predict a continuous variable's value.
- The representation for the CART model is a *binary tree*.
- Each root node represents a single input variable (x) and a split point on that variable.
- The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.

- Given a dataset with two inputs (x) of height in centimeters and weight in kilograms the output of gender as male or female, below is a crude example of a binary decision tree
- For example, given the input of [height = 160 cm, weight = 65 kg], we would traverse the above tree as follows:

- Height > 180 cm: No
- Weight > 80 kg: No
- Therefore: Female



Learn a CART Model From Data

- Creating a CART model involves selecting input variables and split points on those variables until a suitable tree is constructed.
- *CART (Classification and Regression Trees) → uses Gini Index(Classification) as metric.*
- If all the data belong to a single class, then it can be called pure.
- Its Degree will be always between 0 and 1. ***If 0, means all data belongs to the single class/variable***. ***If 1, the data belong to the different class/field.***

GINI INDEX

- Gini Index is the metric for classification task in CART. It is mostly used when the target variable is categorical. It stores the sum of squared probabilities of each class.
- Gini Impurity= 1- Gini
- Gini impurity = $$1 - \sum_{i=1}^{C} (p_i)^2$$
- where pi is the probability of an object being classified to a particular class.

Steps in CART

- CART for classification works by recursively splitting the training data into smaller and smaller subsets based on certain criteria
- The best-split point of each input is obtained.
- Based on the best-split points of each input in Step 1, the new “best” split point is identified.
- Split the chosen input according to the “best” split point.
- Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.

Data set

There are 14 instances of golf playing decisions based on outlook, temperature, humidity and wind factors.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- **Gini index**
- Gini index is a metric for classification tasks in CART. It stores sum of squared probabilities of each class. We can formulate it as illustrated below.
    - Gini = $1 – \Sigma (Pi)^2$, for i=1 to number of classes

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

$Gini(Outlook=Sunny) = 1 - (2/5)^2 - (3/5)^2 = 1 - 0.16 - 0.36 = 0.48$

$Gini(Outlook=Overcast) = 1 - (4/4)^2 - (0/4)^2 = 0$

- $Gini(Outlook=Rain) = 1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$
- Then, we will calculate weighted sum of gini indexes for outlook feature.
- $Gini(Outlook) = (5/14) \times 0.48 + (4/14) \times 0 + (5/14) \times 0.48 = 0.171 + 0 + 0.171 = 0.342$

CART

- **Temperature**
- Similarly, temperature is a nominal feature and it could have 3 different values: Cool, Hot and Mild. Let's summarize decisions for temperature feature.
- Gini(Temp=Hot) = $1 – (2/4)^2 – (2/4)^2 = 0.5$
- Gini(Temp=Cool) = $1 – (3/4)^2 – (1/4)^2 = 1 – 0.5625 – 0.0625 = 0.375$
- Gini(Temp=Mild) = $1 – (4/6)^2 – (2/6)^2 = 1 – 0.444 – 0.111 = 0.445$
- We'll calculate weighted sum of gini index for temperature feature
- Gini(Temp) = $(4/14) \times 0.5 + (4/14) \times 0.375 + (6/14) \times 0.445 = 0.142 + 0.107 + 0.190 = 0.439$

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

CART

- **Humidity**
- Humidity is a binary class feature. It can be high or normal.
- Gini(Humidity=High) = $1 – (3/7)^2 – (4/7)^2 = 1 – 0.183 – 0.326 = 0.489$
- Gini(Humidity=Normal) = $1 – (6/7)^2 – (1/7)^2 = 1 – 0.734 – 0.02 = 0.244$
- Weighted sum for humidity feature will be calculated next
- Gini(Humidity) = (7/14) x 0.489 + (7/14) x 0.244 = 0.367

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

- **Wind**
- Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

- Gini(Wind=Weak) = $1 – (6/8)^2 – (2/8)^2 = 1 – 0.5625 – 0.062 = 0.375$
- Gini(Wind=Strong) = $1 – (3/6)^2 – (3/6)^2 = 1 – 0.25 – 0.25 = 0.5$
- Gini(Wind) = $(8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$

CART

- We’ve calculated gini index values for each feature. The winner will be outlook feature because **its cost is the lowest.**

Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

Outlook

Sunny

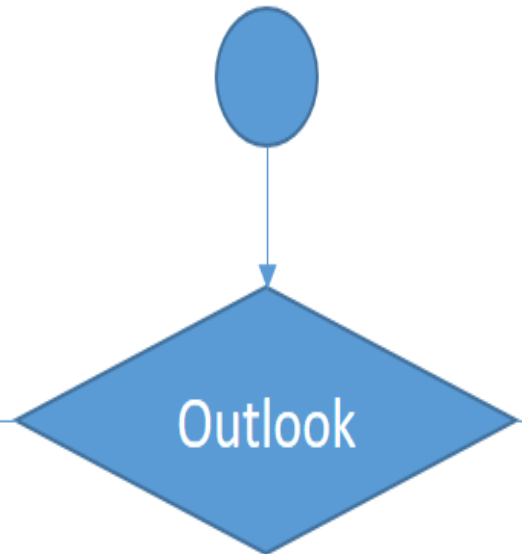
Overcast

Rain

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes



Sunny

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Overcast

Yes

Rain

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- Focus on the **sub dataset for sunny outlook**. We need to find the gini index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

Gini(Outlook=Sunny and Temp.=Hot) = $1 - (0/2)^2 - (2/2)^2 = 0$
Gini(Outlook=Sunny and Temp.=Cool) = $1 - (1/1)^2 - (0/1)^2 = 0$
Gini(Outlook=Sunny and Temp.=Mild) = $1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$

