Machine Learning

Part-1 (Machine Learning Preprocessing)

Machine Learning Libraries

- NumPy - is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- Pandas - is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. Can easily read data file like csv, json
- Matpoltlib - Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- SciPy : provides a large menu of libraries for scientific computation, such as integration, interpolation, signal processing, linear algebra etc. It is built upon the infrastructure of Numpy.
- Sklearn – (scikit-learn) This is a popular ML library, built on NumPy, SciPy and matplotlib. Please note that sklearn is used to build machine learning models

- TensorFlow - TensorFlow is widely used in the field of deep learning research
- Keras : Keras is a Deep Learning library that wraps around the functionalities of othe libraries like Tensorflow, Theano or CNTK. Written in Python.
- Theano - is a Python library for fast numerical computation that can be run on the CPU or GPU. It was developed by the LISA (now MILA) group at the University of Montreal, Quebec, Canada. It is named after a Greek mathematician, Theano
- PyTorch : This is a Facebook deep-learning library. PyTorch was built by Facebook, as it name implies it was written in Python.
- Caffe2 : an improved version of Caffe, is an open machine learning framework build by Facebook for the streamline and flexible deep learning of complex models and support for mobile deployment.

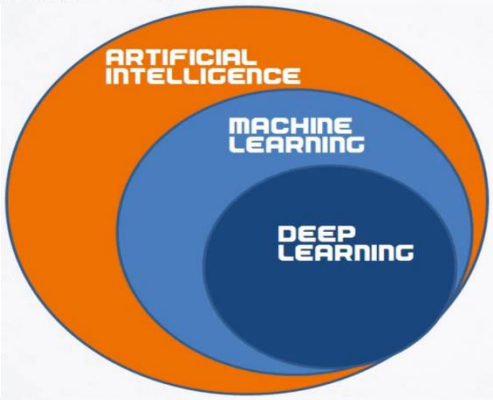
IDE - Integrated Development Environment

- PyCharm :
- Spyder
- Visual Studio Code
- Vim
- GNU/Emacs
- Atom/Atom-IDE
- IDLE - Integrated Development and Learning Environment
- Thonny
- Colab Notebook
- Jupyter Notebook

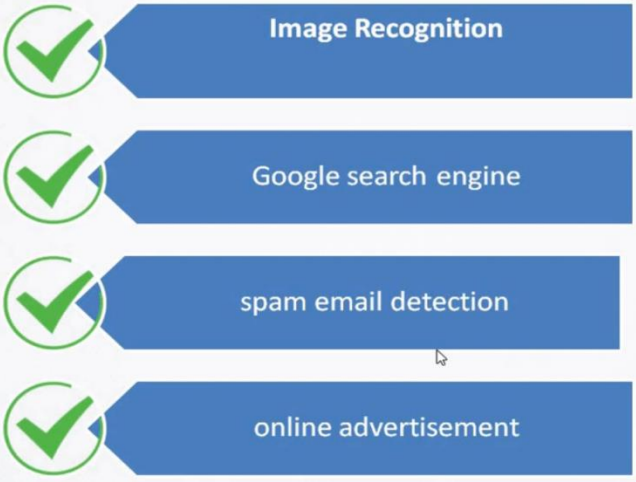
choice of your IDE is based on the performance and advanced features of IDEs for large projects

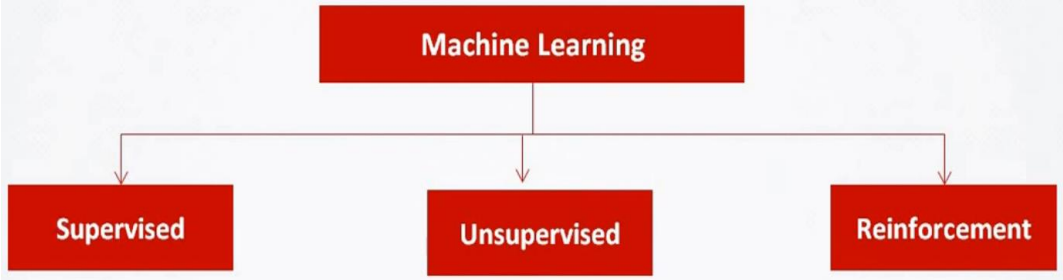
Machine Learning

Machine learning is the process of teaching a computer system how to make accurate predictions when fed data.



machine learning use cases



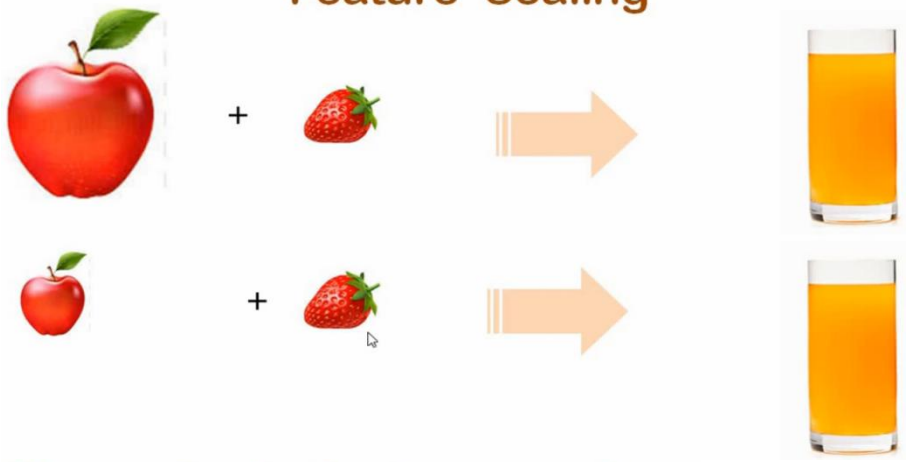


Labels and Features

Gender	Age	Height	Weight	Heart Rate	Calories
female	70	176	67	90.00	200
male	65	190	87	95.00	170
female	29	186	60	87.00	50
male	40	172	80	89.00	65

Case 2 : Gender, Height, and Weight to predict the ***Heart Rate*** .
Heart Rate here is a **Label**. And Gender, Height, and Weight are features

Feature Scaling



Ways to do feature scaling

1) Min Max Scaler
2) Standard Scaler
3) Max Abs Scaler
4) Robust Scaler
5) Quantile Transformer Scaler
6) Power Transformer Scaler
7) Unit Vector Scaler

Normalization

$$X_{nr} = \frac{X - \min(x)}{\max(x) - \min(x)}$$

Normalization rescale the feature in fixed range between 0 to 1, (or -1 to 1 if there are negative values).
Normalization prefers for Image processing because of pixel intensity between 0 to 255.
It is called Min Max Scaler

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3$$

Before Feature Scaling

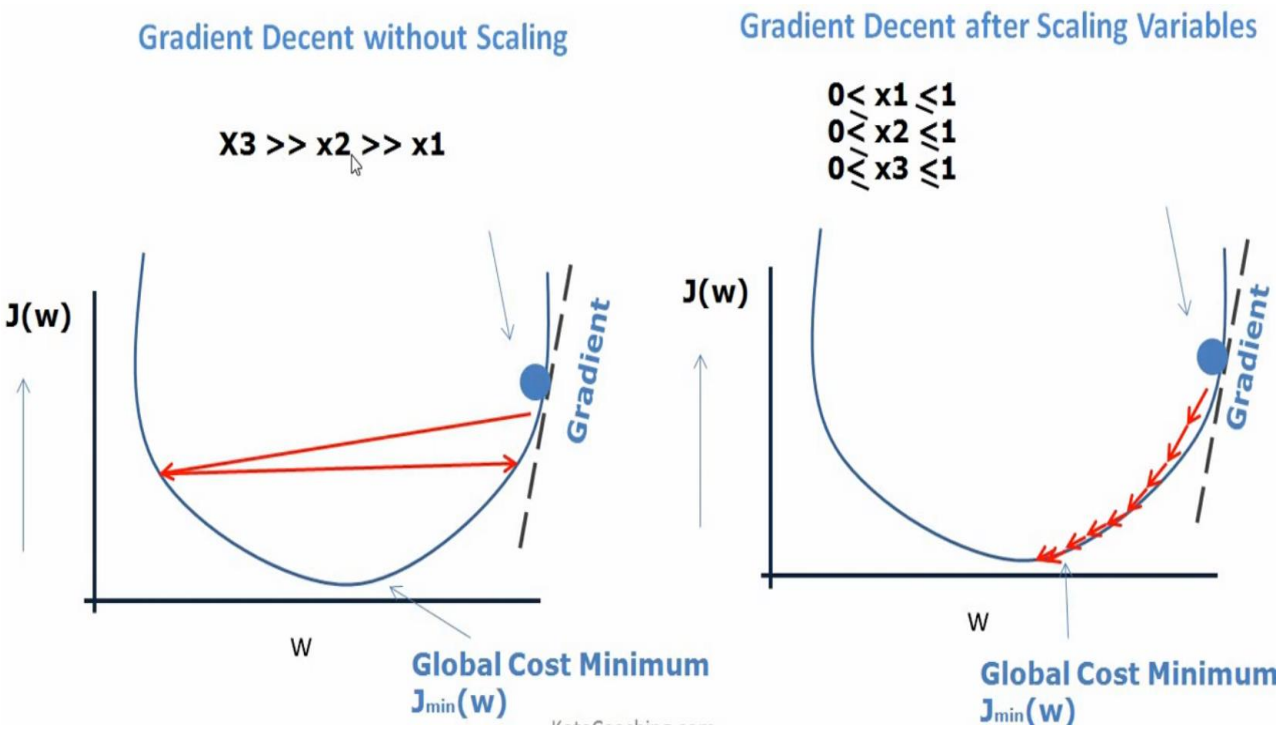
x1	x2	x3	y
5	787	0.01	1
1	654	0.09	0
3	300	0.06	0
5	543	0.01	1

After Feature Scaling

x1	x2	x3	y
1.00	1.00	0.00	1
0.00	0.79	1.00	0
0.50	0.00	0.62	0
1.00	0.54	0.00	1

Normalization...

x1	x2	x3	y
(5-1)/(5-1)	(787 – 300)/(748-300)	(.01 – .01)/(.09-.01)	1
(1-1)/(5-1)	(654 – 300)/(748-300)	(.09 – .01)/(.09-.01)	0
(3-1)/(5-1)	(300 – 300)/(748-300)	(.06 – .01)/(.09-.01)	0
(5-1)/(5-1)	(543– 300)/(748-300)	(.01 – .01)/(.09-.01)	1



Where we should apply Feature Scaling?

Gradient Descent Based Algorithms →

- Linear Regression
- Logistic Regression
- Neural Network

Those Algorithms Calculate Distance →

- K-Nearest Neighbours
- K-Means
- SVM
- PCA
- Linear discriminant Analysis

Where we should not apply Feature Scaling?

Tree Based Algorithms →

- Decision Tree
- Random Forest
- XGBoost

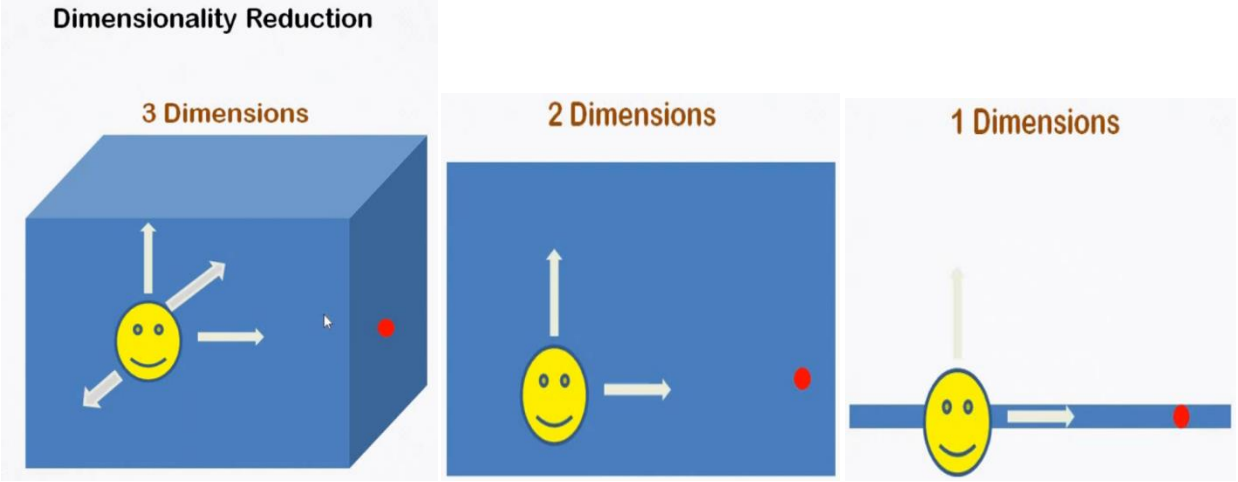
Dimensionality Reduction

The number of input variables or features for a dataset is referred to as its dimensionality.

Dimensionality reduction refers to techniques that reduce the number of input variables in a dataset

Why?

The higher the number of features, the harder it gets to visualize the training set and then work on it



Most popular technique for dimensionality reduction in machine learning is Principal Component Analysis (PCA)

Principal Component analysis also known as PCA is such a feature extraction method. where we create new independent features from the old features and from combination of both keep only those features that are most important in predicting the target.

PCA Steps

1. **Standardization**
2. **Computing Covariance Matrix**
3. **Calculating Eigenvector and eigenvalue**
4. **Compute Principal Components**
5. **Reduce data dimension**

Standardization

$$X_{st} = \frac{X - \text{mean}(x)}{\text{Standard deviation}(x)}$$

This technique to rescale features value with the distribution value between 0 and 1.
It is called Standard Scaler

$$cov(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

	X	Y	Z
A =	2	3	3
	4	3	2
	3	6	4

$\overline{X} = 3 \quad \overline{Y} = 4 \quad \overline{Z} = 3$

$$Cov(X,X) = \frac{(2-3)(2-3)+(4-3)(4-3)+(3-4)(3-4)}{3} = 1$$

$$Cov(X,Y) = \frac{(2-3)(3-4)+(4-3)(3-4)+(3-3)(6-4)}{3} = 0$$

$$Cov(X,Z) = \frac{(2-4)(3-3)+(4-4)(2-3)+(3-4)(4-3)}{3} = 0.33$$

$$Cov(Y,Y) = \frac{(3-4)(3-4)+(3-4)(3-4)+(6-4)(6-4)}{3} = 2$$

$$Cov(Y,X) = \frac{(3-4)(2-3)+(3-4)(4-3)+(6-4)(3-3)}{3} = 0$$

$$Cov(Y,Z) = \frac{(3-4)(3-3)+(3-4)(2-3)+(6-4)(4-3)}{3} = 1$$

$$Cov(Z,X) = \frac{(3-3)(2-3)+(2-3)(3-3)+(4-3)(4-3)}{3} = 0.33$$

$$Cov(Z,Y) = \frac{(3-3)(3-4)+(2-3)(3-4)+(4-3)(6-4)}{3} = 1$$

$$Cov(Z,Z) = \frac{(3-3)(3-3)+(2-3)(2-3)+(4-3)(4-3)}{3} = 1$$

Covariance matrix

$$COV = \begin{pmatrix} XX & XY & XZ \\ YX & YY & YZ \\ ZX & ZY & ZZ \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0.33 \\ 0 & 2 & 1 \\ 0.33 & 1 & 1 \end{pmatrix}$$

Deviation Matrix -> a

Covariance Matrix

$a = A - [].A.(1/5)$

$$A = \begin{matrix} M & P & E \end{matrix} \begin{pmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 60 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 20 & 90 \end{pmatrix}$$

$$a = \begin{pmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 60 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 20 & 90 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 60 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 20 & 90 \end{pmatrix} \frac{1}{5} = \begin{pmatrix} 30 & 30 & -30 \\ 30 & 10 & 10 \\ 0 & 0 & 0 \\ -30 & -10 & 0 \\ -30 & -30 & 20 \end{pmatrix}$$

$$CM = a^T a = \begin{pmatrix} 30 & 30 & 0 & -30 & -30 \\ 30 & 10 & 0 & -10 & -30 \\ -30 & 10 & 0 & 0 & 20 \end{pmatrix} \begin{pmatrix} 30 & 30 & -30 \\ 30 & 10 & 10 \\ 0 & 0 & 0 \\ -30 & -10 & 0 \\ -30 & -30 & 20 \end{pmatrix} = \begin{pmatrix} 3600 & 2400 & -1200 \\ 2400 & 2000 & -1400 \\ -1200 & -1400 & 1400 \end{pmatrix}$$

Compute Eigenvectors and corresponding Eigenvalues

$$|A - \lambda.I| = 0$$

λ *a scalar*

A *be a square matrix*

Equation is called characteristic equation of **A**,

$$\begin{pmatrix} 3600 & 2400 & -1200 \\ 2400 & 2000 & -1400 \\ -1200 & -1400 & 1400 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{determinant} \begin{pmatrix} 3600 - \lambda & 2400 & -1200 \\ 2400 & 2000 - \lambda & -1400 \\ -1200 & -1400 & 1400 - \lambda \end{pmatrix}$$

$(3600 - \lambda)((2000 - \lambda)(1400 - \lambda) - (-1400)(-1400)) - 2400(2400(1400 - \lambda) - (-1200 * (-1400)) + (-1200)\,2400(-1400) - (-1200)*(2000 - \lambda) = 0$

$-\lambda 3 - 15996240000 - 3600\lambda 2 + 6360000\lambda = 0$

$\lambda = 874.18062, 873.18062, -5348.36125,$

For each eigenvalue there will be an eigenvector for which the eigenvalue.

