

# **REGRESSION**

- **Numerical prediction** is similar to **classification**
  - construct a model
  - use model to predict continuous or ordered value for a given input
- **Numeric prediction vs. classification**
  - Classification refers to predict categorical class label
  - Numeric prediction models continuous-valued functions

- **Regression analysis** is the major method for numeric prediction
- **Regression analysis** model the relationship between
  - one or more **independent** or **predictor variables** and
  - a **dependent** or **response** variable
- **Regression analysis** is a good choice when all of the **predictor variables** are **continuous** valued as well.

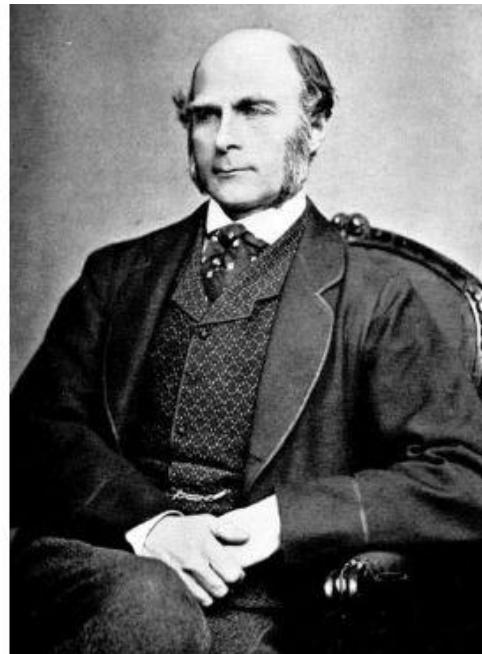
- **Regression analysis methods:**

- Linear regression
  - ◆ Straight-line linear regression
  - ◆ Multiple linear regression
- Non-linear regression
- Generalized linear model
  - ◆ Poisson regression
  - ◆ Logistic regression
- Log-linear models
- Regression trees and Model trees

# LINEAR REGRESSION

# History

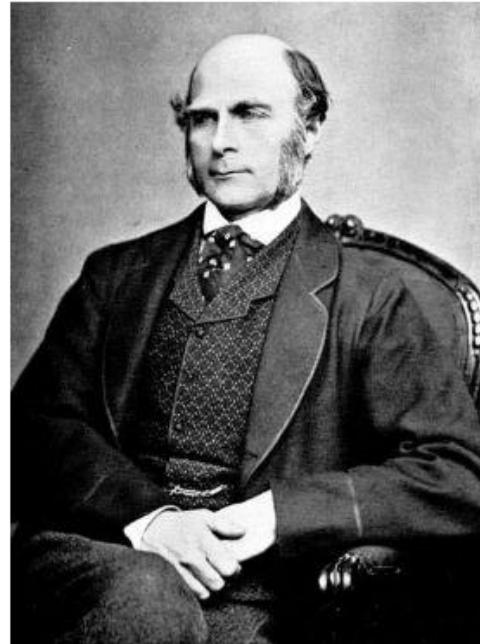
This all started in the 1800s with a guy named [Francis Galton](#). Galton was studying the relationship between parents and their children. In particular, he investigated the relationship between the heights of fathers and their sons.



# History

What he discovered was that a man's son tended to be roughly as tall as his father.

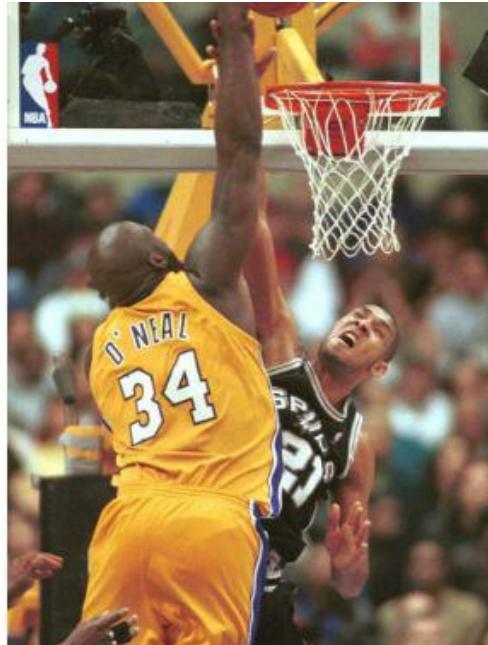
However Galton's breakthrough was that the son's height **tended to be closer to the overall average** height of all people.



# Example

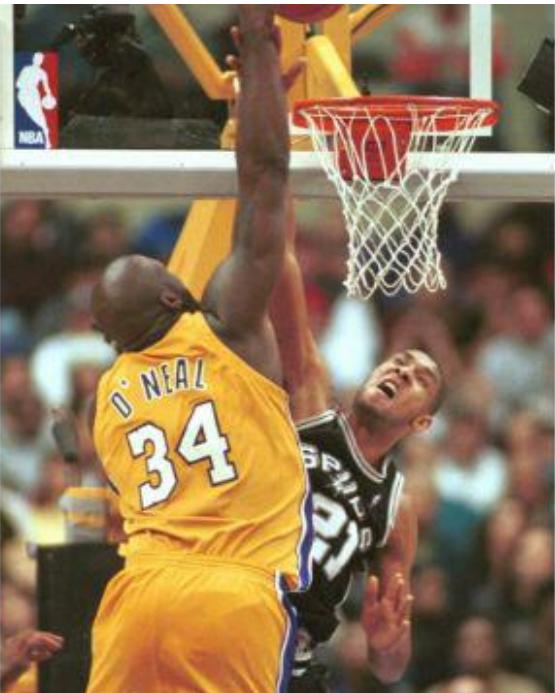
Let's take Shaquille O'Neal as an example. Shaq is really tall: 7ft 1in (2.2 meters).

If Shaq has a son, chances are he'll be pretty tall too. However, Shaq is such an anomaly that there is also a very good chance that his son will be **not be as tall as Shaq.**

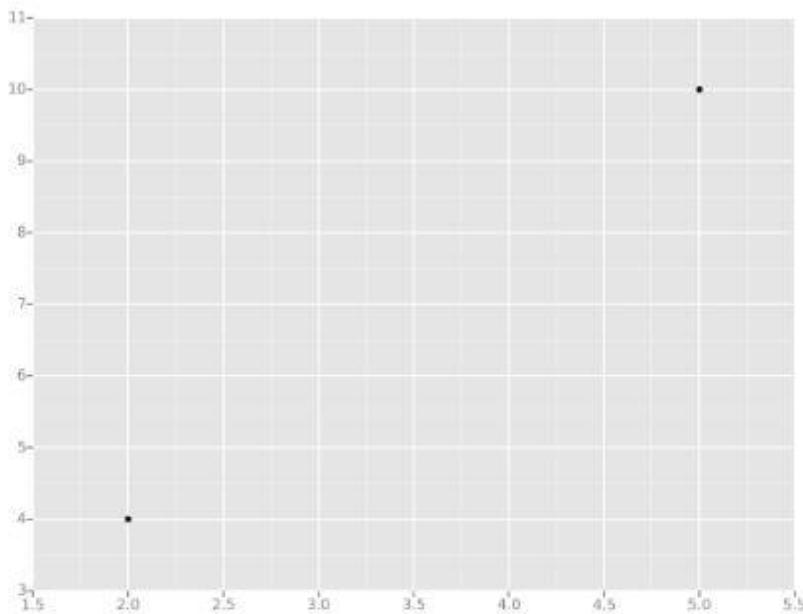


Turns out this is the case:  
Shaq's son is pretty tall (6 ft 7 in), but not nearly as tall as his dad.

Galton called this phenomenon **regression**, as in "A father's son's height tends to regress (or drift towards) the mean (average) height."

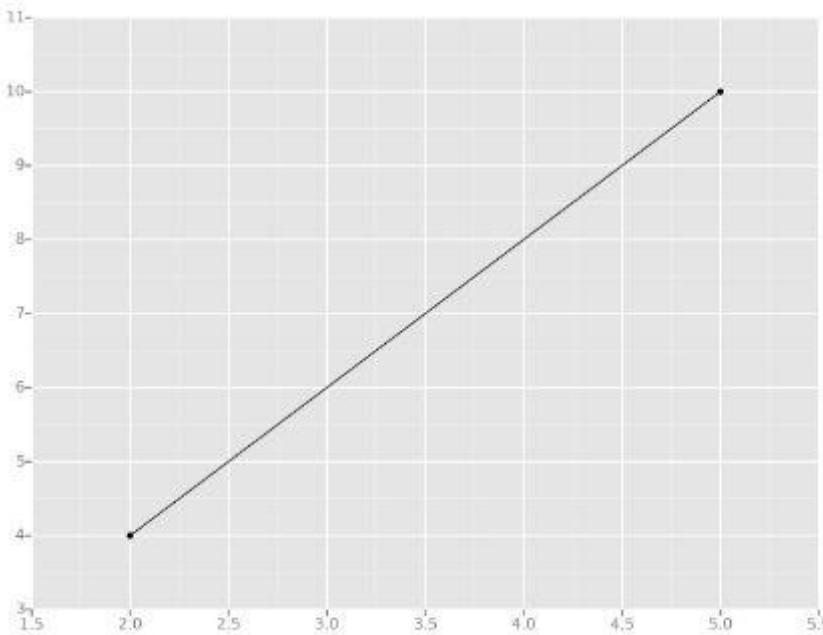


Let's take the simplest possible example:  
calculating a regression with only 2 data points.



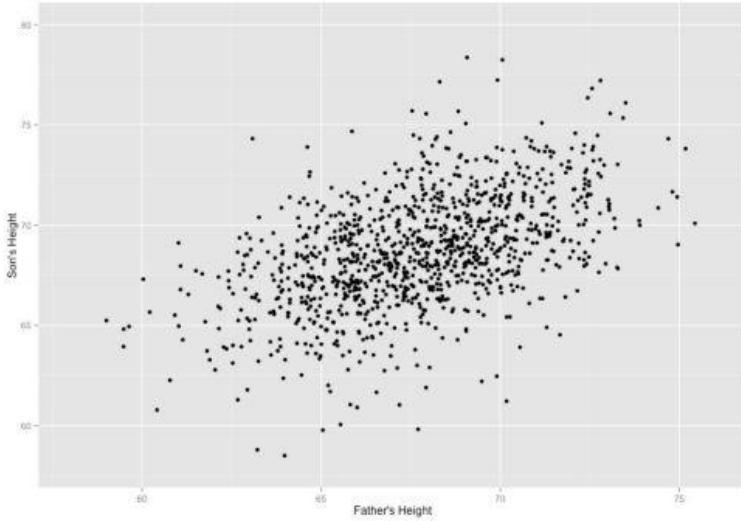
All we're trying to do when we calculate our regression line is draw a line that's as close to every dot as possible.

For classic linear regression, or "Least Squares Method", you only measure the closeness in the "up and down" direction



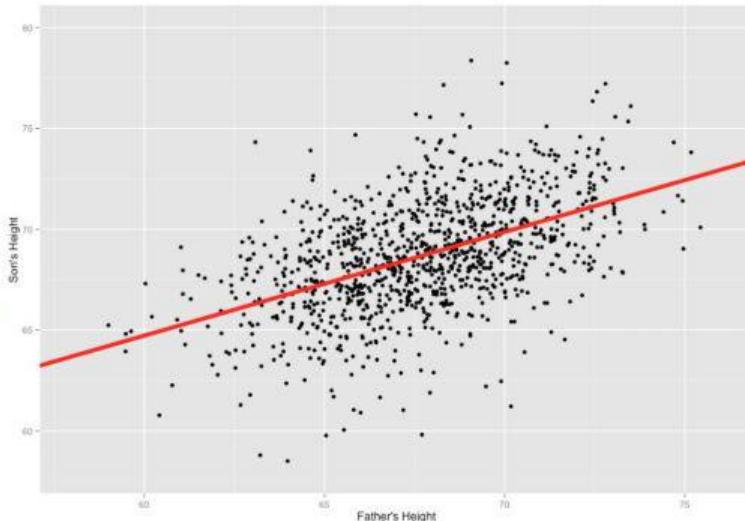
Now wouldn't it be great if we could apply this same concept to a graph with more than just two data points?

By doing this, we could take multiple men and their son's heights and do things like tell a man how tall we expect his son to be...before he even has a son!

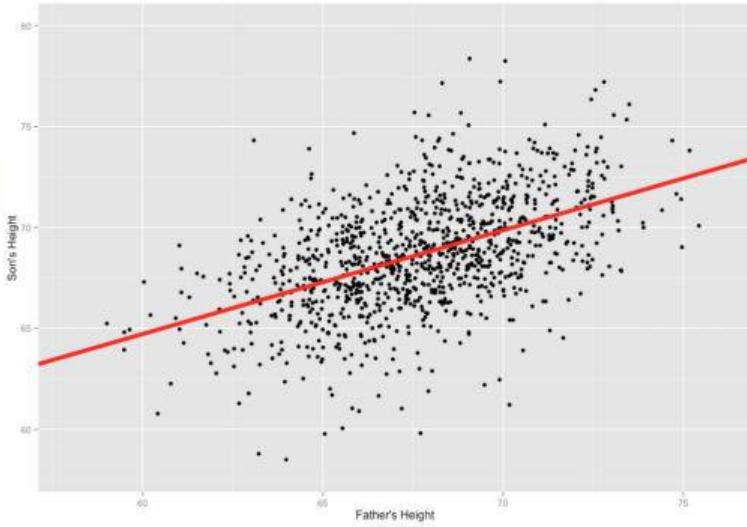


Our goal with linear regression is to **minimize the vertical distance** between all the data points and our line.

So in determining the **best line**, we are attempting to minimize the distance between **all** the points and their distance to our line.

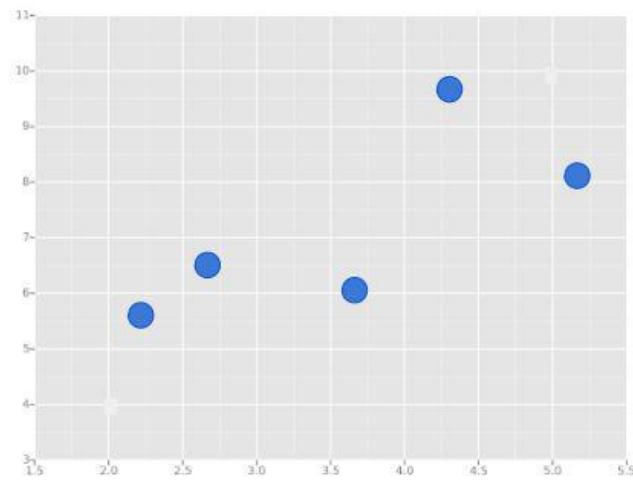


There are lots of different ways to minimize this, (sum of squared errors, sum of absolute errors, etc), but all these methods have a general goal of minimizing this distance.



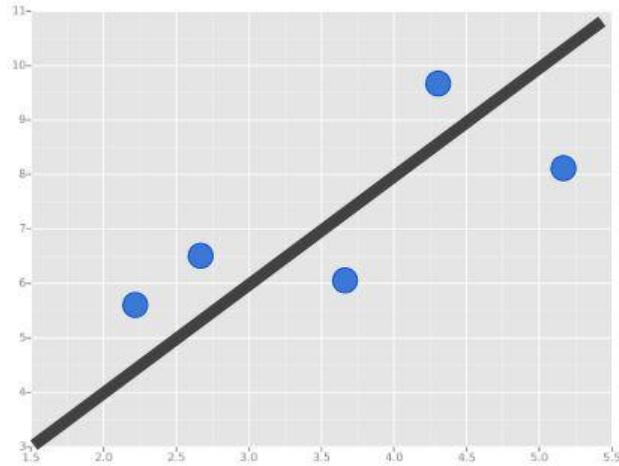
For example, one of the most popular methods is the least squares method.

Here we have blue data points along an x and y axis.



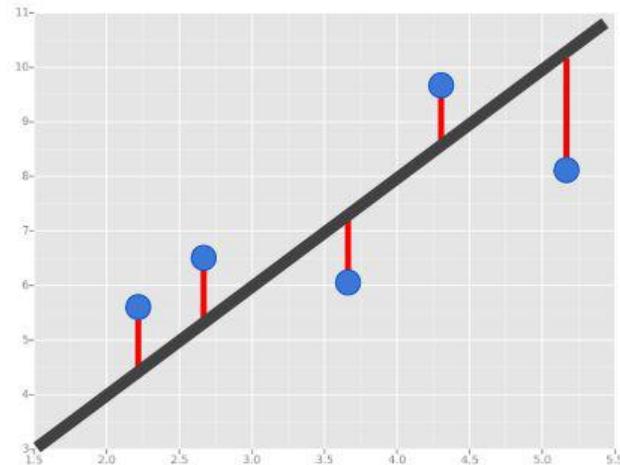
Now we want to fit a linear regression line.

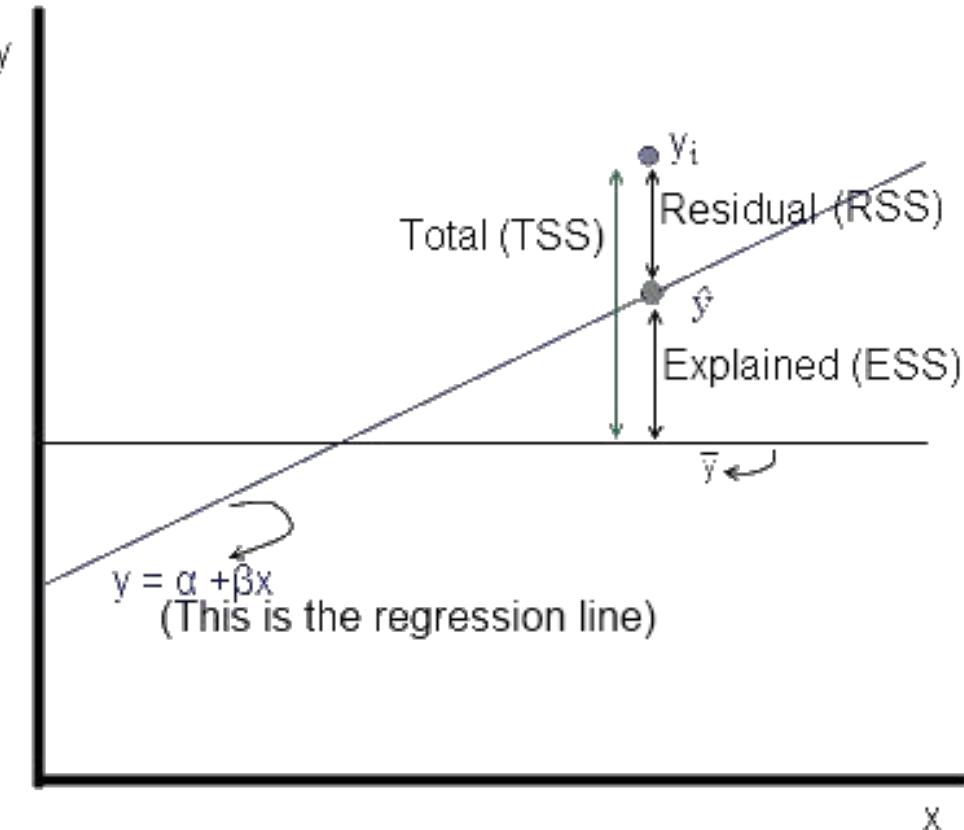
The question is, how do we decide which line is the best fitting one?



We'll use the Least Squares Method, which is fitted by minimizing the ***sum of squares of the residuals***.

The residuals for an observation is the difference between the observation (the y-value) and the fitted line.





$\hat{y}$  is the predicted value of  $y$  given  $x$ , using the equation  $\hat{y} = \alpha + \beta x$ .

$y_i$  is the actual observed value of  $y$ .

$\bar{y}$  is the mean of  $y$ .

The distances that RSS, ESS and TSS represent are shown in the diagram to the left - but remember that the actual calculations are squares of these distances.

$$TSS = \sum (y_i - \bar{y})^2$$

$$RSS = \sum (y_i - \hat{y})^2$$

$$ESS = \sum (\hat{y} - \bar{y})^2$$

# **SS**

For linear regression with one explanatory variable like this analysis, R-squared is the same as the square of r, the correlation coefficient.

The sum of squares (SS) represents variation from several sources.

**SS(regression)** describes the variation within the fitted values of Y, and is the sum of the squared difference between each fitted value of Y and the mean of Y. The squares are taken to 'remove' the sign (+ or -) from the residual values to make the calculation easier.

**SS(error)** describes the variation of observed Y from estimated (fitted) Y. It is derived from the cumulative addition of the square of each residual, where a residual is the distance of a data point above or below the fitted

**SS(total)** describes the variation within the values of Y, and is the sum of the squared difference between each value of Y and the mean of Y.

## Total Sum of Squares

In linear regression, TSS stands for Total Sum of Squares, which is a measure of the total variation in the response variable that is explained by the model. Mathematically, TSS is calculated as the sum of the squared differences between the observed response values and the mean of the response variable:

$$TSS = \Sigma(y_i - \bar{y})^2$$

where  $y_i$  is the observed response value for the  $i$ -th observation,  $\bar{y}$  is the mean of the response variable, and  $\Sigma$  represents the summation over all observations.

## **R<sup>2</sup>**

TSS is used in conjunction with other measures such as **RSS (Residual Sum of Squares)** and **ESS (Explained Sum of Squares)** to calculate the **coefficient of determination (R<sup>2</sup>)**, which is a measure of how well the linear regression model fits the data. Specifically, **R<sup>2</sup> is the proportion of the total variation in the response variable that is explained by the linear regression model** and is calculated as:

$$\mathbf{R^2 = ESS / TSS}$$

where ESS is the Explained Sum of Squares, which is the sum of the squared differences between the predicted response values and the mean of the response variable.

# Optimization Problem

Linear regression becomes an optimization problem when we want to estimate the coefficients (or parameters) of the linear regression model that best fit the data. The goal is to find the values of the coefficients that minimize the difference between the predicted response values and the actual response values.

The RSS function is defined as :

$$\text{RSS} = \sum (y_i - \hat{y}_i)^2$$

where  $y_i$  is the observed response value for the  $i$ -th observation, and  $\hat{y}_i$  is the predicted response value for the  $i$ -th observation based on the linear regression model.

## Parabolic shape

If we express the predicted response values  $\hat{y}_i$  in terms of the coefficients, we can write the RSS function as a quadratic function of the coefficients. Specifically, the RSS function can be expressed as:

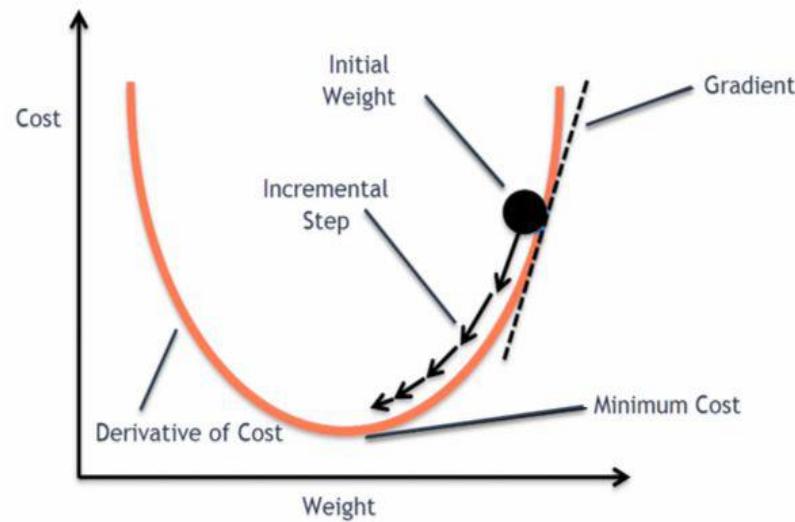
$$\text{RSS} = (Y - X\beta)^T(Y - X\beta)$$

where  $Y$  is the vector of observed response values,  $X$  is the matrix of predictor variables,  $\beta$  is the vector of coefficients, and  $^T$  denotes the transpose of a matrix. The expression  $(Y - X\beta)^T(Y - X\beta)$  expands to a quadratic function of the coefficients  $\beta$ .

# Parabolic shape

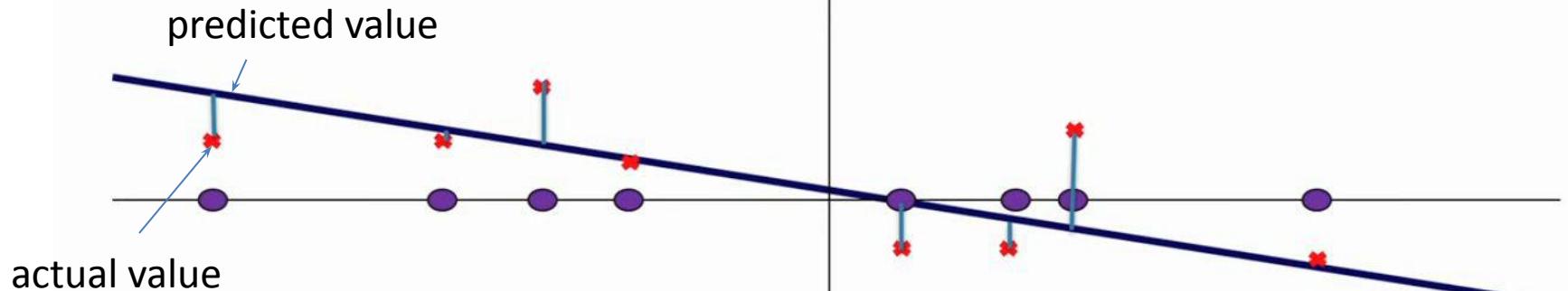
Since the RSS function is a quadratic function of the coefficients, it has a **parabolic shape** in the space of the coefficients.

The minimum of the parabolic function corresponds to the **optimal values of the coefficients** that **minimize the RSS**, which is what we want to find in linear regression optimization.



# Linear Regression

Minimize  
**sum of squared error**



# Linear Regression

- **Straight-line linear regression:**

- involves a response variable  $y$  and a single predictor variable  $x$

$$y = w_0 + w_1 x$$

- $w_0$  :  $y$ -intercept
  - $w_1$  : slope
  - $w_0$  &  $w_1$  are **regression coefficients**

# Least Square Error

- **Method of least squares**: estimates the best-fitting straight line as the one that minimizes the error between the actual data and the estimate of the line.

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

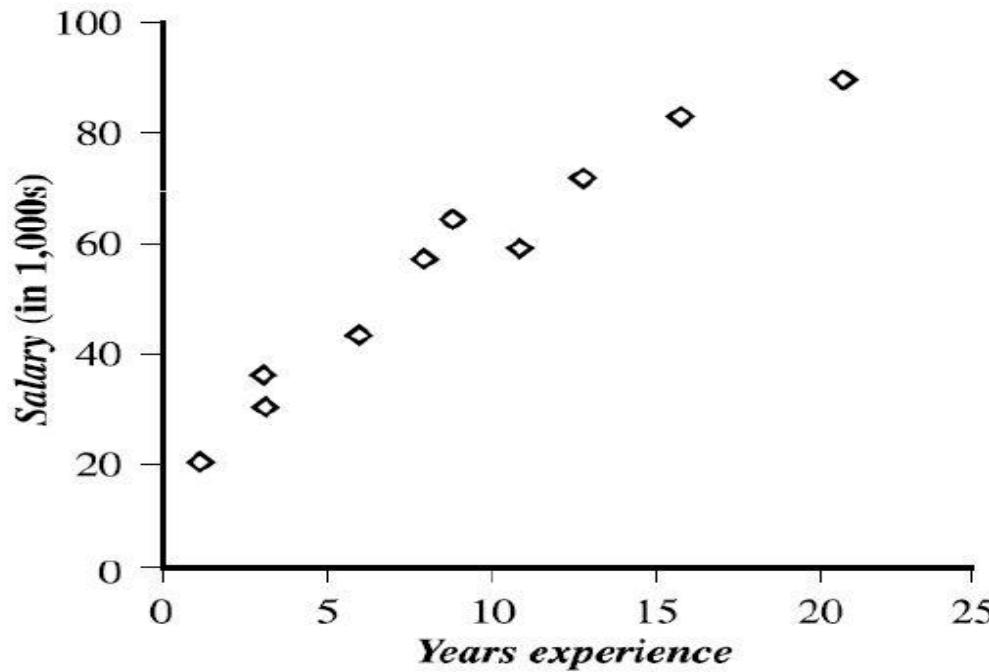
- $D$ : a training set
- $x$ : values of predictor variable
- $y$ : values of response variable
- $|D|$  : data points of the form  $(x_1, y_1), (x_2, y_2), \dots, (x/|D|, y/|D|)$ .
- $\bar{x}$  : the mean value of  $x_1, x_2, \dots, x/|D|$
- $\bar{y}$  : the mean value of  $y_1, y_2, \dots, y/|D|$

The table shows a set of paired data where  $x$  is the number of years of work experience of a college graduate and  $y$  is the corresponding salary of the graduate.

$x$ years experience	$y$ salary (in \$1000s)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

The 2-D data can be graphed on a **scatter plot**.

The plot suggests a linear relationship between the two variables,  $x$  and  $y$ .



- Given the above data, we compute

$$\bar{x} = 9.1 \text{ and } \bar{y} = 55.4$$

- we get

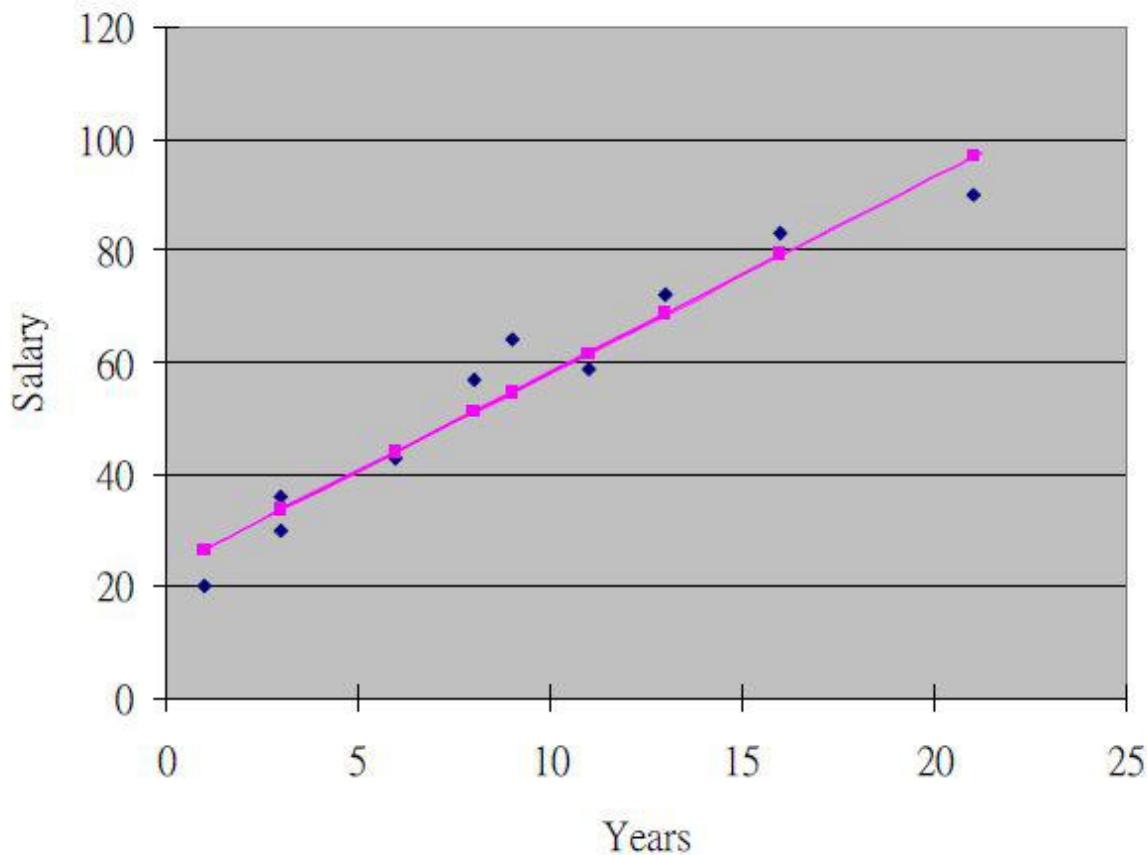
$$w_1 = \frac{(3 - 9.1)(30 - 55.4) + (8 - 9.1)(57 - 55.4) + \dots + (16 - 9.1)(83 - 55.4)}{(3 - 9.1)^2 + (8 - 9.1)^2 + \dots + (16 - 9.1)^2} = 3.5$$

$$w_0 = 55.4 - (3.5)(9.1) = 23.6$$

- The equation of the least squares line is estimated by

$$y = 23.6 + 3.5x$$

Linear Regression:  $Y=3.5*X+23.2$



# Problem Statement

Last year, five randomly selected students took a math aptitude test before they began their statistics course. The Statistics Department has three questions.