Gini(Outlook=Sunny and Temp.) = (2/5)x0 + (1/5)x0 + (2/5)x0.5 = 0.2

Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

Gini(Outlook=Sunny and Humidity=High) = $1 - (0/3)^2 - (3/3)^2 = 0$

Gini(Outlook=Sunny and Humidity=Normal) = $1 - (2/2)^2 - (0/2)^2 = 0$

Gini(Outlook=Sunny and Humidity) = $(3/5)x0 + (2/5)x0 = 0$

Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

$$Gini(Outlook=Sunny \text{ and } Wind=Weak) = 1 - (1/3)^2 - (2/3)^2 = 0.266$$

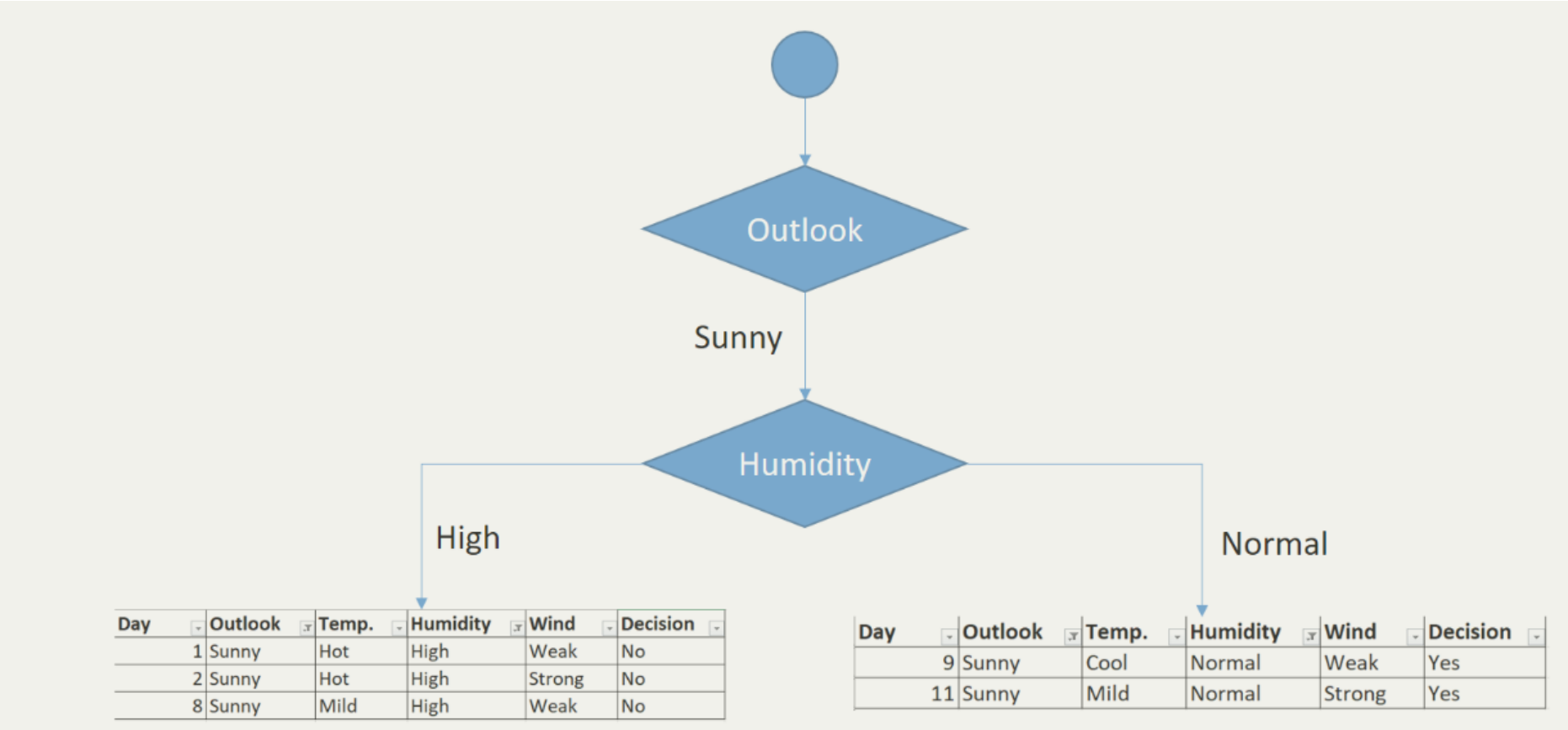
$$Gini(Outlook=Sunny \text{ and } Wind=Strong) = 1 - (1/2)^2 - (1/2)^2 = 0.2$$

$$Gini(Outlook=Sunny \text{ and } Wind) = (3/5) \times 0.266 + (2/5) \times 0.2 = 0.466$$

Decision for sunny outlook

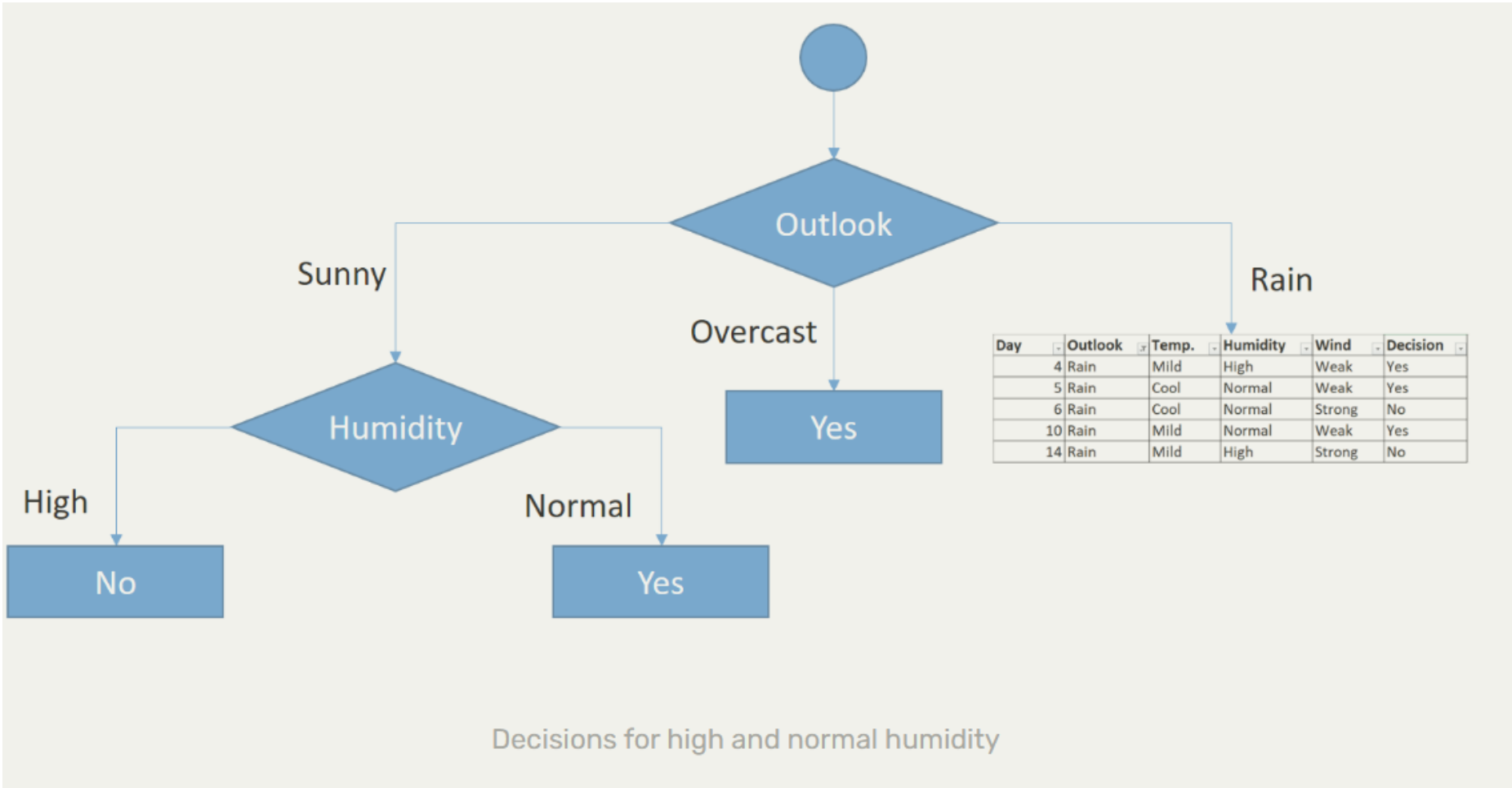
We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decisions for high and normal humidity

Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Gini of temprature for rain outlook

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

Gini(Outlook=Rain and Temp.=Cool) = $1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Temp.=Mild) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Temp.) = $(2/5)\times0.5 + (3/5)\times0.444 = 0.466$

Gini of humidity for rain outlook

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

Gini(Outlook=Rain and Humidity=High) = $1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Humidity=Normal) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Humidity) = $(2/5)\text{x}0.5 + (3/5)\text{x}0.444 = 0.466$