$Av = \lambda v$

λ is Eagen Value

A is Covariance matrix,

v is Eigen vector and

$$\begin{pmatrix} 3600 & 2400 & -1200 \\ 2400 & 2000 & -1400 \\ -1200 & -1400 & 1400 \end{pmatrix} \begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix} = 874.18062 \begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix}$$

$$\begin{pmatrix} 3600 & 2400 & -1200 \\ 2400 & 2000 & -1400 \\ -1200 & -1400 & 1400 \end{pmatrix} \begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix} = 873.18062 \begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix}$$

$$\begin{pmatrix} 3600 & 2400 & -1200 \\ 2400 & 2000 & -1400 \\ -1200 & -1400 & 1400 \end{pmatrix} \begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix} = -5348.36125 \begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix}$$

Arrange the eigen pairs in decreasing order of respective eigenvalues and pick the value which has the maximum value, this is the first principal component that protects the maximum information from the original data.

Calculate eigenvectors that are the principal component

If we choose the top 2 eigenvectors, the matrix will look like this:

$$\begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix} = \begin{pmatrix} 23 \\ 56 \\ 73 \end{pmatrix}$$

$$\begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix} = \begin{pmatrix} 42 \\ 86 \\ 91 \end{pmatrix}$$

$$\begin{pmatrix} X1 \\ X2 \\ X3 \end{pmatrix} = \begin{pmatrix} 43 \\ 52 \\ 33 \end{pmatrix}$$

$$\begin{pmatrix} 23 & 42 & 43 \\ 56 & 86 & 52 \\ 73 & 91 & 33 \end{pmatrix} \Rightarrow \begin{pmatrix} 23 & 42 \\ 56 & 86 \\ 73 & 91 \end{pmatrix}$$

Transform the original matrix.

A

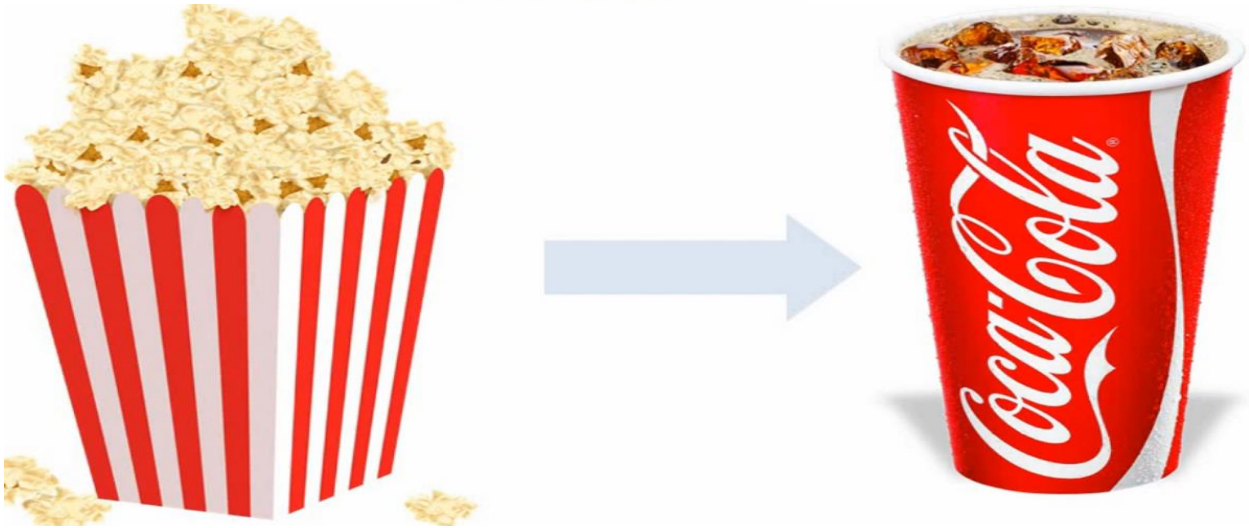
$$\begin{pmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 60 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 20 & 90 \end{pmatrix} \begin{pmatrix} 23 & 42 \\ 56 & 86 \\ 73 & 91 \end{pmatrix} \Rightarrow \begin{pmatrix} 90*23+80*56+40*73 & 90*42+80*86+40*91 \\ 90*23+60*56+80*73 & 90*42+60*86+80*91 \\ 60*23+50*56+70*73 & 60*42+50*86+70*91 \\ 60*23+50*56+70*73 & 30*42+40*86+70*91 \\ 30*23+20*56+90*73 & 30*42+20*86+90*91 \end{pmatrix}$$

$$\Downarrow$$

$$\begin{pmatrix} 6663 & 14300 \\ 11270 & 16220 \\ 11271 & 34565 \\ 34223 & 34534 \\ 43444 & 87554 \end{pmatrix} \Rightarrow \begin{pmatrix} 0.28 & 0.21 \\ -0.92 & 0.34 \\ 0.76 & -0.43 \\ 0.45 & -0.56 \\ 0.76 & 0.82 \end{pmatrix}$$

Principal components for K=2 - >

Association Rule



Association Rule Learning

Association rule learning (Association rule mining) is a rule-based machine learning method for discovering interesting relations between variables in large databases. In simply how items are associated to each other.

Association analysis which attempts to find common patterns of items in large data sets. being used in **market basket analysis**

Association Rule Mining

A => B

IF THEN

An association rule has two parts: an antecedent (if) and a consequent (then). An antecedent is an item found within the data. A consequent is an item found in combination with the antecedent.

If you buy A Then there is possibility you will buy B

There are three common ways to measure association.

$$\text{Support} = \frac{\text{Freq}(A,B)}{N}$$

$$\text{Confidence} = \frac{\text{Freq}(A,B)}{\text{Freq}(A)}$$

$$\text{Lift} = \frac{\text{Support}}{\text{Support}(A) \times \text{Support}(B)}$$

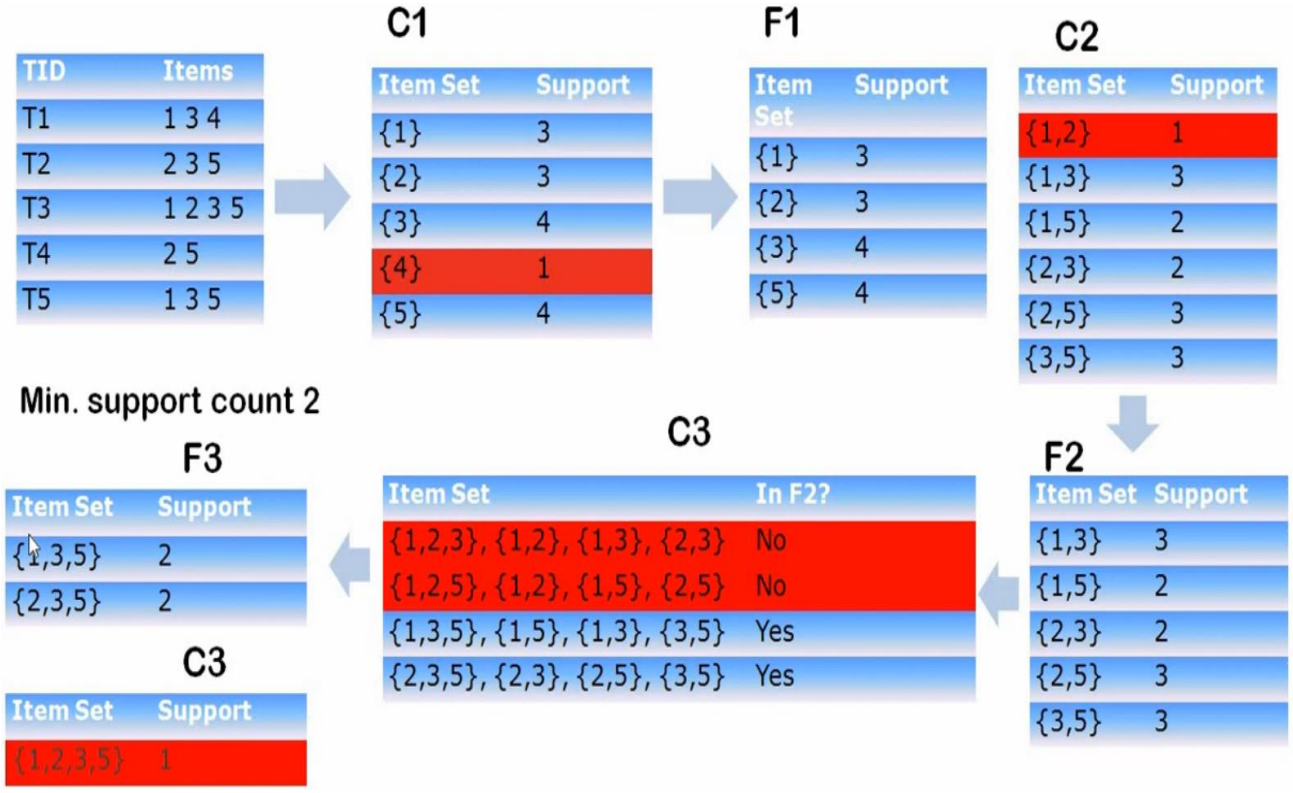
T1	A	B	C
T2	A	C	D
T3	B	C	D
T4	A	D	E
T5	B	C	E

Transactions at a local Market

Rule	Support	Confidence	Lift
A=>D	2/5	2/3	10/9
C=>A	2/5	2/4	5/6
A=>C	2/5	2/3	5/6
B,C=>A	1/5	1/3	5/9

Apriori Algorithm

Apriori algorithm is a classical algorithm in data mining. It is used for mining frequent itemsets and relevant association rules.



Apriori Algorithm – Applying rules to Item set F3

F3

Item Set	Support
{1,3,5}	2
{2,3,5}	2

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}

S –> (I – S) **means** S recommends I-S
If support(I)/support(S) >= min conf val

Rule 1: {1,3} -> ({1,3,5} - {1,3}) means 1 & 3 -> 5
Confidence = support(1,3,5)/support(1,3) = 2/3 = 66.66% > 60%
Rule 1 is selected

Rule 2: {1,5} -> ({1,3,5} - {1,5}) means 1 & 5 -> 3
Confidence = support(1,3,5)/support(1,5) = 2/2 = 100% > 60%
Rule 2 is selected

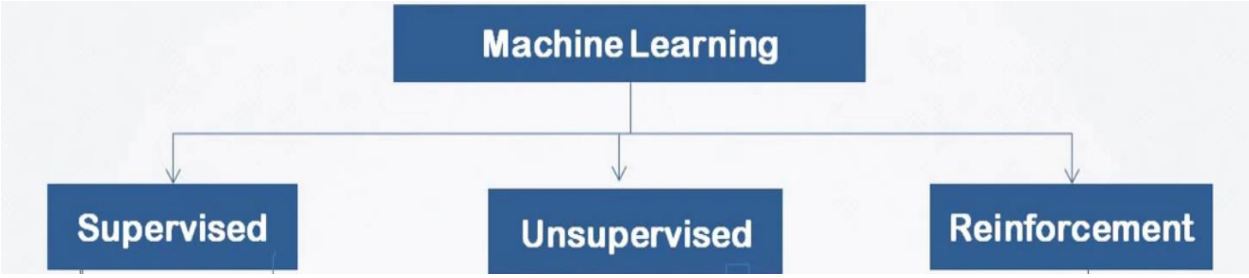
Rule 3: {3,5} -> ({1,3,5} - {3,5}) means 3 & 5 -> 1
Confidence = support(1,3,5)/support(3,5) = 2/3 = 66.66% > 60%
Rule 3 is selected

Rule 4: {1} -> ({1,3,5} - {1}) means 1 - > 3 & 5
Confidence = support(1,3,5)/support(1) = 2/3 = 66.66% > 60%
Rule 4 is selected

Rule 5: {3} -> ({1,3,5} - {3}) means 3 -> 1 & 5
Confidence = support(1,3,5)/support(3) = 2/4 = 50% < 60%
Rule 5 is rejected

Rule 6: {5} -> ({1,3,5} - {5}) means 5 -> 1 & 3
Confidence = support(1,3,5)/support(5) = 2/4 = 50% < 60%
Rule 6 is rejected

Part-2 (Machine learning Algorithm)



Supervised: Supervised algorithm divided into two parts.
(1) Classification (also called Logistic Regression)
(2)Regression

Classification

Height	Age	Weight	Class
14	13 Years	16	Dog
12	15Years	10	Cat
120	60 Year	6k	Elephant
12	15Years	9	Cat

Feature based classification

Binary Classification: Classification tasks with two classes.
Multi-class Classification: Classification tasks with more than two classes.

Example : spam emails, Speech Recognition, Identification of cancer cells

Regression

Year	Expenses	Branches	Price
2005	100	5	300
2010	110	6	400
2015	105	6	390
2020	200	7	500

Product Price with time

The regression Algorithm can be further divided into
Linear and **Non-linear** Regression.