- What linear regression equation best predicts statistics performance, based on math aptitude scores?
- If a student made an 80 on the aptitude test, what grade would we<sup>32</sup> expect her to make in statistics?
- How well does the regression equation fit the data?

In the table below, the  $x_i$  column shows scores on the aptitude test. Similarly, the  $y_i$  column shows statistics grades. The last two rows show sums and mean scores that we will use to conduct the regression analysis.

Student	$x_i$	$y_i$	$(x_i - \bar{x})$	$(y_i - \bar{y})$	$(x_i - \bar{x})^2$	$(y_i - \bar{y})^2$	$(x_i - \bar{x})(y_i - \bar{y})$
1	95	85	17	8	289	64	136
2	85	95	7	18	49	324	126
3	80	70	2	-7	4	49	-14
4	70	65	-8	-12	64	144	96
5	60	70	-18	-7	324	49	126
<b>Sum</b>	390	385			730	630	470
<b>Mean</b>	78	77					

The regression equation is a linear equation of the form:  $\hat{y} = b_0 + b_1 x$ . To conduct a regression analysis, we need to solve for  $b_0$  and  $b_1$ . Computations are shown below.

33

$b_1 = \frac{\sum [(x_i - \bar{x})(y_i - \bar{y})]}{\sum (x_i - \bar{x})^2}$	$b_0 = \bar{y} - b_1 * \bar{x}$
$b_1 = 470/730 = 0.644$	$b_0 = 77 - (0.644)(78) = 26.768$

Therefore, the regression equation is:  $\hat{y} = 26.768 + 0.644x$ .

Once you have the regression equation, using it is a snap. Choose a value for the independent variable ( $x$ ), perform the computation, and you have an estimated value ( $\hat{y}$ ) for the dependent variable.

In our example, the independent variable is the student's score on the aptitude test. The dependent variable is the student's statistics grade. If a student made an 80 on the aptitude test, the estimated statistics grade would be:

$$\hat{y} = 26.768 + 0.644x = 26.768 + 0.644 * 80 = 26.768 + 51.52 = 78.288$$

34

Warning: When you use a regression equation, do not use values for the independent variable that are outside the range of values used to create the equation. That is called extrapolation, and it can produce unreasonable estimates.

Whenever you use a regression equation, you should ask how well the equation fits the data. One way to assess fit is to check the coefficient of determination, which can be computed from the following formula.

$$R^2 = \{ ((1 / N) * \sum [(x_i - \bar{x}) * (y_i - \bar{y})]) / (\sigma_x * \sigma_y) \}^2$$

where  $N$  is the number of observations used to fit the model,  $\Sigma$  is the summation symbol,  $x_i$  is the  $x$  value for observation  $i$ ,  $\bar{x}$  is the mean  $x$  value,  $y_i$  is the  $y$  value for observation  $i$ ,  $\bar{y}$  is the mean  $y$  value,  $\sigma_x$  is the standard deviation of  $x$ , and  $\sigma_y$  is the standard deviation of  $y$ . Computations for the sample problem of this lesson are shown below.

$$\sigma_x = \sqrt{[\sum (x_i - \bar{x})^2 / N]}$$

$$\sigma_x = \sqrt{730/5} = \sqrt{146} = 12.083$$

$$\sigma_y = \sqrt{[\sum (y_i - \bar{y})^2 / N]}$$

$$\sigma_y = \sqrt{630/5} = \sqrt{126} = 11.225$$

$$R^2 = ((1/N) * \sum [(x_i - \bar{x}) * (y_i - \bar{y})])^2 / (\sigma_x * \sigma_y)^2$$

$$R^2 = [(1/5) * 470 / (12.083 * 11.225)]^2 = (94 / 135.632)^2 = (0.693)^2 = 0.48$$

A coefficient of determination equal to 0.48 indicates that about 48% of the variation in statistics grades (the dependent variable) can be explained by the relationship to math aptitude scores (the independent variable). This would be considered a good fit to the data, in the sense that it would substantially improve an educator's ability to predict student performance in statistics class.

# Outliers and Influential Points

An outlier is an extreme observation that does not fit the general correlation or regression pattern. In the regression setting, outliers will be far away from the regression line in the y-direction. Since it is an unusual observation, the inclusion of an outlier may affect the slope and the y-intercept of the regression line.

When examining a scatterplot graph and calculating the regression equation, it is worth considering whether extreme observations should be included or not.

Let's use our example above to illustrate the effect of a single outlier. Say that we have a student who has a high GPA but who suffered from test anxiety the morning of the SAT verbal test and scored a 410. Using our original regression equation, we would expect the student to have a GPA of 2.2. But, in reality, the student has a GPA equal to 3.9. The inclusion of this value would change the slope of the regression equation from 0.0055 to 0.0032, which is quite a large difference.

There is no set rule when trying to decide whether or not to include an outlier in regression analysis. For univariate data, we can use the IQR rule to determine whether or not a point is an outlier. We should consider values that are 1.5 times the inter-quartile range below the first quartile or above the third quartile as outliers.

Extreme outliers are values that are 3.0 times the inter-quartile range below the first quartile or above the third quartile.

# PYTHON CODE

# Linear Regression

```
from sklearn.linear_model import LinearRegression  
model = LinearRegression()  
plt.scatter(x,y)
```

```
x = np.array(x).reshape((-1, 1))  
y= np.array(y)
```

```
model.fit(x,y)
```

```
r_sq = model.score(x,y)  
print("Variance for model",r_sq )  
print('intercept model:', model.intercept_)  
print('slope: model', model.coef_)
```

# MULTIPLE REGRESSION

# Multiple Linear Regression

- **Multiple linear regression** involves more than one predictor variable
- Training data is of the form  $(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \dots, (\mathbf{X}_{|\mathcal{D}|}, y_{|\mathcal{D}|})$
- where the  $\mathbf{X}_i$  are the  $n$ -dimensional training data with associated class labels,  $y_i$
- An example of a multiple linear regression model based on two predictor attributes:

$$y = w_0 + w_1x_1 + w_2x_2$$

# CPU Performance Data

Cycle time (ns) MYCT	Main memory (KB)		Cache (KB) CACH	Channels			Performance PRP
	Min. MMIN	Max. MMAX		Min. CHMIN	Max. CHMAX		
1	125	256	6000	256	16	128	198
2	29	8000	32000	32	8	32	269
3	29	8000	32000	32	8	32	220
4	29	8000	32000	32	8	32	172
5	29	8000	16000	32	8	16	132
...							
207	125	2000	8000	0	2	14	52
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

$$\begin{aligned} \text{PRP} = & -55.9 + 0.0489 \text{ MYCT} + 0.0153 \text{ MMIN} + 0.0056 \text{ MMAX} \\ & + 0.6410 \text{ CACH} - 0.2700 \text{ CHMIN} + 1.480 \text{ CHMAX}. \end{aligned}$$

Various statistical measures exist for determining how well the proposed model can predict  $y$  (described later).

Obviously, the greater the number of predictor attributes is, the slower the performance is.

Before applying regression analysis, it is common to perform attribute subset selection to eliminate attributes that are unlikely to be good predictors for  $y$ .

In general, regression analysis is accurate for numeric prediction, except when the data contain outliers.

## Non linear regression

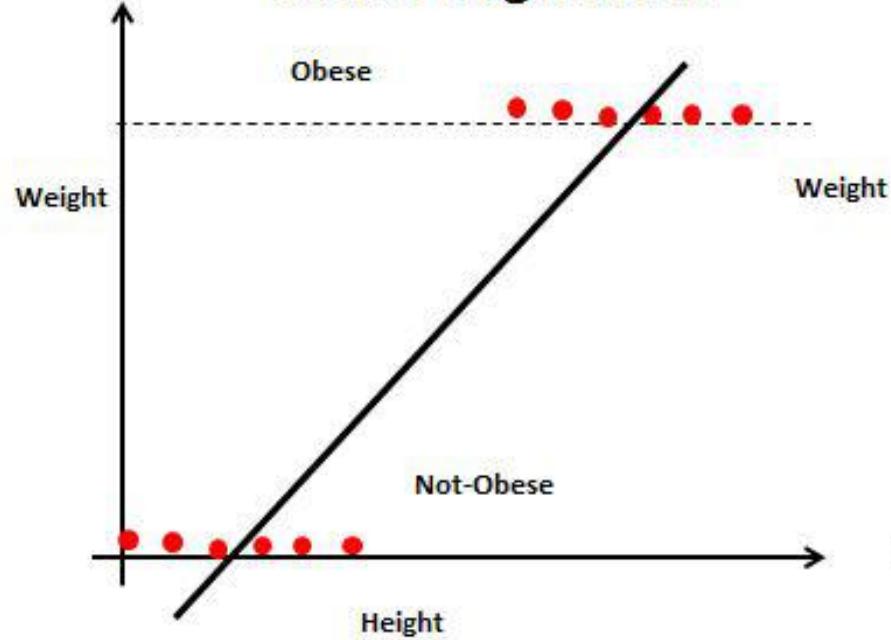
If data that does not show a linear dependence  
we can get a more accurate model using a  
**nonlinear regression** model

For example,

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

# LOGISTIC REGRESSION

## Linear Regression



## Logistic Regression



Logistic regression vs Linear regression

# Logistic Regression

Logistic regression is a statistical method used to analyze the relationship between a categorical dependent variable and one or more independent variables. The dependent variable is typically binary (e.g., success/failure, yes/no), and the independent variables can be categorical or continuous.

In logistic regression, the relationship between the dependent variable and the independent variables is modeled using a logistic function, which is an S-shaped curve that maps any input value to a value between 0 and 1. The output of the logistic function represents the probability of the dependent variable being in a particular category.

Logistic regression is commonly used in various fields, including medicine, marketing, finance, and social sciences, to predict the likelihood of an event occurring based on a set of input variables. For example, it can be used to predict the likelihood of a customer buying a product, the likelihood of a patient developing a particular disease, or the likelihood of an employee quitting their job. It is also commonly used in machine learning applications, such as image and text classification.

## Linear v/s Logistic Regression

- Consider the following example
- If X contains the area in square feet of houses, and Y contains the corresponding sale price of those houses
  - **Linear regression** - to predict selling price as a function of house size. The output will be the price value
  - **Logistic regression** - to predict, whether a house would sell for more than \$200K, The possible outputs are either Yes, the house will sell for more than \$200K, or No, the house will not

Linear Regression	Logistic Regression
In linear regression, the outcome (dependent variable) is continuous. It can have any one of an infinite number of possible values	In logistic regression, the outcome (dependent variable) has only a limited number of possible values. Mostly two

# Logistic Regression

---

- Logistic Regression produces results in a binary format
- It is used for predicting the outcome of a categorical dependent variable based on one or more features
- It is most widely used when the dependent variable is binary i.e., the number of available categories is two
- Some of the usual outputs of logistic regression are,
  - Yes and No
  - True and False
  - High and Low
  - Pass and Fail

## Mathematically:

The mathematics behind logistic regression involves the use of the logistic function, which is also known as the sigmoid function. The logistic function is defined as:

$$f(x) = 1 / (1 + e^{-x})$$

where  $x$  is the input variable.

In logistic regression, the goal is to model the probability of the dependent variable ( $y$ ) taking on a particular value (usually 0 or 1) given the values of the independent variables ( $x$ ). This probability can be expressed mathematically as:

$$P(y=1 | x) = f(\beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_nx_n)$$

where  $\beta_0, \beta_1, \beta_2, \dots, \beta_n$  are the coefficients of the independent variables, and  $n$  is the number of independent variables.

## MLE

The equation above represents the logistic regression model. The coefficients of the independent variables are estimated using a maximum likelihood estimation (MLE) method, which involves finding the values of the coefficients that maximize the likelihood of observing the data given the model.

Once the coefficients are estimated, they can be used to predict the probability of the dependent variable taking on a particular value given the values of the independent variables. If the predicted probability is greater than or equal to 0.5, the dependent variable is predicted to be 1; otherwise, it is predicted to be 0.

## Use Case – Hands On

---

- A car company has released a new suv in the market, using the data they have about the earlier sales of their suv's they want to predict what type of people will be interested in buying this
- The dataset contains

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0

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

dataset2 = pd.read_csv("cars")

X = dataset2.iloc[:, [2,3]].values

Y = dataset2.iloc[:,4].values

from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =0.25,  
random_state =0)
```

```
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.fit_transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
```

```
classifier = LogisticRegression(random_state=0)
```

```
classifier.fit(X_train, Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,  
                    intercept_scaling=1, l1_ratio=None, max_iter=100,  
                    multi_class='auto', n_jobs=None, penalty='l2',  
                    random_state=0, solver='lbfgs', tol=0.0001, verbose=0,  
                    warm_start=False)
```

```
Y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(Y_test, Y_pred)
```

```
print(cm)
```

```
[[63  5]  
 [ 8 24]]
```

Predicted Values

Actual Values

	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

**LET'S SEE IF GENDER PLAYS A  
ROLE**

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

dataset2 = pd.read_csv("cars")

X = dataset2.iloc[:, [1,2,3]].values

Y = dataset2.iloc[:,4].values

Xgender = X[:,0]

from sklearn.preprocessing import LabelEncoder,OneHotEncoder

Xgender= LabelEncoder().fit_transform(Xgender)

onen = OneHotEncoder(categories ='auto')
```

```
onen = OneHotEncoder(categories ='auto')
```

```
xg = pd.DataFrame(Xgender)
```

```
xg
```

**0**

**0** 1

**1** 1

**2** 0

**3** 0

**4** 1

60

```
onen = OneHotEncoder(categories ='auto')

a= onen.fit_transform(xg).toarray()

df = pd.DataFrame(data=a, columns=["Male", "Female"])

df
```

	Male	Female
0	0.0	1.0
1	0.0	1.0
2	1.0	0.0
3	1.0	0.0

```
In [27]: X
```

```
Out[27]: array([['Male', 19, 19000],  
                 ['Male', 35, 20000],  
                 ['Female', 26, 43000],  
                 ...,  
                 ['Female', 50, 20000],  
                 ['Male', 36, 33000],  
                 ['Female', 49, 36000]], dtype=object)
```

```
In [28]: X = X[:, [1,2]]
```

```
In [30]: X = pd.DataFrame(X)
```

```
In [31]: X
```

```
Out[31]:
```

	0	1
0		
1		

400 rows × 2 columns

```
XF = pd.concat([X, df], axis=1)
```

```
XF
```

	0	1	Male	Female
0	19	19000	0.0	1.0
1	35	20000	0.0	1.0
2	26	43000	1.0	0.0
3	27	57000	1.0	0.0
4	19	76000	0.0	1.0

```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, Y_train, Y_test = train_test_split(XF,Y, test_size  
=0.25, random_state =0)
```

```
from sklearn.preprocessing import StandardScaler  
  
sc = StandardScaler()  
  
X_train = sc.fit_transform(X_train)  
  
X_test = sc.fit_transform(X_test)  
  
from sklearn.linear_model import LogisticRegression  
  
classifier = LogisticRegression(random_state=0)  
  
classifier.fit(X_train, Y_train)  
  
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,  
intercept_scaling=1, l1_ratio=None, max_iter=100,  
multi_class='auto', n_jobs=None, penalty='l2',  
random_state=0, solver='lbfgs', tol=0.0001, verbose=0)
```

```
Y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Y_test, Y_pred)

print(cm)

[[64  4]
 [ 5 27]]
```

# Model Performance

## Model performance parameters

Model performance parameters are metrics used to evaluate the performance of a machine learning model.

These metrics help to determine the accuracy and reliability of a model, and provide a way to compare different models and choose the best one for a given task.

We can use a confusion matrix to evaluate our model.  
For example, imagine testing for disease.

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Example: Test for presence of disease  
NO = negative test = False = 0  
YES = positive test = True = 1

# Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

## Basic Terminology:

- True Positives (TP)
- True Negatives (TN)
- False Positives (FP)
- False Negatives (FN)

**Confusion Matrix:** A confusion matrix is a table that summarizes the performance of a binary classifier. It shows the number of true positive, false positive, false negative, and true negative predictions made by the model.

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

## Accuracy:

- Overall, how often is it **correct?**
- $(TP + TN) / \text{total} = 150/165 = 0.91$

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

## Misclassification Rate (Error Rate):

- Overall, how often is it **wrong?**
- $(FP + FN) / \text{total} = 15/165 = 0.09$

		Real Label	
		Positive	Negative
Predicted Label	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)
		$\downarrow$	
		$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FN}}$	

$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$$

**Precision:** Precision is the fraction of true positive predictions out of all positive predictions. It measures how many of the positive predictions made by the model are actually true.

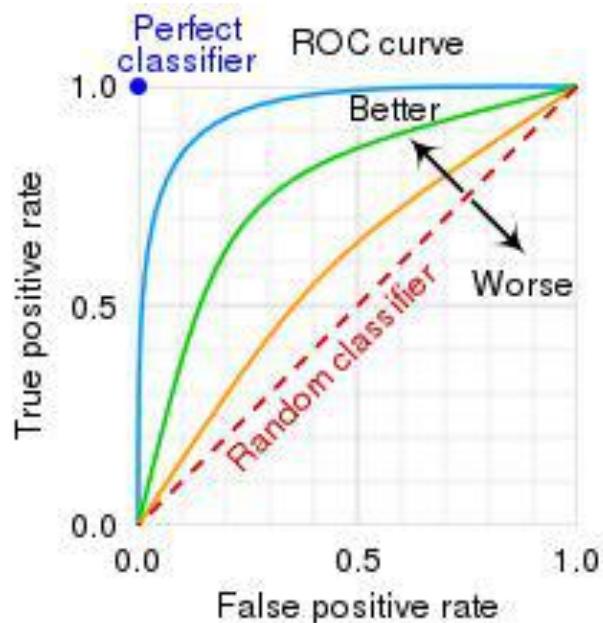
$$\text{Accuracy} = \frac{\sum \text{TP} + \text{TN}}{\sum \text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

**Accuracy:** Accuracy is the fraction of correctly classified samples out of the total number of samples. It is a commonly used metric for classification problems and is expressed as a percentage.

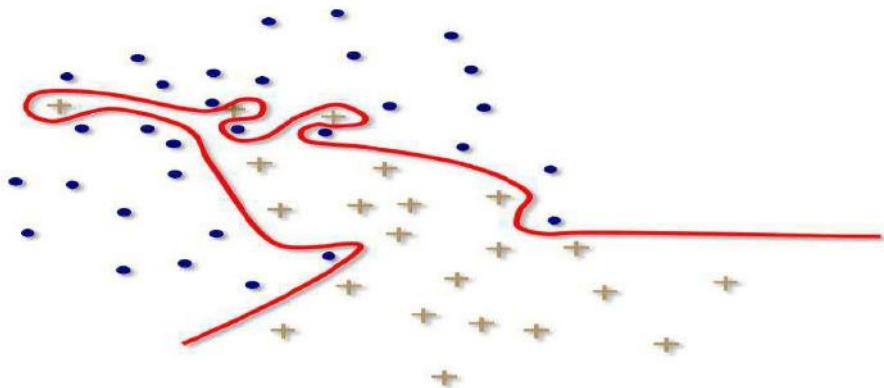
**Recall (Sensitivity):** Recall is the fraction of true positive predictions out of all actual positive cases. It measures how well the model is able to detect positive cases.

# Metrics

- **F1 Score:** The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances precision and recall and is useful when the positive class is rare.
- **ROC Curve:** The ROC (receiver operating characteristic) curve is a plot of the true positive rate versus the false positive rate for different thresholds. It provides a visual representation of the trade-off between precision and recall and is useful for comparing different classifiers.
- **AUC (Area under the ROC Curve):** The AUC is the area under the ROC curve and provides a single scalar metric that summarizes the performance of a classifier. A higher AUC indicates a better-performing classifier.



# Overfitting

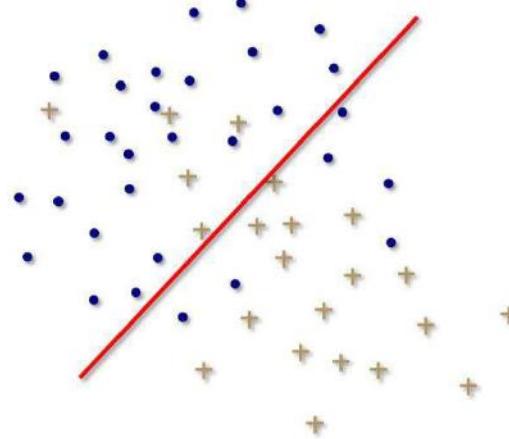


Overfitting and underfitting are two common problems in machine learning that occur when a model is either too complex or too simple for the data it is trained on.

Overfitting occurs when a **model is too complex** and has **learned the noise in the data instead of the underlying relationship between the inputs and outputs**.

As a result, the model **performs well on the training data but poorly on new, unseen data**. Overfitting can be thought of as a model that has memorized the training data, but has not learned to generalize to new data.

# Underfitting



Underfitting occurs when a **model is too simple** and cannot capture the underlying relationship between the inputs and outputs. As a result, the **model performs poorly** on both the training data and new, **unseen data**.

Underfitting can be thought of as a model that has **not learned the pattern in the data**.

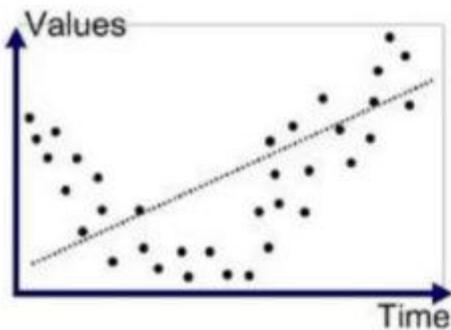
In both cases, it is important to find the right balance between model complexity and simplicity. A model that is too complex will overfit the data, while a model that is too simple will underfit the data. The goal is to find a model that is complex enough to capture the underlying relationship in the data, but not so complex that it memorizes the noise in the data.

# Model (Function) Fitting

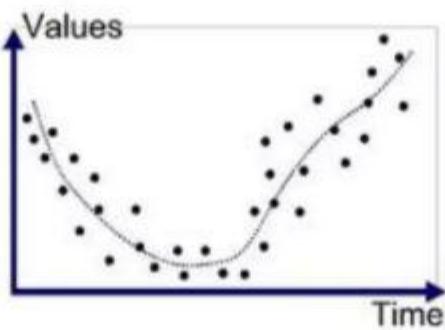
- How well a model performs on training/evaluation datasets will define its characteristics

	Underfit	Overfit	Good Fit
Training Dataset	Poor	Very Good	Good
Evaluation Dataset	Very Poor	Poor	Good

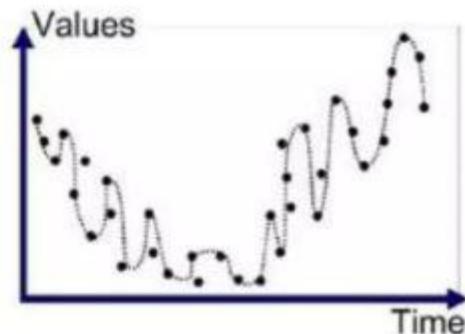
## Model Fitting – Visualization



Underfitted



Good Fit/R robust

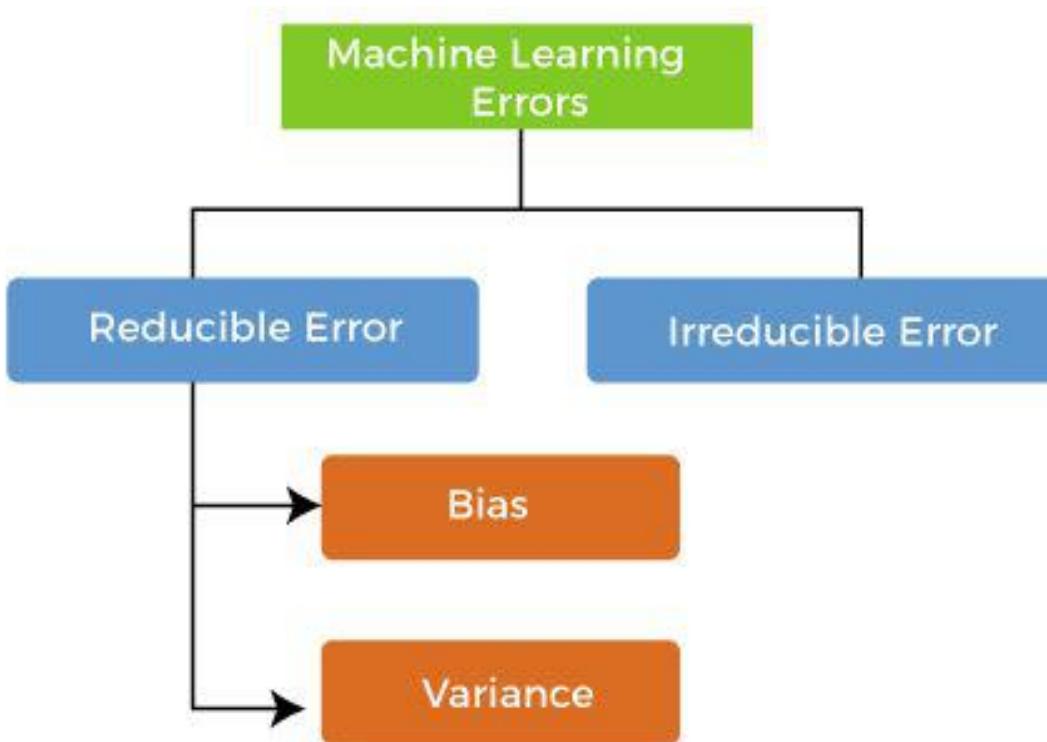


Overfitted

Variations of model fitting [1]

# Bias Variance

- **Bias**
  - Represents the extent to which average prediction over all data sets differs from the desired regression function
- **Variance**
  - Represent the extent to which the model is sensitive to the particular choice of data set



# Bias and Variance

In machine learning, variance and bias are two important concepts that describe the errors that can occur in a model.

**Bias** refers to the **difference between the predicted values of a model and the true values**. A model with high bias tends to consistently predict the same value, regardless of the input. This can happen when the model is too simple and doesn't have enough capacity to capture the underlying relationships in the data. High bias can lead to underfitting, where the model is not flexible enough to fit the training data well.

**Variance**, on the other hand, refers to the **variability of a model's predictions for a given input**. A model with high variance is sensitive to small fluctuations in the training data and will produce different predictions for similar inputs. This can lead to overfitting, where the model becomes too complex and fits the training data too well, but performs poorly on unseen data.

# Example

Suppose you are building a model to predict the height of a person based on their weight.

A model with **high bias** might make a very simple assumption, such as: "**every person is exactly 5 feet tall**." This model would have a **low variance**, because it would give the same prediction for any input. However, it would also have high bias, because it would **consistently overestimate or underestimate the true height** of a person.

On the other hand, a model with **high variance** might make a **very complex assumption**, such as: "**the height of a person is equal to their weight multiplied by a random value**." This model would have a high variance, because the randomness in the model would lead to **different predictions for similar inputs**. However, it would have low bias, because it would fit the training data very well, but would likely perform poorly on unseen data.

A good model should strive to find the **right balance between bias and variance**, so that it generalizes well to unseen data, while still fitting the training data accurately.

# How to reduce Bias?

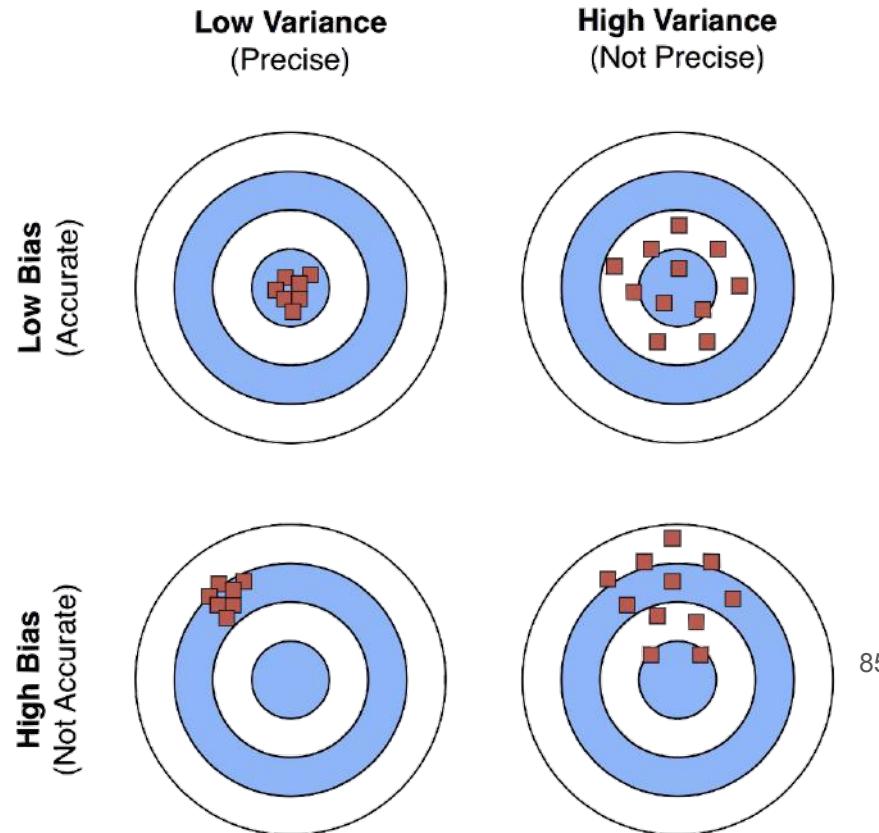
There are several ways to reduce bias in machine learning models:

1. **Increasing model complexity:** By adding more parameters, hidden layers, or increasing the capacity of the model, you can reduce the bias. This can be done by using more complex algorithms like deep neural networks.
2. **Adding more features:** Adding relevant features to the model can help capture more of the underlying relationship between inputs and outputs, which can reduce bias.
3. **Increasing the size of the training data:** With more data, a model can learn the true relationship between inputs and outputs, which can reduce bias.
4. **Regularization:** Regularization is a technique used to prevent overfitting by adding a penalty term to the loss function. This term discourages the model from assigning too much importance to any one feature or from making the model too complex. Some common forms of regularization are L1 regularization, L2 regularization, and dropout.
5. **Cross-validation:** Cross-validation is a technique used to evaluate a model's performance by splitting the data into training and validation sets. By evaluating the model on the validation set, you can get a better idea of how well the model will perform on unseen data, which can help reduce overfitting and bias.