Gini of wind for rain outlook

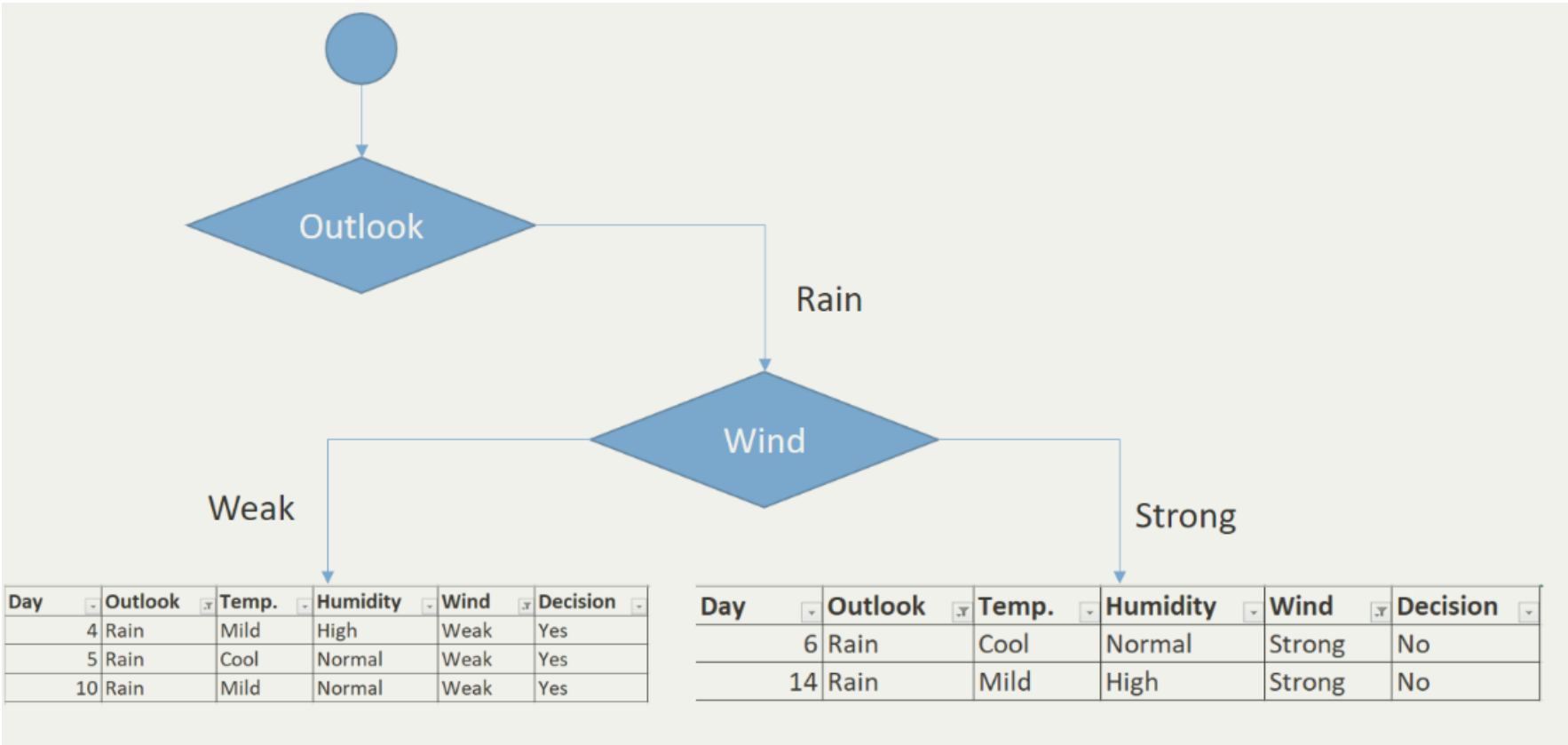
Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

Gini(Outlook=Rain and Wind=Weak) = $1 - (3/3)^2 - (0/3)^2 = 0$

Gini(Outlook=Rain and Wind=Strong) = $1 - (0/2)^2 - (2/2)^2 = 0$

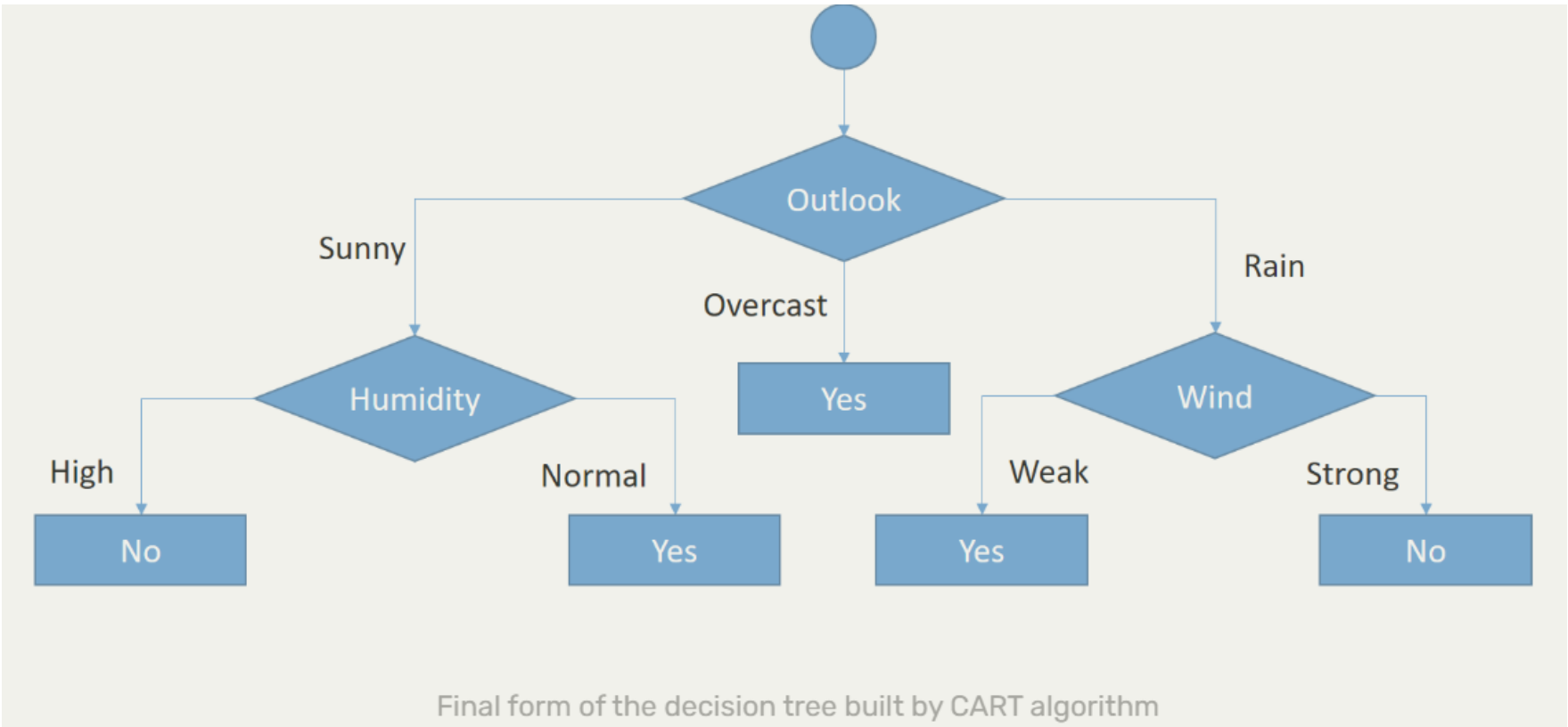
Gini(Outlook=Rain and Wind) = $(3/5)x0 + (2/5)x0 = 0$

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No



Final form of the decision tree built by CART algorithm

Entropy

- **Entropy** is a measure of disorder or *impurity in the given dataset.*
- In the decision tree, messy data are split based on values of the feature vector associated with each data point.
- With each *split, the data becomes more homogenous* which will decrease the entropy.
- The higher the entropy, the harder it is to draw any conclusion. When the tree finally reaches the terminal or leaf node maximum purity is added.