Example : Weather Prediction, House price prediction

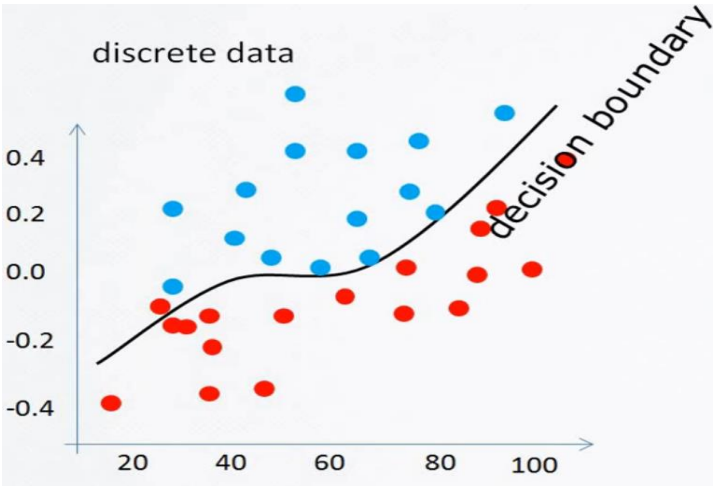


Figure: Classification

C

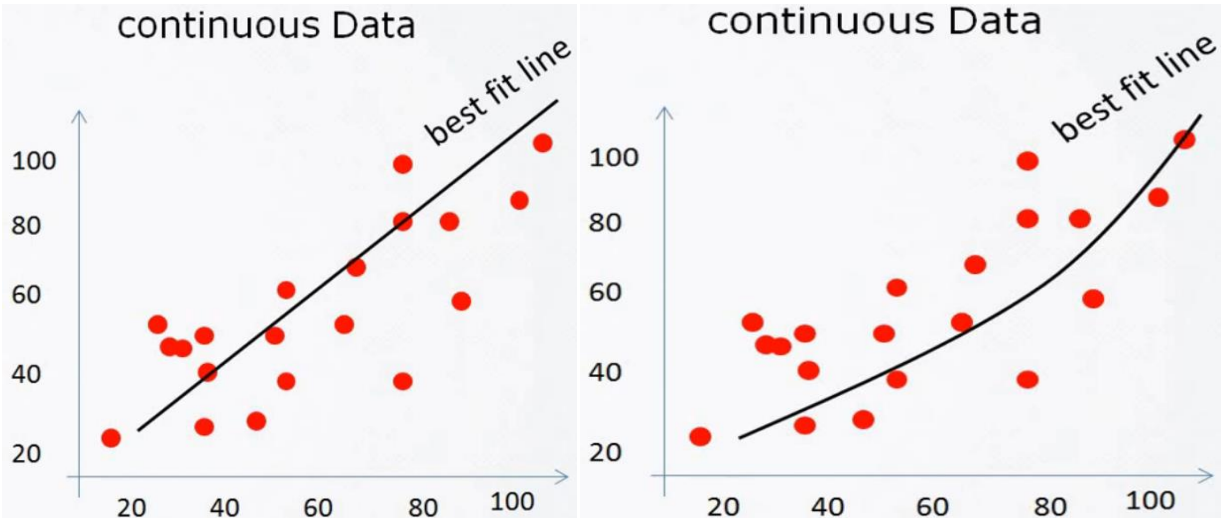


Figure: Regression

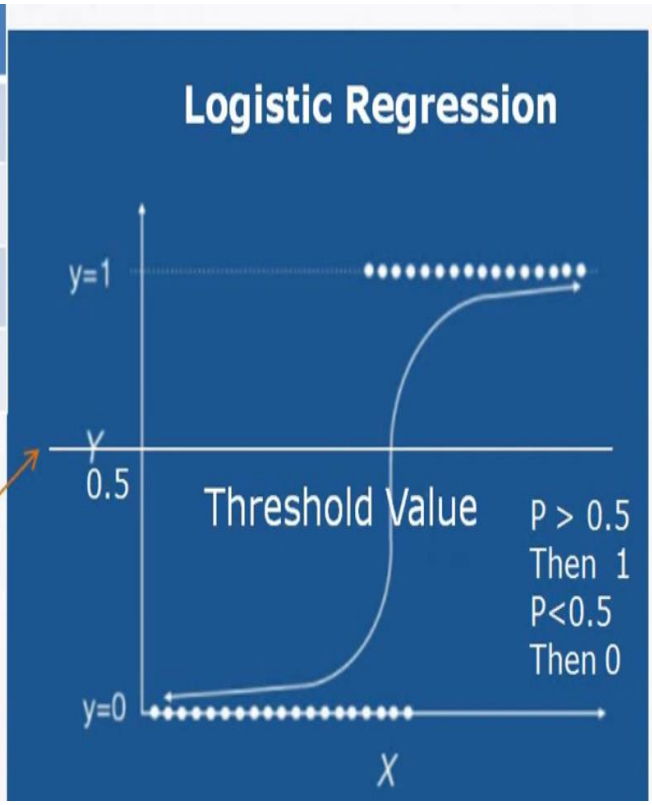
Difference in Linear and Logistic

$$Y = MX + c \qquad P(x) = \frac{1}{1+e^{-(b0+b1x)}}$$

Linear	Logistic
Solve Regression Problem	Solve Classification Problem
Response is Continuous in Nature	Categorical Response
Straight line	S curve
Ex . Weather to know temp, Share market price.	Ex weather Rain or not. Image classification . Like image of Dog or not.

Logistic Regression

Hair Length	(Y)Gender
Medium	Male
Longhair	Female
Longhair	Female
short	Male



probability of gender given longhair

P(gender=female|longhair)

p(longhair)

Binary Response range between 0 and 1.

The logistic function will always produce an S-shaped curve

ODDS

$$\text{Odds} = \frac{P(\text{occurring})}{P(\text{not occurring})} \qquad \frac{P}{1-P}$$

Coin Flip

$$\text{Odds} = \frac{(1/2)}{1-(1/2)}$$

0.5/0.5 = 1 or 1:1

Die Roll

$$\text{Odds(1 or 2)} = \frac{(2/6)}{1-(2/6)}$$

0.33/0.66 = 1 / 2 = 0.5 or 1:2

Math behind logistic regression

Predict the odds of success

$$\text{Log}\left(\frac{P(x)}{1-P(x)}\right) = b_0 + b_1X$$

Exponentiation both side

$$e^{\text{Log}\left(\frac{P(x)}{1-P(x)}\right)} = e^{b_0 + b_1X}$$

$$\left(\frac{P(x)}{1-P(x)}\right) = e^{b_0 + b_1X}$$

$$Y = \left(\frac{P(x)}{1-P(x)}\right)$$

$$Y - Y(P(x) = P(x)$$

$$Y = P(x) + Y(P(x))$$

$$Y = P(x)(1 + Y)$$

$$P(x) = Y/(1+Y)$$

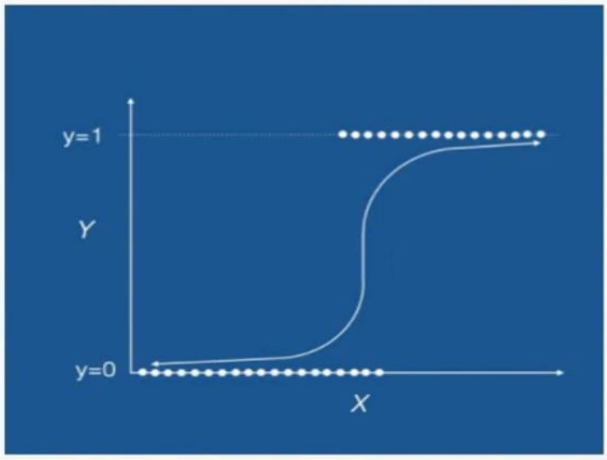
$$P(x) = \frac{e^{b0+b1x}}{1+e^{b0+b1x}}$$

$$P(x) = \frac{1}{1+e^{-(b0+b1x)}}$$

Sigmoid function

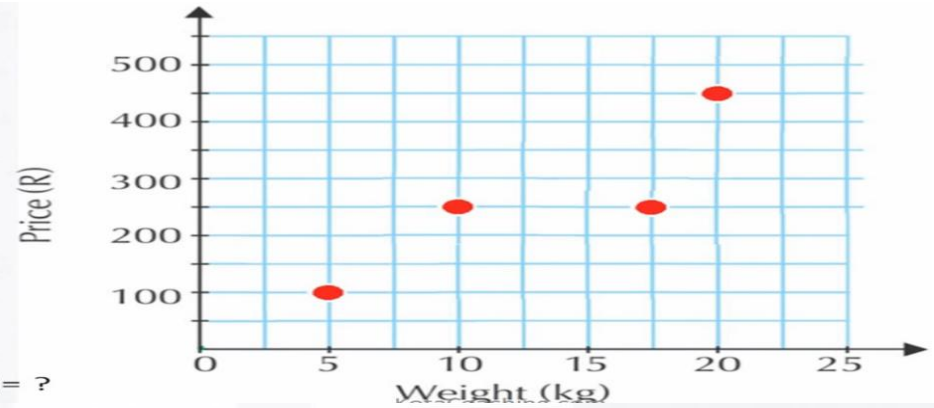
$$P(x) = \frac{1}{1+e^{-(b0+b1x)}}$$

Sigmoid Curve



Simple Linear Regression

x	y
5	100
10	250
17.5	250
20	450



Y = MX + C

Slope(M) = ?
Intercept (C)= ?

$$m = r\left(\frac{s_y}{s_x}\right) \qquad s = \sqrt{\frac{\sum (x-\bar{x})^2}{n-1}} \qquad s = \sqrt{\frac{\sum (y-\bar{y})^2}{n-1}}$$

$C = \bar{y}(\text{mean}) - m\bar{x}(\text{mean})$

Slope Standard Deviation of X Standard Deviation of Y

where r is the correlation between X and Y,
and s_x and s_y are the standard deviations of the x-values and the y-values

$$r = \frac{\sum_{i=1}^{n}\left(y_i-\bar{y}\right)\left(x_i-\bar{x}\right)}{\sqrt{\sum_{i=1}^{n}\left(y_i-\bar{y}\right)^2 \times \sum_{i=1}^{n}\left(x_i-\bar{x}\right)^2}}$$

x (mean)= (5+ 10+ 15+20)/4 = **12.5**

Y(mean) = (100 + 200 + 300 +400)/4 = **250**

$$r = \frac{((100-250)\,(5-12.5))+((100-250)\,(10-12.5))+((100-250)\,(15-12.5))+((100-250)\,(20-12.5))}{\text{sqRt}((100-250)^{**}2\,(5-12.5))^{**}2+((100-250)^{**}2\,(10-12.5))^{**}2+(100-250)^{**}2+(15-12.5)^{**}2+((100-250)^{**}2\,(20-12.5)^{**}2)}$$

m = 1 X(129.099/6.4549) = 20

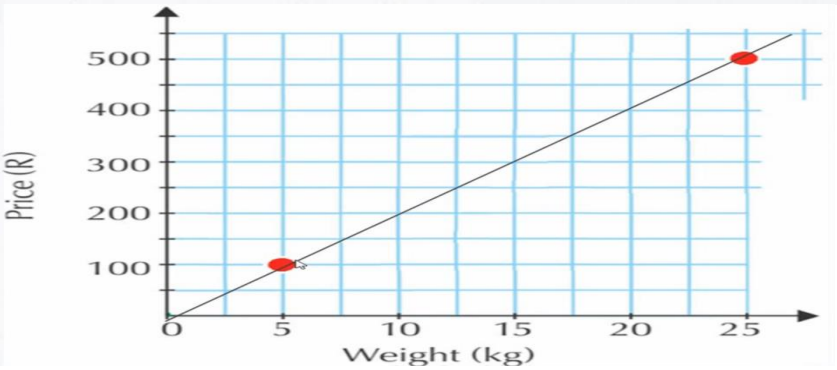
C = 250 – 20x(12.5) = 0

Y = MX + C

Slope(M) = 20
Intercept (C)= 0

Y = 20X + 0
Y = 20*5 = 100 (5, 100)

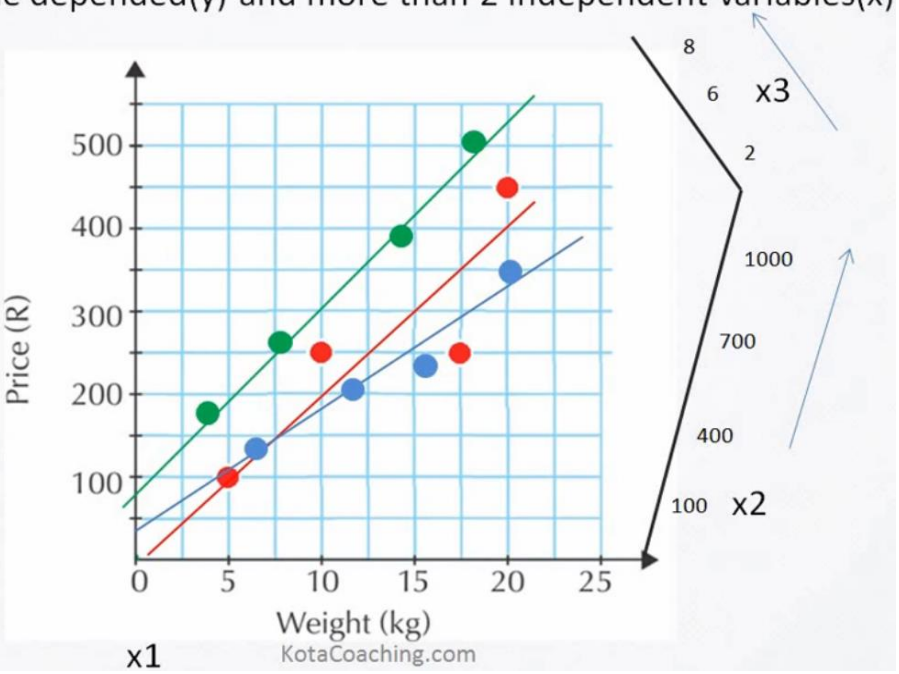
Y = 20*25 = 500 (25,500)



Multiple Linear Regression

One depended(y) and more than 2 independent variables(x)

x1	x2	x3	y
5	100	2	100
10	200	3	250
17.5	700	5	250
20	400	7	450



$Y = b0 + b1X1 + b2x2 + b3x3$

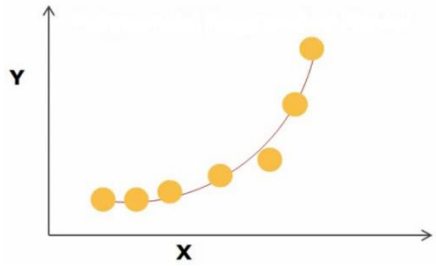
$b0 = \bar{y} - b1\bar{X}1 - b2\bar{x}2 - b3\bar{x}3$

Dimensions = P + 1

Polynomial Linear Regression

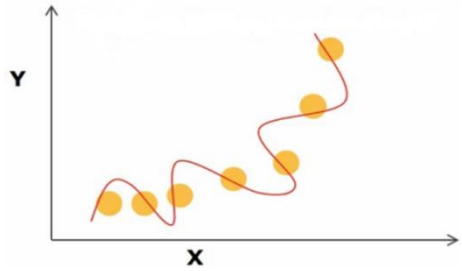
Variable = 1 & Degree = 2 :

$$Y = b_0 + b_1x + b_2x^2$$



Variable = 1 & Degree = n :