# How to reduce Variance?

There are several ways to reduce variance in machine learning models:

1. **Simplifying the model:** By reducing the number of parameters, hidden layers, or reducing the capacity of the model, you can reduce the variance. This can be done by using simpler algorithms like linear regression or decision trees.
2. **Feature selection:** By removing irrelevant or redundant features from the model, you can reduce the variance caused by noise in the data.
3. **Ensemble methods:** Ensemble methods are techniques that combine the predictions of multiple models to produce a more robust prediction. This can reduce variance by combining the strengths of multiple models and reducing the impact of any one model's weaknesses. Examples of ensemble methods include random forests, gradient boosting, and bagging.
4. **Early stopping:** Early stopping is a technique used to prevent overfitting by stopping the training process when the model's performance on a validation set starts to degrade. By stopping the training early, you can prevent the model from learning the noise in the data, which can reduce variance.
5. **Regularization:** Regularization is a technique used to prevent overfitting by adding a penalty term to the loss function. This term discourages the model from assigning too much importance to any one feature or from making the model too complex. Some common forms of regularization are L1 regularization, L2 regularization, and dropout.



# Generalization and Over Fitting

- Generalization is the model's ability to give sensible outputs to sets of input that it has never seen before
- Arguably, Machine Learning models have one sole purpose; to generalize well
- A model that generalizes well is a model that is neither underfit nor overfit
- When we run our training algorithm on the data set, we allow the overall cost (i.e. distance from each point to the line) to become smaller with more iterations. Leaving this training algorithm run for long leads to minimal overall cost. However, this means that the line will be fit into all the points (including noise), catching secondary patterns that may not be needed for the generalizability of the model
- If we leave the learning algorithm running for long, it could end up fitting the line in the following manner:

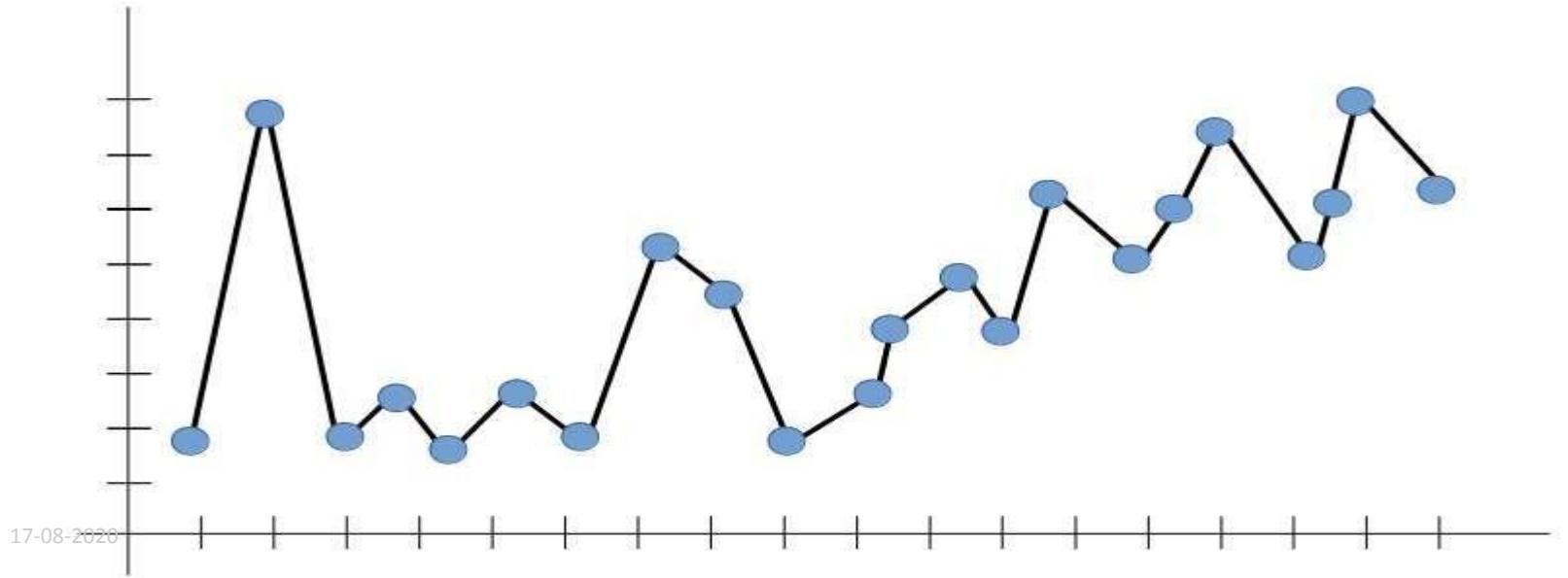
# Generalization

Generalization refers to the ability of a machine learning model to make accurate predictions on new, unseen data.

A model that has learned to generalize well can be used to make predictions on new data points without having seen them before.

The goal of any machine learning model is to generalize well, meaning it can make accurate predictions on new data.

# Generalization and Over Fitting



- This looks good, right? Yes, but is it reliable? Well, not really

# Generalization and Over Fitting

- In the figure above, the algorithm captured all trends — but not the dominant one. If we want to test the model on inputs that are beyond the line limits we have (i.e. generalize), what would that line look like? There is really no way to tell. Therefore, the outputs aren't reliable
- If the model does not capture the dominant trend that we can all see (positively increasing, in our case), it can't predict a likely output for an input that it has never seen before — defying the purpose of Machine Learning to begin with

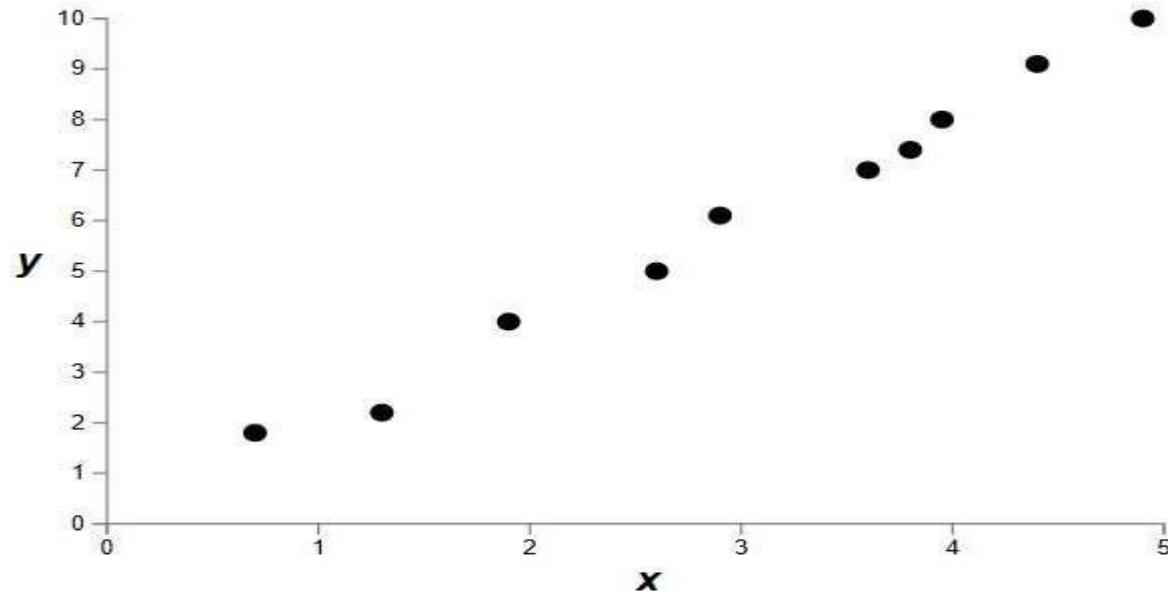
# Regularization

- Too many parameters = overfitting
- Not enough parameters = underfitting
- More data = less chance to overfit
- How do we know what is required?

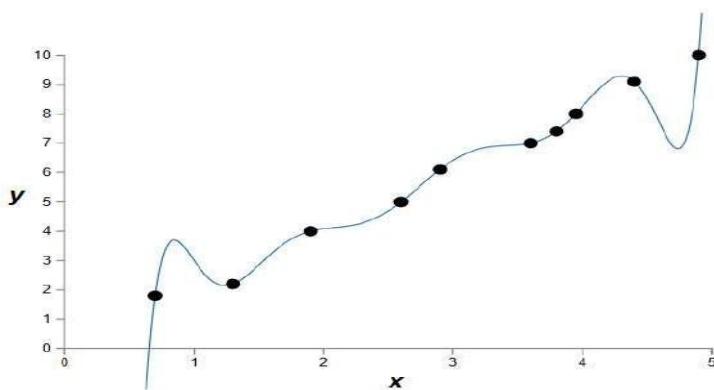
# Regularization

- Attempt to guide solution to not overfit
- But still give freedom with many parameters

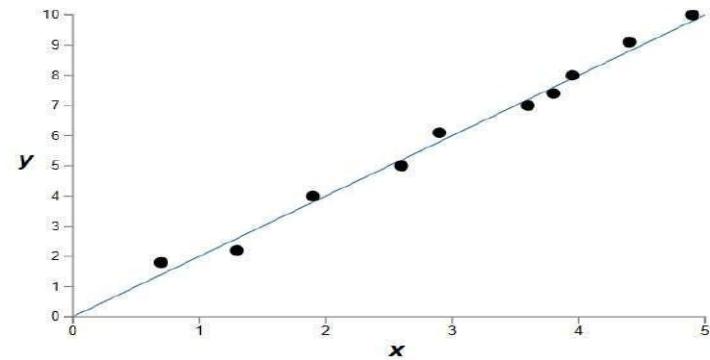
# Data fitting problem



# Which is better a priori?



9<sup>th</sup> order polynomial



1<sup>st</sup> order polynomial

# Regularization

- Regularization in machine learning refers to a set of techniques used to prevent overfitting of a model to the training data.
- Overfitting occurs when a model is too complex and fits the training data too closely, resulting in poor generalization to new, unseen data.
- Regularization methods typically involve adding a penalty term to the loss function of a model during training, which discourages the model from learning complex patterns in the training data that may not generalize well to new data.

There are several types of regularization techniques used in machine learning, including L1 regularization (also known as Lasso regularization), L2 regularization (also known as Ridge regularization), and dropout regularization.

# Forms of Regularization

- Adding more data is a kind of regularization
- Pooling is a kind of regularization
- Data augmentation is a kind of regularization



# Regularization

L1, L2, and L3 regularization are different types of regularization that are used to prevent overfitting in machine learning models.

**L1 Regularization:** Also known as **Lasso regularization**, L1 regularization adds a **penalty term to the loss function that is proportional to the absolute value of the coefficients**.

This leads to sparse solutions, where many of the coefficients are exactly zero.

This can be **useful for feature selection**, as it effectively removes the least important features from the model.

# L2 Regularization

Also known as **Ridge regularization**, L2 regularization adds a penalty term to the loss function that is proportional to the square of the coefficients.

This leads to **small coefficients**, which can prevent overfitting.

The effect of L2 regularization is to shrink the coefficients towards zero, but not to zero.

L1 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

L2 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M W_j^2$$

Loss function

Regularization Term

# L3 Regularization

L3 regularization is a type of regularization that is less commonly used, but works similarly to L2 regularization. It adds a **penalty term to the loss function that is proportional to the cube of the coefficients**. Like L2 regularization, L3 regularization leads to small coefficients and can prevent overfitting.

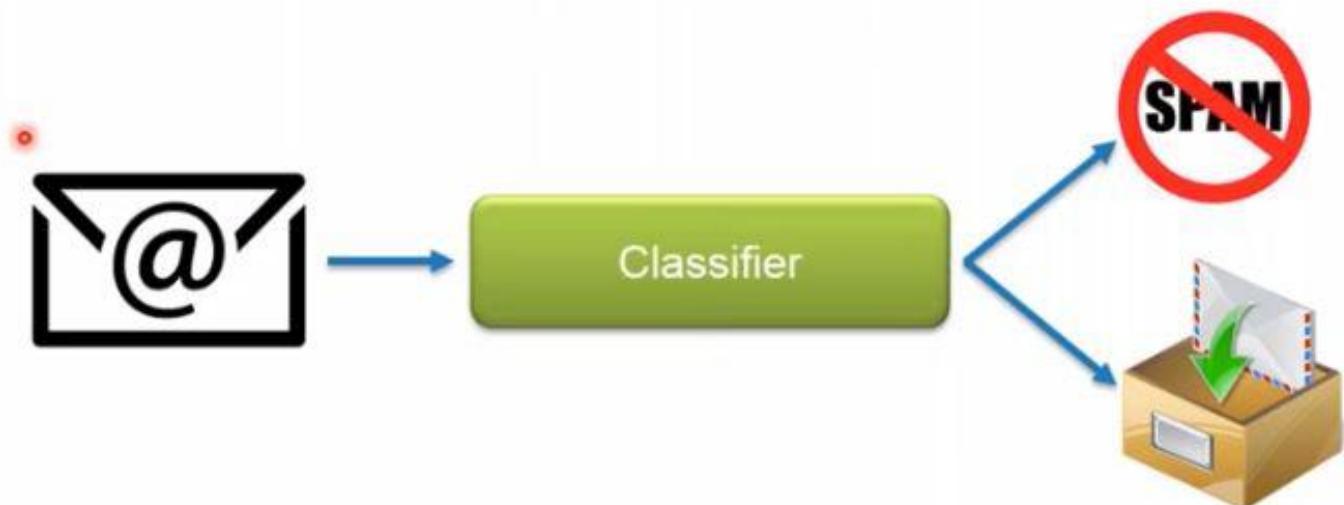
The choice of L1, L2, or L3 regularization will depend on the specific problem and the goals of the model. In general, **L1 regularization is useful for feature selection**, while **L2 regularization is useful for preventing overfitting**. L3 regularization is less commonly used and may be less effective than L1 or L2 regularization.

# SUPERVISED LEARNING

# What is Classification?

---

- Classification is the problem of identifying to which set of categories a new observation belongs
- The goal of classification is to find boundaries that best separate different categories of data. These 'decision boundaries' then allow you to differentiate between classes of data, and classify any new value



# **Applications of Classification**

---

- Medical Diagnostics
  - predict whether a patient is sick or not
- Animal Recognition
  - Classifying a set of animal images
- Machine vision
  - Classify faces based on patterns(face detection)
- Market segmentation
  - Predict if customer will respond to promotion or not
- Bioinformatics
  - Classify proteins according to their function

# Nearest Neighbours Classification

K-Nearest Neighbors (KNN) is a type of **instance-based or lazy learning algorithm**, which is a type of **supervised learning** in machine learning. In KNN, the algorithm does not learn a model from the training data but instead, it stores the entire training dataset in memory and uses it to make predictions on new, unseen data.

In instance-based learning, the model directly memorizes the training examples and uses them to make predictions. KNN uses a **distance metric, such as Euclidean distance**, to find the **k closest neighbors** to a **new data point** in the training dataset, and the class of the new data point is determined based on the **majority class among the k neighbors**.

KNN is a **non-parametric algorithm** because it **does not make any assumptions about the underlying distribution of the data**. It is also a simple and easy-to-understand algorithm that can be used for **both classification and regression tasks**. However, it can be **computationally expensive for large datasets**, and the choice of **k** can have a significant impact on the performance of the algorithm.

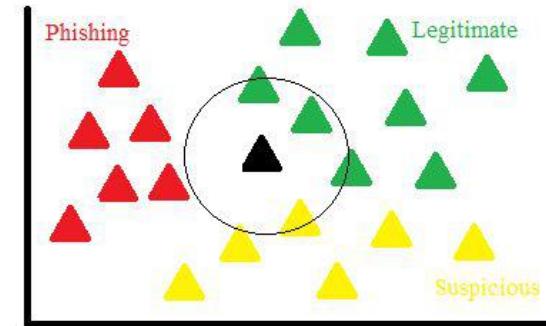


Table 1: Distance Metrics for kNN

Distance	Equation
Euclidean	$d = \sqrt{\sum_{j=1}^n (x_{sj} - x_{tj})^2}$
City Block	$d = \sum_{j=1}^n  x_{sj} - x_{tj} $
Chebyshev	$d = \max_j \{  x_{sj} - x_{tj}  \}$
Cosine	$d = 1 - \frac{\sum_{j=1}^n x_{sj}x_{tj}}{\sqrt{\sum_{j=1}^n x_{sj}x_{sj}}\sqrt{\sum_{j=1}^n x_{tj}x_{tj}}}$
Correlation	$d = 1 - \frac{(x_s - \tilde{x}_s)(x_t - \tilde{x}_t)'}{\sqrt{(x_s - \tilde{x}_s)(x_s - \tilde{x}_s)'}\sqrt{(x_t - \tilde{x}_t)(x_t - \tilde{x}_t)'}}$

# Types of Classifiers

---

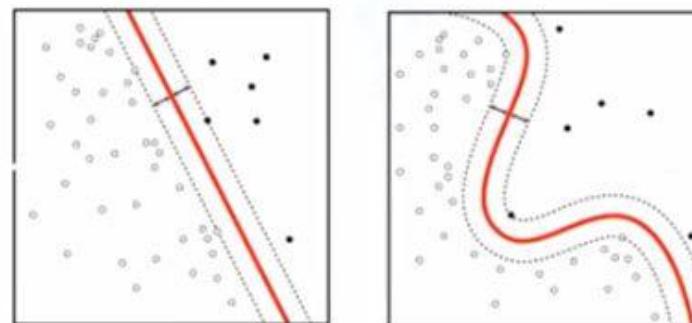
- Some of the popular classifiers used in Machine Learning are:



# Support Vector Machine

---

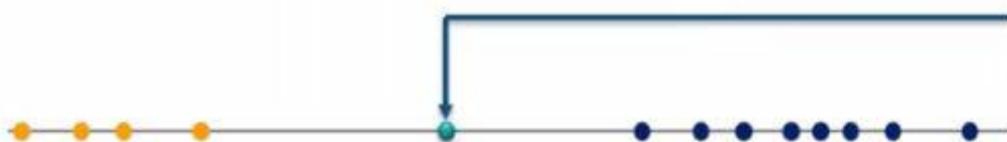
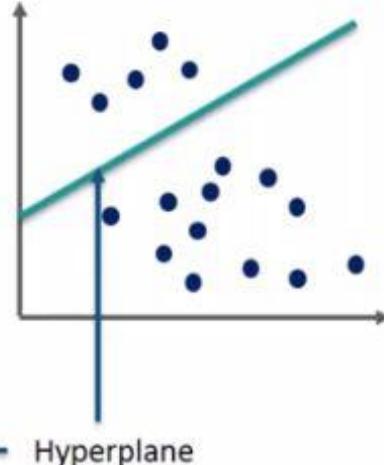
- Support Vector Machine(SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges
- It tries to define a hyperplane which can split the data in the most optimal way such that there is a wide margin among the hyperplane and the observations
- It is one of the most efficient algorithm in Machine Learning



# What is Hyperplane?

---

- An hyperplane is a generalization of a plane
  - in one dimension, an hyperplane is called a point
  - in two dimensions, it is a line
  - in three dimensions, it is a plane
  - in more dimensions you can call it an hyperplane



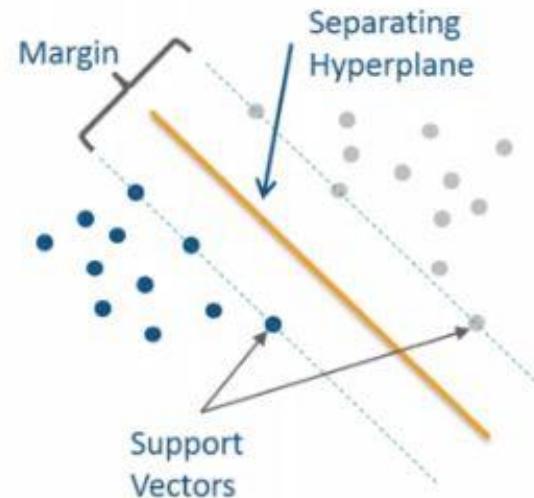
# SVM

- In SVM, the goal is to find the best possible boundary or hyperplane that separates the data into different classes.
- This hyperplane is chosen such that the distance between the nearest data point from both classes is maximized. These nearest data points are called support vectors.
- SVM can be used for both linear and non-linear classification and regression problems.
- In non-linear problems, SVM uses a kernel function to transform the data into a higher-dimensional space where it becomes possible to find a linear boundary that separates the classes.

# Support Vector Machine

- An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible
- New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall

Support Vectors are simply the co-ordinates that lie on the margins of the hyperplane



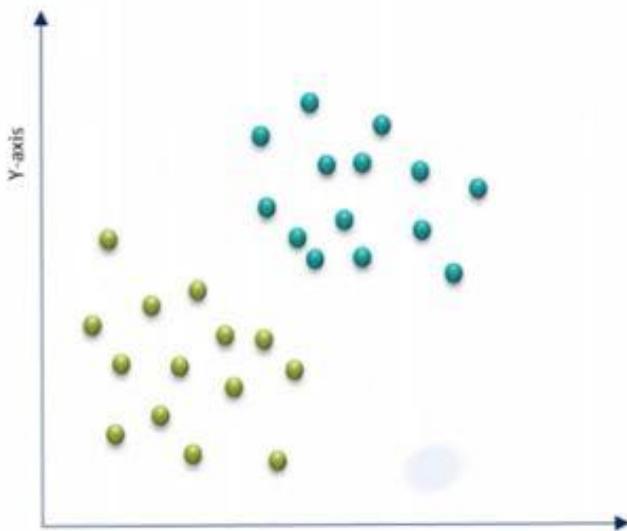
## SVM (contd.)

- SVM is a powerful algorithm because it can handle high-dimensional data and can find the optimal solution even when the data is not linearly separable.
- SVM is widely used in many applications, including text classification, image classification, and bioinformatics.
- Some of the advantages of SVM include its ability to handle high-dimensional data, its flexibility in choosing different kernel functions, and its ability to find the global minimum, which leads to better generalization performance.
- However, SVM can be computationally expensive for large datasets and may not perform well when there is noise or outliers in the data.

# How it Works?

---

- Consider the following example
  - Suppose we have two classes plotted as:

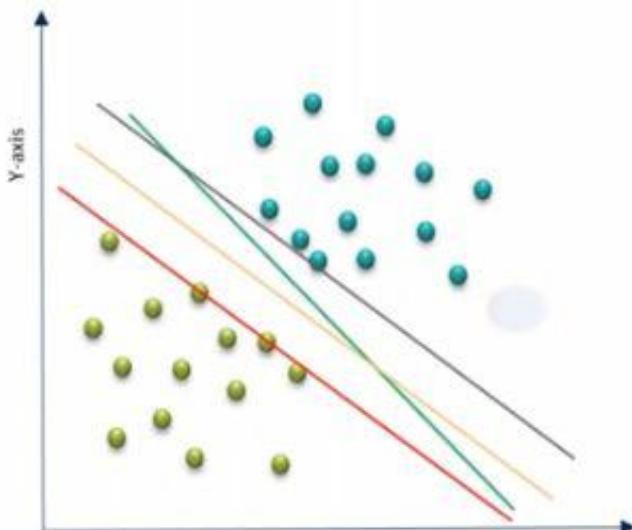


Just by looking at the plot, we can see that it is possible to separate the data using a straight line

## How it Works?

---

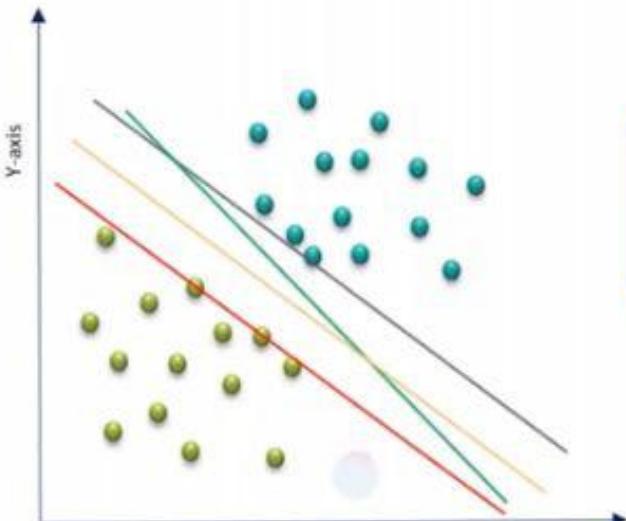
- We can draw a separating lines as:



Multiple separating lines can be drawn here

# How it Works?

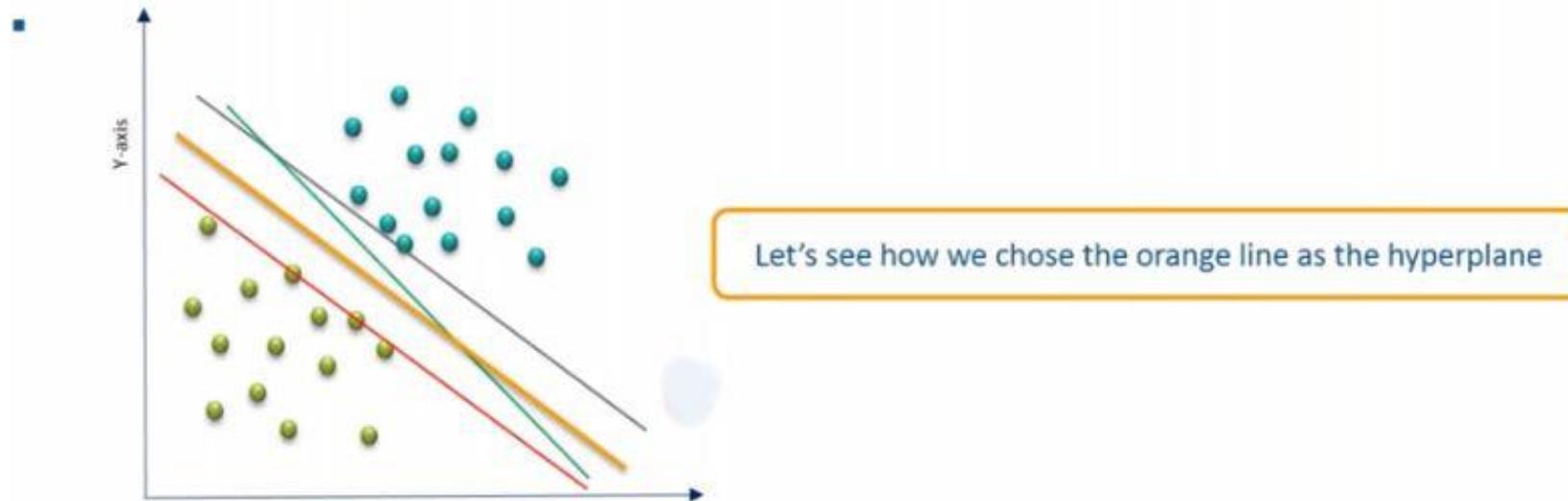
- Suppose we choose the red line as the hyperplane, then we can see that some the observations will be misclassified



Intuitively, we can see that if we select an hyperplane which is close to the data points of one class, then it might not generalize well. So we will try to select an hyperplane as far as possible from data points from each category

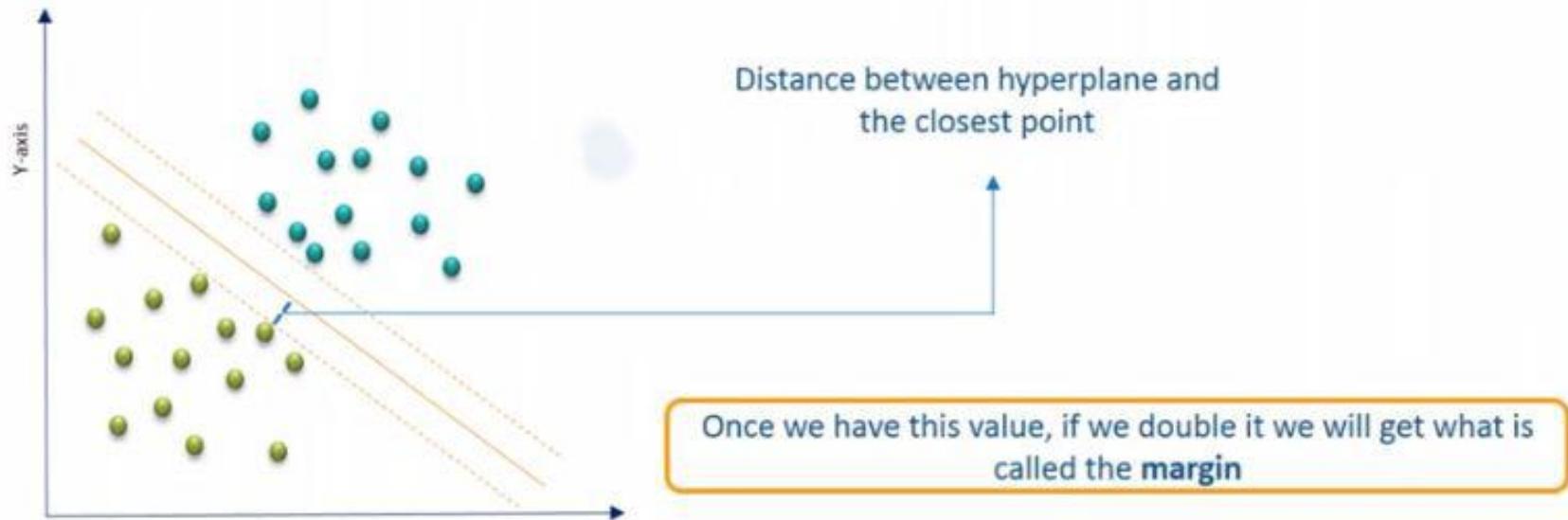
# How it Works?