- For a dataset that has C classes and the probability of randomly choosing data from class, i is Pi. Then entropy E(S) can be mathematically represented as

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

  - If we have a dataset of 10 observations belonging to two classes YES and NO. If 6 observations belong to the class, YES, and 4 observations belong to class NO, then entropy can be written as below.

$$E(S) = -(P_{yes} \log_2 P_{yes} + P_{no} \log_2 P_{no})$$

- *Pyes* is the probability of choosing Yes and *Pno* is the probability of choosing a No. Here *Pyes* is 6/10 and *Pno* is 4/10.

$$E(S) = -\left(6/10 * \log_2 * 6/10 + 4/10 * \log_2 * 4/10\right) \approx 0.971$$

- If all the 10 observations belong to 1 class then entropy will be equal to zero. Which implies the node is a pure node.

$$E(S) = - (1 \log_2 1) = 0$$

- If both classes YES and NO have an equal number of observations, then entropy will be equal to 1.

$$E = - (\frac{5}{10} * \log_2 \frac{5}{10} + \frac{5}{10} * \log_2 \frac{5}{10}) = -2(0.5 \log_2 0.5) = 1$$

Calculate Entropy

Age	Mileage	Road Tested	Buy
Recent	Low	Yes	Buy
Recent	High	Yes	Buy
Old	Low	No	Don’t buy
Recent	High	No	Don’t buy

Calculating Entropy for the root node

$$E = -\left(P(\checkmark) * \log_2(P(\checkmark)) + P(\times) * \log_2(P(\times))\right)$$

Probability formula:

$$P(\checkmark) = \frac{\text{count of } \checkmark}{\text{total examples}}$$

$$P(\checkmark) = 2/4 = 0.5$$

$$P(\times) = 2/4 = 0.5$$

Plugging these values in the formula we get:

$$E = -\left(0.5 * \log_2(0.5) + 0.5 * \log_2(0.5)\right)$$

$$E = 1$$

Information Gain

- The Information Gain measures the expected reduction in entropy.
- Entropy measures impurity in the data and information gain measures ***reduction in impurity in the data***.
- The feature which has minimum impurity will be considered as the root node.
- Information gain is used to decide which feature to split on at each step in building the tree.

- Information gain of a parent node can be calculated as the entropy of the parent node is the subtracted entropy of the weighted average of the child node.
- As per the above example, the dataset has 10 observations belonging to two classes YES and NO. Where 6 observations belong to the class, YES, and 4 observations belong to class NO.

Observations	COLOR	Outcome
1	Red	Yes
2	Red	No
3	Yellow	Yes
4	Yellow	Yes
5	Red	Yes
6	Yellow	Yes
7	Red	No
8	Red	No
9	Red	Yes
10	Yellow	No

- Red color has 3 Yes outcome and 3 No outcome whereas yellow has 3 Yes outcome and 1 No outcome.
- E(S) is approximately equal to 0.971
-

$$E(S_{Red}) = -\left(\frac{3}{6} * \log_2 \frac{3}{6} + \frac{3}{6} * \log_2 \frac{3}{6}\right) = 1$$

$$E(S_{Yellow}) = -\left(\frac{3}{4} * \log_2 \frac{3}{4} + \frac{1}{4} * \log_2 \frac{1}{4}\right) \approx 0.811$$

$$\text{Weighted average} = \frac{6}{10} * E(S_{Red}) + \frac{4}{10} * E(S_{Yellow})$$

$$= \frac{6}{10} * 1 + \frac{4}{10} * 0.811$$

$$= 0.924$$

$$\text{Information Gain (S, Color)} = E(S) - \text{Weighted Average}$$

$$= 0.971 - 0.924 \approx - 0.047$$

- For a dataset having many features, the information gain of each feature is calculated. The feature having maximum information gain will be the most important feature which will be the root node for the decision tree.

Example

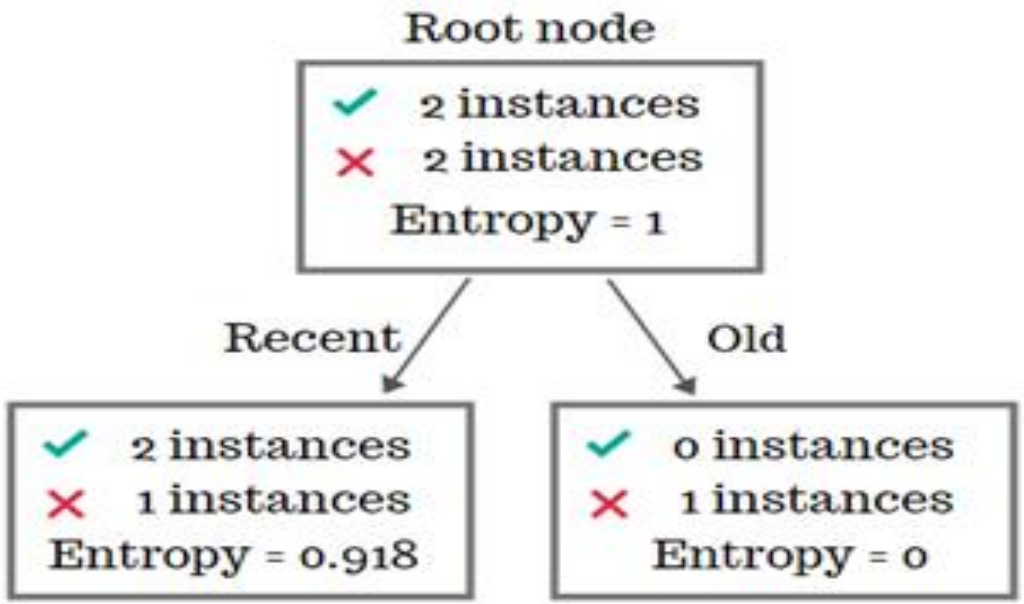
Information Gain (IG)