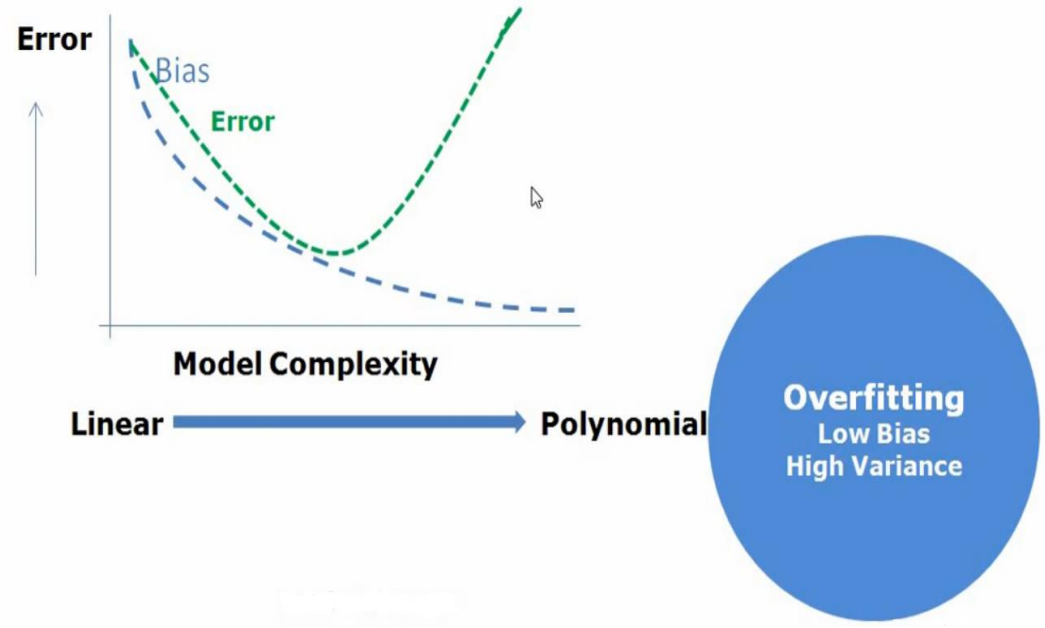
$$Y = b_0 + b_1x + b_2x^2 + b_3x^3 + \ldots\ldots + b_nx^n$$



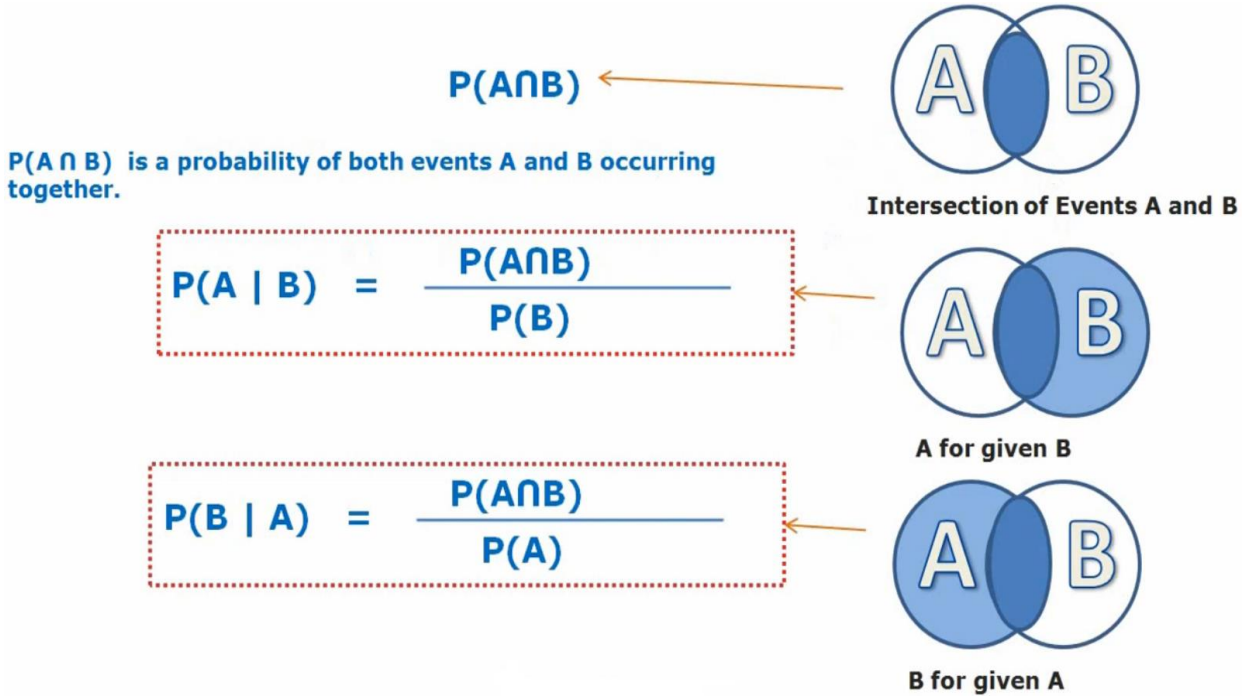
Variable = 2 & Degree = 2 :

$$Y = b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2 + b_4x_1^2 + b_5x_2^2$$

Relation:



Naive Bayes - Conditional Probability



$$P(A \cap B) = P(A)P(B \mid A)$$

$$P(A \cap B) = P(B)P(A \mid B)$$

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

Prior

Posterior

Likelihood

Marginal

$$P(A|B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

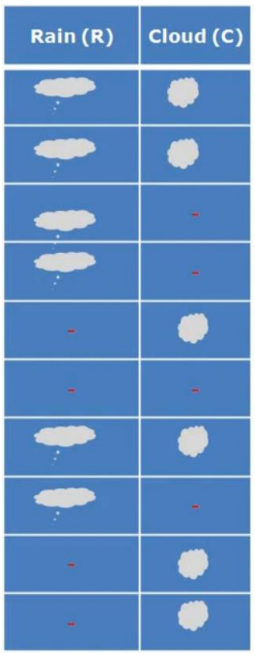
| - given that

A and B are events

P(A | B) is a **conditional** probability : Event A occurring given that B

P(B | A) is a **conditional** probability : Event B occurring given that A

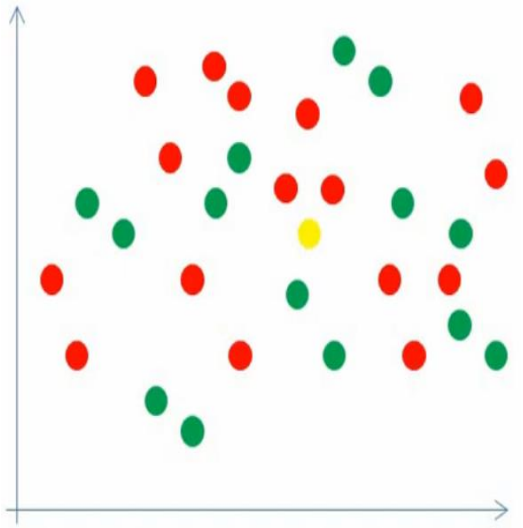
P(A) and P(B) are **marginal** probability for Events occurring of A and B



$$P(R|C) = \frac{P(C \mid R)\, P(R)}{P(C)} \qquad = \frac{(3/6)*(6/10)}{6/10}$$

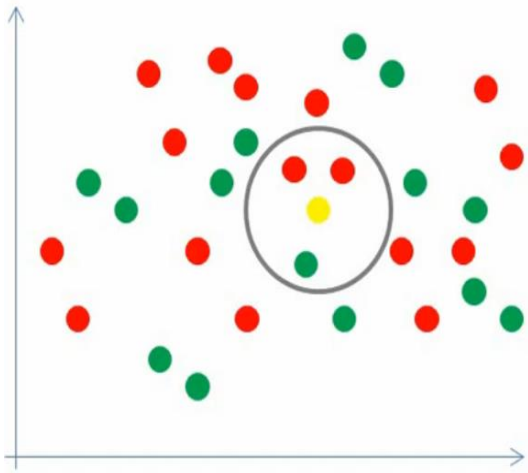
$$P(R|C) = 0.5$$

K-Nearest Neighbors (KNN) algorithm



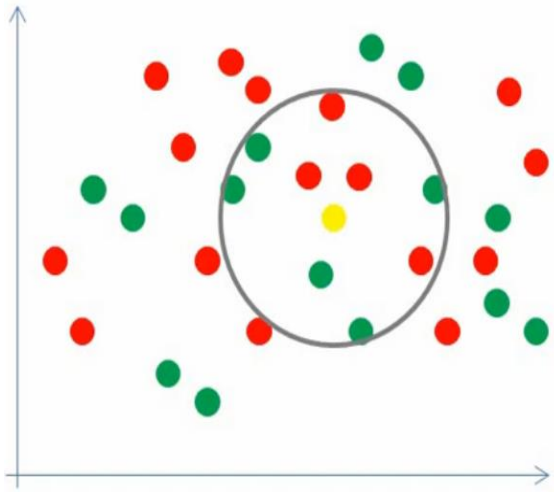
In KNN, finding the value of k is not easy.
Data scientists generally choose as an odd number if the number of classes is 2
another simple approach to select k is set k=sqrt(n).
there is not physical or biological way to determine the best value of K
Choosing the right value of k is a process Called **Parameter tuning**

Small k – low bias, high variance
Large k – high bias, low variance



For K = 3

Red – 2
Green - 1



For K = 9

Red – 4
Green - 5

Prediction : Height 161 cms weight 61kg

Step1 : Calculate the Euclidean distance between the new point and the existing points

Euclidean Distance

$$d^E(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Distance of all points from = (161 ,61)

SquareRoute((161-159))**2 + (61-57))**2)) = 4.47

SquareRoute((161-159))**2 + (61-58))**2)) = 3.60

K = 5

Step2 : Choose the value of K and select K neighbours closet to the new point.

Step3 : Count the votes of all the K neighbors Values

Here we are finding nearest neighbours For value k = 5

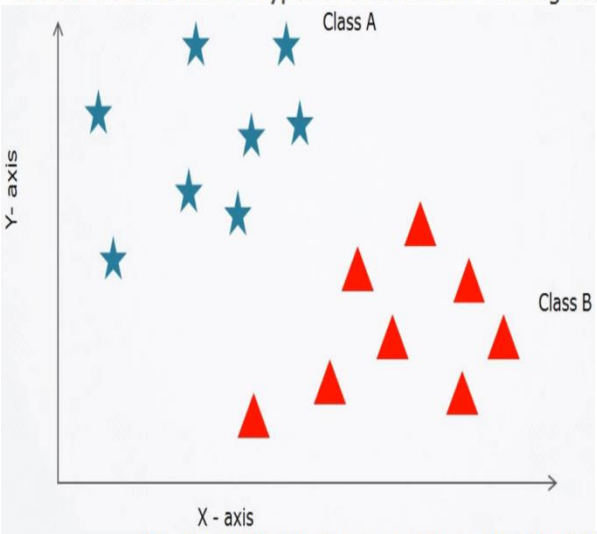
4 T-Shirt of size M and 1 T-Shirt of size L

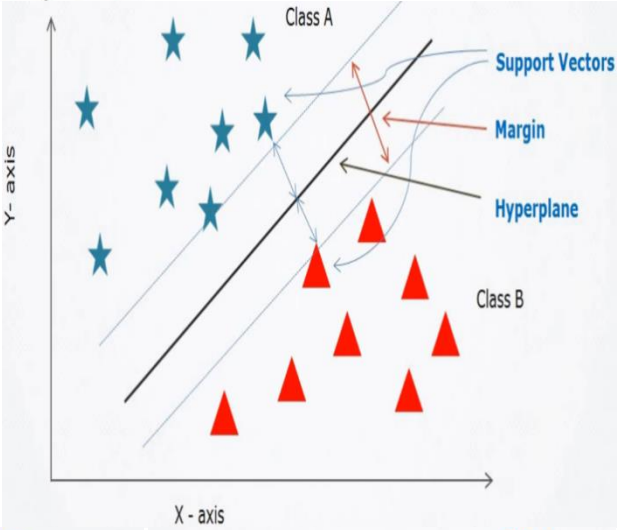
Output would be size M for provided input

Height in cms	Weight in kgs	T –Shirt Size	Euc. Distance
159	57	M	4.47
159	58	M	3.60
159	62	M	2.23
161	61	M	1.00
161	60	M	1.00
163	60	M	2.23
163	61	L	2.00
160	64	L	3.16
163	64	L	3.60
165	61	L	4.00
165	62	L	4.12
165	63	L	4.47
168	62	L	7.07
168	63	L	7.28
168	66	L	8.60

Support Vector Machine

- Consider for classification approach,
  but can be used in both types of classification and regression problems.

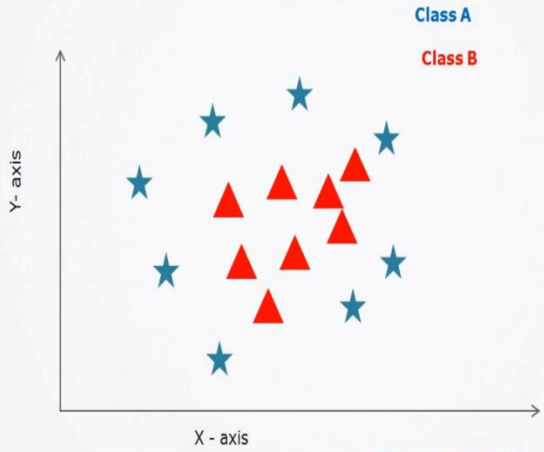


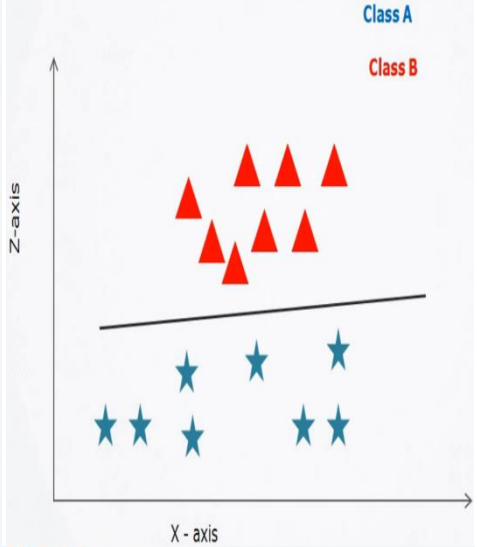


Dealing with non-linear and inseparable planes

Linear hyperplane cannot solve some problems

SVM uses a **kernel trick** to transform the input space(LD) to a higher dimensional(HD)





Tuning Hyperparameters

Kernel transform the given dataset input data into the required form.

- Linear - for linear hyperplane. can be used as normal dot product any two given observations
- Polynomial - for non-linear hyperplane . curved or nonlinear input space.
- Radial basis function (RBF) - for non-linear hyperplane RBF can map an input space in infinite dimensional space

Regularization : In SKLearn Python Library represent with "C"
smaller value - small-margin hyperplane
larger value - larger-margin hyperplane.

Gamma If value is high - exactly fit the training dataset
If value is high - loosely fit the training dataset

SVC(C=1.0, gamma=0.0, kernel='rbf'..)

SVM Advantages

SVM provide very high accuracy compared to other classifiers such as logistic regression, and decision trees.