- After going through all the possible hyperplane, we can see that the orange lines separates the data in an optimal way



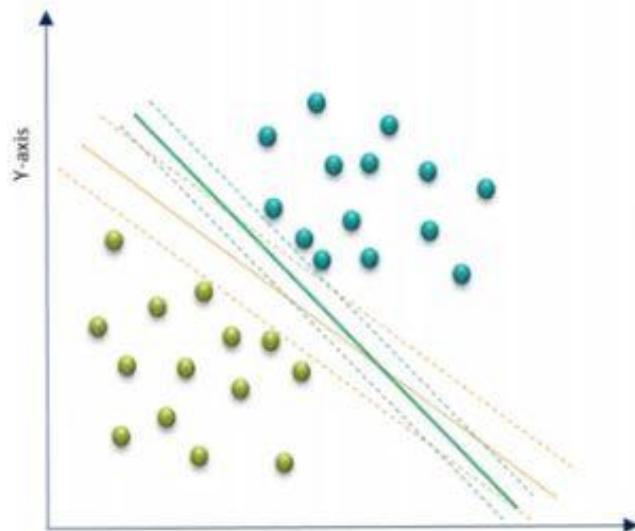
# Choosing Optimal Hyperplane

- Given a particular hyperplane, we can compute the distance between the hyperplane and the closest data point



# Choosing Optimal Hyperplane

- Similarly we will find the margin for every other hyperplanes

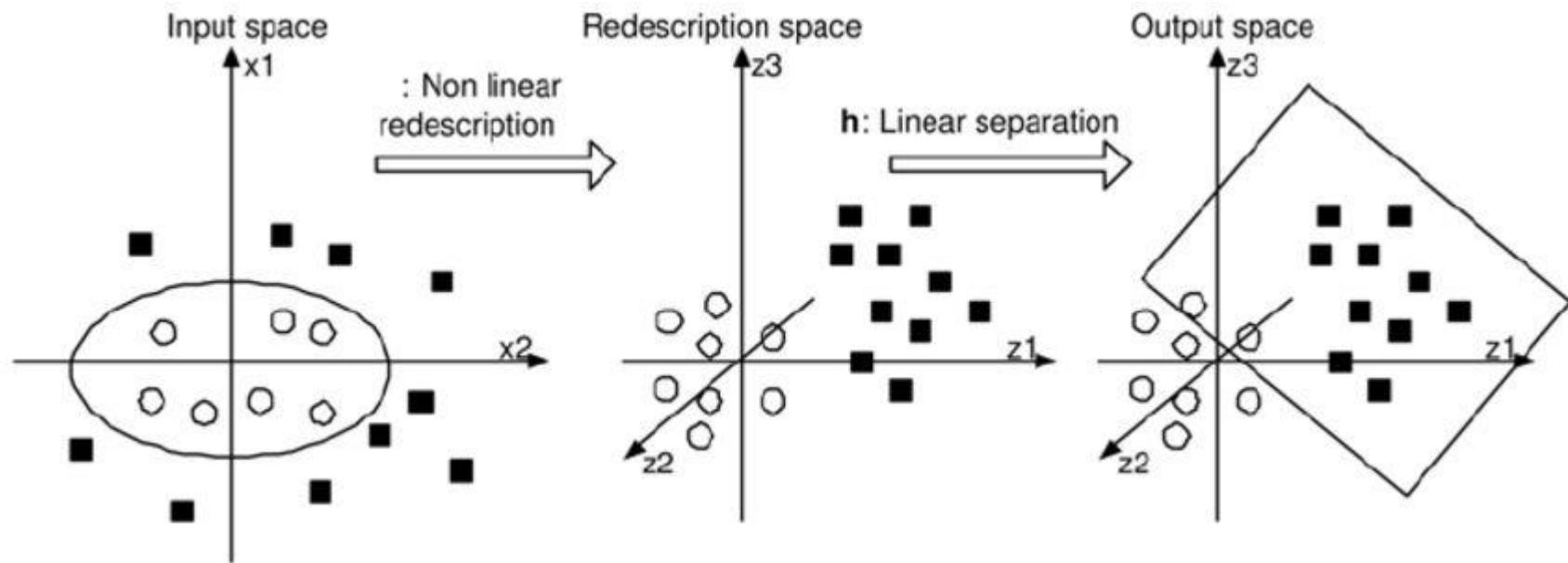


Basically the margin is a no man's land. There will never be any data point inside the margin

# Choosing Optimal Hyperplane

- After we find the margins of all hyperplanes, we will select the hyperplane having the largest margin as our separating hyperplane





## Use Case – Hands On

---

- Here we will use SVM on the iris dataset and we will create a model which can classify the flowers based on their features
- The dataset looks like:

sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	Species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa

```
from sklearn import datasets
from sklearn import svm

iris = datasets.load_iris()

X = iris.data[:, :3]

Y = iris.target

from sklearn.model_selection import train_test_split
```

```
X_train,X_test,Y_train,Y_test =  
    train_test_split(X, Y, test_size=0.2,random_state=0)
```

```
model = svm.SVC(kernel="linear")
```

```
model
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,  
decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',  
max_iter=-1, probability=False, random_state=None, shrinking=True,  
tol=0.001, verbose=False)
```

```
model.fit(X_train, Y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,  
decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',  
max_iter=-1, probability=False, random_state=None, shrinking=True,  
tol=0.001, verbose=False)
```

```
accuracy = model.score(X_test, Y_test)
```

```
accuracy
```

```
0.9666666666666667
```

# UNSUPERVISED LEARNING

The following scenarios implement Clustering:

- A telephone company needs to establish its network by putting its towers in a particular region it has acquired. The location of putting these towers can be found by clustering algorithm so that all its users receive maximum signal strength.
- Cisco wants to open its new office in California. The management wants to be cordial to its employees and want their office in a location so that its employees' commutation is reduced to minimum.
- The Miami DEA wants to make its law enforcement more stringent and hence have decided to make their patrol vans stationed across the area so that the areas of high crime rates are in vicinity to the patrol vans.
- A Hospital Care chain wants to open a series of Emergency-Care wards, keeping in mind the factor of maximum accident prone areas in a region.



Performance tuning technique number 41:  
clustering to reduce overheads

### Why Clustering?

Organizing data into clusters such that there is:

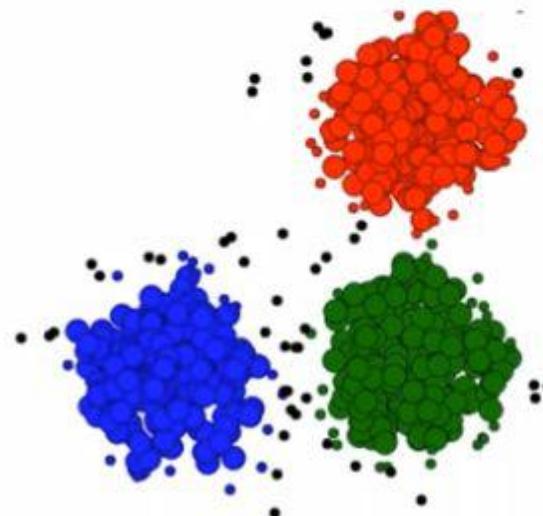
- High intra-cluster similarity
- Low inter-cluster similarity
- Informally, finding natural groupings among objects

- Organizing data into clusters shows internal structure of the data  
Ex. Clusty and clustering genes
- Sometimes the partitioning is the goal  
Ex. Market segmentation
- Prepare for other AI techniques  
Ex. Summarize news (cluster and then find centroid)
- Discovery in data  
Ex. Underlying rules, reoccurring patterns, topics, etc.

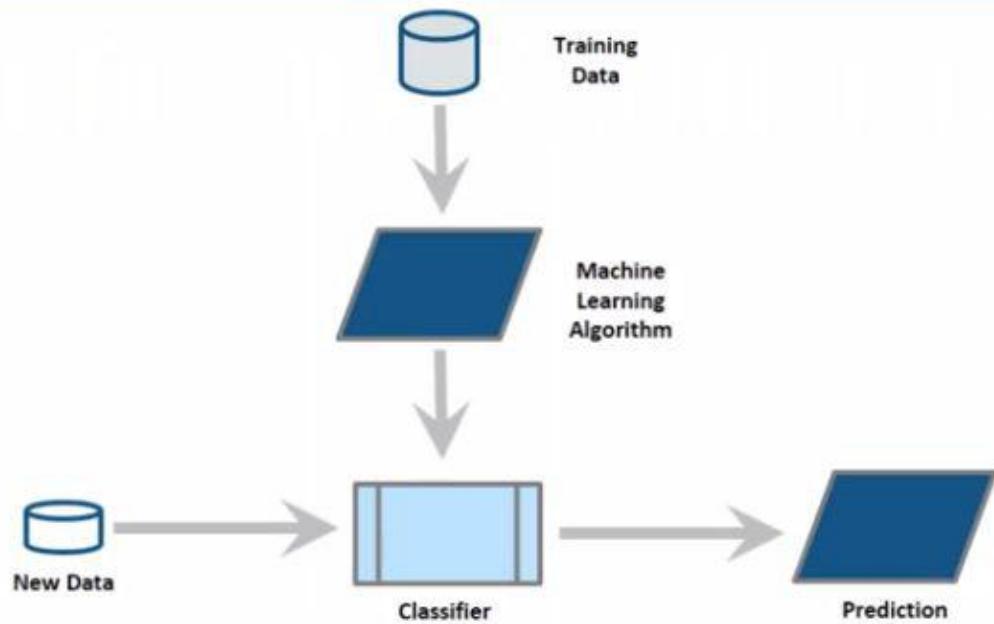
# Unsupervised Learning

---

- Unsupervised learning is the training of a model using information that is neither classified nor labelled
- This model can be used to cluster the input data in classes on the basis of their statistical properties



# Unsupervised Learning - Process Flow

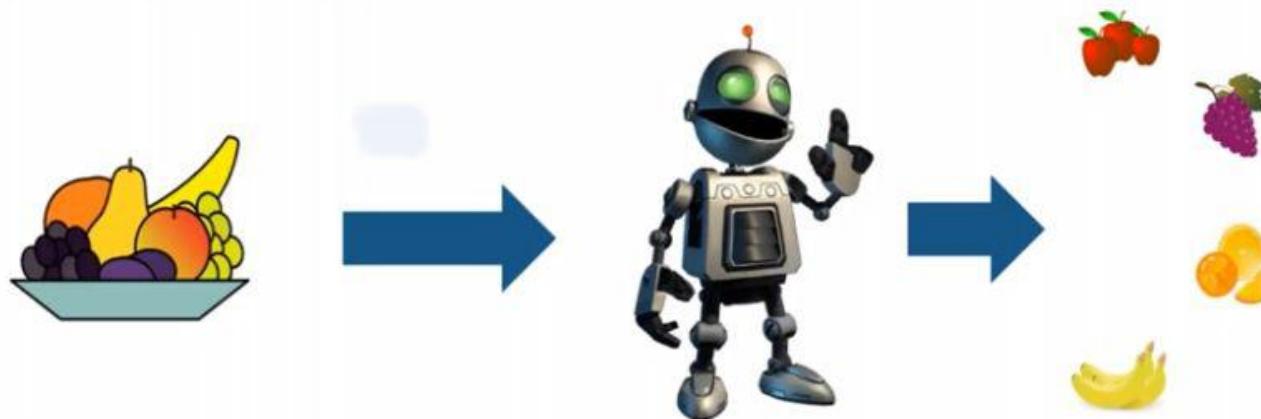


The machine learns from training data and classifies new data based on it

## Unsupervised Learning - Process Flow

---

- A set of fruit images is first fed into the system
- The system identifies different fruits using features like color, size, surface type etc, and it categorizes them
- When a new fruit is shown, it analyses its features and puts it into the category having similar featured items

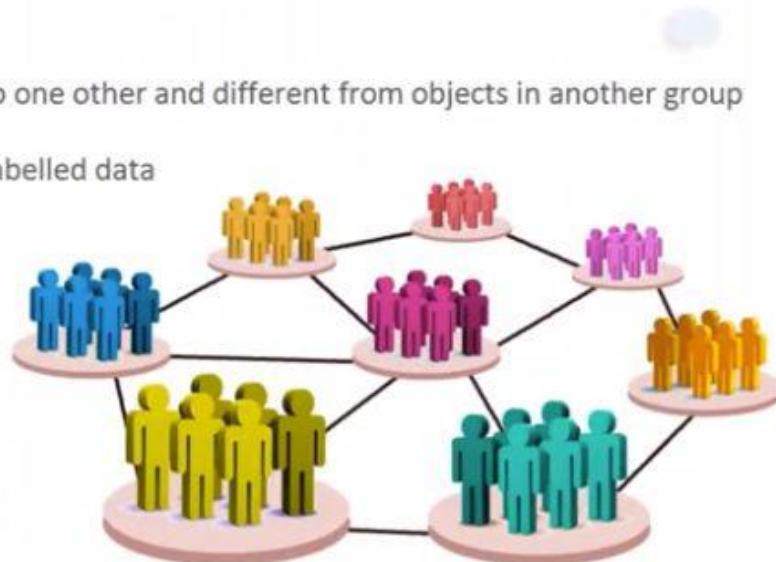


# CLUSTER ANALYSIS

# What is Clustering?

---

- Clustering means grouping of objects based on the information found in the data, describing the objects or their relationship
- The goal is that objects in one group will be similar to one other and different from objects in another group
- It deals with finding a structure in a collection of unlabelled data
- Some Examples of clustering methods are :
  - K-means Clustering
  - Fuzzy/ C-means Clustering
  - Hierarchical Clustering



# Why Clustering?

---

- The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data
- Organizing data into clusters shows internal structure of the data
- Sometimes partitioning is the goal
- Some Examples are
  - finding groups of customers with similar behavior
  - classification of animals given their features, example classification of animals into reptiles, mammals, birds, fish etc.

The purpose of clustering algorithm is to make sense of and extract value from large sets of structured and unstructured data

# Clustering Use Cases



## Marketing

Discovering distinct groups in customer databases, such as customers who make lot of long-distance calls.

## Insurance

Identifying groups of crop insurance policy holders with a high average claim rate. Farmers crash crops, when it is "profitable".

## Search Engine

Better the clustering algorithm used, better are the chances of getting the required result on the front page.

## Seismic studies

Identifying probable areas for oil/gas exploration based on seismic data

# Types of Data used for Clustering

- data matrix

- the “classic” data input

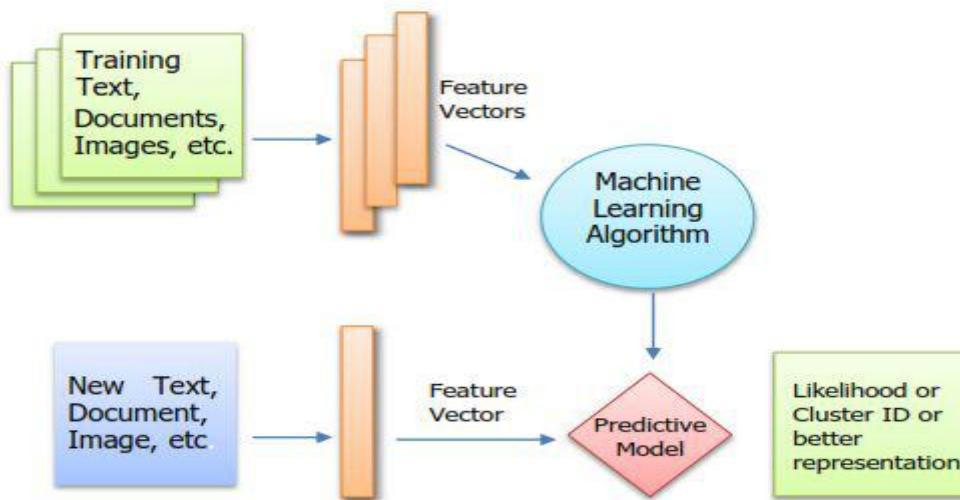
		attributes/dimensions				
		$x_{11}$	$\dots$	$x_{1f}$	$\dots$	$x_{1p}$
		$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
tuples/objects		$x_{i1}$	$\dots$	$x_{if}$	$\dots$	$x_{ip}$
		$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
		$x_{n1}$	$\dots$	$x_{nf}$	$\dots$	$x_{np}$
		objects				

- distance or dissimilarity matrix

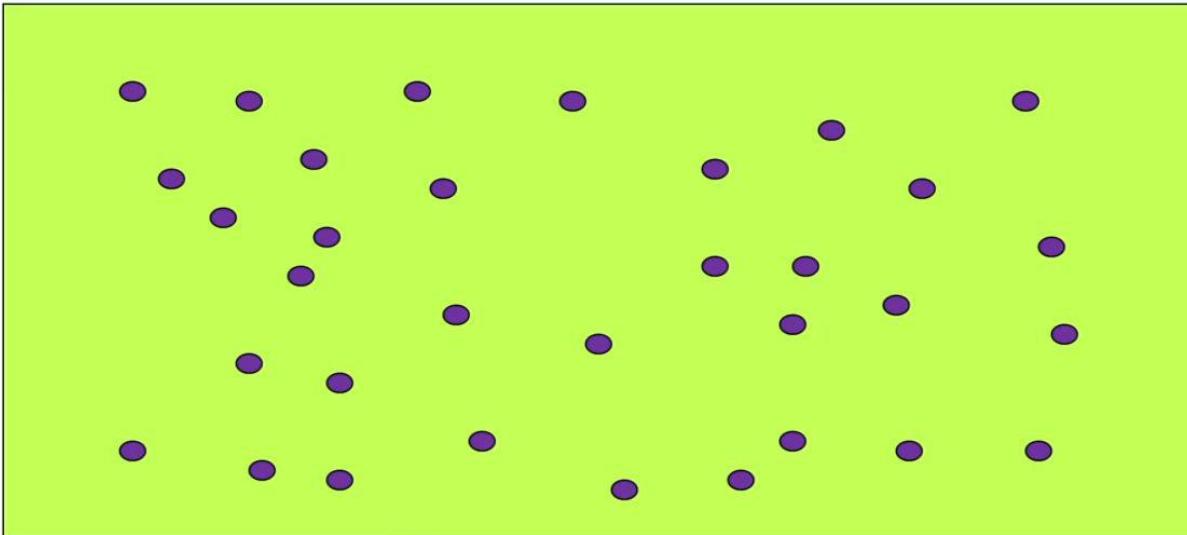
- The desired input to some clustering algorithms like Hierarchical clustering

objects	0		
	$d(2,1)$	0	
	$d(3,1)$	$d(3,2)$	0
	:	:	:
	$d(n,1)$	$d(n,2)$	$\dots$
			0

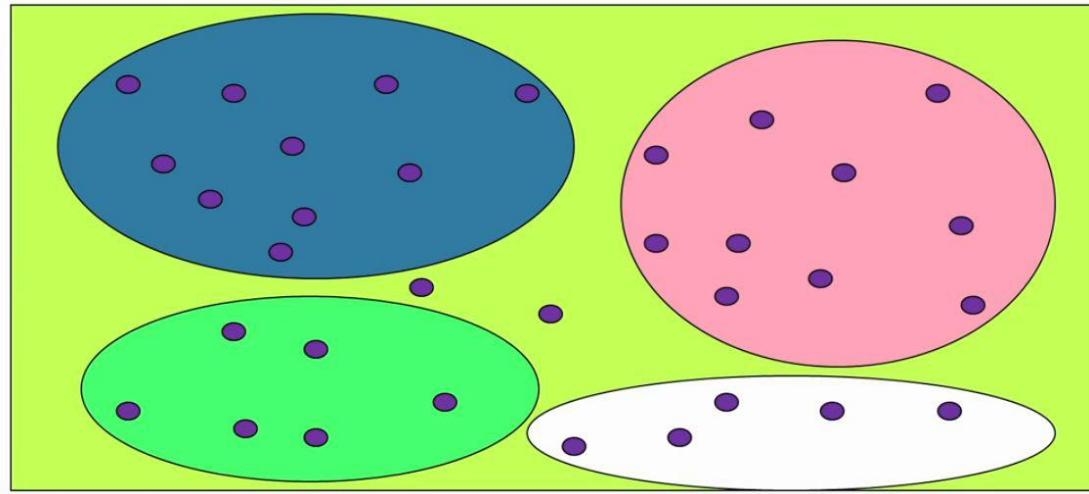
# The flowchart:



# Unlabelled Training Data



# Possible Clusters

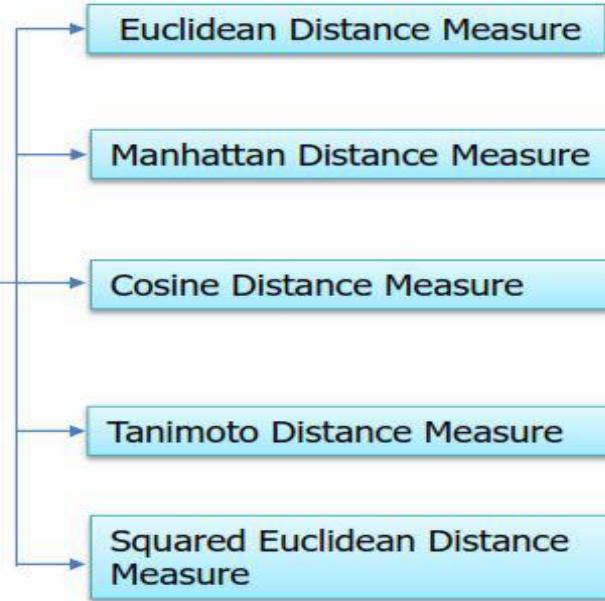


We are finding cohesive groups of points and points which are outliers  
I need some form of bias: here I have assumed all clusters as ellipsoids  
Not all points fall into clusters

# Applications

- Customer data (targeted promotions for group of customers)
  - Discover classes of customers
- Image pixels (segmentation)
  - Discover regions
- Words
  - Synonyms
- Documents
  - Topics

## Distance Measures



Similarity can also be measured in terms of the placing of data points.

By finding the distance between the data points , the distance/difference of the point to the cluster can be found.

From this image we can say that, The Euclidean distance measure gives 5.65 as the distance between (2, 2) and (6, 6) whereas the Manhattan distance is 8.0

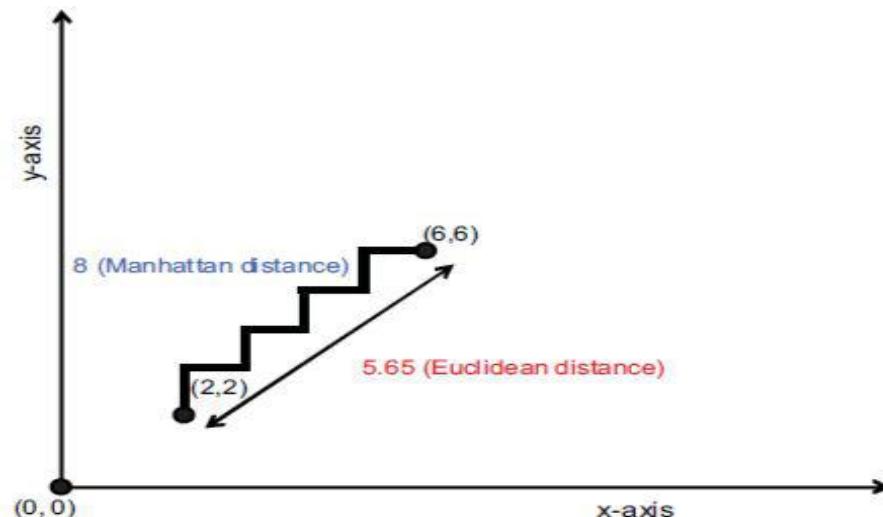
Mathematically, Euclidean distance between two n-dimensional vectors

$(a_1, a_2, \dots, a_n)$  and  $(b_1, b_2, \dots, b_n)$  is:

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

Manhattan distance between two n-dimensional vectors

$$d = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$$



The formula for the cosine distance between  $n$ -dimensional vectors  
 $(a_1, a_2, \dots, a_n)$  and  $(b_1, b_2, \dots, b_n)$  is

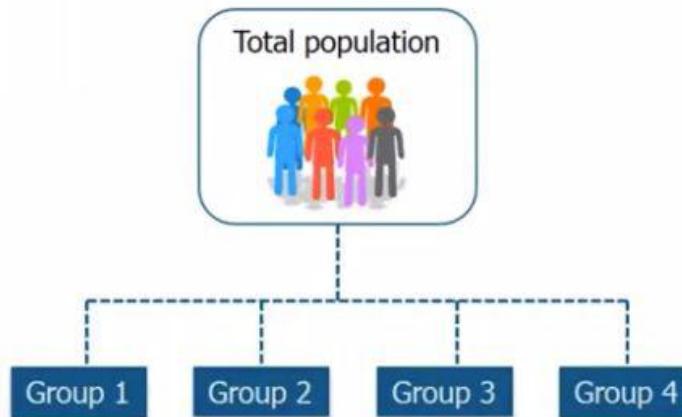
$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \dots + a_n b_n)}{(\sqrt{a_1^2 + a_2^2 + \dots + a_n^2}) \sqrt{(b_1^2 + b_2^2 + \dots + b_n^2)})}$$

# K-MEANS CLUSTERING

# K-Means Clustering

- The process by which objects are classified into a predefined number of groups so that they are as much dissimilar as possible from one group to another group, but as much similar as possible within each group

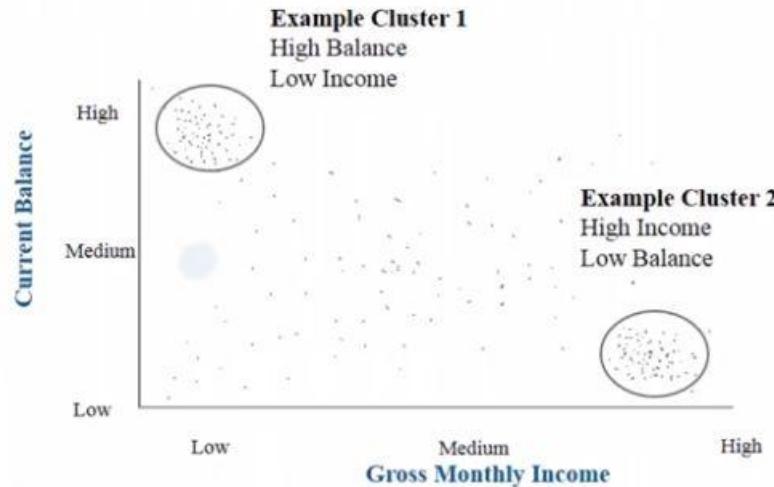
- The objects in group 1 should be as similar as possible
- But there should be much difference between an object in group 1 and group 2
- The attributes of the objects are allowed to determine which objects should be grouped together



# K-Means Clustering

- Consider the following example

- The objects in Cluster 1 have similar characteristics (High Income and Low balance)
- Also the objects in Cluster 2 have the same characteristic (High Balance and Low Income)
- But there are much differences between an object in Cluster 1 and an object in Cluster 2



## Example

---

- The plot of students in an area is as given below



## Initialization

1 Initialization

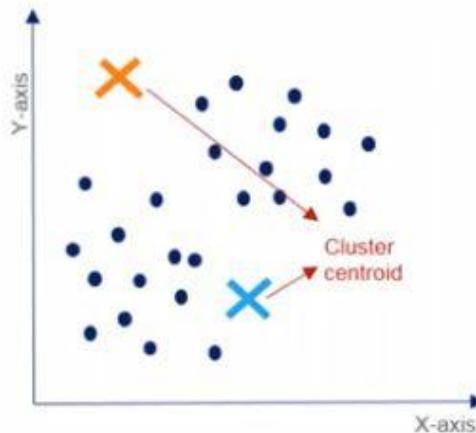
2 Cluster assignment

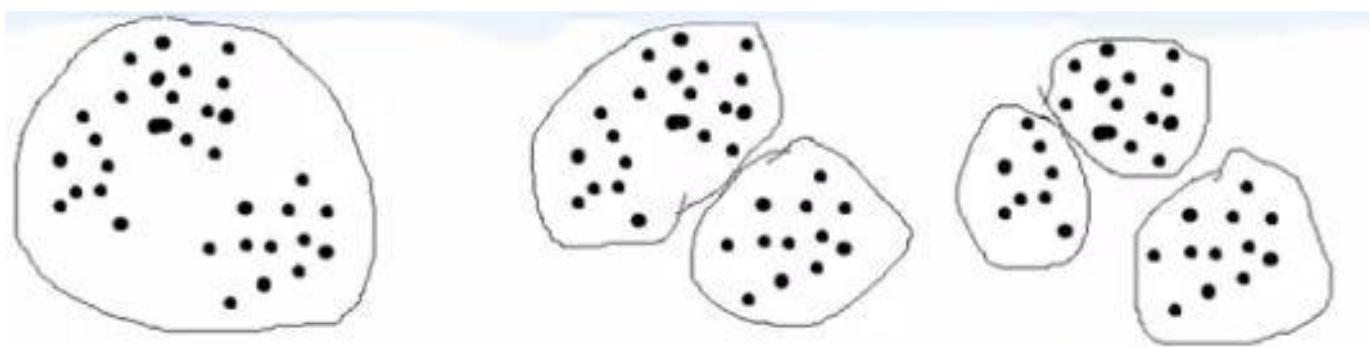
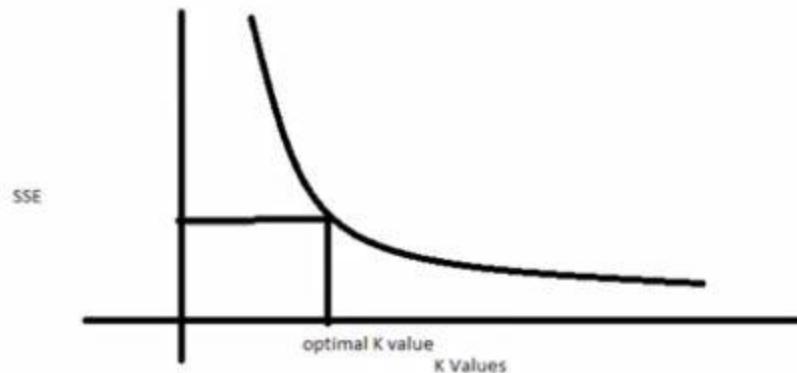
3 Move centroid

4 Optimization

5 Convergence

- Randomly initialize k points called the cluster centroids  
Here, k = 2
- Value of k(number of clusters) can be determined by the elbow curve





SSE1,  $k=1$

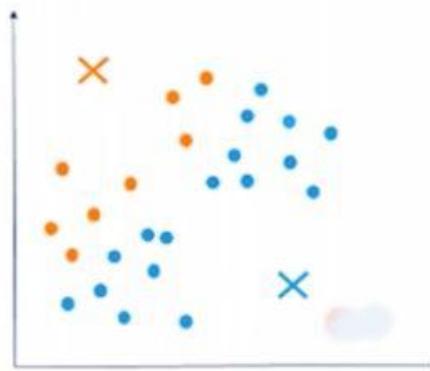
SSE2,  $k=2$

SSE3,  $k=3$

## Cluster assignment

- 1 Initialization
- 2 Cluster assignment
- 3 Move centroid
- 4 Optimization
- 5 Convergence

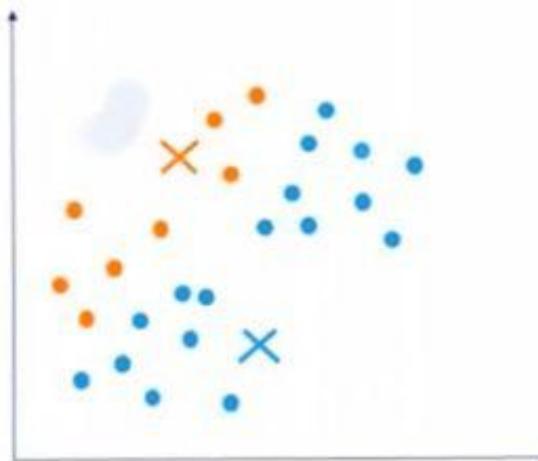
- Compute the distance between the data points and the cluster centroid initialized
- Depending upon the minimum distance, data points are divided into two groups