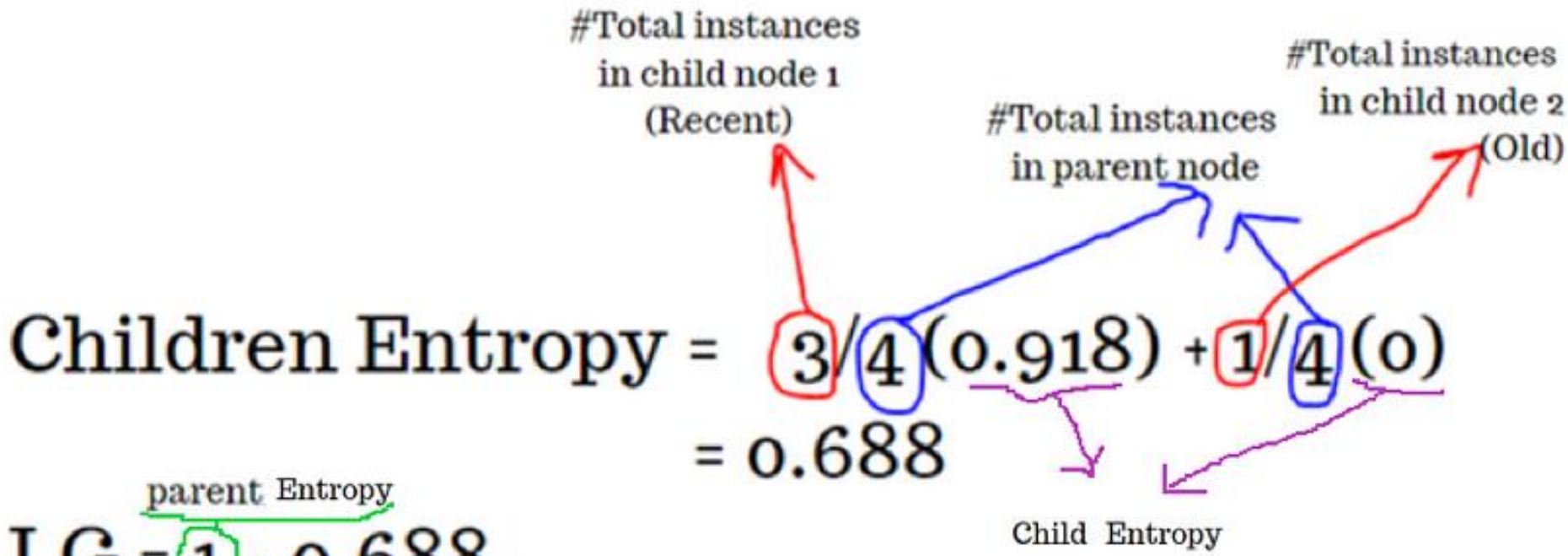
IG = Entropy (Parent) - weighted_avg * Entropy (Children)

Entropy (Parent) = 1

Information gain for Age

Age	Mileage	Road Tested	Buy
Recent	Low	Yes	Buy ✓
Recent	High	Yes	Buy ✓
Old	Low	No	Don't buy ✗
Recent	High	No	Don't buy ✗




#Total instances in child node 1 (Recent)
#Total instances in parent node
#Total instances in child node 2 (Old)
Children Entropy = 3/4 (0.918) + 1/4 (0)
= 0.688
Child Entropy
parent Entropy
I.G = 1 - 0.688
= 0.3112
Information gain if we split at "AGE"

Decision Tree (ID3 Algorithm)

- ID3 or **Iterative Dichotomiser3 Algorithm** is used in machine learning for building decision trees from a given dataset. It was developed in **1986** by **Ross Quinlan**.
- It is a greedy algorithm that builds a decision tree by recursively partitioning the data set into smaller and smaller subsets until all data points in each subset belong to the same class.
- It employs a top-down approach, recursively selecting features to split the dataset based on information gain.
- used for both classification and regression tasks.ID3 deals primarily with categorical properties, which means that it can efficiently handle objects with a discrete set of values. This property is consistent with its suitability for problems where the input features are categorical rather than continuous

Steps of the ID3 Algorithm

1. **Determine entropy** for the overall the dataset using class distribution.
2. For each feature.
    2. Calculate **Entropy for Categorical Values**.
    3. Assess **information gain** for each unique categorical value of the feature.
3. Choose the feature that generates **highest information gain**.
4. Iteratively apply all above steps to build the decision tree structure.

```
def ID3(D, A):
  if D is pure or A is empty:
    return a leaf node with the majority class in D
  else:
    A_best = argmax(InformationGain(D, A))
    root = Node(A_best)
    for v in values(A_best):
      D_v = subset(D, A_best, v)
      child = ID3(D_v, A - {A_best})
      root.add_child(v, child)
    return root
```

Applications of ID3

1. **Fraud detection:** ID3 can be used to develop models that can detect fraudulent transactions or activities.
2. **Medical diagnosis:** ID3 can be used to develop models that can diagnose diseases or medical conditions.
3. **Customer segmentation:** ID3 can be used to segment customers into different groups based on their demographics, purchase history, or other factors.
4. **Risk assessment:** ID3 can be used to assess risk in a variety of different areas, such as insurance, finance, and healthcare.
5. **Recommendation systems:** ID3 can be used to develop recommendation systems that can recommend products, services, or content to users based on their past behavior or preferences.

Example:

- Build a **decision tree** using ID3 algorithm for the given training data in the table (Buy Computer data), and predict the class of the following new example: **age<=30, income=medium, student=yes, credit-rating=fair**

age	income	student	Credit rating	Buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

ID3

- First, check which attribute provides the highest Information Gain in order to split the training set based on that attribute. We need to calculate the expected information to classify the set and the entropy of each attribute.
- The mutual information of the two classes,
- Entropy(S)= E(9,5)= -9/14 $\log_2(9/14)$ – 5/14 $\log_2(5/14)$=0.94
- **Now Consider the Age attribute**
- For Age, we have three values $age_{<=30}$ (2 yes and 3 no), $age_{31..40}$ (4 yes and 0 no), and $age_{>40}$ (3 yes and 2 no)
- Entropy(age) = 5/14 (-2/5 $\log_2(2/5)$-3/5$\log_2(3/5)$) + 4/14 (0) + 5/14 (-3/5$\log_2(3/5)$-2/5$\log_2(2/5)$)
- = 5/14(0.9709) + 0 + 5/14(0.9709) = 0.6935
- Gain(age) = 0.94 – 0.6935 = 0.2465

ID3

- **Next, consider Income Attribute**
- For Income, we have three values $income_{high}$ (2 yes and 2 no), $income_{medium}$ (4 yes and 2 no), and $income_{low}$ (3 yes 1 no)
- Entropy(income) = $4/14(-2/4\log_2(2/4)-2/4\log_2(2/4)) + 6/14 (-4/6\log_2(4/6)-2/6\log_2(2/6)) + 4/14 (-3/4\log2(3/4)-1/4\log2(1/4))$
- = 4/14 (1) + 6/14 (0.918) + 4/14 (0.811)
- = 0.285714 + 0.393428 + 0.231714 = 0.9108
- Gain(income) = 0.94 – 0.9108 = 0.0292

- **Next, consider Student Attribute**
- For Student, we have two values $student_{yes}$ (6 yes and 1 no) and $student_{no}$ (3 yes 4 no)

$Entropy(student) = 7/14(-6/7\log_2(6/7)-1/7\log_2(1/7)) + 7/14(-3/7\log_2(3/7)-4/7\log_2(4/7)$
$= 7/14(0.5916) + 7/14(0.9852)$
$= 0.2958 + 0.4926 = 0.7884$
$Gain\ (student) = 0.94 – 0.7884 = 0.1516$

Finally, consider Credit_Rating Attribute

For Credit_Rating we have two values credit_ratingfair (6 yes and 2 no) and credit_ratingexcellent (3 yes 3 no)