SVM works well with a clear margin of separation and with high dimensional space

SVM Disadvantages

SVM is not suitable for large datasets because of its high training time

works poorly with overlapping classes

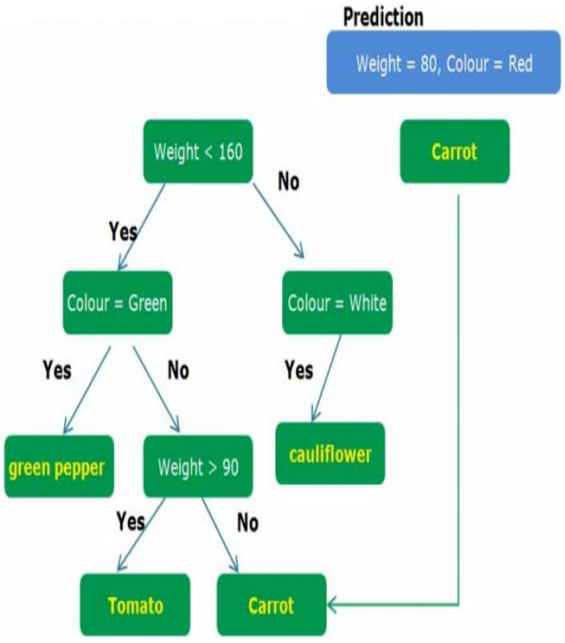
Sensitive to the type of kernel used

Decision tree

Decisions are based on **conditions**

Can be used in Classification and Regression Problem

Weight in gm	Colour	Vegetable
61	Red	carrot
800	White	cauliflower
160	Green	cucumber
165	Green	green pepper
100	Red	Tomato
2000	Orange	pumpkin



Root Node: The top-most decision node in a decision tree.

Decision Node: A tree node or parent node that splits into one ore more child nodes is called a decision node.

Leaf or Terminal Node: Bottom nodes that (generally speaking) don't split any further.

Splitting: Process of dividing a node into two or more child nodes.

Pruning: The opposite process of splitting. Removing the child nodes of a decision node is called pruning.

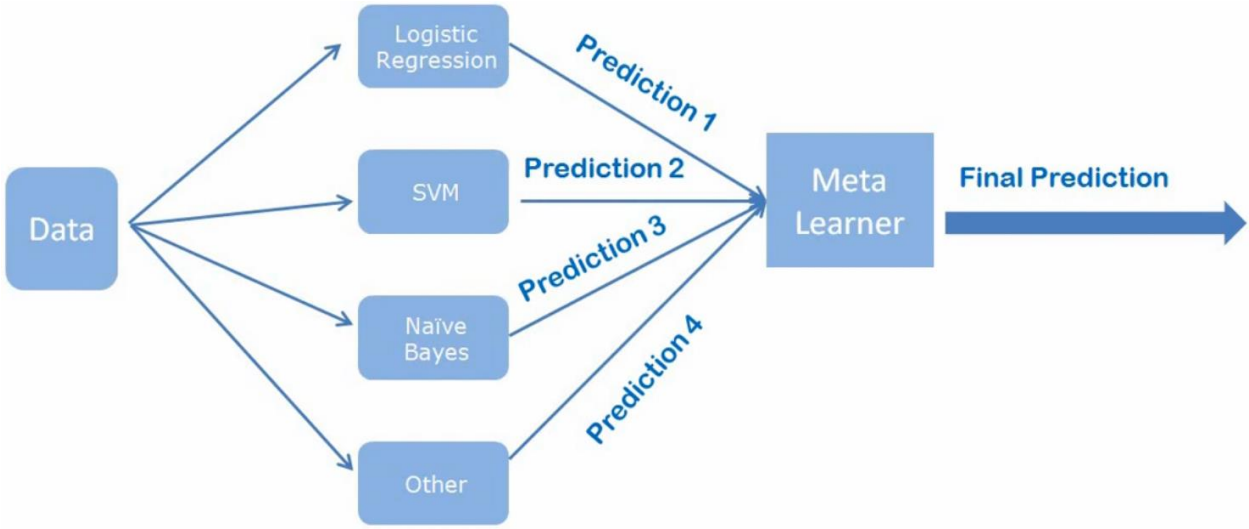
Ensemble learning - Random Forest

We use Ensemble learning to improve machine learning results.

An ensemble is a group of predictors that are trained and used for predictions

Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting)

Ensemble learning improves predictions .

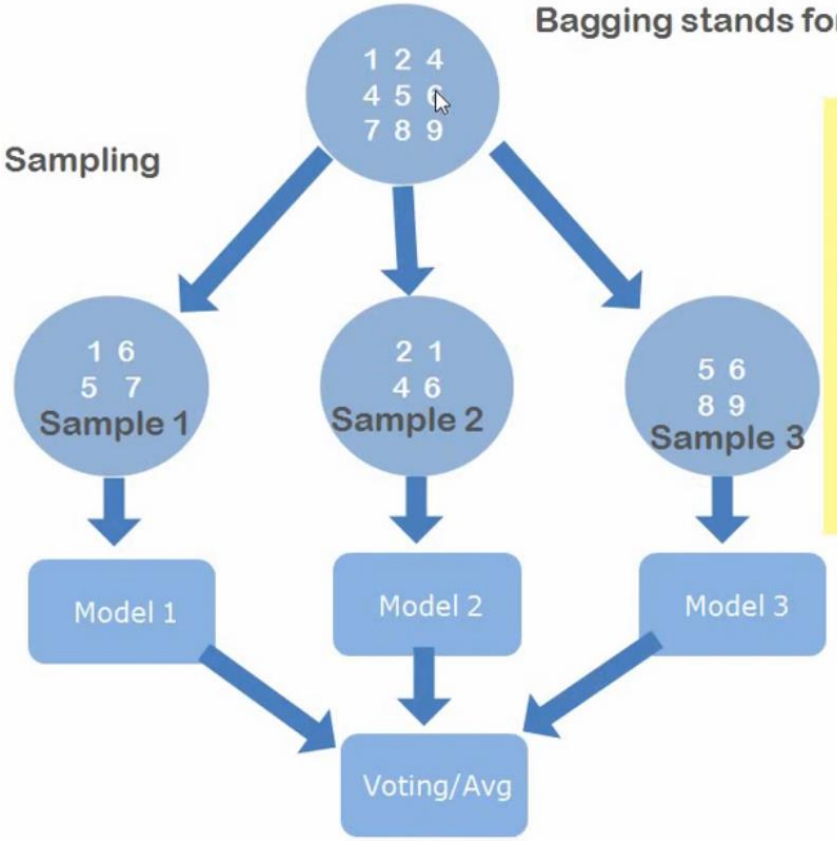


Basic Ensemble Techniques

- Max Voting
- Averaging
- Weighted Average
- Stacking
- Blending
- Bagging
- Boosting

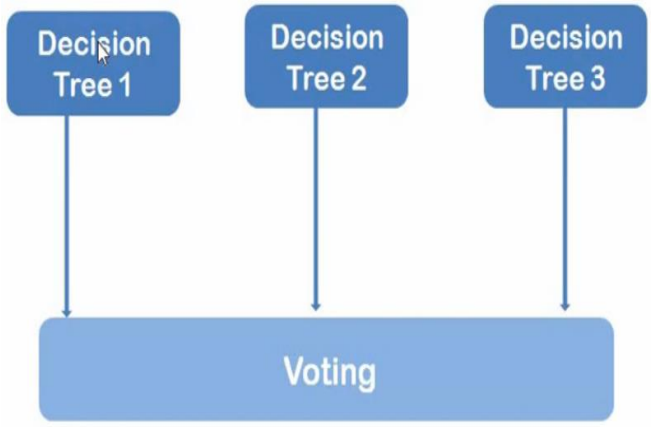
Bagging

Bagging stands for bootstrap aggregation.



Bootstrapping: This is a statistical technique that's used to generate random samples or bootstrap samples with replacement.

Random forest



Random forest is a supervised learning algorithm.

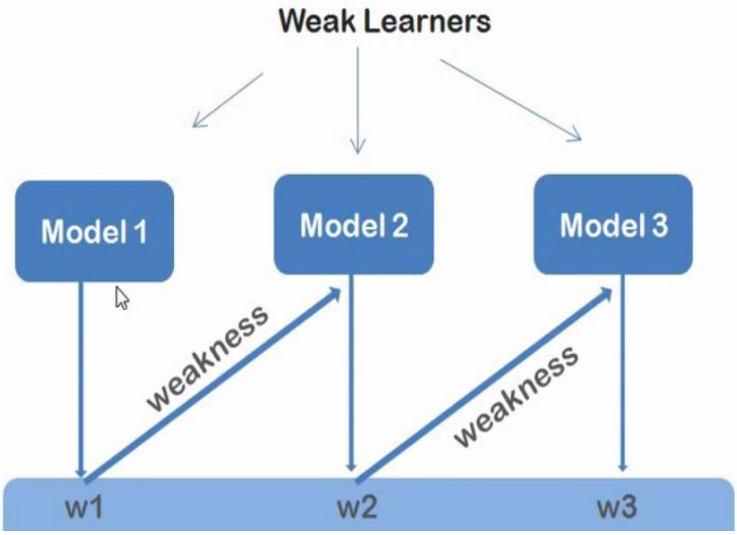
The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

You can also deal with regression tasks

Boosting

AdaBoost is a specific Boosting algorithm developed for classification problems

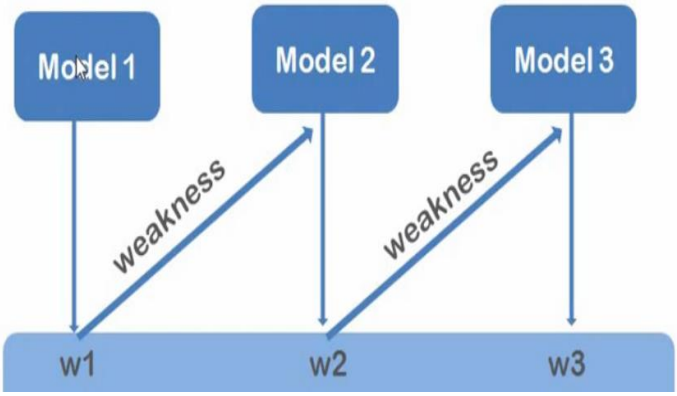


Boosting needs you to specify a weak model (e.g. regression, shallow decision trees, etc) and then improves it.

Adaboost combines multiple weak learners into a single strong learner.

AdaBoost

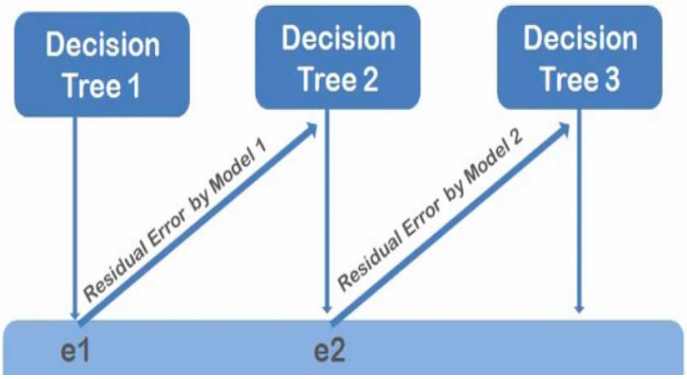
Adaptive Boosting



In AdaBoot Ensemble learning happens by adjusting the weights

1. AdaBoost identifies miss-classified data points
2. Increasing their weights
3. Next classifier pay extra attention to get them right.

Gradient Boosting

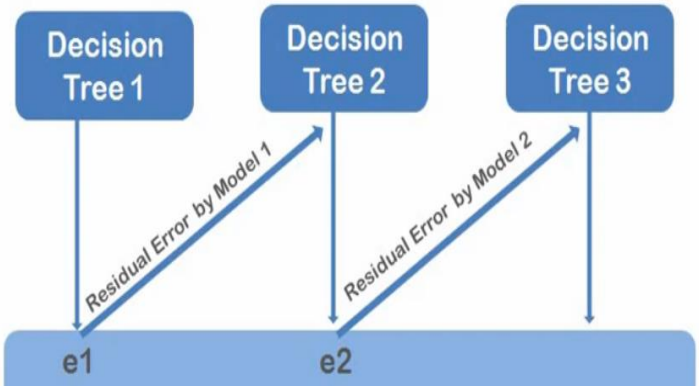


In Gradient Boosting Ensemble learning happens by optimizing the Loss.

Each new model gradually minimizes the loss function

1. Gradient boosting calculate loss instead of weights
2. Each time we fit a new estimator
3. Next classifier pay extra attention to get them right.

XGBoost
Extreme Gradient Boosting



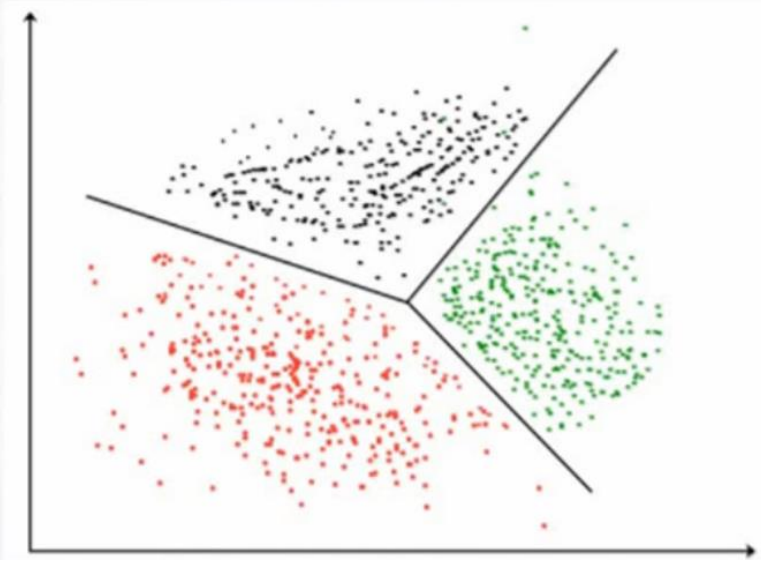
Gradient Boost is slow in performance to increase performance XGBoost was introduced.

Training is Fast
It use much hyperpermeters to tune the model

K-means Clustering

It is basically a type of Unsupervised learning method

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar to each other than to those in other groups (**clusters**)



K-means clustering algorithm

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data.

Goal

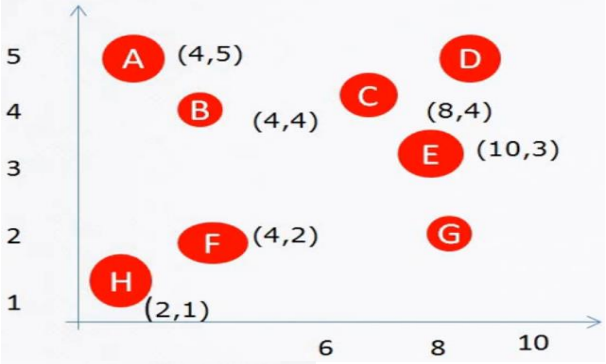
goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*

How does it works?