## Move centroid

- 1 Initialization
- 2 Cluster assignment
- 3 Move centroid
- 4 Optimization
- 5 Convergence

- Compute the mean of blue dots
- Reposition blue cluster centroid to this mean
- Compute the mean of orange dots
- Reposition orange cluster centroid to this mean



## Optimization

1 Initialization

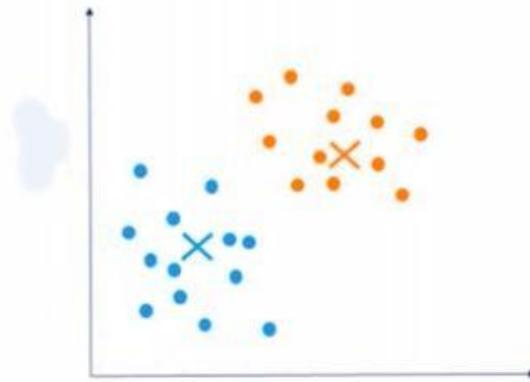
2 Cluster assignment

3 Move centroid

4 Optimization

5 Convergence

- Repeat previous two steps iteratively till the cluster centroids stop changing their positions



## Convergence

1 Initialization

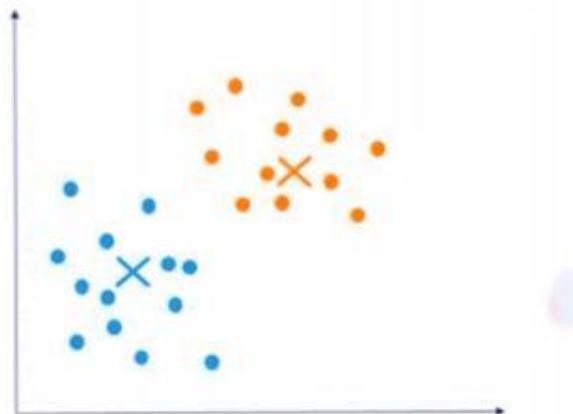
2 Cluster assignment

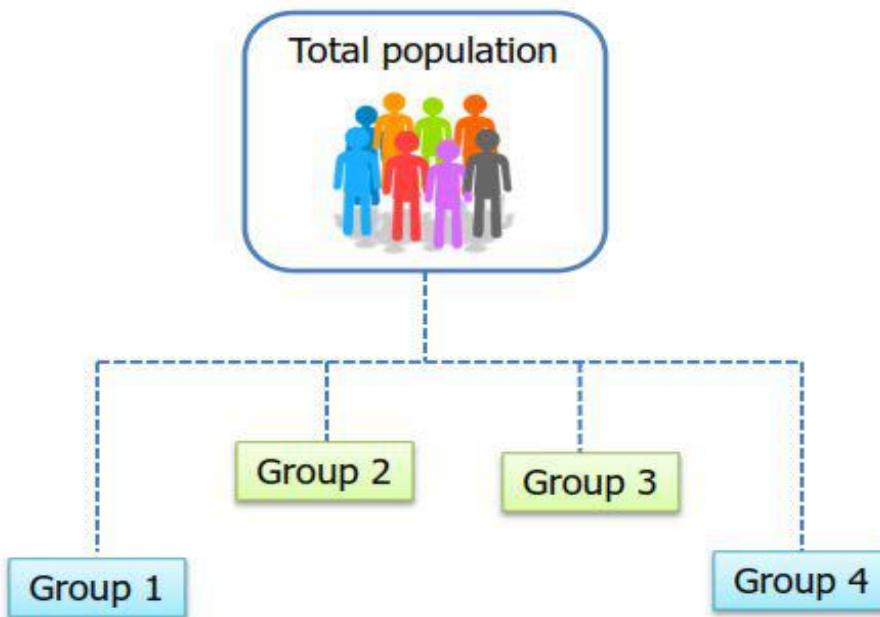
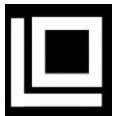
3 Move centroid

4 Optimization

5 Convergence

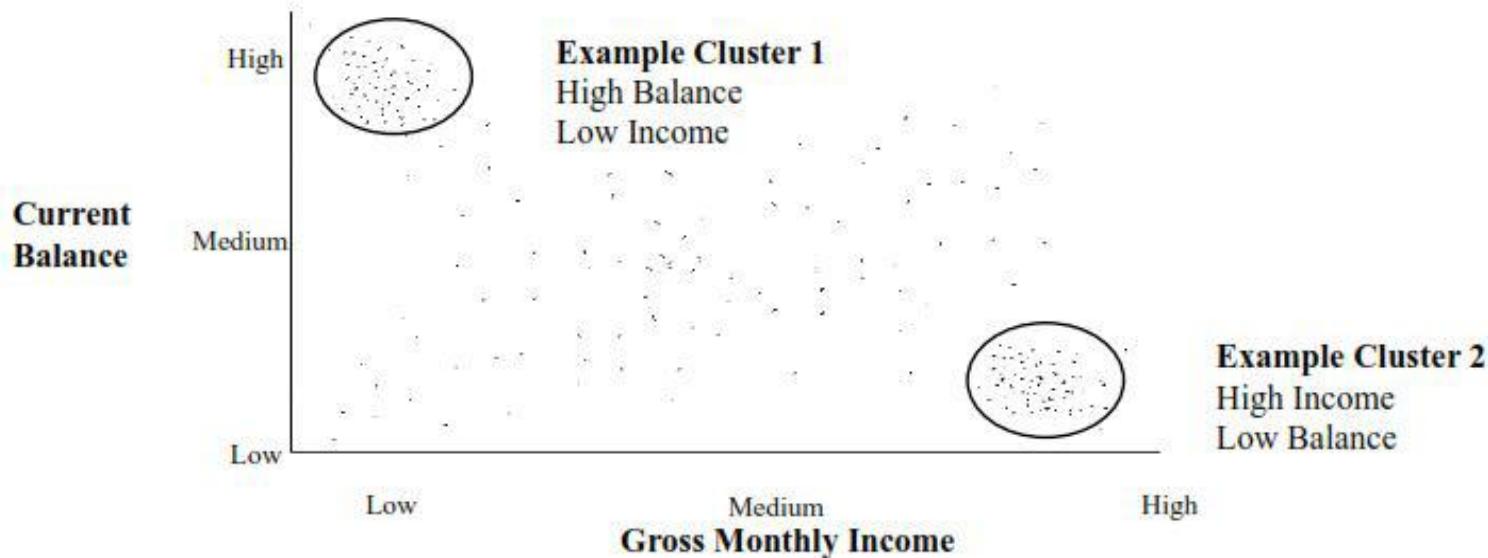
- Finally, k-means clustering algorithm converges
- Divides the data points into two clusters clearly visible in orange and blue



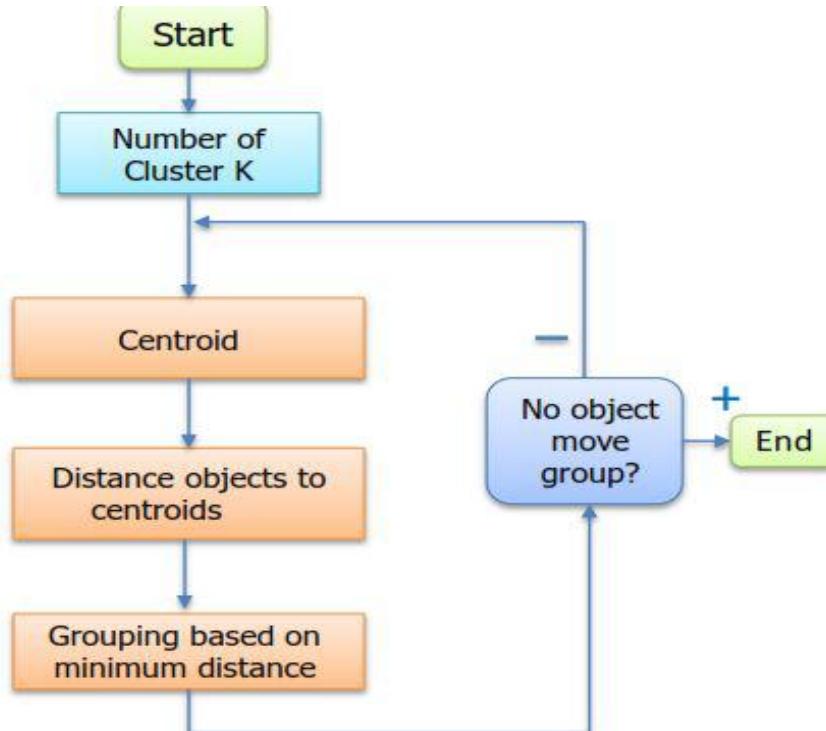


- The process by which objects are classified into a number of groups so that they are as much dissimilar as possible from one group to another group, but as much similar as possible within each group.
- The objects in group 1 should be as similar as possible.
- But there should be much difference between an object in group 1 and group 2.
- The attributes of the objects are allowed to determine which objects should be grouped together.

## Basic concepts of Cluster Analysis using two variables

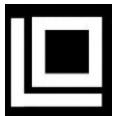


- Cluster 1 and Cluster 2 are being differentiated by Income and Current Balance.
- The objects in Cluster 1 have similar characteristics (High Income and Low balance), on the other hand the objects in Cluster 2 have the same characteristic (High Balance and Low Income).
- But there are much differences between an object in Cluster 1 and an object in Cluster 2.



Iterate until *stable* (cluster centers converge):

1. Determine the centroid coordinate.
2. Determine the distance of each object to the centroids.
3. Group the object based on minimum distance (find the closest centroid)



### Problem Statement:

The newly appointed Governor has finally decided to do something for the society and wants to open a chain of schools across a particular region, keeping in mind the distance travelled by children is minimum, so that the percentage turnout is more.

Poor fella cannot decide himself and has asked its Data Science team to come up with the solution.

Bet, these guys have the solution to almost everything!!



1. If  $k=4$ , we select 4 random points in the 2d space and assume them to be cluster centers for the clusters to be created.



2. We take up a random data point from the space and find out its distance from all the 4 clusters centers.

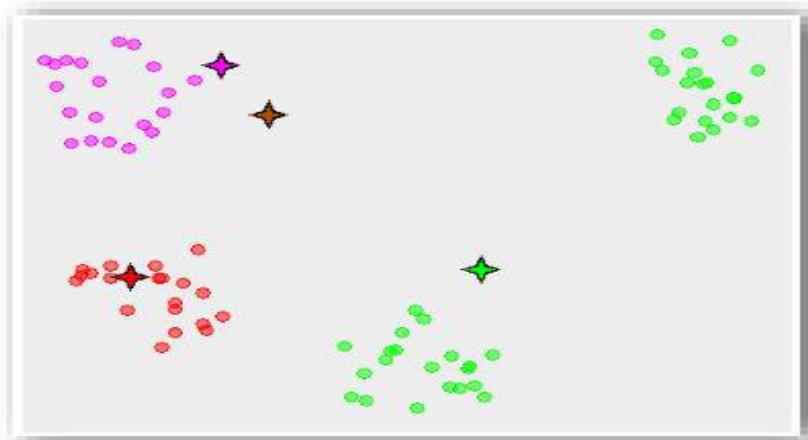
If the data point is closest to the pink cluster center, it is colored pink.



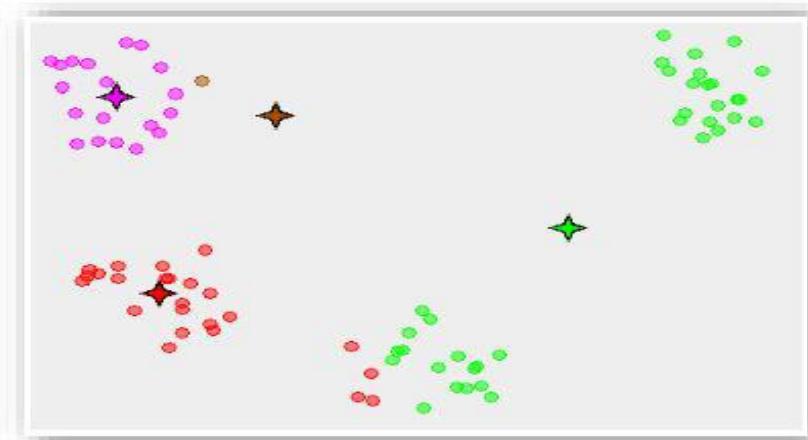
3. Now we calculate the centroid of all the pink points and assign that point as the cluster center for that cluster.

Similarly, we calculate centroids for all the 4 colored(clustered) points and assign the new centroids as the cluster centers.

4. Step-2 and step-3 are run iteratively, unless the cluster centers converge at a point and no longer move.

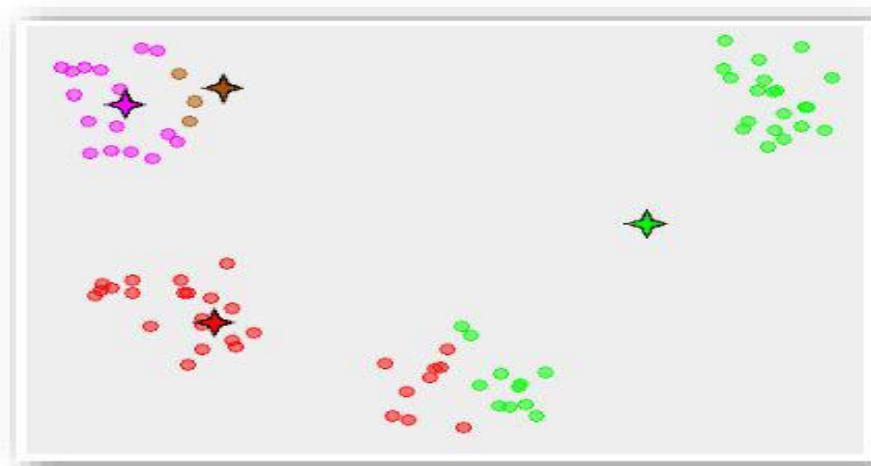


Iteration-1

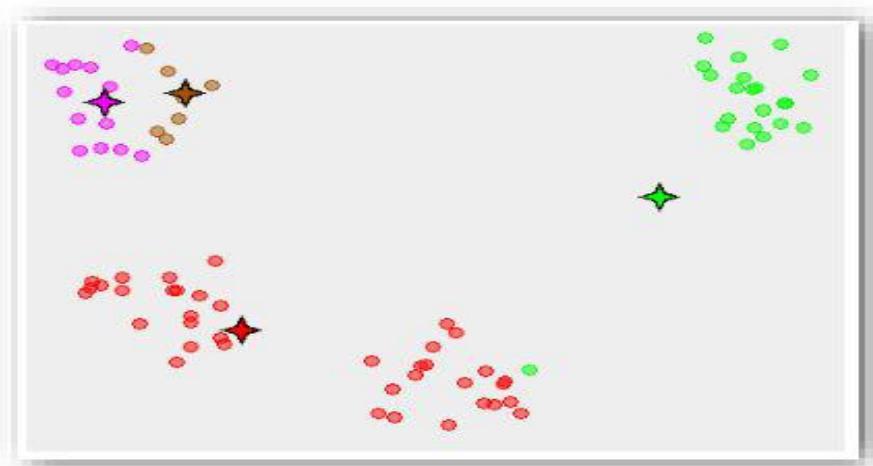


Iteration-2

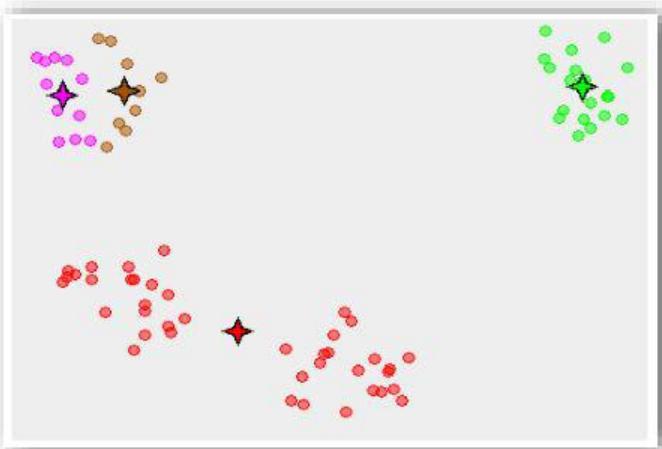
5. We can see that the cluster centers are still not converged so we go ahead and iterate it more.



Iteration-3



Iteration-4



Finally, after multiple iterations, we reach a stage where the cluster centers converge and the clusters look like as:

Here we have performed:

Iterations: 5



# Mathematical Formula

$D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_m\}$  → data set of  $m$  records

$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})$  → each record is an  $n$ -dimensional vector

$$\mathbf{c}_i = \\ \text{cluster}(\mathbf{x}_i) = \arg \min_j \|x_i - \mu_j\|^2$$

$$Distortion = \sum_{i=1}^m (x_i - c_i)^2 = \sum_{j=1}^k \sum_{i \in \text{OwnedBy}(\mu_j)} (x_i - \mu_j)^2$$

*(within cluster sum of squares)*

Owned By(.): set of records that belong to the specified cluster center

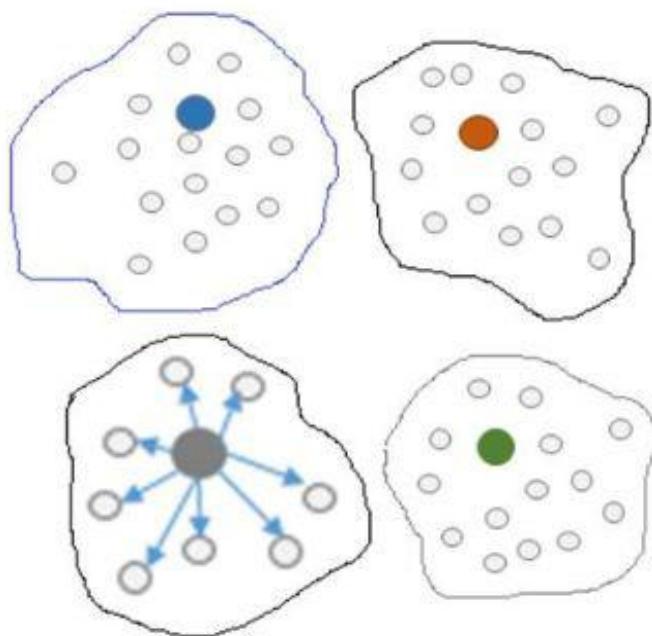


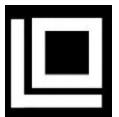
Goal: Find cluster centers that *minimize* Distortion

Solution can be found by setting the partial derivative of Distortion w.r.t. each cluster center to zero

$$\frac{\partial \text{Distortion}}{\partial \mu_j} = \frac{\partial}{\partial \mu_j} \sum_{i \in \text{OwnedBy}(\mu_j)} (x_i - \mu_j)^2 = -2 \sum_{i \in \text{OwnedBy}(\mu_j)} (x_i - \mu_j) = 0 \text{ (for minimum)}$$

$$\Rightarrow \mu_j = \frac{1}{|\text{OwnedBy}(\mu_j)|} \sum_{i \in \text{OwnedBy}(\mu_j)} x_i$$

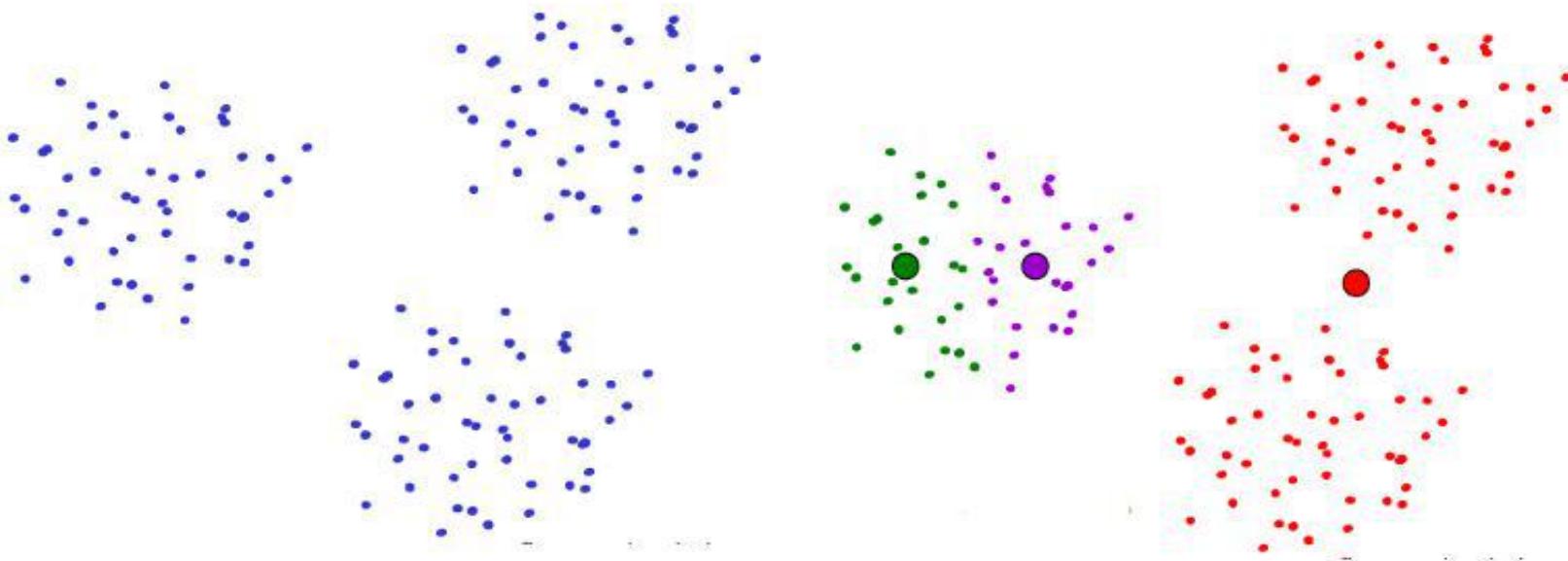


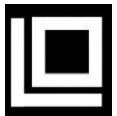


Not necessarily!

We might get stuck in local minimum, and not a global minimum

Try to come up with a converged solution, but does not have minimum distortion:





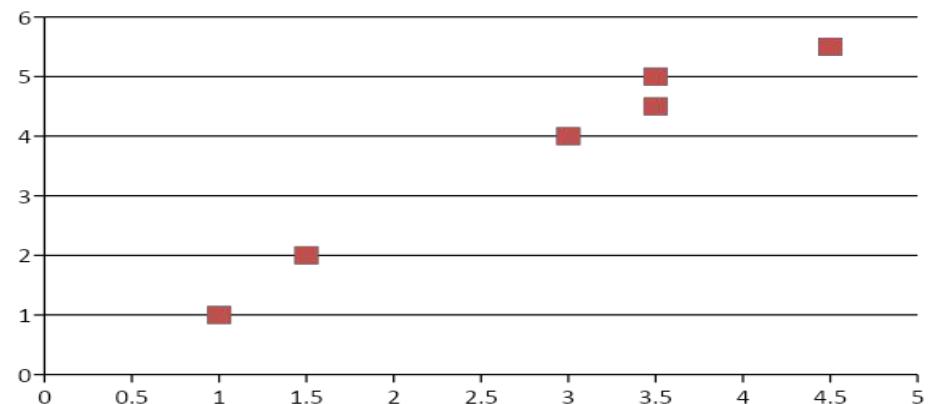
# How to get optimal solution

Idea 1: careful about where we start

- Choose first center at random
- Choose second center that is far away from the first center
- ... Choose  $j^{\text{th}}$  center as far away as possible from the closest of centers 1 through  $(j-1)$

Idea 2: Do many runs of K-means, each with different random starting point

I	X1	X2
A	1	1
B	1.5	2
C	3	4
D	4.5	5.5
E	3.5	5
F	3.5	4.5



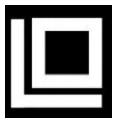
Let's take 2 points as cluster centers for the k=2

We can take point A and point B randomly

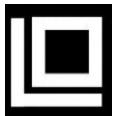
Now find distance of each point from these cluster centers

e.g. Distance of B from A =  $((1.5-1)^2 + (2-1)^2))^{1/2} = 1.118$

Distance of C from A =  $((3-1)^2 + (4-1)^2))^{1/2} = 3.6$  and so on...



i	Cluster1	Cluster2
A	0	1.18
B	1.18	0
C		
D		
E		
F		
G		



# Example 2

Cluster the following eight points (with  $(x, y)$  representing locations) into three clusters:

A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).

The distance function between two points  $a = (x_1, y_1)$  and  $b = (x_2, y_2)$  is defined as-

$$P(a, b) = |x_2 - x_1| + |y_2 - y_1|$$

Use K-Means Algorithm to find the three cluster centers after the second iteration.



Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

From here, New clusters are-



### Cluster-01:

First cluster contains points-

- A1(2, 10)

### Cluster-02:

Second cluster contains points-

- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- A8(4, 9)

### Cluster-03:

Third cluster contains points-

- A2(2, 5)
- A7(1, 2)

Now,

- We re-compute the new cluster clusters.
- The new cluster center is computed by taking mean of all the points contained in that cluster.

### **Cluster-03:**



Third cluster contains points-

- A2(2, 5)
- A7(1, 2)

Now,

- We re-compute the new cluster clusters.
- The new cluster center is computed by taking mean of all the points contained in that cluster.

### **For Cluster-01:**

- We have only one point A1(2, 10) in Cluster-01.
- So, cluster center remains the same.

### **For Cluster-02:**

Center of Cluster-02

$$\begin{aligned} &= ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5) \\ &= (6, 6) \end{aligned}$$

### **For Cluster-03:**

Center of Cluster-03

$$\begin{aligned} &= ((2 + 1)/2, (5 + 2)/2) \\ &= (1.5, 3.5) \end{aligned}$$

This is completion of Iteration-01.



## **Iteration-02:**

- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.

The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-



Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

## K-Means: Hands on

---

- Here we will use k-means clustering to group a set of flowers using their features
- We will be using iris dataset for this purpose

sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	Species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import datasets
import pandas as pd
import numpy as np
```

```
iris = datasets.load_iris()
```

```
X = pd.DataFrame(iris.data)
```

```
X.columns =['sl','sw','pl','pw']
```

```
Y= pd.DataFrame(iris.target)
```

```
Y.columns =[ 'target' ]
```

```
model = KMeans(n_clusters=3)
```

```
model = KMeans(n_clusters=3)
```

```
model.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,  
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',  
       random_state=None, tol=0.0001, verbose=0)
```

```
model.labels_
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
      1, 1, 1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
      0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 2, 2, 2, 2, 0, 2,  
      2, 2, 2, 0, 0, 2, 2, 2, 0, 2, 0, 2, 2, 2, 0, 0, 2, 2, 2, 2,  
      2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0])
```

```
colormap= np.array(['red','blue','green'])
```

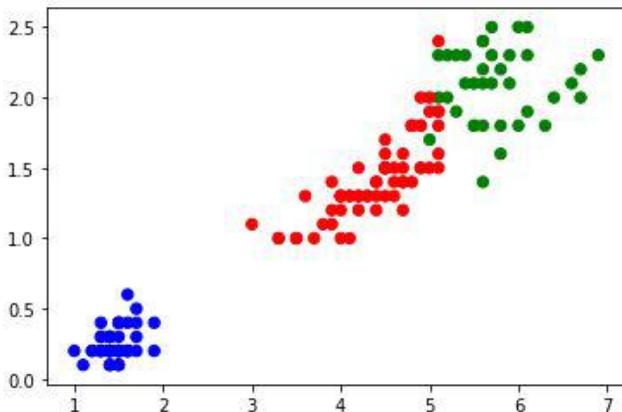
```
plt.scatter(X.pl, X.pw, c = colormap[model.labels_], s=40)
```

```
<matplotlib.collections.PathCollection at 0x1a4c14d3fc8>
```

```
colormap= np.array(['red','blue','green'])
```

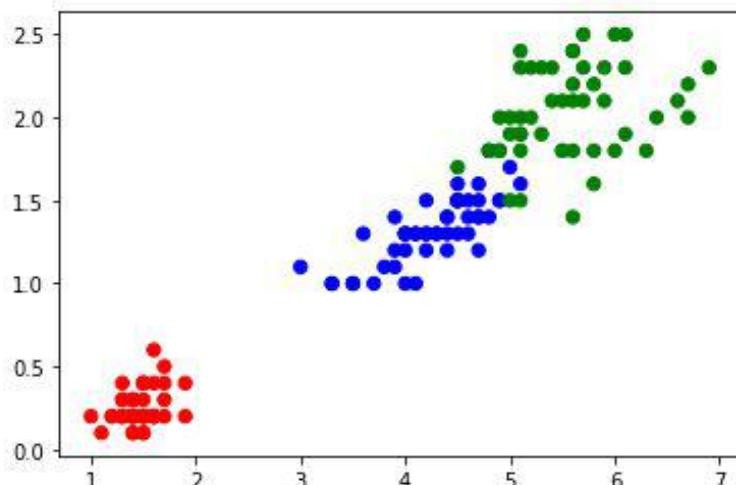
```
plt.scatter(X.pl, X.pw, c = colormap[model.labels_], s=40)
```