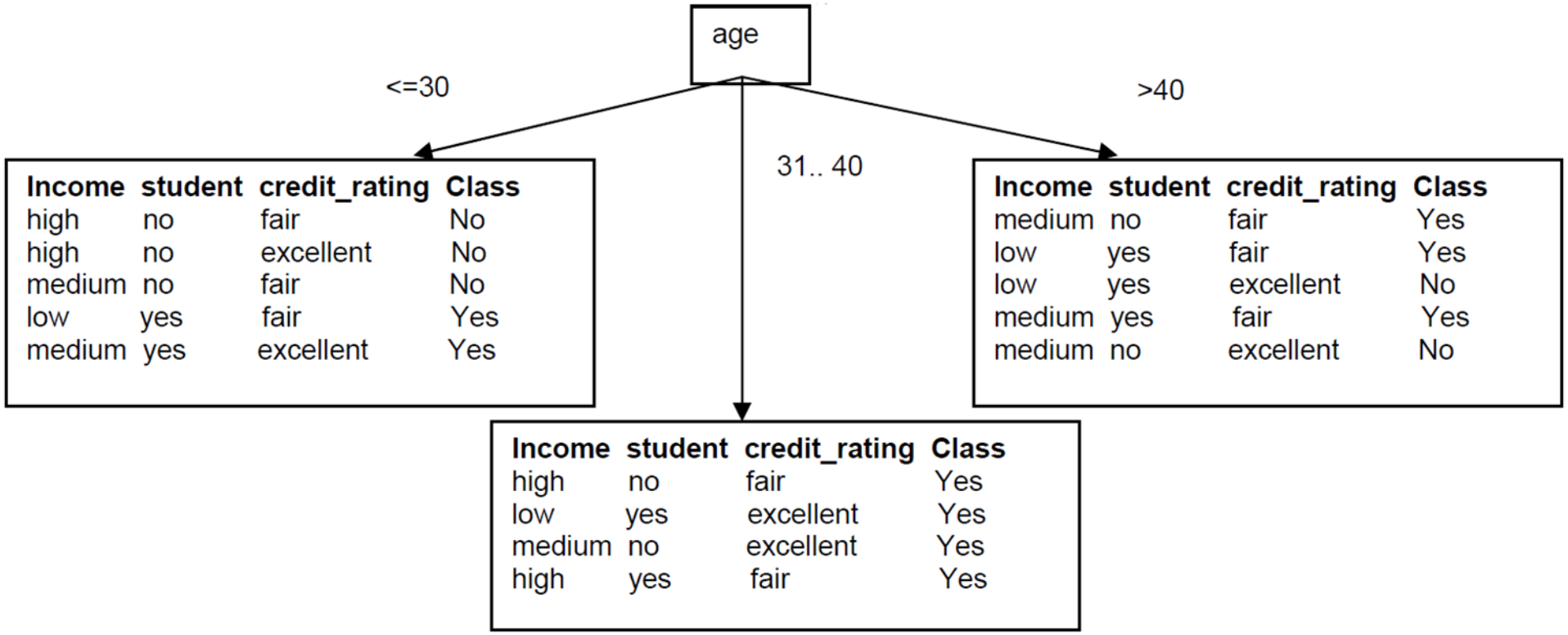
Entropy(credit_rating) = $8/14(-6/8\log_2(6/8)-2/8\log_2(2/8)) + 6/14(-3/6\log_2(3/6)-3/6\log_2(3/6))$

= $8/14(0.8112) + 6/14(1)$

= $0.4635 + 0.4285 = 0.8920$

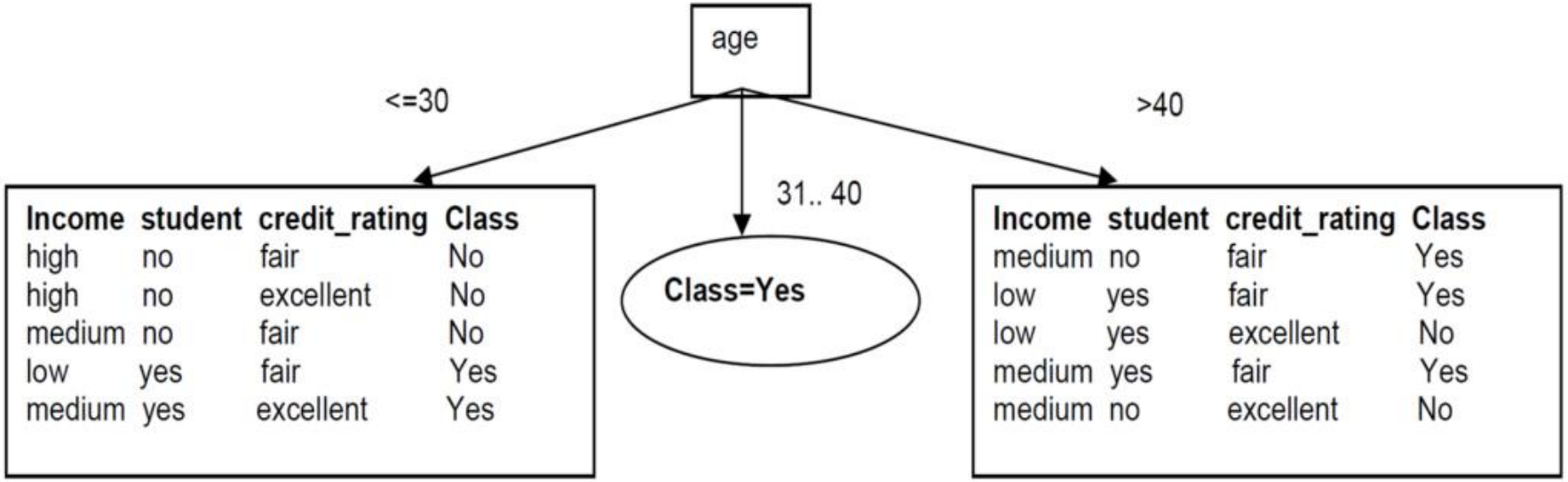
- Gain(credit_rating) = 0.94 – 0.8920 = 0.479
- **Since Age has the highest Information Gain we start splitting the dataset using the age attribute.**

Decision Tree after step 1



Decision Tree after step 1_1

- Since all records under the branch age31..40 are all of the class, Yes, we can replace the leaf with Class=Yes



Income	student	credit_rating	Class
high	no	fair	No
high	no	excellent	No
medium	no	fair	No
low	yes	fair	Yes
medium	yes	excellent	Yes

Income	student	credit_rating	Class
medium	no	fair	Yes
low	yes	fair	Yes
low	yes	excellent	No
medium	yes	fair	Yes
medium	no	excellent	No

- **Now build the decision tree for the left subtree**
- **The same process of splitting has to happen for the two remaining branches.**