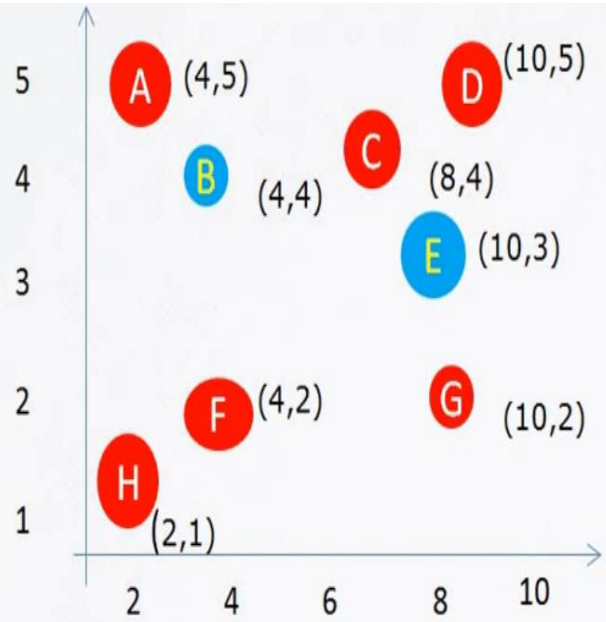
- Specify number of clusters *K*
- Initialize centroids randomly
- Keep iterating until there is no change to the centroids

- Compute the Euclidean distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

Targeted Marketing



Need to open 2 Branches



N = 2 Centroid

C1 = (4,4)
C2= (3,10)

C1 = (4,4), (4,5), (8,4),(2,1),(4,2))

C2 = (10,2),(10,2),(10,5)

Euclidean Distance

$$d^E(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Distance of all points from C1 = (4,4)

B - SquareRoute(Sum((4-4))**2 + (4-4))**2)) = 0

A - SquareRoute(Sum((4-4))**2 + (5-4))**2)) = 1

G- SquareRoute(Sum((10-4))**2 + (2-4))**2)) = 6.32

C - SquareRoute(Sum((8-4))**2 + (4-4))**2)) = 4

H- SquareRoute(Sum((2-4))**2 + (1-4))**2)) = 3.6

E- SquareRoute(Sum((4-4))**2 + (2-4))**2)) = 2

D- SquareRoute(Sum((10-4))**2 + (5-4))**2)) = 6.08

F- SquareRoute(Sum((4-4))**2 + (2-4))**2)) = 2

Distance of all points from C2 = (3,10)

B - SquareRoute(Sum((4-10))**2 + (4-3))**2)) = 6.08

A - SquareRoute(Sum((4-10))**2 + (5-3))**2)) = 8.24

G- SquareRoute(Sum((10-10))**2 + (2-3))**2)) = 1

C - SquareRoute(Sum((8-10))**2 + (4-3))**2)) = 8.06

H- SquareRoute(Sum((2-10))**2 + (1-3))**2)) = 8.06

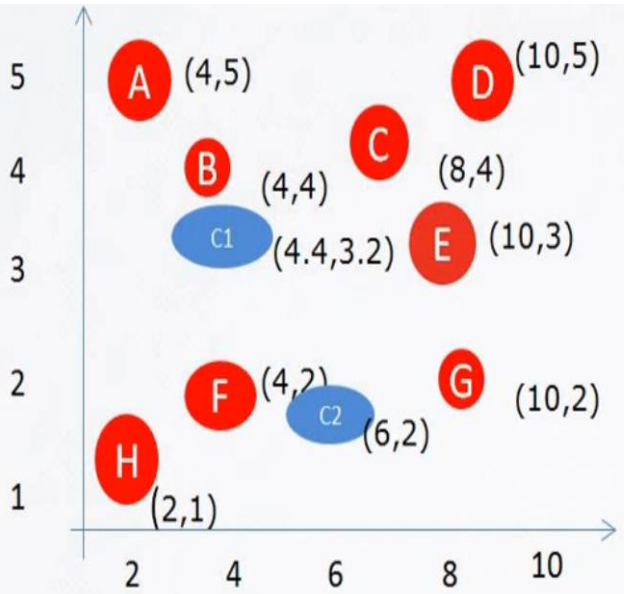
E- SquareRoute(Sum((4-10))**2 + (2-3))**2)) = 6.08

D- SquareRoute(Sum((10-10))**2 + (5-3))**2)) = 2

F- SquareRoute(Sum((4-10))**2 + (2-3))**2)) = 8.06

C1 mean = (4+4+8+2+4)/5, (4+5+4+1+2)/5 = **(4.4,3.2)**

C2 mean = (10+10+10)/5, (5+3+2)/5 = **(6,2)**



N = 2

Centroid

C1 = (4.4,3.2)

C2= (6,2)

C1 = (4,4), (4,5), (2,1),(4,2))

C2 = (10,2), (8,4),(10,3),(10,5)

Distance of all points from C1 = (4.3,3.2)

B - 0

A - 1

G- 6.32

C - 4

H- 3.6

E- 2

D- 6.08

F- 2

Distance of all points from C2 = (6,2)

B – 4.4

A – 3.6

G- 4

C - 2

H- 4.42

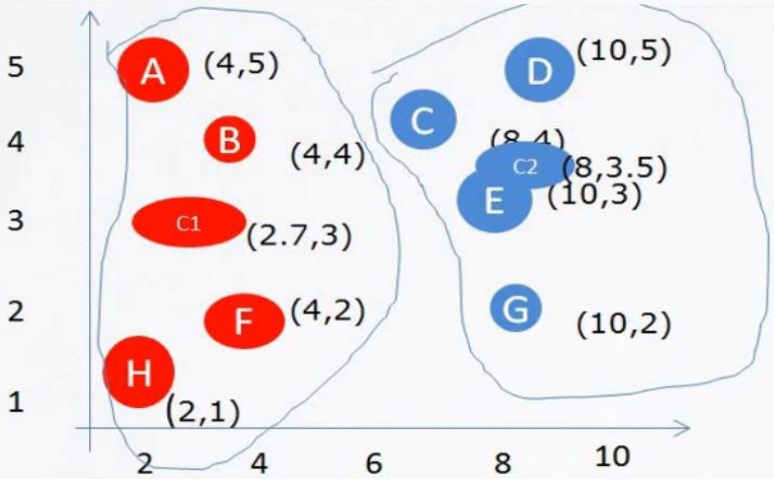
E- 4.2

D- 5

F- 2

C1 mean = (4+4+2+1)/4, (4+5+1+2)/4 = (2.7, 3)

C2 mean = (10+8+10+4)/4, (2+4+3+5)/4 = (8,3.5)



N = 2

Centroid

C1 = (2.7,3)

C2= (8,3.5)

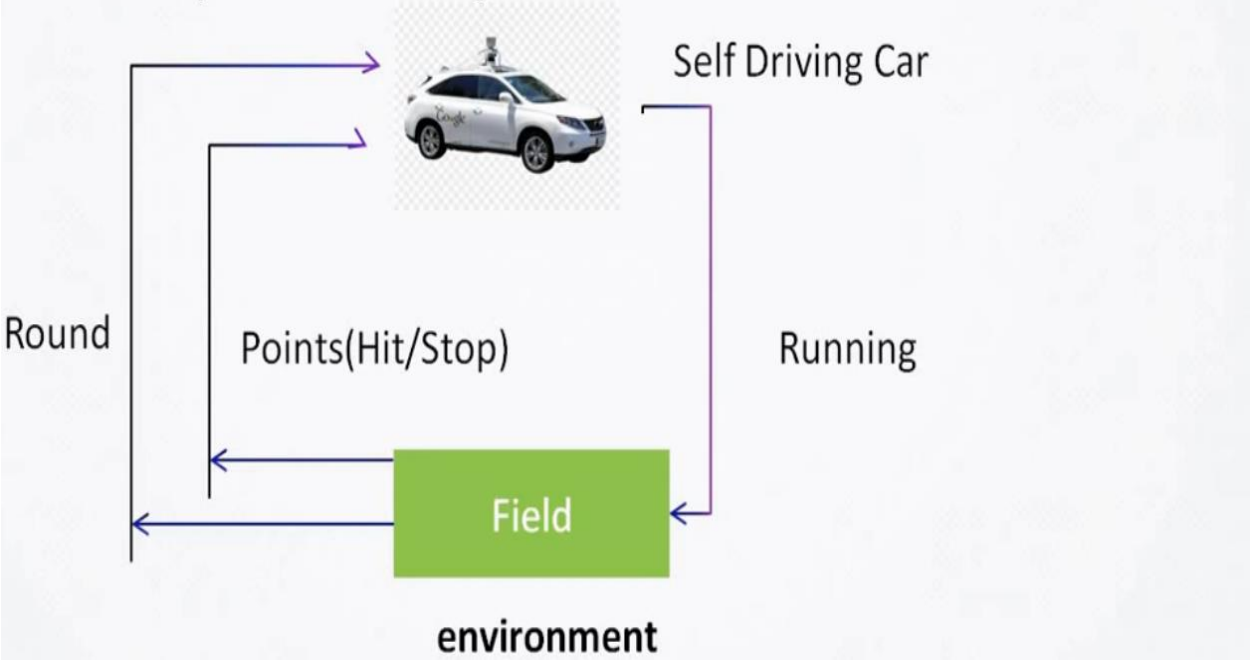
C1 = (4,4), (4,5), (2,1),(4,2))

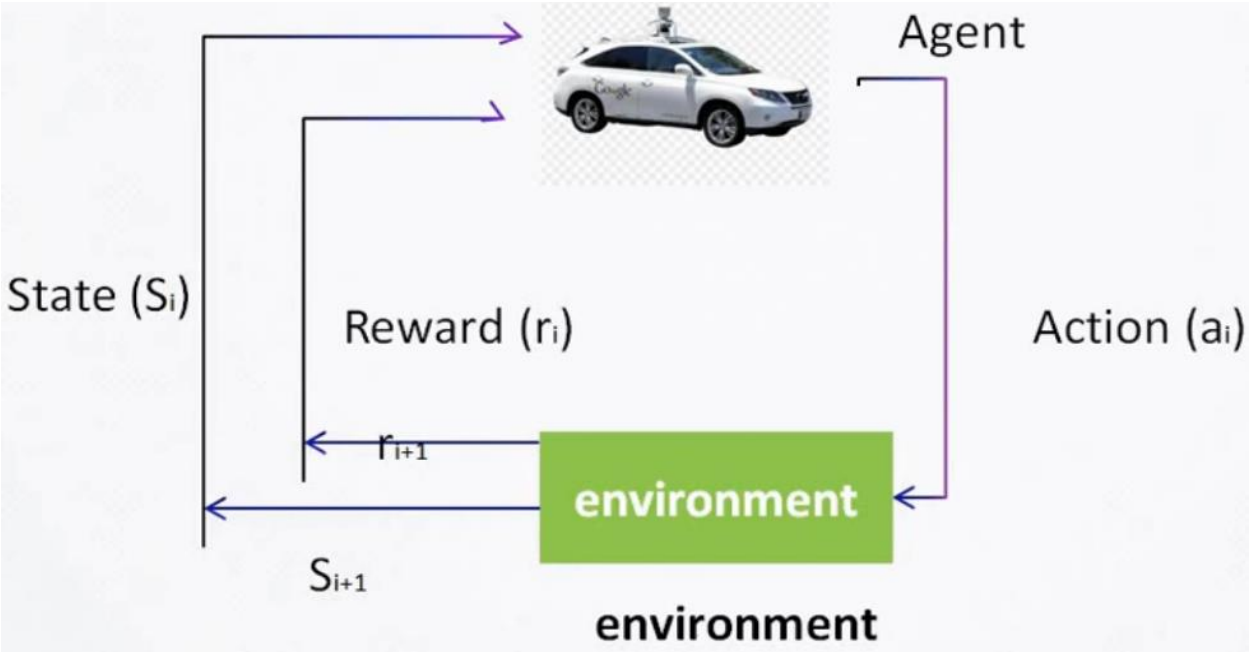
C2 = (10,2), (8,4),(10,3),(10,5)

Reinforcement

-> Learn From Mistakes

Reinforcement learning is fairly different when compared to supervised and unsupervised learning.





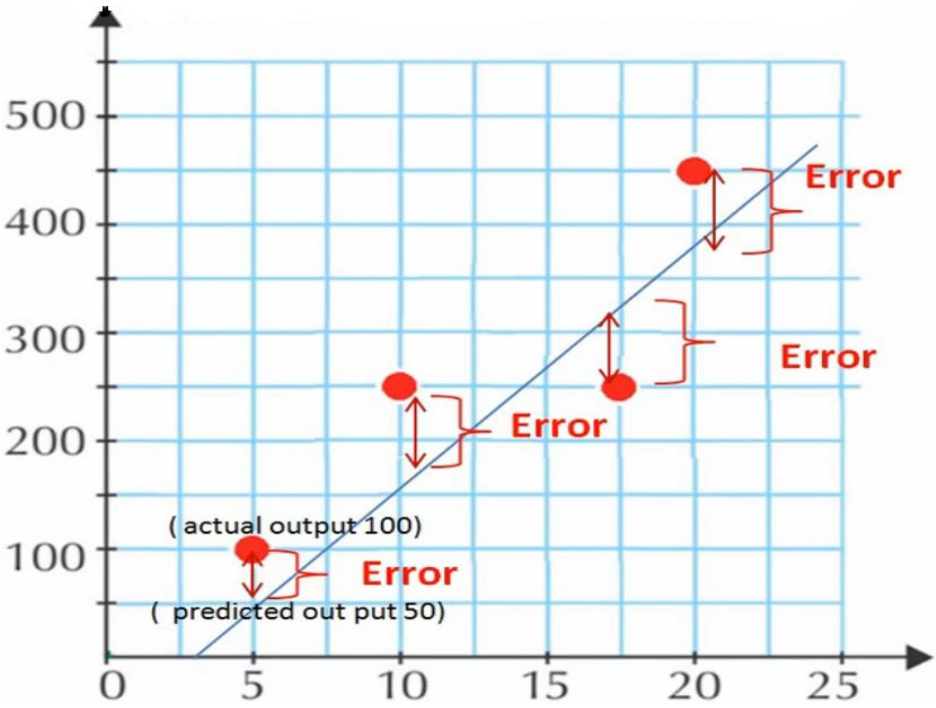
Part-3 (Machine Learning Evaluation)

x	y
5	100
10	250
17.5	250
20	450

Mean absolute error (MAE)

$$\frac{1}{\text{Total data points}} \sum \left| \text{Actual output} - \text{predicted output} \right|$$

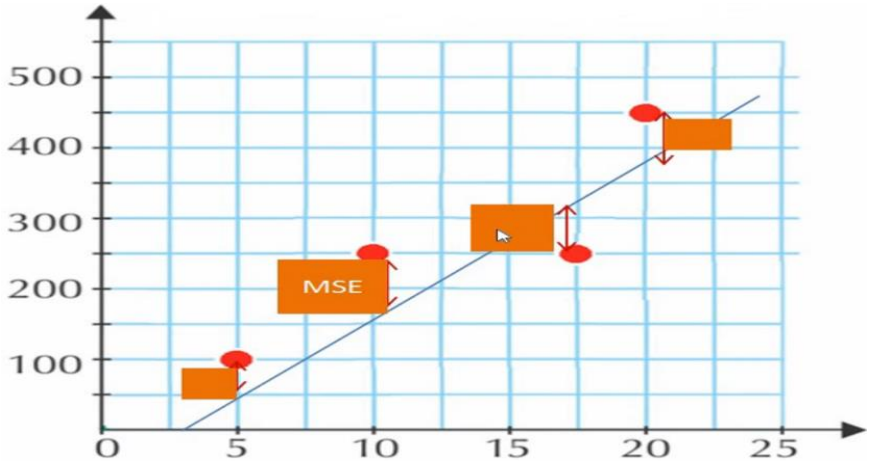
$$\frac{1}{n} \sum \left| y - \hat{y} \right|$$