```
<matplotlib.collections.PathCollection at 0x1a4c14d3fc8>
```



```
plt.scatter(X.pl, X.pw, c = colormap[Y.target], s=40)
```

```
<matplotlib.collections.PathCollection at 0x1a4c272a848>
```



# Pros and Cons of K-means

---

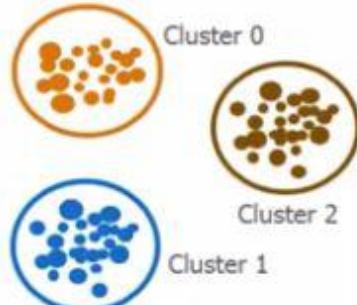
- Pros:
  - Simple, understandable
  - Items automatically assigned to clusters
- Cons:
  - Must define number of clusters
  - All items forced into clusters
  - Unable to handle noisy data and outliers

# Types of Clustering

## Clustering

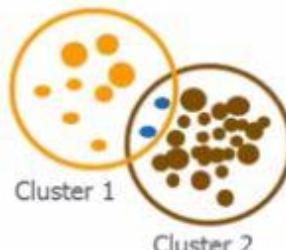
### Exclusive Clustering

Here, an item belongs exclusively to one cluster, not several. K-means does this sort of exclusive clustering.



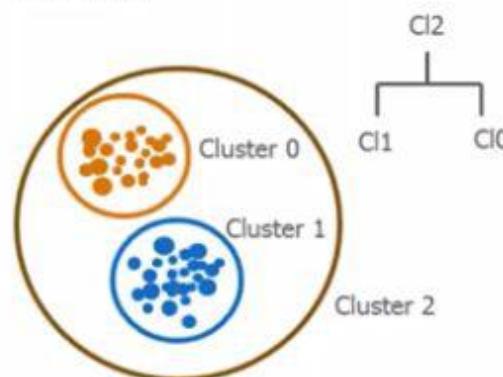
### Overlapping Clustering

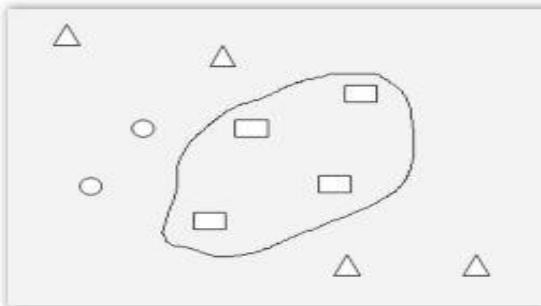
Here, an item can belong to multiple clusters and its degree of association with each cluster is shown. fuzzy/c-means is of this type.



### Hierarchical Clustering

When two cluster have a parent-child relationship or a tree-like structure then it is Hierarchical clustering.

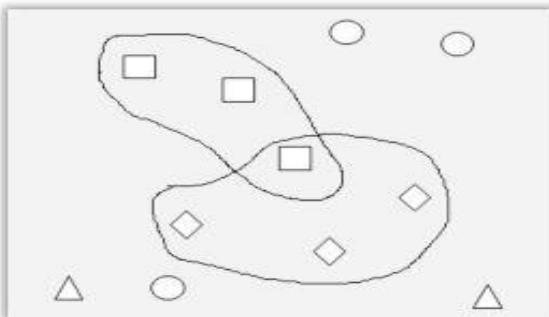




### Exclusive Clustering:

Data is grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it could not be included in another cluster.

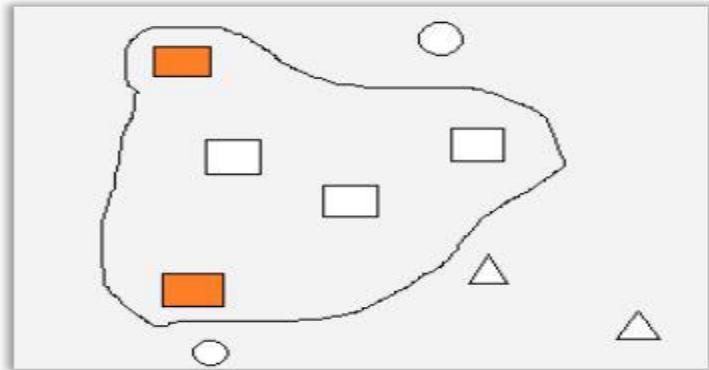
E.g. K-means



### Overlapping Clustering:

The overlapping clustering, uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership.

E.g. Fuzzy C-means



### Hierarchical Clustering:

It is based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster.

There are certain properties which one cluster receives in hierarchy from another cluster.

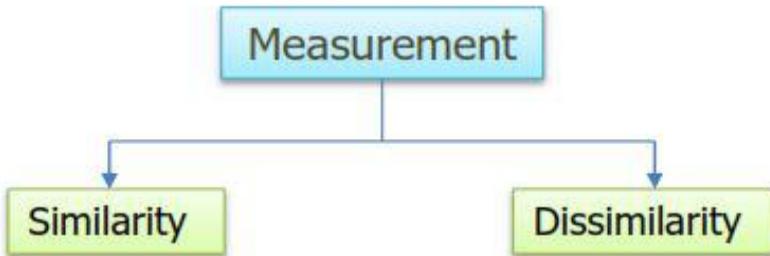
The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another.

It should be noted that an algorithm that is designed for one kind of model has no chance on a data set that contains a radically different kind of model.

For example, k-means cannot find non-convex clusters.

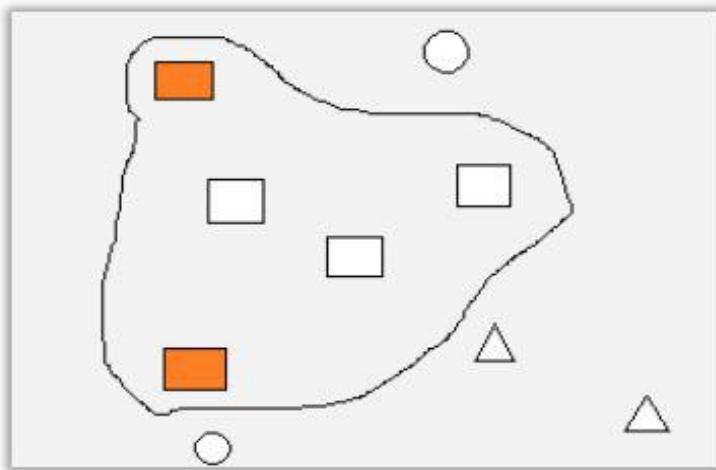
To achieve Clustering, a similarity/dissimilarity measure must be determined so as to cluster the data points based either on :

1. Similarity in the data or
2. Dissimilarity in the data



The measure reflects the degree of closeness or separation of the target objects and should correspond to the characteristics that are believed to distinguish the clusters embedded in the data.

Similarity measures the degree to which a pair of objects are alike.

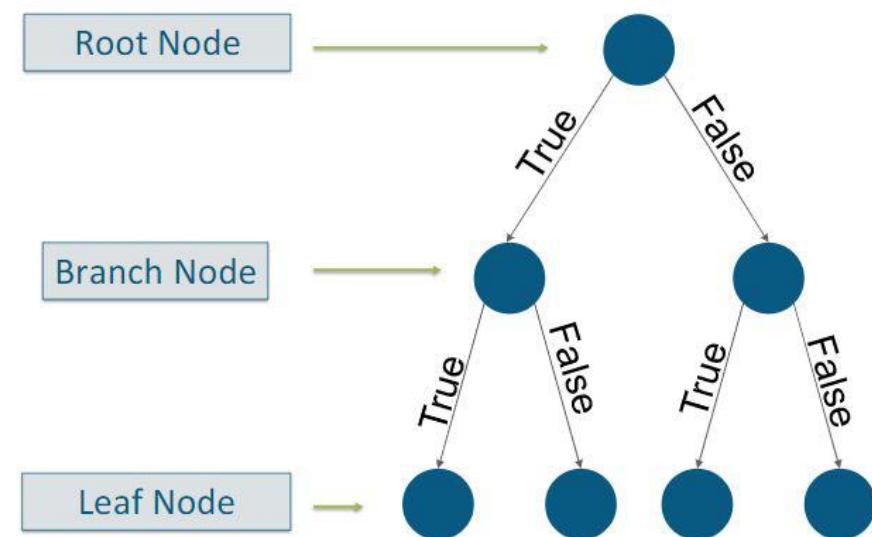


Concerning structural patterns represented as strings or sequences of symbols, the concept of pattern resemblance has typically been viewed from three main perspectives:

- Similarity as matching, according to which patterns are seen as different viewpoints, possible instantiations or noisy versions of the same object;
- Structural resemblance, based on the similarity of their composition rules and primitives;
- Content-based similarity.

# Decision Tree

- A decision tree is a tree-like structure in which internal node represents test on an attribute.
- Each branch represents outcome of test and each leaf node represents class label (decision taken after computing all attributes).
- A path from root to leaf represents classification rules.

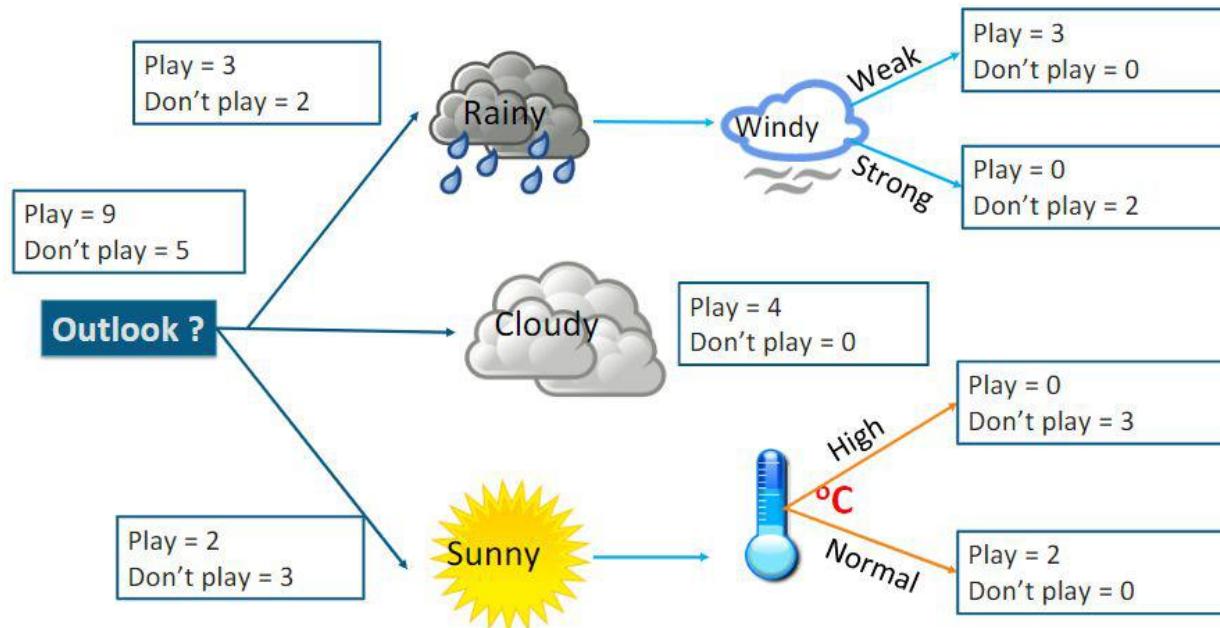




Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

# Decision Tree

A Decision Tree is made from our data by analyzing the variables. From the Decision Tree we can easily find out whether there will be game tomorrow if the conditions are rainy and less windy.



Let's now understand how this decision tree was made



# Building a Decision Tree

From each of the outlooks' we can divide the data as,



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

2 Yes / 3 No

Split further

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

Pure subset  
Yes(will play)

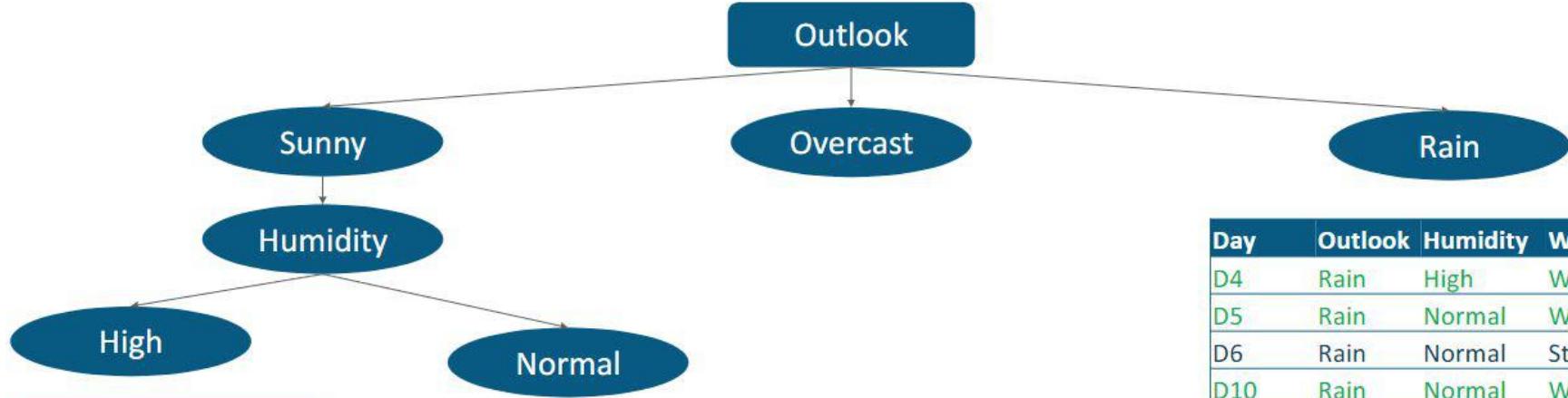
Day	Outlook	Humidity	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

3 Yes / 2 No

Split further

# Building a Decision Tree

We will use Humidity column to split the subset sunny further,



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

Pure subset

NO(will not play)

Day	Humidity	Wind
D9	Normal	Weak
D11	Normal	Strong

Pure subset

Yes(will play)

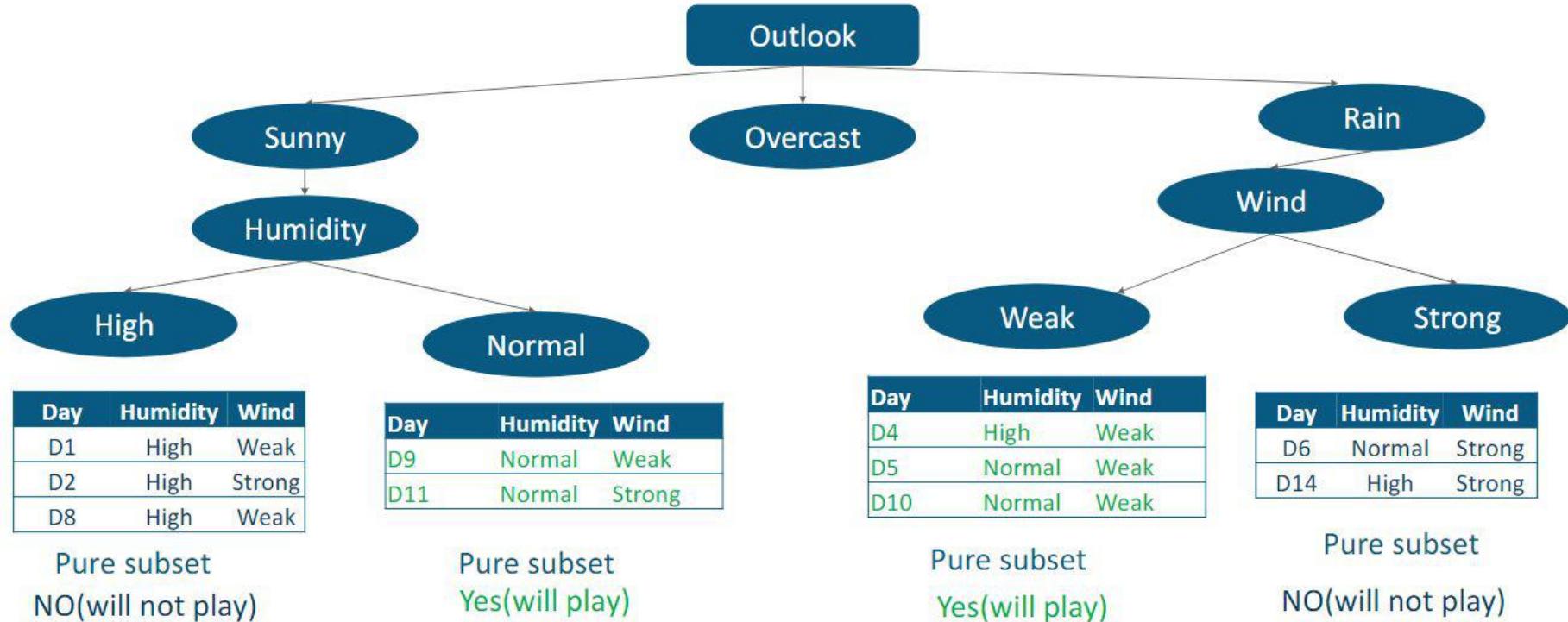
Day	Outlook	Humidity	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

3 Yes / 2 No

Split further

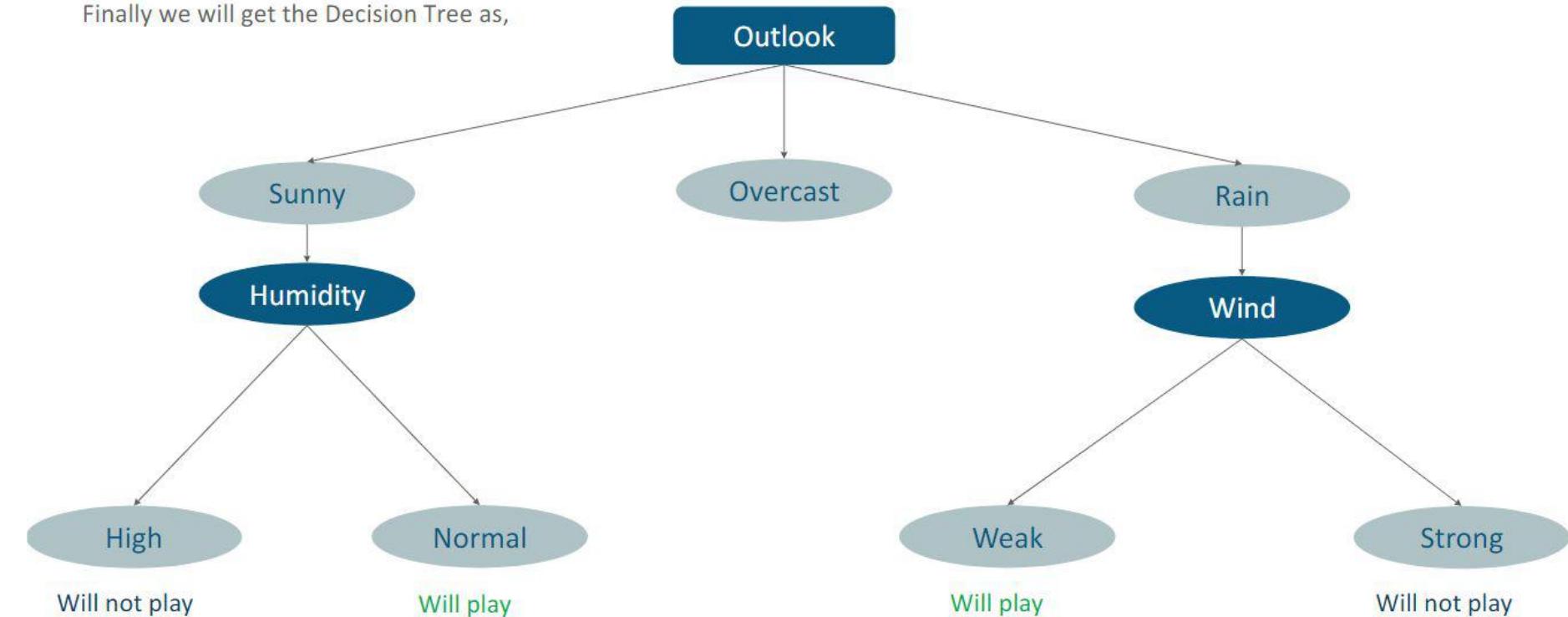
# Building a Decision Tree

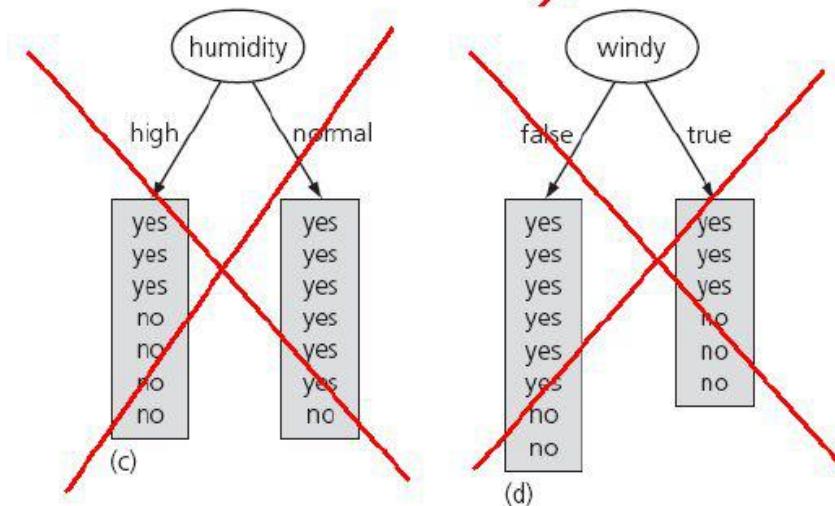
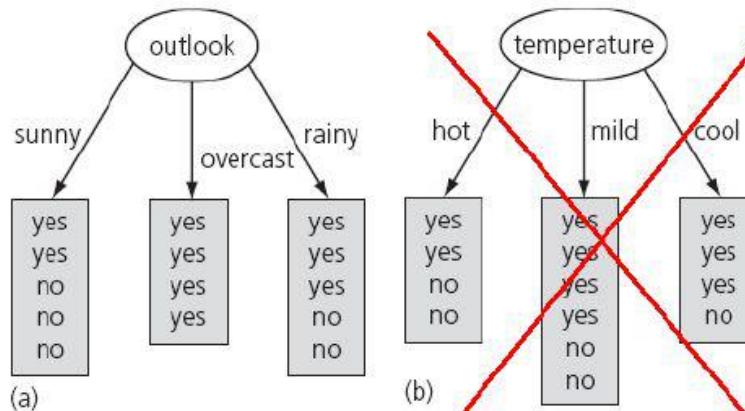
Similarly we will divide rain subset using Humidity,

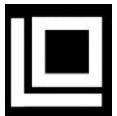


# Building a Decision Tree

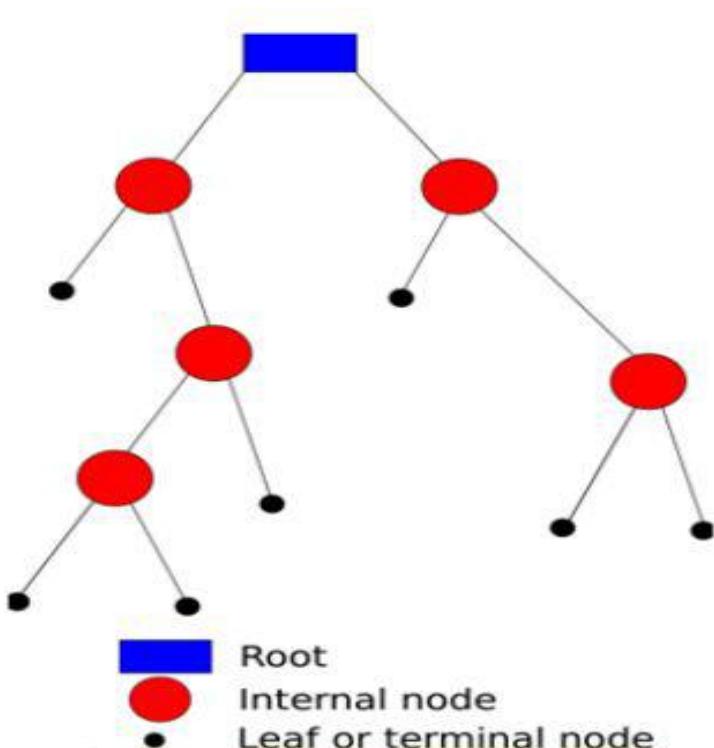
Finally we will get the Decision Tree as,







# How to build?



Use training data to build model Tree generator determines:

- Which variable to split at a node and what will be the value of the split.
- Decision to stop (make a terminal note) or split again has to be made.
- Assign terminal nodes to a label.



# Algorithm

## Basic algorithm (a greedy algorithm)

- Tree is constructed in a top-down recursive divide-and-conquer manner
- At start, all the training examples are at the root
- Attributes are categorical (if continuous-valued, they are discretized in advance)
- Input data is partitioned recursively based on selected attributes
- Test attributes at each node are selected on the basis of a heuristic or statistical measure (e.g., information gain)

## Conditions for stopping partitioning

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
- There are no samples left

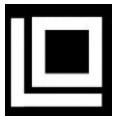


- **Classification by Decision tree**

- the learning of decision trees from class-labeled training instances.

- A **decision tree** is a flowchart-like tree structure, where

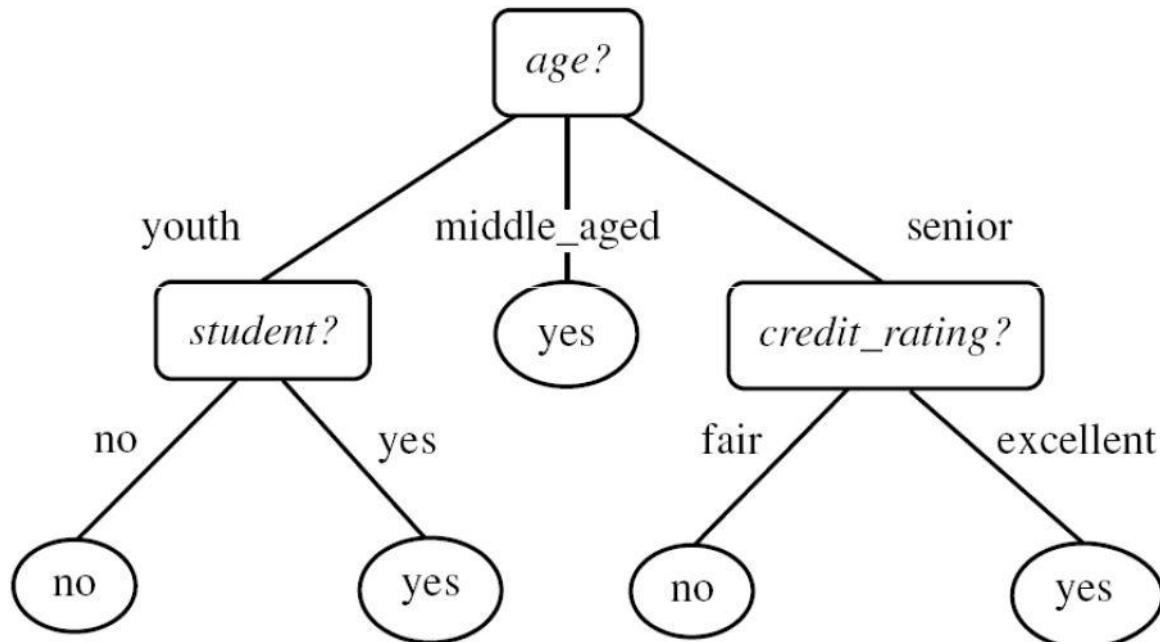
- each **internal node** (non-leaf node) denotes a test on an attribute
  - each **branch** represents an outcome of the test
  - each **leaf node** (or *terminal node*) holds a class label.
  - The topmost node in a tree is the **root node**.



# Training Dataset

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

# Sample output





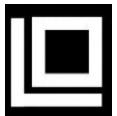
# ATTRIBUTE SELECTION MEASURES



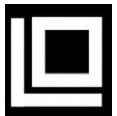
- Which is the best attribute?
  - Want to get the smallest tree
  - choose the attribute that produces the “purest” nodes
- Attribute selection measure
  - a heuristic for selecting the splitting criterion that “best” separates a given data partition,  $D$ , of class-labeled training instances into individual classes.
  - If we were to split  $D$  into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be **pure** (i.e., all of the instances that fall into a given partition would belong to the same class).



- Attribute selection measures are also known as **splitting rules** because they determine how the instances at a given node are to be split.
- The attribute selection measure provides a **ranking** for each attribute describing the given training instances.
- The attribute having the best score for the measure is chosen as the **splitting attribute** for the given instances.



- If the splitting attribute is continuous-valued or if we are restricted to binary trees then, respectively, either a **split point** or a **splitting subset** must also be determined as part of the splitting criterion.
- Three popular attribute selection measures:
  - Information gain
  - Gain ratio
  - Gini index

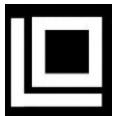


- The notation used herein is as follows.

- Let  $D$ , the data partition, be a training set of class-labeled instances.
- Suppose the class label attribute has  $m$  distinct values defining  $m$  distinct classes,  $C_i$  (for  $i = 1, \dots, m$ )
- Let  $C_{i,D}$  be the set of instances of class  $C_i$  in  $D$ .
- Let  $|D|$  and  $|C_{i,D}|$  denote the number of instances in  $D$  and  $C_{i,D}$ , respectively.



# INFORMATION GAIN



- Select the attribute with the highest **information gain** as the splitting attribute
- This attribute minimizes the information needed to classify the instances in the resulting partitions and reflects the **least impurity** in these partitions.
- **ID3** uses **information gain** as its attribute selection measure.
- ***Entropy (impurity)***
  - *High Entropy* means X is from a uniform (boring) distribution
  - *Low Entropy* means X is from a varied (peaks and valleys) distribution



- Need a measure of node impurity:

C0: 5
C1: 5

**Non-homogeneous,**  
**High degree of impurity**

C0: 9
C1: 1

**Homogeneous,**  
**Low degree of impurity**



- Let  $p_i$  be the probability that an arbitrary instance in  $D$  belongs to class  $C_i$ , estimated by  $|C_i, D|/|D|$
- **Expected information (entropy)** needed to classify an instance in  $D$  is given by:

$$Info(D) = - \sum_{i=1}^m p_i \log_2(p_i)$$

- $Info(D)$  (entropy of  $D$ )
  - the average amount of information needed to identify the class label of an instance in  $D$ .
  - The smaller information required, the greater the purity.



- At this point, the information we have is based solely on the **proportions of instances of each class**.
- A log function to the base 2 is used, because the information is encoded in bits (It is measured in bits).



- Need a measure of node impurity:

C0: 5  
C1: 5

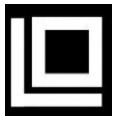
Non-homogeneous,  
High degree of impurity

$$Info(D) = 1$$

C0: 9  
C1: 1

Homogeneous,  
Low degree of impurity

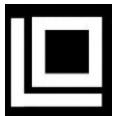
$$Info(D) = 0.469$$



- Suppose attribute A can be used to split D into v partitions or subsets,  $\{D_1, D_2, \dots, D_v\}$ , where  $D_j$  contains those instances in D that have outcome  $a_j$  of A.
- Information needed (after using A to split D) to classify D:

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

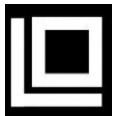
- The smaller the expected information (still) required, the greater the purity of the partitions.



- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

- Information gain increases with the average purity of the subsets
- Information gain: information needed before splitting – information needed after splitting
  - The attribute that has the highest information gain among the attributes is selected as the splitting attribute.



This table presents a training set,  $D$ .

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

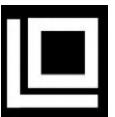


- The class label attribute, *buys\_computer*, has two distinct values (namely, {yes, no}); therefore, there are two distinct classes (that is,  $m = 2$ ).
- Let class  $C1$  correspond to *yes* and class  $C2$  correspond to *no*.
- The expected information needed to classify an instance in  $D$ :

$$Info(D) = I(9,5) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$



- Next, we need to compute the expected information requirement for each attribute.
- Let's start with the attribute *age*. We need to look at the distribution of *yes* and *no* instances for each category of *age*.
  - For the age category *youth*,
    - ◆ there are two *yes* instances and three *no* instances.
  - For the category *middle\_aged*,
    - ◆ there are four *yes* instances and zero *no* instances.
  - For the category *senior*,
    - ◆ there are three *yes* instances and two *no* instances.

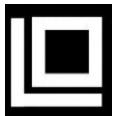


- The expected information needed to classify an instance in  $D$  if the instances are partitioned according to  $age$  is

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2)$$

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14} \times \left( -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) \\ &\quad + \frac{4}{14} \times \left( -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} \right) \\ &\quad + \frac{5}{14} \times \left( -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right) \\ &= 0.694 \text{ bits.} \end{aligned}$$

age	Class: buys_computer
youth	no
youth	no
middle_aged	yes
senior	yes
senior	yes
senior	no
middle_aged	yes
youth	no
youth	yes
senior	yes
youth	yes
middle_aged	yes
middle_aged	yes
senior	no



- The gain in information from such a partitioning would be

$$Gain(\text{age}) = \text{Info}(D) - \text{Info}_{\text{age}}(D) = 0.940 - 0.694 = 0.246 \text{ bits}$$

- Similarly

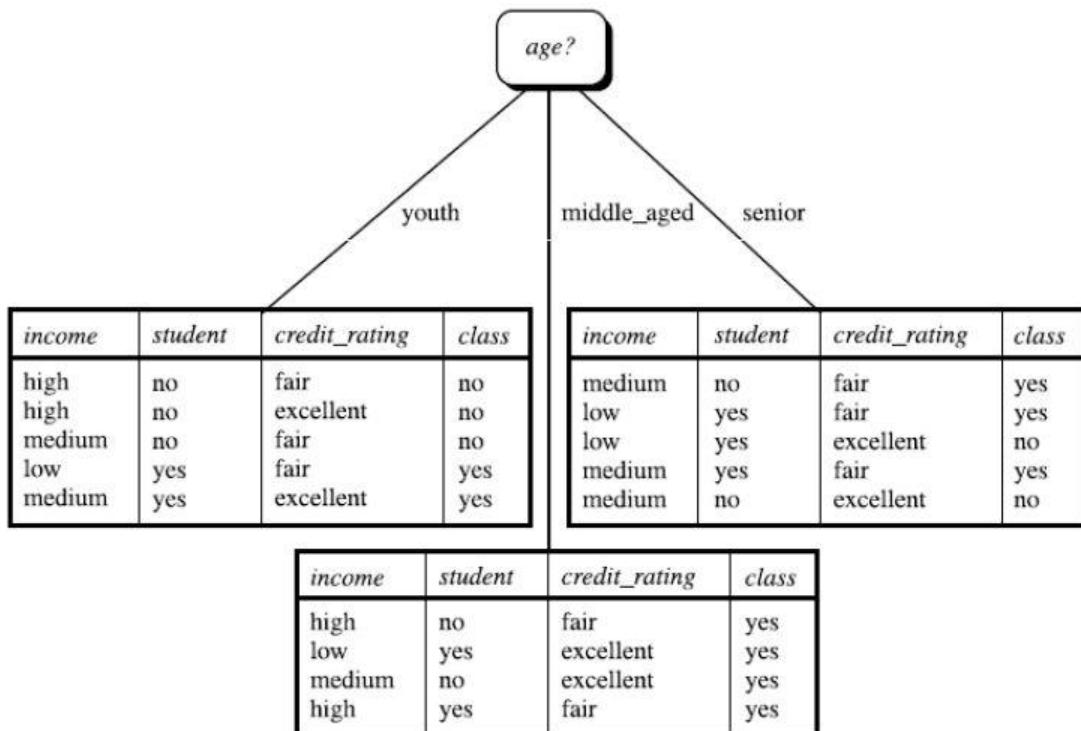
$$Gain(\text{income}) = 0.029$$

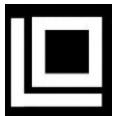
$$Gain(\text{student}) = 0.151$$

$$Gain(\text{credit\_rating}) = 0.048$$

- Because *age* has the highest information gain among the attributes, it is selected as the splitting attribute.

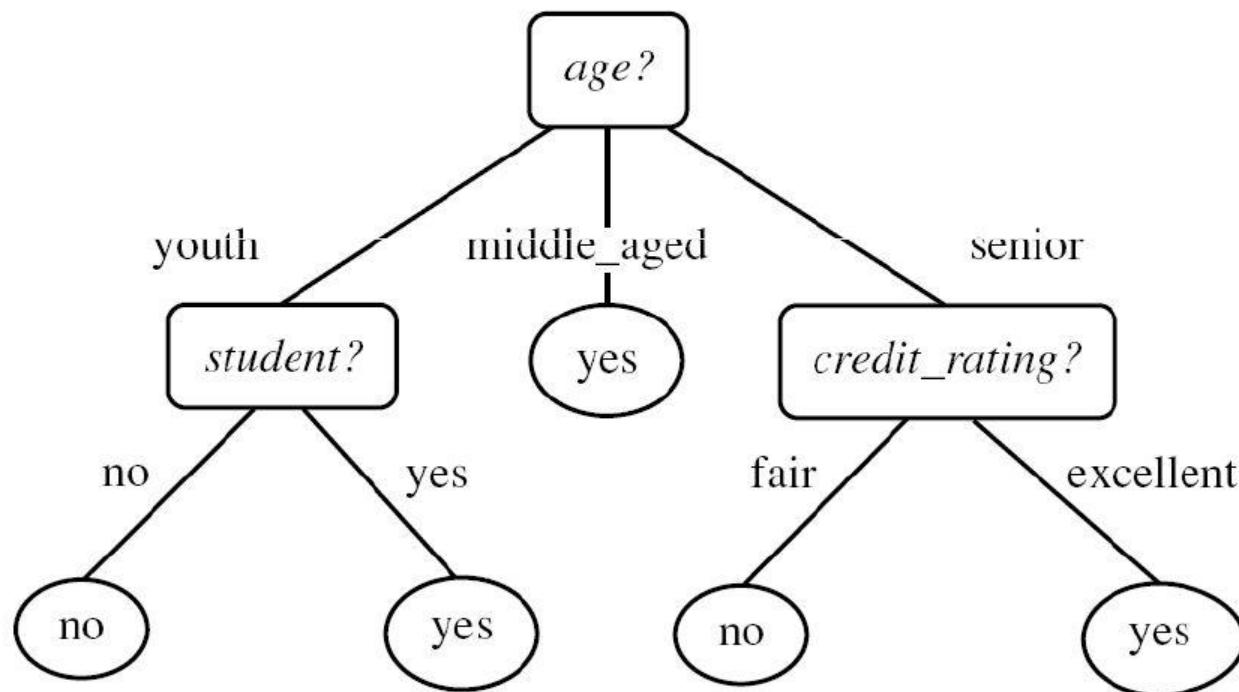
- Branches are grown for each outcome of *age*. The instances are shown partitioned accordingly.

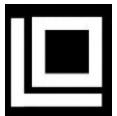




- Notice that the instances falling into the partition for  $age = middle\_aged$  all belong to the same class.
- Because they all belong to class “yes,” a leaf should therefore be created at the end of this branch and labeled with “yes.”

- The final decision tree returned by the algorithm





- Problem of information gain

- When there are attributes with a large number of values
- Information gain measure is biased towards attributes with a large number of values
- This may result in selection of an attribute that is non-optimal for prediction



- *Gain ratio*

- a modification of the information gain
  - C4.5 uses gain ratio to overcome the problem

- Gain ratio applies a kind of normalization to information gain using a **split information**

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right)$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

- The attribute with the maximum gain ratio is selected as the splitting attribute.

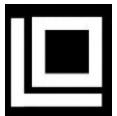


- Example

- Computation of gain ratio for the attribute *income*.
- A test on *income* splits the data into three partitions, namely *low*, *medium*, and *high*, containing four, six, and four instances, respectively.
- Computation of the gain ratio of *income*:

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926$$

- Gain(income) = 0.029
- GainRatio(income) = 0.029/0.926 = 0.031



- Gini index
  - is used in CART algorithm.
  - measures the impurity of  $D$
  - considers a binary split for each attribute.
- If a data set  $D$  contains examples from  $m$  classes, gini index,  $gini(D)$  is defined as

$$gini(D) = 1 - \sum_{i=1}^m p_i^2$$

- where  $p_i$  is the relative frequency of class  $i$  in  $D$



- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition.
- If a data set  $D$  is split on A into two subsets  $D_1$  and  $D_2$ , the *gini* index  $gini(D)$  is defined as

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

- First we calculate Gini index for all subsets of an attribute, then the subset that gives the minimum Gini index for **that attribute** is selected.



- The reduction in impurity that would be incurred by a binary split on attribute  $A$  is

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute.



- Example:

- $D$  has 9 instances in `buys_computer = "yes"` and 5 in `"no"`
  - The impurity of  $D$ :

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- the attribute *income* partitions:
    - ◆ {low, medium} & {high}
    - ◆ {low, high} & {medium}
    - ◆ {low} & {medium, high}

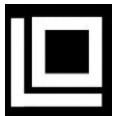


- Example:

- Suppose the attribute *income* partitions  $D$  into 10 in  $D_1$ : {low, medium} and 4 in  $D_2$

$$\begin{aligned}Gini_{income \in \{low,medium\}}(D) \\&= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\&= \frac{10}{14} \left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14} \left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\&= 0.450 \\&= Gini_{income \in \{high\}}(D).\end{aligned}$$

- Similarly, the Gini index values for splits on the remaining subsets are:
  - ◆ For {low, high} and {medium} is 0.315
  - ◆ For {low} and {medium, high} is 0.300



- The attribute *income* and splitting subsets  $\{\text{low}\}$  and  $\{\text{medium}, \text{high}\}$  *and* give the minimum Gini index overall, with a reduction in impurity of:

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

$$\Delta Gini(\text{income}) = 0.459 - 0.300 = 0.159$$

- Now we should calculate  $\Delta Gini$  for other attributes including age, student, and credit rate.
- Then we can choose the best attribute for splitting.

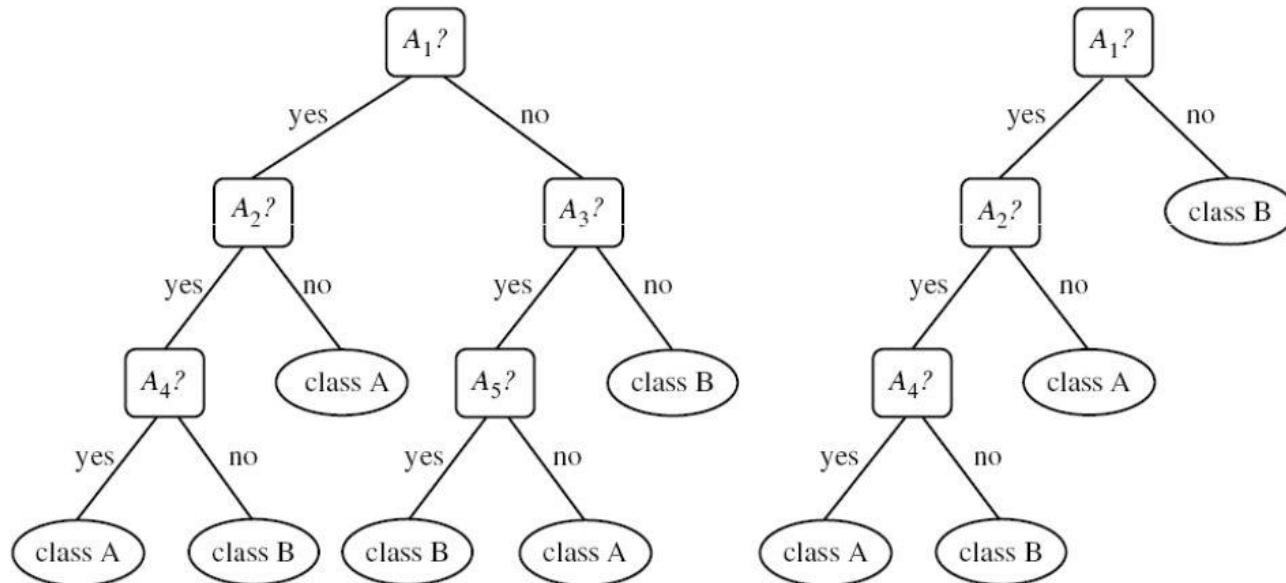


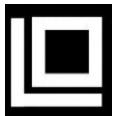
- The three measures, in general, return good results but
  - Information gain:
    - ◆ biased towards multivalued attributes
  - Gain ratio:
    - ◆ tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - ◆ biased to multivalued attributes
    - ◆ has difficulty when # of classes is large



- **Overfitting:** An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- **Tree Pruning**
  - To prevent overfitting to noise in the data
  - Pruned trees tend to be smaller and less complex and, thus, easier to comprehend.
  - They are usually faster and better at correctly classifying independent test data.

- An unpruned decision tree and a pruned version of it.

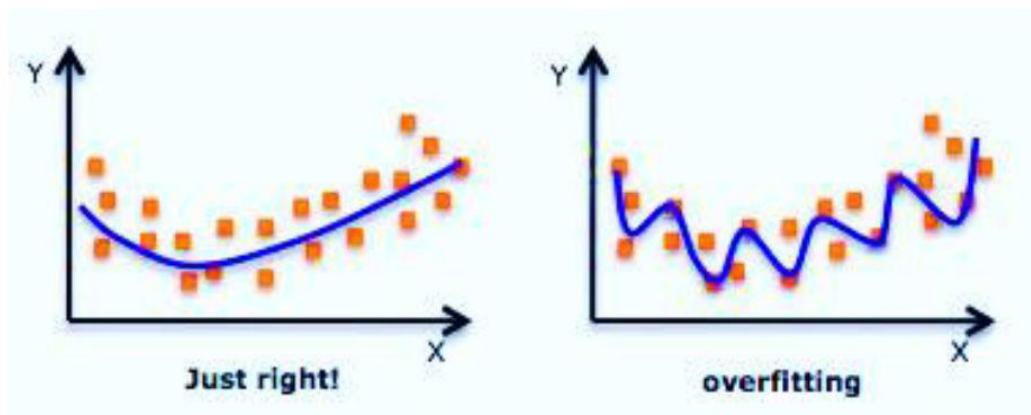




# Overfitting and tree pruning

Overfitting: An induced tree may overfit the training data

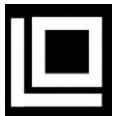
- Too many branches, some may reflect anomalies due to noise or outliers
- Poor accuracy for unseen samples





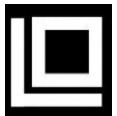
## Two approaches to avoid overfitting

- Prepruning
  - ◆ stop growing a branch when information becomes unreliable