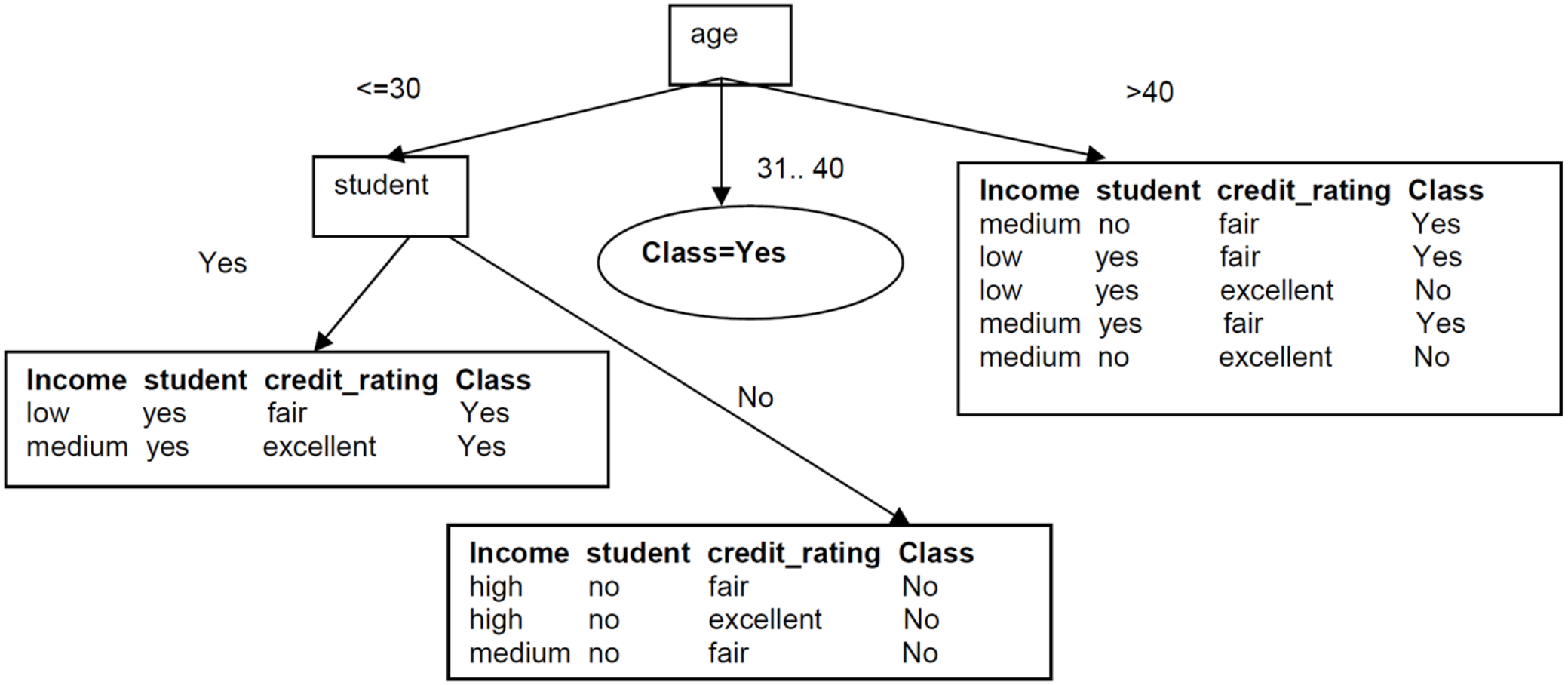
Income	student	credit_rating	Class
high	no	fair	No
high	no	excellent	No
medium	no	fair	No
low	yes	fair	Yes
medium	yes	excellent	Yes

ID3

- For branch age<=30 we still have attributes **income, student, and credit_rating. Which one should be used to split the partition**?
- The mutual information is $E(S_{age<=30})= E(2,3)= -2/5 \log_2(2/5) – 3/5 \log_2(3/5)=$**0.97**
- For **Income,** we have three values $income_{high}$ (0 yes and 2 no), $income_{medium}$ (1 yes and 1 no) and $income_{low}$ (1 yes and 0 no)
- Entropy(income) = $2/5(0) + 2/5 (-1/2\log_2(1/2)-1/2\log_2(1/2)) + 1/5 (0) = 2/5 (1) = 0.4$
- Gain(income) = 0.97 – 0.4 = 0.57

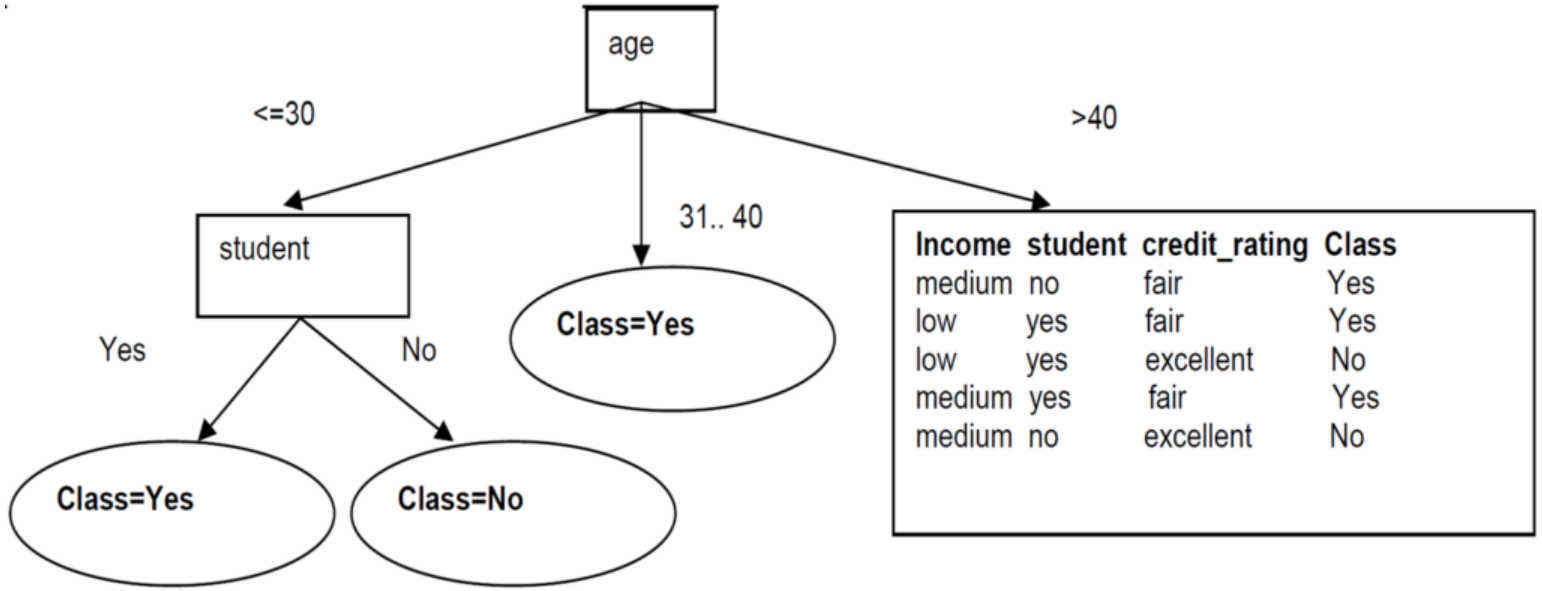
- For Student, we have two values student yes (2 yes and 0 no) and studentno (0 yes 3 no)
- Entropy(student) = 2/5(0) + 3/5(0) = 0
- Gain (student) = 0.97 – 0 = 0.97
- We can then safely split on attribute student without checking the other attributes since the information gain is maximized.

Decision Tree after step 2



Decision Tree after step 2_2

- Since these two new branches are from distinct classes, we make them into leaf nodes with their respective class as label:



Right sub-branch

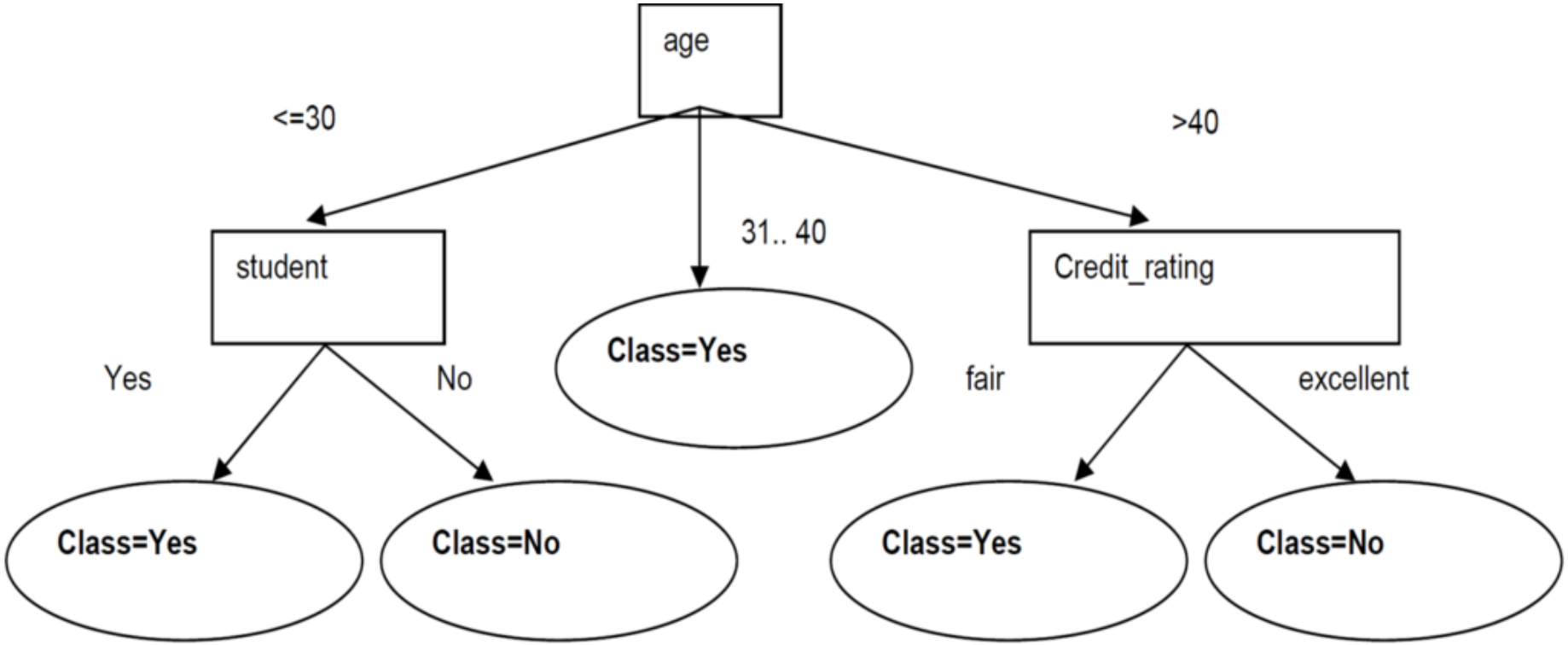
- **Now build the decision tree for right left subtree**

Income	student	credit_rating	Class
medium	no	fair	Yes
low	yes	fair	Yes
low	yes	excellent	No
medium	yes	fair	Yes
medium	no	excellent	No

ID3

- The mutual information is Entropy($S_{age>40}$)= I(3,2)= -3/5 $\log_2$(3/5) – 2/5 $\log_2$(2/5)=0.97
- For **Income**, we have two values $income_{medium}$ (2 yes and 1 no) and $income_{low}$ (1 yes and 1 no)
- Entropy(income) = 3/5(-2/3log2(2/3)-1/3log2(1/3)) + 2/5 (-1/2log2(1/2)-1/2log2(1/2))
- = 3/5(0.9182)+2/5 (1) = 0.55+0. 4= 0.95
- Gain(income) = 0.97 – 0.95 = 0.02
- For **Student**, we have two values $student_{yes}$ (2 yes and 1 no) and $student_{no}$ (1 yes and 1 no)
- Entropy(student) = 3/5(-2/3$\log_2$(2/3)-1/3$\log_2$(1/3)) + 2/5(-1/2$\log_2$(1/2)-1/2$\log_2$(1/2)) = 0.95
- Gain (student) = 0.97 – 0.95 = 0.02
- For **Credit_Rating**, we have two values credit_ratingfair (3 yes and 0 no) and credit_ratingexcellent (0 yes and 2 no)
- Entropy(credit_rating) = 0
- Gain(credit_rating) = 0.97 – 0 = 0.97

- We then split based on credit_rating. These splits give partitions each with records from the same class. We just need to make these into leaf nodes with their class label attached:



- New example: age<=30, income=medium, student=yes, credit-rating=fair
- Follow branch(age<=30) then student=yes we predict Class=yes
- **Buys_computer = yes**

Tree Pruning

- Decision tree pruning is a technique used to prevent decision trees from overfitting the training data.
- Pruning aims to simplify the decision tree by removing parts of it that do not provide significant predictive power, thus improving its ability to generalize to new data.
- Decision Tree Pruning removes unwanted nodes from the overfitted decision tree to make it smaller in size which results in more fast, more accurate and more effective predictions.

How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree.
- Some common pre-pruning techniques include:
- **Maximum Depth**: It limits the maximum level of depth in a decision tree.
- **Minimum Samples per Leaf**: Set a minimum threshold for the number of samples in each leaf node.
- **Minimum Samples per Split**: Specify the minimal number of samples needed to break up a node.
- **Maximum Features:** Restrict the quantity of features considered for splitting.

Pre Pruning Example

```
from sklearn.tree import DecisionTreeClassifier
parameter = {
    'criterion' :['entropy','gini','log_loss'],
    'splitter':['best','random'],
    'max_depth':[1,2,3,4,5],
    'max_features':['auto','sqrt','log2']
}
model = DecisionTreeClassifier()
from sklearn.model_selection import GridSearchCV
cv = GridSearchCV(model,param_grid = parameter,cv = 5)
cv.fit(X_train,Y_train)
```

- In pruning technique hyperparameter tuning through cross-validation using GridSearchCV.
- Hyperparameter tuning involves searching for the optimal hyperparameters for a machine learning model to improve its performance.
- It does not directly prune the decision tree, but it helps in finding the best combination of hyperparameters, such as max_depth, max_features, criterion, and splitter, which indirectly controls the complexity of the decision tree and prevents overfitting.

How to Address Overfitting…

- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Eg: Cost complexity pruning-helps to improve test accuracy and get a better model

- Cost complexity pruning is all about finding the right parameter for **alpha**.We will get the alpha values for this tree and will check the accuracy with the pruned trees.
- This parameter controls the trade-off between the tree's size (complexity) and its performance. A larger α will lead to a simpler tree, while a smaller α may retain more detail but risk overfitting.

Post Pruning Example

Load the Iris dataset

data = load_iris()

X = data.data

y = data.target

Split the dataset into training and validation sets

- X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

Fit a decision tree classifier

dt = DecisionTreeClassifier(random_state=42)

dt.fit(X_train, y_train)

 # Perform cost complexity pruning

path = dt.cost_complexity_pruning_path(X_train, y_train)

ccp_alphas = path.ccp_alphas

impurities = path.impurities

Fit trees for each alpha value

for ccp_alpha in ccp_alphas:

    tree = DecisionTreeClassifier(random_state=42, ccp_alpha=ccp_alpha)

    tree.fit(X_train, y_train)

Advantages

Prevents Overfitting-simplify the tree structure, remove branches that capture noise or outliers in the training data.

Improves Generalization-A pruned decision tree is more likely to capture underlying patterns in the data rather than memorizing specific instances

Reduces Model Complexity-Pruning results in a simpler decision tree with fewer branches and nodes

Enhances Interpretability-Pruning produces decision trees with fewer branches and nodes, which are easier to interpret and understand.

Speeds Up Training and Inference-Pruned decision trees require less computational resources during both training and inference phases.

Facilitates Model Maintenance-s new data becomes available or the problem domain evolves, pruned decision trees are easier to update and adapt compared to overly complex, unpruned trees.