Mean square error (MSE)

$$\frac{1}{\text{Total data points}}\sum (\text{Actual output} - \text{predicted output})^2$$

$$\frac{1}{n}\sum (y - \hat{y})^2$$



Root Mean square error (RMSE)

$$\sqrt{\frac{1}{\text{Total data points}}\sum (\text{Actual output} - \text{predicted output})^2}$$

$$\sqrt{\frac{1}{n}\sum (y - \hat{y})^2}$$

Mean bias error (MBE)

$$\frac{1}{\text{Total data points}}\sum (\text{Actual output} - \text{predicted output})$$

$$\frac{1}{n}\sum (y - \hat{y})$$

Max Error

$$\text{Max}\left(\left| \text{Actual output} - \text{predicted output} \right| \right)$$

$$\text{Max}\left(\left| y - \hat{y} \right| \right)$$

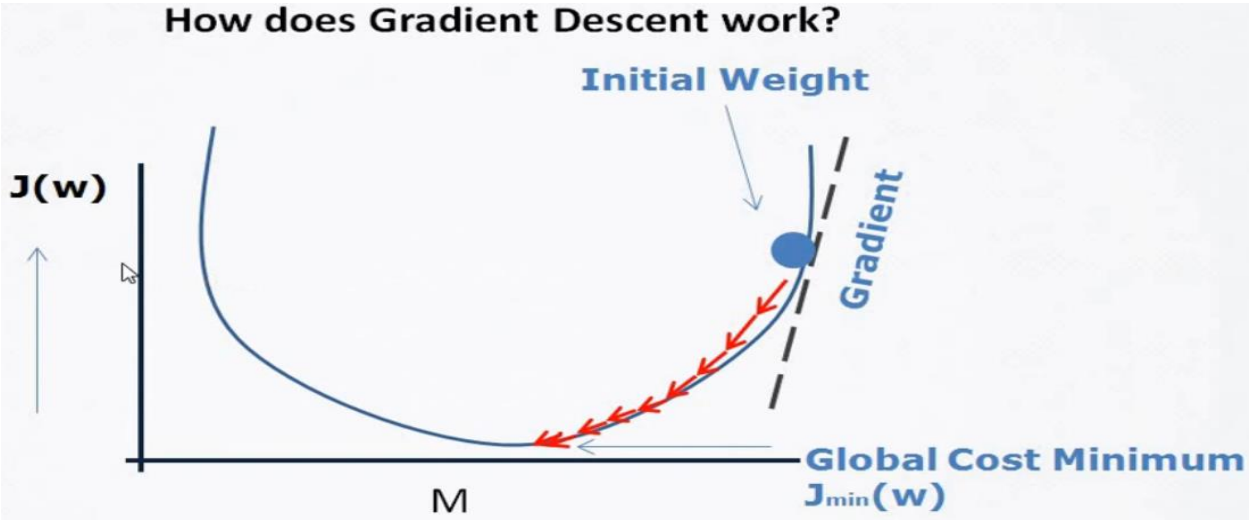
Gradient descent

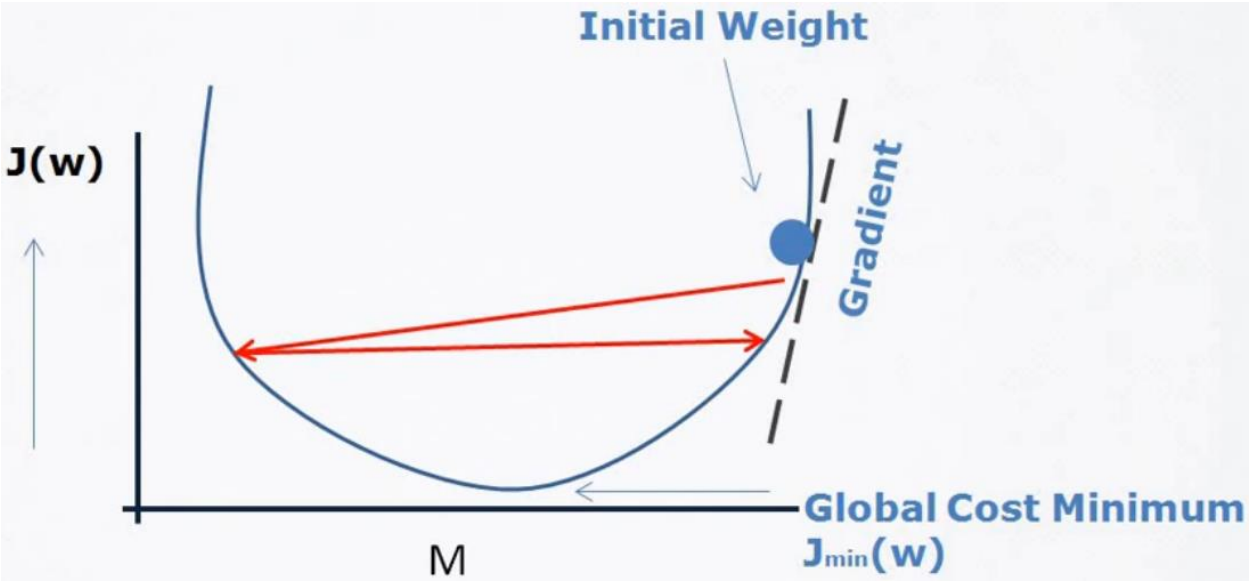
Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function(cost).

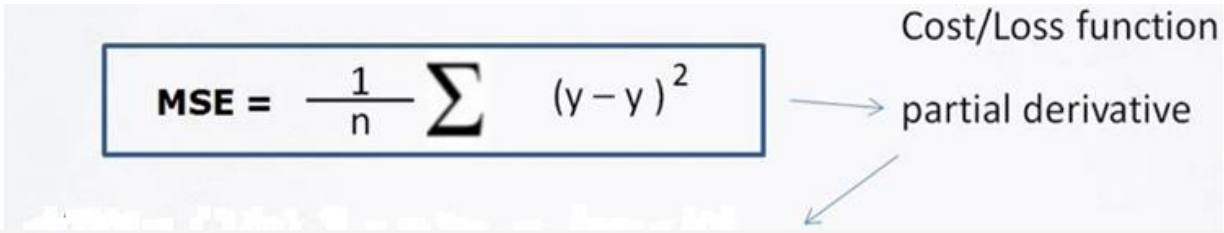
Gradient descent is an algorithm that finds best fit line for given training dataset.

Y = MX + C

$$M = \frac{y2 - y1}{x2 - x1}$$








```
import numpy as np

def Gradient_Descent(x,y):
    M = 0
    B0 = 0
    iterations = 10000
    n = len(x)
    learning_rate = 0.08

    for i in range(iterations):
        y_predicted = M * x + B0
        cost = (1/n) * sum ([val**2 for val in (y - y_predicted)])
        md = -(2/n)* sum(x*(y - y_predicted))
        bd = -(2/n)* sum(y - y_predicted)
        M = M - learning_rate * md
        B0 = B0 - learning_rate * bd
        print("m {}, b {}, cost {} iteration {}". format(M,B0,cost, i))

x = np.array([1,2,3,4,5])
y = np.array([5,7,9,11,13])

Gradient_Descent(x,y)
```

$d/dm = (2/n) \sum - x_i (y_i - (mx_i + b))$

$d/db = (2/n) \sum - (y_i - (mx_i + b))$

$m = m - \alpha \frac{d}{dm}$

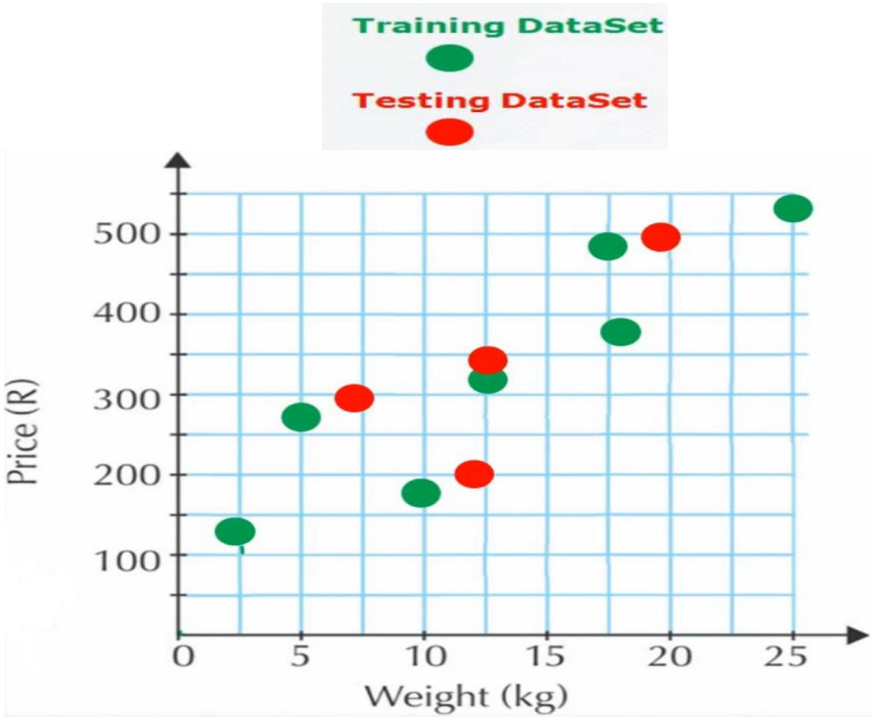
$c = c - \alpha \frac{d}{dc}$

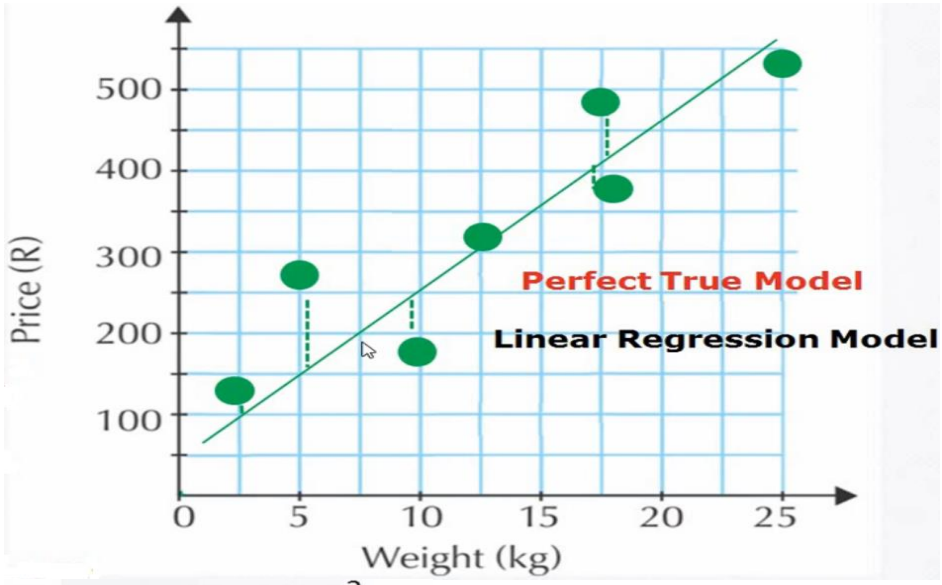
α = learning rate

Output:

```
m 2.162896162432927, b 2.4118931669446906, cost 0.06297135809402594 iteration 9991
m 2.16287413260285, b 2.4119727016212957, cost 0.0629543269437137 iteration 9992
m 2.1628521057520538, b 2.412052225541752, cost 0.06293730039962381 iteration 9993
m 2.1628300818801356, b 2.412131738707514, cost 0.06292027846051029 iteration 9994
m 2.162808060986692, b 2.4122112411200356, cost 0.06290326112512823 iteration 9995
m 2.1627860430713213, b 2.4122907327807717, cost 0.06288624839223193 iteration 9996
m 2.16276402813362, b 2.412370213691176, cost 0.06286924026057726 iteration 9997
m 2.162742016173185, b 2.4124496838527025, cost 0.06285223672891908 iteration 9998
m 2.162720007189615, b 2.412529143266805, cost 0.0628352377 601374 iteration 9999
```

Bias and Variance



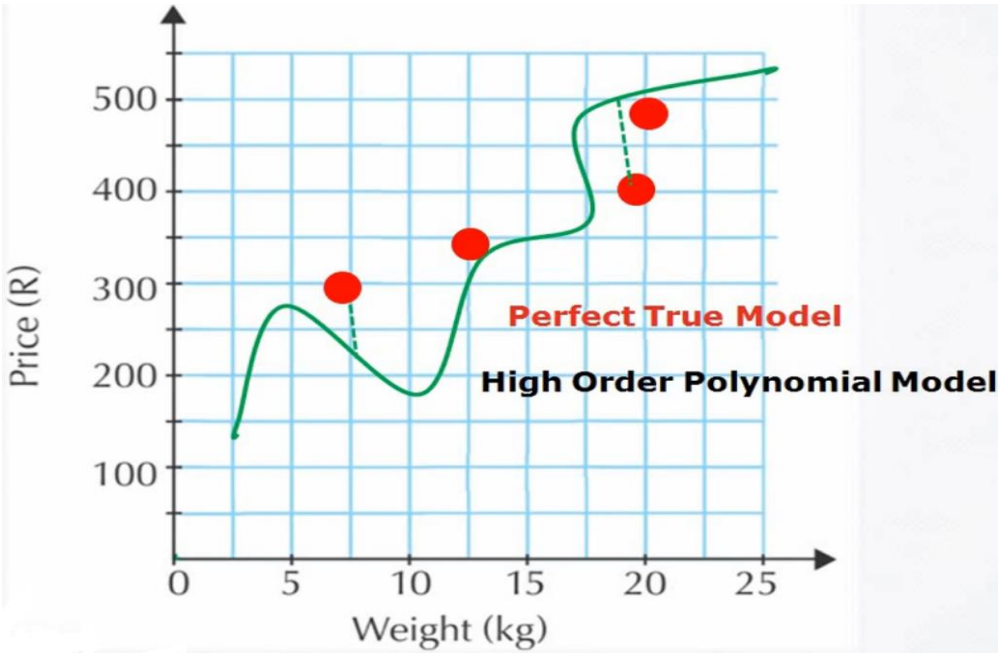


$$\sum (\text{Actual - predicted})^2$$

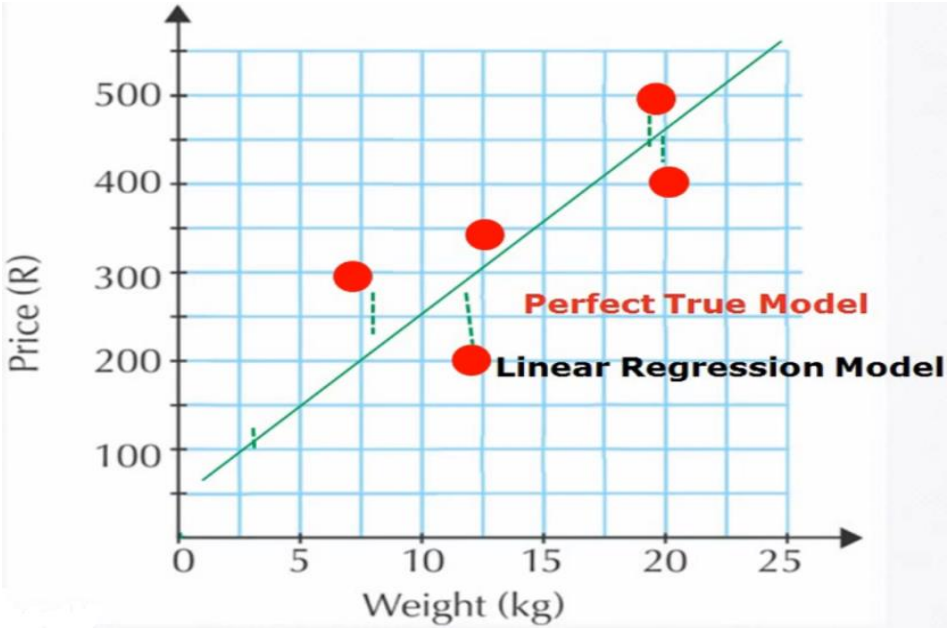
$$\sum (y - \hat{y})^2$$

SUM Of SQUARE (Large)