- Postpruning
  - ◆ take a fully-grown decision tree and remove unreliable branches
  - ◆ Postpruning preferred in practice



# Prepruning

- Based on statistical significance test
  - Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
- Most popular test: *chi-squared test*
- ID3 used chi-squared test in addition to information gain
  - Only statistically significant attributes were allowed to be selected by information gain procedure



# Postpruning

- Postpruning: first, build full tree & Then, prune it
- Two pruning operations:
  - *Subtle replacement*
  - *Subtree raising*
- Possible strategies: error estimation and significance testing



# NAÏVE BAYE'S



Statistics and Probability are two related but separate academic disciplines.

Statistical analysis often uses probability distributions, and the two topics are often studied together. Now we will discuss about probability.





# Probability

- Probability is the measure of how likely something will occur.
- It is the ratio of desired outcomes to total outcomes.

$$(\# \text{ desired}) / (\# \text{ total})$$

- Probabilities of all outcomes sums to 1.



Example:

- ✓ If I roll a dice, there are six total possibilities. (1,2,3,4,5,6)
- ✓ Each possibility only has one outcome, so each has a PROBABILITY of 1/6.
- ✓ For instance, the probability of getting a numeric 2 is 1/6, since there is only a single 2 on the dice.



# Bayes Theorem

- Bayes' theorem (also known as Bayes' rule) is a useful tool for calculating conditional probabilities.

*Bayes' theorem :*

$$P(A | B) = \frac{P(A \text{ and } B)}{P(B)}$$

- For independent events  $P(A | B) = P(A)$ , so by rearranging the formula we can see,

$$P(A \text{ and } B) = P(A) \times P(B)$$



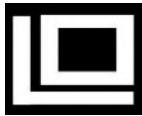
# Probability Distribution

- A probability distribution assigns a probability to each measurable subset of the possible outcomes of a random experiment

One Toss	Head	Tail	Two tosses	Head-Head	Tail-Tail	Head-Tail	Tail-Head
Probability	0.5	0.5	Probability	0.25	0.25	0.25	0.25

- Rules:

1. The outcomes listed must be disjoint
2. Each probability must be between 0 and 1
3. The probabilities must sum to 1



Consider a school with a total population of 100 persons. These 100 persons can be seen either as 'Students' and 'Teachers' or as a population of 'Males' and 'Females'.

With below tabulation of the 100 people, what is the conditional probability that a certain member of the school is a 'Teacher' given that he is a 'Man'?

	Female	Male	Total
Teacher	8	12	20
Student	32	48	80
Total	40	60	100



$$P(\text{Teacher} \mid \text{Male}) = \frac{P(\text{Teacher} \cap \text{Male})}{P(\text{Male})} = 12/60 = 0.2$$

This can be represented as the intersection of Teacher (A) and Male (B) divided by Male (B). Likewise, the conditional probability of B given A can be computed. The Bayes Rule that we use for Naive Bayes, can be derived from these two notations.

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \quad (1)$$

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} \quad (2)$$



Bayes Rule is a way to go from  $P(X | Y)$  to find  $P(Y | X)$

$$P(X | Y) = \frac{P(X \cap Y)}{P(Y)}$$

*Known*

1

$$P(Y | X) = \frac{P(X \cap Y)}{P(X)}$$

*Unknown*

2

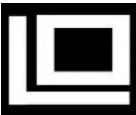
$P(\text{Evidence} | \text{Outcome})$   
(Known from training data)



$P(\text{Outcome} | \text{Evidence})$   
(To be predicted for test data)

**Bayes** Rule

$$P(Y | X) = \frac{P(X | Y) * P(Y)}{P(X)}$$



When there are multiple X variables, we simplify it by  
assuming the X's are independent, so the **Bayes** rule

$$P(Y=k | X) = \frac{P(X | Y=k) * P(Y=k)}{P(X)}$$

where, k is a class of Y

becomes, Naive **Bayes**

$$P(Y=k | X_1..X_n) = \frac{P(X_1 | Y=k) * P(X_2 | Y=k) ... * P(X_n | Y=k) * P(Y=k)}{P(X_1) * P(X_2) ... * P(X_n)}$$



$$P(Y=k | X_1..X_n) = \frac{P(X_1 | Y=k) * P(X_2 | Y=k) ... * P(X_n | Y=k) * P(Y=k)}{P(X_1) * P(X_2) ... * P(X_n)}$$

can be understood as ..

$$\text{Probability of Outcome I Evidence (Posterior Probability)} = \frac{\text{Probability of Likelihood of evidence} * \text{Prior}}{\text{Probability of Evidence}}$$



Probability of Evidence is same for all classes of Y



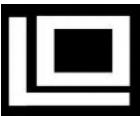
# Problem

- Say you have 1000 fruits which could be either ‘banana’, ‘orange’ or ‘other’. These are the 3 possible classes of the Y variable. We have data for the following X variables, all of which are binary (1 or 0).
  - Long
  - Sweet
  - Yellow



Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

So the objective of the classifier is to predict if a given fruit is a ‘Banana’ or ‘Orange’ or ‘Other’ when only the 3 features (long, sweet and yellow) are known.



- Let's say you are given a fruit that is: Long, Sweet and Yellow, can you predict what fruit it is?
- This is the same of predicting the Y when only the X variables in testing data are known.
- Let's solve it by hand using Naive Bayes. The idea is to compute the 3 probabilities, that is the probability of the fruit being a banana, orange or other. Whichever fruit type gets the highest probability wins.



## Step 1: Compute the ‘Prior’ probabilities for each of the class of fruits.

That is, the proportion of each fruit class out of all the fruits from the population.

You can provide the ‘Priors’ from prior information about the population. Otherwise, it can be computed from the training data. For this case, let’s compute from the training data. Out of 1000 records in training data, you have 500 Bananas, 300 Oranges and 200 Others.

So the respective priors are 0.5, 0.3 and 0.2.

$$P(Y=\text{Banana}) = 500 / 1000 = 0.50$$

$$P(Y=\text{Orange}) = 300 / 1000 = 0.30$$

$$P(Y=\text{Other}) = 200 / 1000 = 0.20$$



## **Step 2: Compute the probability of evidence that goes in the denominator.**

This is nothing but the product of P of Xs for all X.

This is an optional step because the denominator is the same for all the classes and so will not affect the probabilities.

$$P(x_1=\text{Long}) = 500 / 1000 = 0.50$$

$$P(x_2=\text{Sweet}) = 650 / 1000 = 0.65$$

$$P(x_3=\text{Yellow}) = 800 / 1000 = 0.80$$



## Step 3: Compute the probability of likelihood of evidences that goes in the numerator.

It is the product of conditional probabilities of the 3 features.

If you refer back to the formula, it says  $P(X_1 | Y=k)$ .

Here  $X_1$  is 'Long' and  $k$  is 'Banana'.

That means the probability the fruit is 'Long' given that it is a Banana.

In the above table, you have 500 Bananas. Out of that 400 is long.

So,  $P(\text{Long} | \text{Banana}) = 400/500 = 0.8$ .

Here, I have done it for Banana alone.



## Probability of Likelihood for Banana

$$P(x_1=\text{Long} \mid Y=\text{Banana}) = 400 / 500 = 0.80$$

$$P(x_2=\text{Sweet} \mid Y=\text{Banana}) = 350 / 500 = 0.70$$

$$P(x_3=\text{Yellow} \mid Y=\text{Banana}) = 450 / 500 = 0.90.$$

So, the overall probability of Likelihood of evidence for Banana =  $0.8 * 0.7 * 0.9 = 0.504$



Step 4: If a fruit is 'Long', 'Sweet' and 'Yellow', what fruit is it?

$$P(\text{Banana} \mid \text{Long, Sweet and Yellow}) = \frac{P(\text{Long} \mid \text{Banana}) * P(\text{Sweet} \mid \text{Banana}) * P(\text{Yellow} \mid \text{Banana}) * P(\text{banana})}{P(\text{Long}) * P(\text{Sweet}) * P(\text{Yellow})}$$

$$= \frac{0.8 * 0.7 * 0.9 * 0.5}{P(\text{Evidence})} = 0.252 / P(\text{Evidence})$$

$$P(\text{Orange} \mid \text{Long, Sweet and Yellow}) = 0, \text{ because } P(\text{Long} \mid \text{Orange}) = 0$$

$$P(\text{Other Fruit} \mid \text{Long, Sweet and Yellow}) = 0.01875 / P(\text{Evidence})$$

Answer: Banana - Since it has highest probability amongst the 3 classes



Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

**Outlook**

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
<b>Total</b>	<b>9</b>	<b>5</b>	<b>100%</b>	<b>100%</b>

**Temperature**

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
<b>Total</b>	<b>9</b>	<b>5</b>	<b>100%</b>	<b>100%</b>

**Humidity**

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
<b>Total</b>	<b>9</b>	<b>5</b>	<b>100%</b>	<b>100%</b>

**Wind**

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
<b>Total</b>	<b>9</b>	<b>5</b>	<b>100%</b>	<b>100%</b>

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
<b>Total</b>	<b>14</b>	<b>100%</b>

For example, probability of playing golf given that the temperature is cool,

i.e  $P(\text{temp.} = \text{cool} | \text{play golf} = \text{Yes}) = 3/9$ .

Also, we need to find class probabilities ( $P(y)$ ) which has been calculated in the table 5.

For example,  $P(\text{play golf} = \text{Yes}) = 9/14$ .



today = (Sunny, Hot, Normal, False)

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

$$P(Yes|today) \propto \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} \approx 0.0141$$

$$P(No|today) \propto \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{5}{14} \approx 0.0068$$



These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

So, prediction that golf would be played is 'Yes'.

# Association Rule Mining

- Association rule mining is a method for discovering interesting relations between variables in large databases
- Pattern that states when an event occurs, one more event occurs with a certain probability in parallel



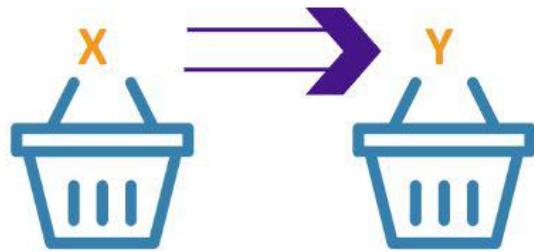
*Customers who purchase a keyboard have 60% likelihood of also purchasing a mouse for their PC as well*



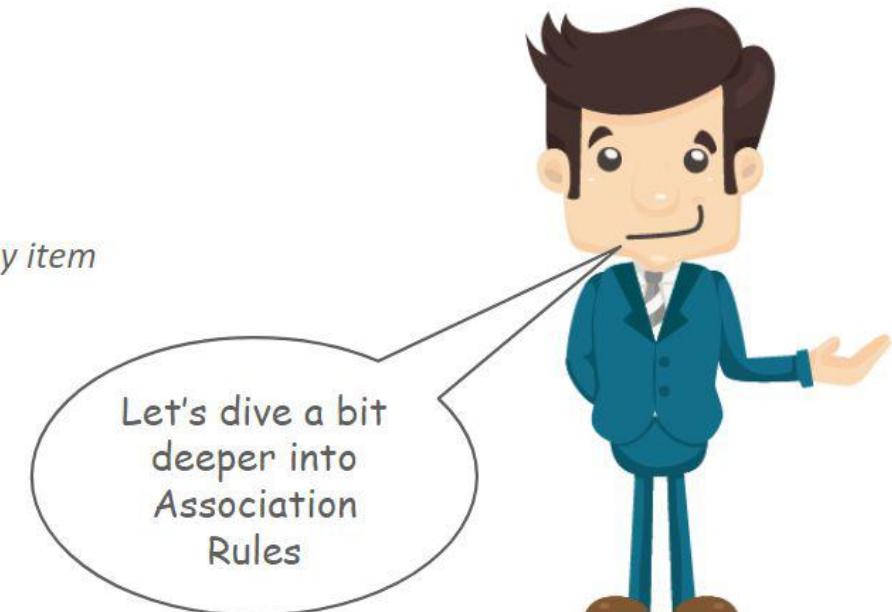
# Association Rule Mining

---

An Example of association rule is given below,



*It means that if a person buys item X then he will also buy item Y*



# Association Rule Mining: Parameters

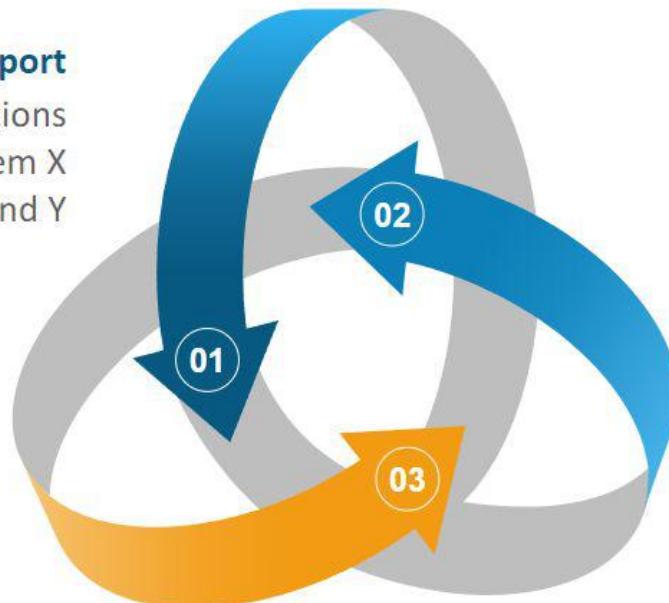
Association rule mining takes care of the following parameters:

## Support

Gives fraction of transactions which contains the item X and Y

## Lift

Lift indicates the strength of a rule over the random co-occurrence of X and Y



## Confidence

Gives how often the items X and Y occurs together, given no. of times X occurs

# Calculating Support, Confidence & Lift

“

Support = no. of times item X occurred / Total number of transactions =

$$(X \cup Y) = \frac{\sigma(x \cup y)}{N}$$



$\sigma$  = support

“

Confidence = no. of times item X & Y occurred / Total occurrence of X =

$$(Y | X) = \frac{\sigma(x \cup y)}{\sigma(x)}$$

Pr



“

Lift = no. of times item X & Y occurred / Total occurrence of X multiplied by Total occurrence of Y =

$$\frac{\sigma(x \cup y)}{\sigma(x) \times \sigma(y)}$$



**Goal:** Find all rules with user-specified **minimum support** (minsup) and **minimum confidence** (minconf)

# Association Rule Mining

---

- Lets take an example,
- Suppose we have five transactions T1,T2,T3,T4,T5 as given below:

T1 : A, B, C

T2 : A, C, D

T3 : B, C, D

T4 : A, D, E

T5 : B, C, E

- Here,

- ❖ A,B,C,D,E are items in a store,  $I = \{A, B, C, D, E\}$
- ❖ Set of all transactions  $T = \{T1, T2, T3, T4, T5\}$
- ❖ Each transaction is a set of items,  $T \subseteq I$



# Association Rule Mining

- Suppose, you made some association rules using our transaction database as given below:

$$A \Rightarrow D$$

$$C \Rightarrow A$$

$$A \Rightarrow C$$

$$B \& C \Rightarrow D$$



- Now we can find support, confidence and lift for these rules using the formula explained earlier:

Rule	Support	Confidence	Lift
$A \Rightarrow D$	2/5	2/3	2/9
$C \Rightarrow A$	2/5	2/4	1/6
$A \Rightarrow C$	2/5	2/3	1/6
$B \& C \Rightarrow D$	1/5	1/3	1/9

# Apriori Algorithm

Uses frequent itemsets to generate association rules,

*"A subset of a frequent itemset must also be a frequent itemset"*



**Note:** *Frequent Itemset*: Support value > Threshold value

# Apriori Algorithm

Consider the following transaction Dataset. We are using a threshold value = 2 for this example:

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

*Minimum support count = 2*

First step is to build a list of itemsets of size one using this dataset



# Apriori Algorithm – First Iteration

List of itemsets of size one is made. Also, its support values are calculated

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

Itemset	Support
{1}	3
{2}	3
{3}	4
{4}	1
{5}	4

C11

Itemset	Support
{1}	3
{2}	3
{3}	4
{5}	4

F11

Since, our threshold value is 2, any itemsets with support less than it are omitted.

# Apriori Algorithm – Second Iteration

- In this iteration we will extend the length of our item set with 1, i.e.  $k = k+1$
- All the combinations of itemsets in F11 is used in this iteration

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

Itemset	Support
{1,2}	1
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

CI2

Itemset	Support
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

F12

# Apriori Algorithm – Third Iteration

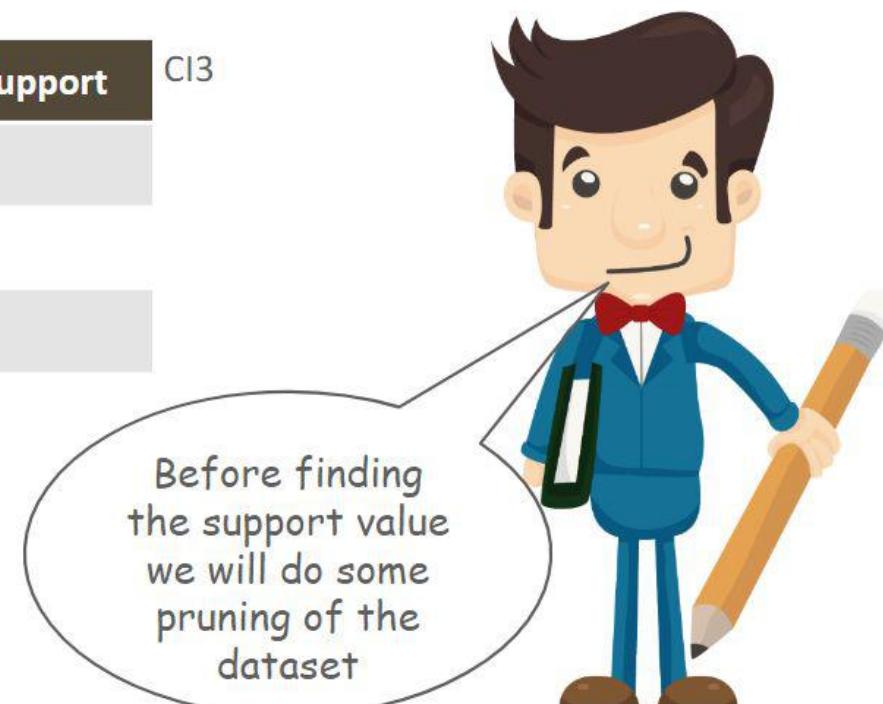
- In this iteration also we will extend the length of our item set. All the combinations of itemsets in F12 is used

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5



Itemset	Support
{1,2,3}	
{1,2,5}	
{1,3,5}	
{2,3,5}	

CI3



# Apriori Algorithm – Pruning

- After the combinations are made you will divide your itemsets to check if there are any other subsets whose support you haven't calculated yet

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

Itemset	In F12?
{1,2,3} <b>{1,2}</b> ,{1,3},{2,3}	No
{1,2,5} <b>{1,2}</b> ,{1,5},{2,5}	No
{1,3,5} <b>{1,5}</b> ,{1,3},{3,5}	Yes
{2,3,5} <b>{2,3}</b> ,{2,5},{3,5}	Yes

CI3

Itemset	Support
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

If any of the subsets of these itemsets are not there in F12 then remove that itemset

# Apriori Algorithm – Fourth Iteration

- Using the itemsets of CI3 you will create new itemset CI4

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5



Itemset	Support
{1,3,5}	2
{2,3,5}	2

F13



Itemset	Support
{1,2,3,5}	1

CI4

Since, support of your CI4 is less than 2, you will stop and return to the previous itemset, i.e. CI3

# Apriori Algorithm – Subset Creation

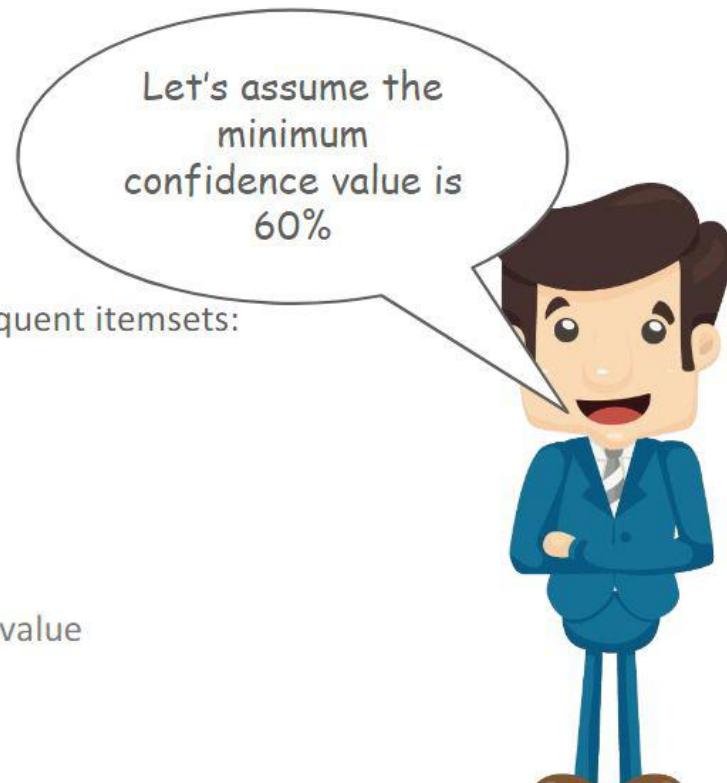
- Now you have the list of frequent itemsets as:

Itemset	Support
{1,3,5}	2
{2,3,5}	2

F13

Let's assume the minimum confidence value is 60%

- Using this you will generate all non empty subsets for each frequent itemsets:
  - For  $I = \{1,3,5\}$ , subsets are  $\{1,3\}$ ,  $\{1,5\}$ ,  $\{3,5\}$ ,  $\{1\}$ ,  $\{3\}$ ,  $\{5\}$
  - For  $I = \{2,3,5\}$ , subsets are  $\{2,3\}$ ,  $\{2,5\}$ ,  $\{3,5\}$ ,  $\{2\}$ ,  $\{3\}$ ,  $\{5\}$
- For every subsets  $S$  of  $I$ , you output the rule
  - $S \rightarrow (I-S)$  (means  $S$  recommends  $I-S$ )
  - if  $\text{support}(I) / \text{support}(S) \geq \text{min\_conf}$  value



# Apriori Algorithm – Applying Rules

- Now we will apply our rules to the itemsets of F13:

1. {1,3,5}

❖ Rule 1:  $\{1,3\} \rightarrow (\{1,3,5\} - \{1,3\})$  means  $1 \& 3 \rightarrow 5$

$$\text{Confidence} = \text{support}(1,3,5)/\text{support}(1,3) = 2/3 = 66.66\% > 60\%$$

Rule 1 is selected

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

❖ Rule 2:  $\{1,5\} \rightarrow (\{1,3,5\} - \{1,5\})$  means  $1 \& 5 \rightarrow 3$

$$\text{Confidence} = \text{support}(1,3,5)/\text{support}(1,5) = 2/2 = 100\% > 60\%$$

Rule 2 is selected

❖ Rule 3:  $\{3,5\} \rightarrow (\{1,3,5\} - \{3,5\})$  means  $3 \& 5 \rightarrow 1$

$$\text{Confidence} = \text{support}(1,3,5)/\text{support}(3,5) = 2/3 = 66.66\% > 60\%$$

Rule 3 is selected

# Apriori Algorithm – Applying Rules

- Now we will apply our rules to the itemsets of F13:

1.  $\{1,3,5\}$

❖ Rule 4:  $\{1\} \rightarrow (\{1,3,5\} - \{1\})$  means  $1 \rightarrow 3 \& 5$

$$\text{Confidence} = \text{support}(1,3,5)/\text{support}(1) = 2/3 = 66.66\% > 60\%$$

Rule 4 is selected

❖ Rule 5:  $\{3\} \rightarrow (\{1,3,5\} - \{3\})$  means  $3 \rightarrow 1 \& 5$

$$\text{Confidence} = \text{support}(1,3,5)/\text{support}(3) = 2/4 = 50\% < 60\%$$

Rule 5 is rejected

❖ Rule 6:  $\{5\} \rightarrow (\{1,3,5\} - \{5\})$  means  $5 \rightarrow 1 \& 3$

$$\text{Confidence} = \text{support}(1,3,5)/\text{support}(5) = 2/4 = 50\% < 60\%$$

Rule 6 is rejected

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 5

# Market Basket Analysis

---

We will be using the following online transactional data of a retail store for generating association rules

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	01-12-2010 08:26	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01-12-2010 08:26	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	01-12-2010 08:26	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	01-12-2010 08:26	4.25	17850	United Kingdom
536366	22633	HAND WARMER UNION JACK	6	01-12-2010 08:28	1.85	17850	United Kingdom
536366	22632	HAND WARMER RED POLKA DOT	6	01-12-2010 08:28	1.85	17850	United Kingdom
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	01-12-2010 08:34	1.69	13047	United Kingdom
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	01-12-2010 08:34	2.1	13047	United Kingdom
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	01-12-2010 08:34	2.1	13047	United Kingdom
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	01-12-2010 08:34	3.75	13047	United Kingdom
536367	22310	IVORY KNITTED MUG COSY	6	01-12-2010 08:34	1.65	13047	United Kingdom

Data can be downloaded from the LMS

# Market Basket Analysis: Step1

First you need to get your pandas and MLxtend libraries imported and read the data

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

df = pd.read_excel('Online_Retail.xlsx')
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

# Market Basket Analysis: Step 2

In this step, you will be doing:

- Data clean up which includes removing spaces from some of the descriptions
- Drop the rows that don't have invoice numbers and remove the credit transactions

```
df['Description'] = df['Description'].str.strip()
df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]
df
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850.0	United Kingdom

# Market Basket Analysis: Step 3

- After the clean-up, we need to consolidate the items into 1 transaction per row with each product
- For the sake of keeping the data set small, we are only looking at sales for France

```
basket = (df[df['Country'] == "France"]
            .groupby(['InvoiceNo', 'Description'])['Quantity']
            .sum().unstack().reset_index().fillna(0)
            .set_index('InvoiceNo'))
```

basket

Description	50'S CHRISTMAS GIFT BAG LARGE	DOLLY GIRL BEAKER	I LOVE LONDON MINI BACKPACK	NINE DRAWER OFFICE TIDY	SET 2 TEA TOWELS I LOVE LONDON	SPACEBOY BABY GIFT SET	TRELLIS COAT RACK	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	...	WRAP VINTAGE PETALS DESIGN	YELLOW COAT RACK PARIS FASHION	
InvoiceNo														
536370	0.0	0.0	0.0	0.0	24.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
536852	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
536974	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
537065	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
537463	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
537468	0.0	0.0	0.0	0.0	0.0	0.0	0.0	24.0	0.0	0.0	0.0	...	0.0	0.0
537693	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
537897	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0

# Market Basket Analysis: Step 4

---

- There are a lot of zeros in the data but we also need to make sure any positive values are converted to a 1 and anything less than 0 is set to 0

```
def encode_units(x):  
    if x <= 0:  
        return 0  
    if x >= 1:  
        return 1  
basket_sets = basket.applymap(encode_units)  
basket_sets.drop('POSTAGE', inplace=True, axis=1)  
basket_sets
```

# Market Basket Analysis: Step 4 (O/P)

Now, you have structured the data properly

Description	50'S CHRISTMAS GIFT BAG LARGE	DOLLY GIRL BEAKER	I LOVE LONDON MINI BACKPACK	NINE DRAWER OFFICE TIDY	SET 2 TEA TOWELS I LOVE LONDON	SPACEBOY BABY GIFT SET	TRELLIS COAT RACK	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	...	WRAP VINTAGE PETALS DESIGN	YELLOW COAT RACK PARIS FASHION
InvoiceNo													
536370	0	0	0	0	1	0	0	0	0	0	0	0	0
536852	0	0	0	0	0	0	0	0	0	0	0	0	0
536974	0	0	0	0	0	0	0	0	0	0	0	0	0
537065	0	0	0	0	0	0	0	0	0	0	0	0	0
537463	0	0	0	0	0	0	0	0	0	0	0	0	0
537468	0	0	0	0	0	0	0	0	1	0	0	0	0

# Market Basket Analysis: Step 5

---

In this step, you will:

- Generate frequent item sets that have a support of at least 7% (this number is chosen so that you can get close enough)
- Generate the rules with their corresponding support, confidence and lift

```
frequent_itemsets = apriori(basket_sets, min_support=0.07,  
use_colnames=True)  
rules = association_rules(frequent_itemsets, metric="lift",  
min_threshold=1)  
rules.head()
```

# Market Basket Analysis: Step 5 (O/P)

	antecedants	consequents	support	confidence	lift
0	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN CIRCUS PARADE )	0.117137	0.648148	4.527217
1	(PLASTERS IN TIN CIRCUS PARADE )	(PLASTERS IN TIN SPACEBOY)	0.143167	0.530303	4.527217
2	(PLASTERS IN TIN CIRCUS PARADE )	(PLASTERS IN TIN WOODLAND ANIMALS)	0.143167	0.606061	4.170059
3	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN CIRCUS PARADE )	0.145336	0.597015	4.170059
4	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.117137	0.759259	5.224157

## Observations:

- A few rules with a high lift value, which means that it occurs more frequently than would be expected, given the number of transaction and product combinations
- Most of the places the confidence is high as well

# Market Basket Analysis: Step 6

- Filter the dataframe using standard pandas code, for a large lift (6) and high confidence (.8)

```
rules[ (rules['lift'] >= 6) &  
       (rules['confidence'] >= 0.8) ]
```

	antecedants	consequents	support	confidence	lift
10	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.108460	0.960000	8.195556
11	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.117137	0.888889	8.195556
12	(SET/6 RED SPOTTY PAPER PLATES, SET/6 RED SPOT...)	(SET/20 RED RETROSPOT PAPER NAPKINS )	0.104121	0.812500	7.203125
13	(SET/20 RED RETROSPOT PAPER NAPKINS , SET/6 RE...)	(SET/6 RED SPOTTY PAPER PLATES)	0.086768	0.975000	8.989500
14	(SET/20 RED RETROSPOT PAPER NAPKINS , SET/6 RE...)	(SET/6 RED SPOTTY PAPER CUPS)	0.086768	0.975000	8.323611

For association rules, the granularity lies at the transaction level. They use transactions as a central entity and hence, do not provide user specific insights

For that we will use  
Recommendation  
Engines



# Recommendation Engines

- A Recommendation engine (sometimes referred to as a recommender system) is a tool, that allows algorithm developers predict what a user may or may not like among a list of given items
- Help users discover products or content that we may not come across otherwise

*This makes recommendation engines a great part of web sites and services such as Facebook, YouTube, Amazon, and more*

Customers who viewed this item also viewed these products



Dualit Food XL1500  
Processor  
\$560

Add to cart



Kenwood kMix Manual  
Espresso Machine  
★★★★★  
\$250

Select options



Weber One Touch Gold  
Premium Charcoal  
Grill-57cm  
\$225

Add to cart



NoMU Salt Pepper and  
Spice Grinders  
\$3

View options

# Recommendation Engine Types

Recommendation engines work ideally in one of two ways:

## User-based filtering

Building a model from a user's past behavior as well as similar decisions made by other users. This model is then used to predict items that the user may have an interest in

## Content-based filtering

Utilizes a series of discrete characteristics of an item in order to recommend additional items to user with similar properties

*It is also possible to combine both these methods to build a much more robust recommendation engine (Hybrid Recommender Systems)*

# Hybrid Recommender System – Example

---

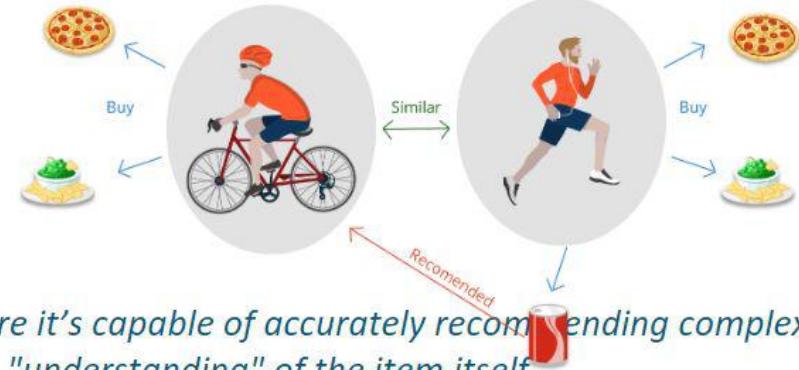
A Hybrid recommender system is based on both UBF and CBF



Netflix, nowadays is using **hybrid recommender systems** for movie/series recommendations to users

# User-Based Collaborative Filtering (UBCF)

- This algorithm searches a large group of people and finds a smaller set with tastes similar to yours
- It looks at other things they like and combines them to create a ranked list of suggestions
- Many algorithms have been used in measuring user similarity or item similarity:
  - ❖ K – nearest neighbor (k-NN) approach[21]
  - ❖ Pearson Correlation



*It does not rely on machine analyzable content and therefore it's capable of accurately recommending complex items such as drinks without requiring an "understanding" of the item itself*

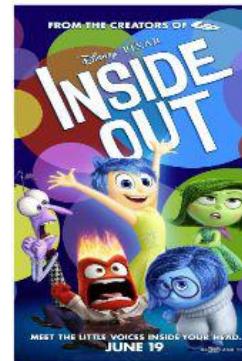
# UBCF Working: Step 1

Consider an example of *Movie Recommendation*

Suppose Sarah has just watched the movie [Inside out](#). Let's see how the recommendation engine works and which are the movies that it thinks she would like to see next

First step

1. Generate a list of users who have seen the following movies



# UBCF Working: Step 2

2. Here we have 4 users who has watched the following movies



John	Yes	Yes	Yes	Yes
Dave	No	Yes	No	Yes
Stuart	Yes	No	Yes	No
Sam	No	No	Yes	Yes

# UBCF Working: Step 3

Now, we find Users similar to Sarah



John	Yes	Yes	Yes	Yes
Dave	No	Yes	No	Yes
Stuart	Yes	No	Yes	No
Sam	No	No	Yes	Yes
Sarah	Yes	??	??	??

# UBCF Working: Step 4

4. Based on the data we have, John and Stuart has also watched the movie inside out, so they are similar to Sarah



John	Yes	Yes	Yes	Yes
Dave	No	Yes	No	Yes
Stuart	Yes	No	Yes	No
Sam	No	No	Yes	Yes
Sarah	Yes	??	??	??

# UBCF Working: Step 5

- Using the data of similar users we can see that the movie **Avengers** gets more vote, so it is recommended to **Sarah**

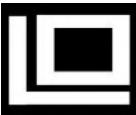


John	Yes	Yes	Yes	Yes
Dave	No	Yes	No	Yes
Stuart	Yes	No	Yes	No
Sam	No	No	Yes	Yes
Sarah	Yes	??	??	??

1 vote

2 votes

1 vote



# Pros & Cons of User-based Filtering

---

Data not a constraint

- Works on consumer item scenario without any user or item feature data availability

Easy to Comprehend

- Easy to explain overall mathematical logic

Differentiated Output

- More differentiated output than associated rule

Cold Start

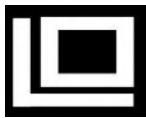
- Need enough users or items to find a match, does not work for new user or item

Sparsity

- User/ratings matrix is sparse and hence, hard to find users that have rated the same items

Popularity Bias

- Tends to recommend popular items, cannot recommend items to one with unique taste



# CONTENT BASED FILTERING

# Content Based Filtering

---

Have the content as the central entities

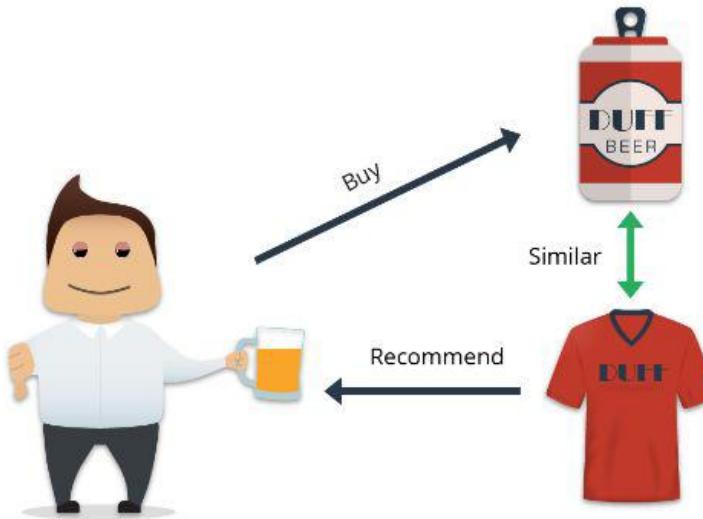


Works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link, purchase history)

Based on that data, a user profile is generated to make suggestions to the user

As the user provides more and more input the engine's accuracy increases

# Content Based Filtering – An Example



*If Sam buys DUFF consumer merchandise, content based filtering considers DUFF beer can as an entity and recommends other DUFF merchandise such as a tee shirt to the buyer*

# CBF Working: Step 1

Consider the same example of Movie Recommendation

Suppose we have watched the movie [Inside out](#), Lets see how the recommendation engine works and which are the movies that it thinks we would like to go see

1. Generate a list of features about the movies like, Actors, Directors, Themes etc



# CBF Working: Step 2

2. Compare columns of each movies with column of the movie Inside out and see which of them matches



Animated	Yes	Yes	No	No
Marvel	No	No	Yes	Yes
Super Villain	No	Yes	Yes	Yes
IMDB rating 8+	Yes	No	Yes	No
Comedy	Yes	Yes	No	Yes

# CBF Working: Step 3

3. The column with the most match is of Minions, so the system will recommend it to watch



Animated	Yes	Yes	No	No
Marvel	No	No	Yes	Yes
Super Villain	No	Yes	Yes	Yes
IMDB rating 8+	Yes	No	Yes	No
Comedy	Yes	Yes	No	Yes



# Pros & Cons of Content Based Filtering



Only user data

- No need for data on other users

No Differentiation

- Able to recommend to users with unique tastes

No first rater problem

- Able to recommend new and unpopular item

Find important features

- Finding the appropriate feature is hard. E.g.: For movies and images which all features are important

Over specialization

- Never recommends items outside user's content profile

No good judgements

- Unable to exploit quality judgements of other users

# Use-Case: E-Commerce Sites

---

*Many of the largest commerce Web sites are already using recommender systems to help their customers find products to purchase*

- The products can be recommended based on the top overall sellers on a site or based on an analysis of the past buying behavior of the customer as a prediction for future buying behavior



# Use-Case : Social Networks

---

*Social networking sites employ recommendation systems in contribution to providing better user experiences*



- Facebook and LinkedIn focus on link recommendation where friend recommendations are presented to users
- Most of the friend suggestion mechanism rely on pre-existing user relationship to pick friend candidates





# BUILDING A RECOMMENDER SYSTEM

# Use Case: Scenario

---

- Consider the ratings dataset below, containing the data on: UserID, MovieID, Rating and Timestamp
- Each line of this file represents one rating of one movie by one user, and has the following format:

UserID::MovieID::Rating::Timestamp

- Ratings are made on a 5 star scale with half star increments

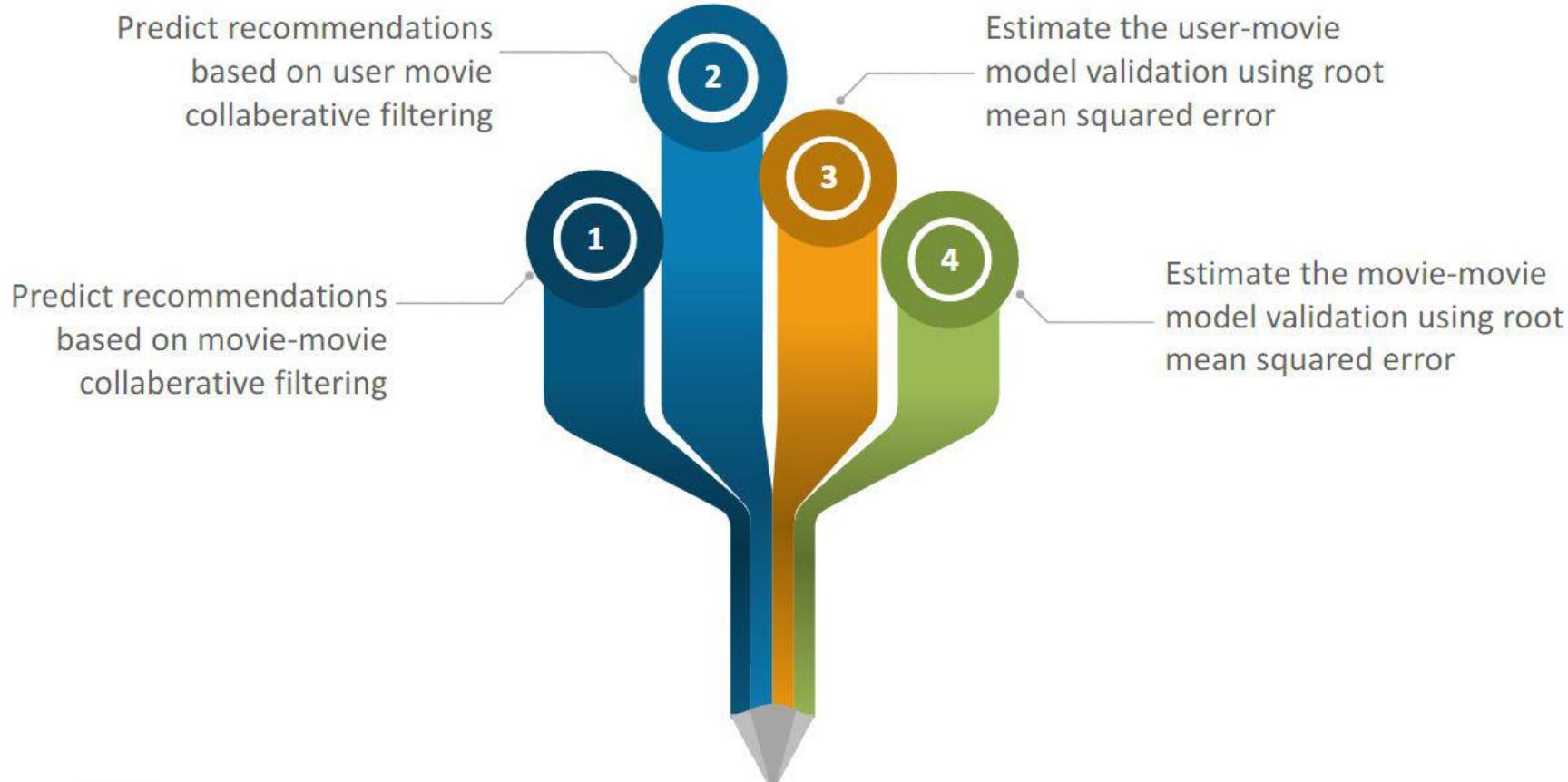
**UserID:** represents ID of the user

**MovieID:** represents ID of the movie

**Timestamp:** represents seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970

196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596
298	474	4	884182806
115	265	2	881171488
253	465	5	891628467
305	451	3	886324817
6	86	3	883603013
62	257	2	879372434
286	1014	5	879781125
200	222	5	876042340
210	40	3	891035994
224	29	3	888104457

# Use Case: Tasks To Do



# Use Case Solution: Step 1

Load the 'Ratings' movie data into pandas with labels

```
df = pd.read_csv('Recommend.csv', names=[ 'user_id',
'movie_id', 'rating', 'timestamp'])
df
```

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
5	298	474	4	884182806
6	115	265	2	881171488
7	253	465	5	891628467
8	305	451	3	886324817

# Use Case Solution: Step 2

---

Declare number of users and movies and create a train test split of 75/25

```
n_users = df.user_id.unique().shape[0]
n_movies = df.movie_id.unique().shape[0]
train_data, test_data = train_test_split(df, test_size=0.25)
```

*The data here now gets split as train data and test data, such that the train data is 75% of the total data*

# Use Case Solution: Step 3

Populate the train matrix (user\_id x movie\_id), containing ratings such that [user\_id index, movie\_id index] = given rating

```
train_data_matrix = np.zeros((n_users, n_movies))
for line in train_data.itertuples():
    #[user_id index, movie_id index] = given rating.
    train_data_matrix[line[1]-1, line[2]-1] = line[3]
train_data_matrix
```

```
array([[ 5.,  3.,  4., ...,  0.,  0.,  0.],
       [ 4.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 5.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  5.,  0., ...,  0.,  0.,  0.]])
```

# Use Case Solution: Step 4

Populate the test matrix (user\_id x movie\_id), containing ratings such that [user\_id index, movie\_id index] = given rating

```
test_data_matrix = np.zeros((n_users, n_movies))
for line in test_data.itertuples():
    #[user_id index, movie_id index] = given rating.
    test_data_matrix[line[1]-1, line[2]-1] = line[3]
test_data_matrix
```

```
array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.]])
```

# Use Case Solution: Step 5

Creates cosine similarity matrices for users and movies and predict a user-movie recommendation model (based on difference from mean rating as it's a better indicator than absolute rating)

```
user_similarity = pairwise_distances(train_data_matrix, metric='cosine')
movie_similarity = pairwise_distances(train_data_matrix.T, metric='cosine')
mean_user_rating = train_data_matrix.mean(axis=1)[:, np.newaxis]
ratings_diff = (train_data_matrix - mean_user_rating)
user_pred = mean_user_rating + user_similarity.dot(ratings_diff) /
np.array([np.abs(user_similarity).sum(axis=1)]).T
user_pred
```

```
array([[ 1.59729744,  0.56219584,  0.49372333, ...,  0.29679791,
       0.2992736 ,  0.29905837],
       [ 1.34378863,  0.27907632,  0.16808849, ..., -0.05775465,
      -0.0545478 , -0.05424075],
       [ 1.34540808,  0.2234773 ,  0.12627917, ..., -0.10610823,
      -0.10287814, -0.10272538],
       ...,
       [ 1.20896726,  0.19028008,  0.08987345, ..., -0.1265226 ,
      -0.1234234 , -0.12332607],
       [ 1.37408572,  0.29130823,  0.21123601, ..., -0.01384715,
      -0.01070536, -0.01068932],
       [ 1.4246277 ,  0.35700773,  0.30011783, ...,  0.09792286,
      0.10016502,  0.1003628 ]])
```

## Use Case Solution: Step 6

Predict the same for the movie based recommendation model (based on difference from mean rating as it's a better indicator than absolute rating)

```
movie_pred = train_data_matrix.dot(movie_similarity) /  
np.array([np.abs(movie_similarity).sum(axis=1)])  
movie_pred
```

```
array([[ 0.37095251,  0.38716765,  0.40025082, ...,  0.44854253,  
       0.4422732 ,  0.43775846],  
       [ 0.09440623,  0.11043915,  0.10521177, ...,  0.11243308,  
       0.11192467,  0.11294353],  
       [ 0.06272422,  0.06545004,  0.06302051, ...,  0.06365259,  
       0.06343483,  0.06429965],  
       ...,  
       [ 0.02838429,  0.03586421,  0.03477146, ...,  0.04045211,  
       0.03992107,  0.03988944],  
       [ 0.12626427,  0.13544609,  0.14149739, ...,  0.14753123,  
       0.14742251,  0.14737875],  
       [ 0.2047876 ,  0.19941526,  0.2216142 , ...,  0.25104105,  
       0.24299932,  0.24511761]])
```

# Use Case Solution: Step 7

---

Define a root mean squared error (RMSE) function to check the validity of the user-based and movie-based recommendation model

```
def rmse(pred, test):
    pred = pred[test.nonzero()].flatten()
    test = test[test.nonzero()].flatten()
    return sqrt(mean_squared_error(pred, test))
```

# Use Case Solution: Step 8

---

Pass the user-based model that you have recently created into the rmse function

```
rmse (user_pred, test_data_matrix)
```

```
3.1205151270317386
```

*The error so obtained is 3.1205151270317386 which is minimal and thus you can conclude that the model is a good model*

# Use Case Solution: Step 9

---

Pass the movie-based model that you have recently created into the rmse function

```
rmse(movie_pred, test_data_matrix)
```

```
3.447038130496627
```

*The error so obtained is 3.447038130496627 which is minimal and thus you can conclude that the model is a good model*