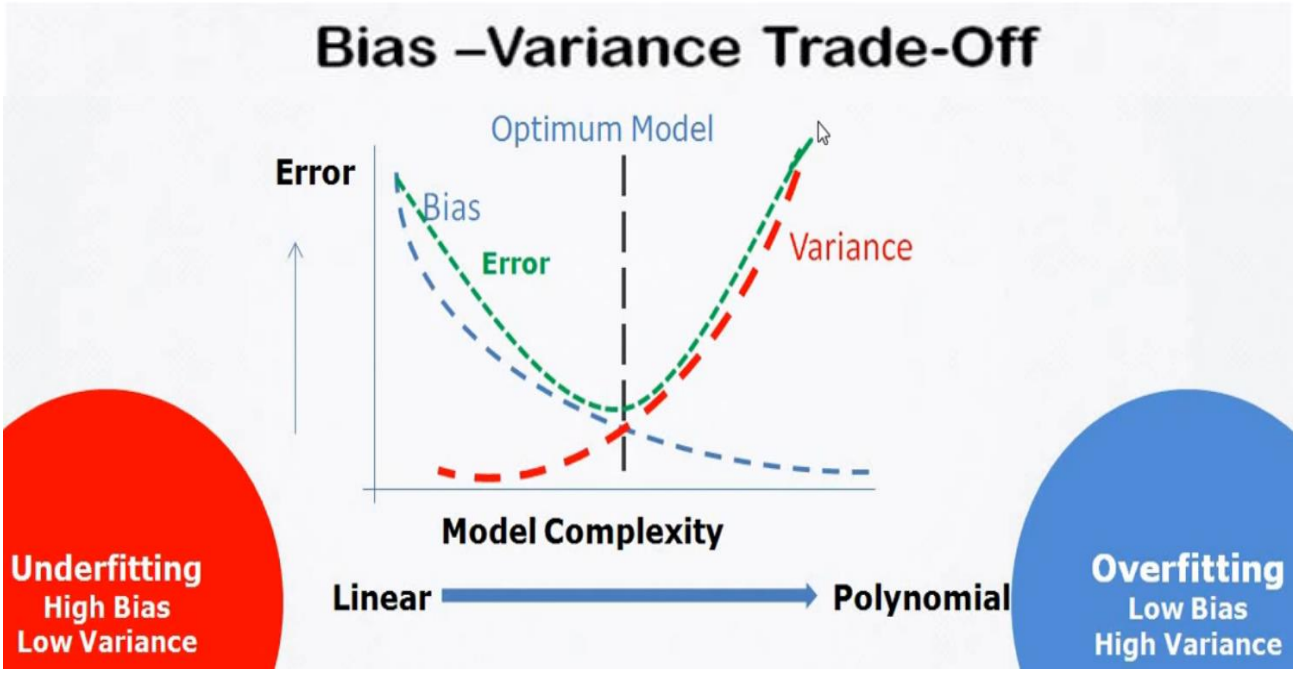
SUM Of SQUARE (Small ~ 0)



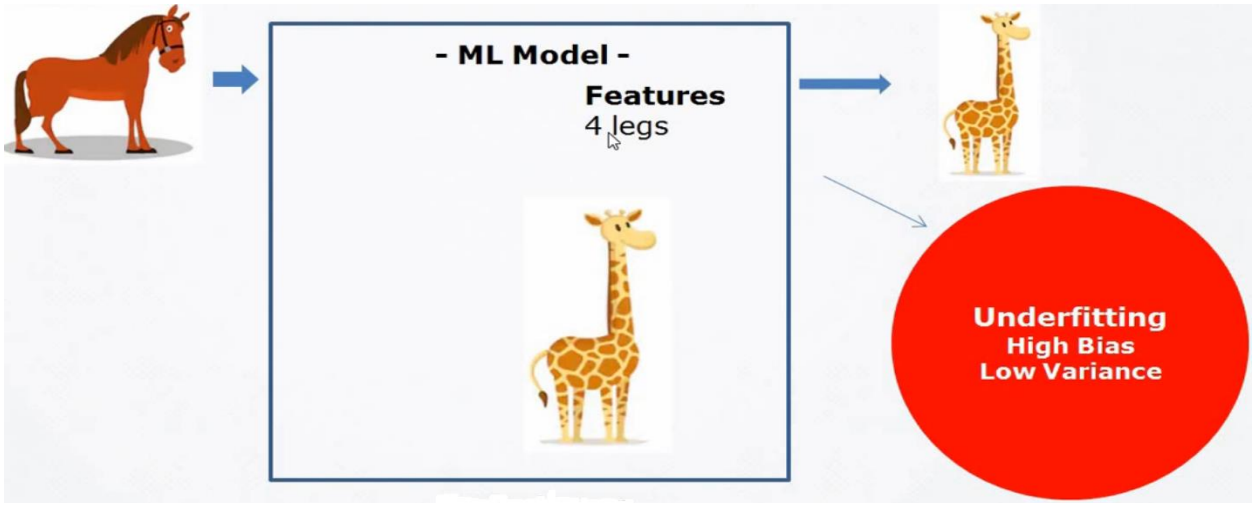
SUM Of SQUARE (Large)

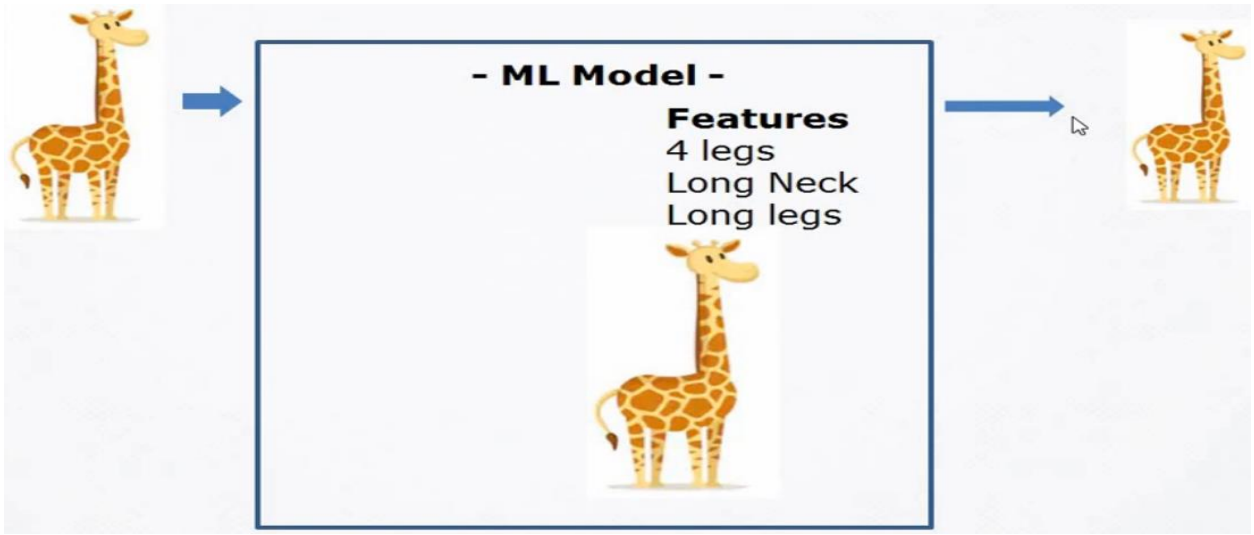


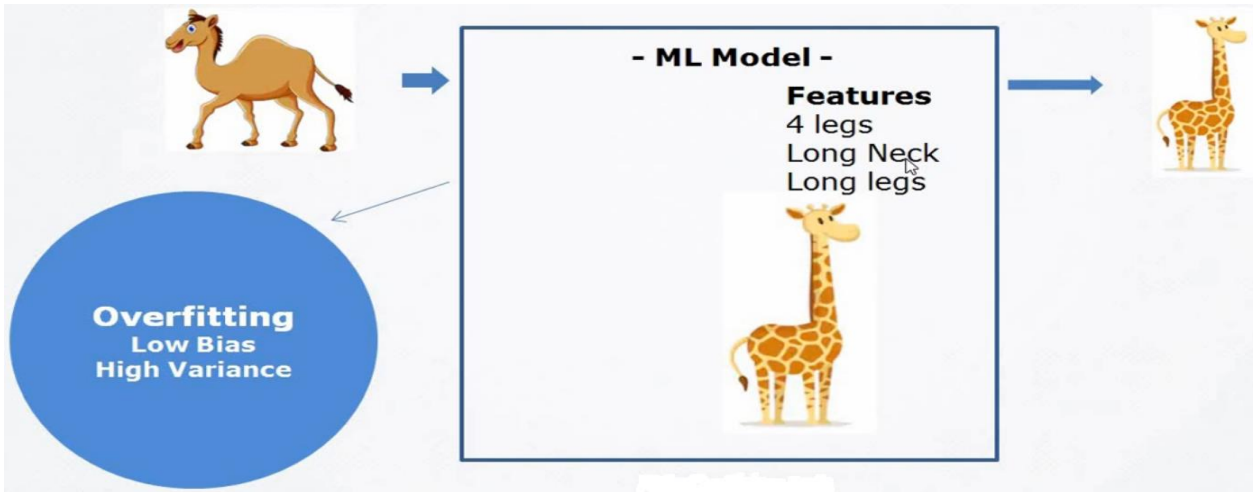
SUM Of SQUARE (Small)



Overfitting and Underfitting







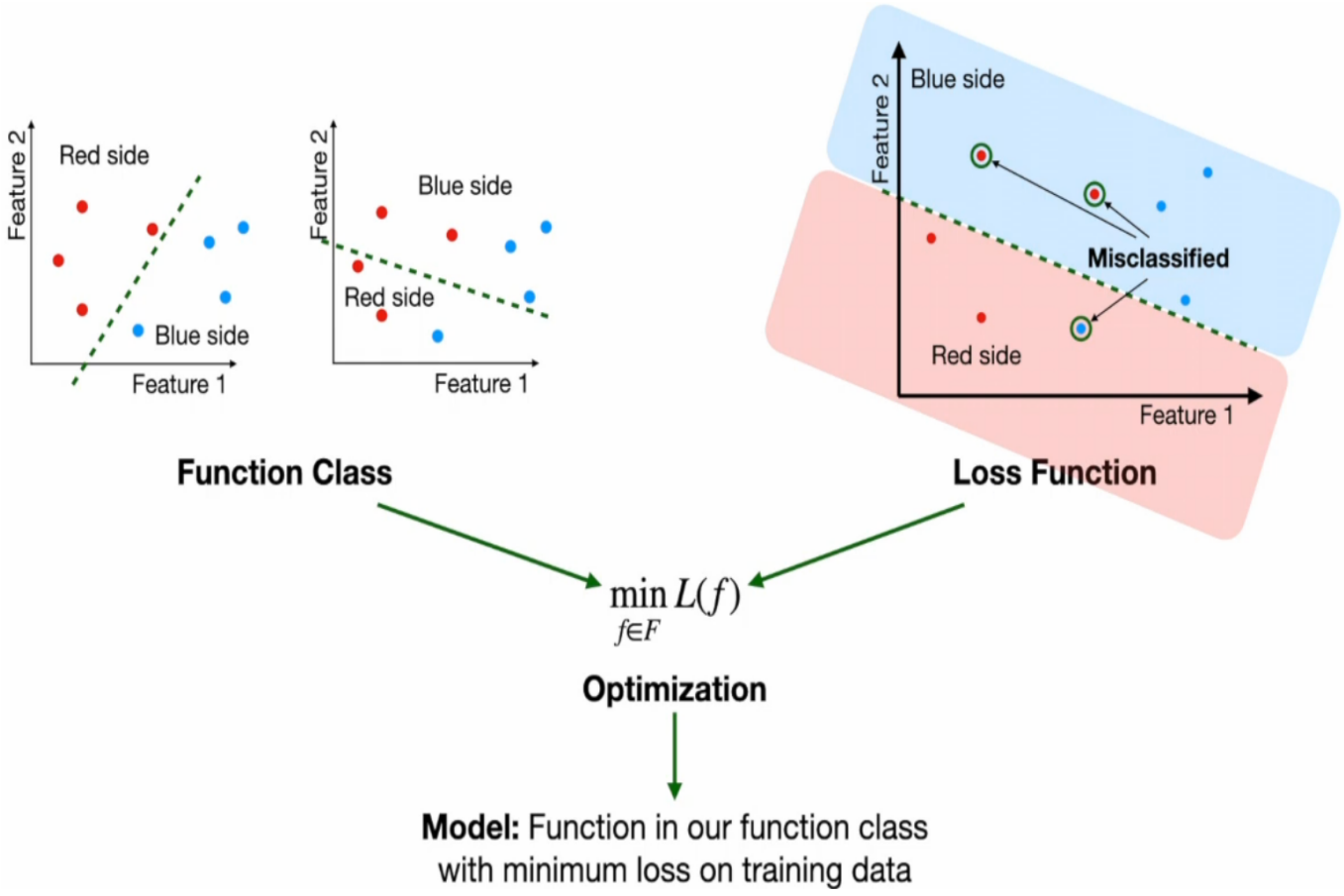
Regularization

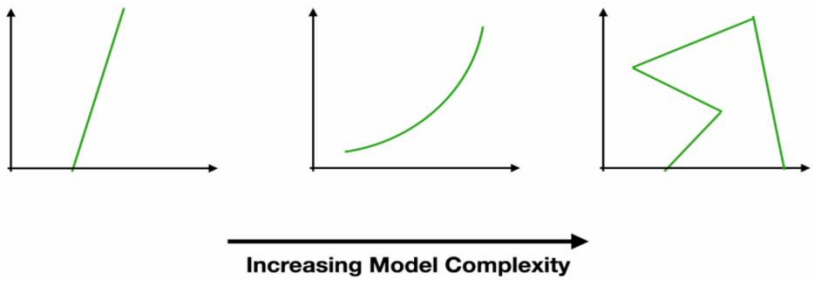
- Regularization is a technique that helps to avoid overfitting and also make a predictive model more understandable.
- to avoid the risk of overfitting / underfitting.

Overfitting ➡ Try to increase the regularization rate.

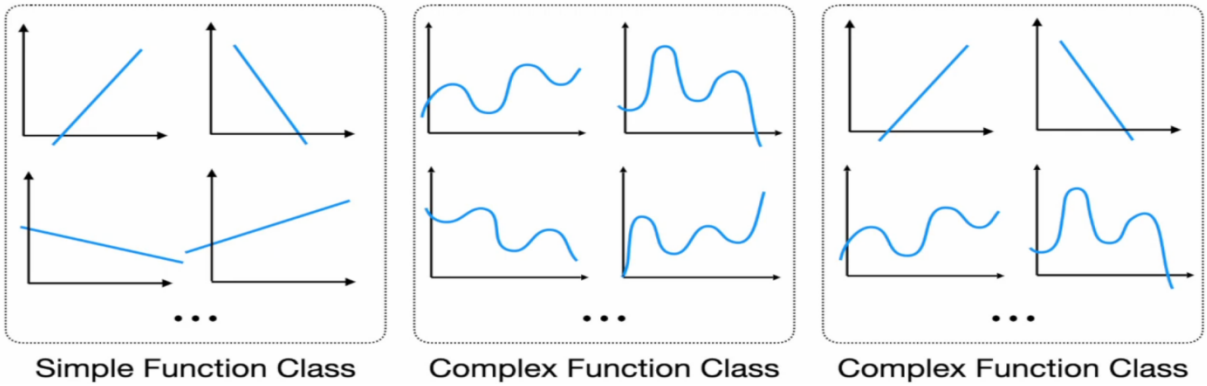
Underfitting ➡ Try to decrease the regularization rate.

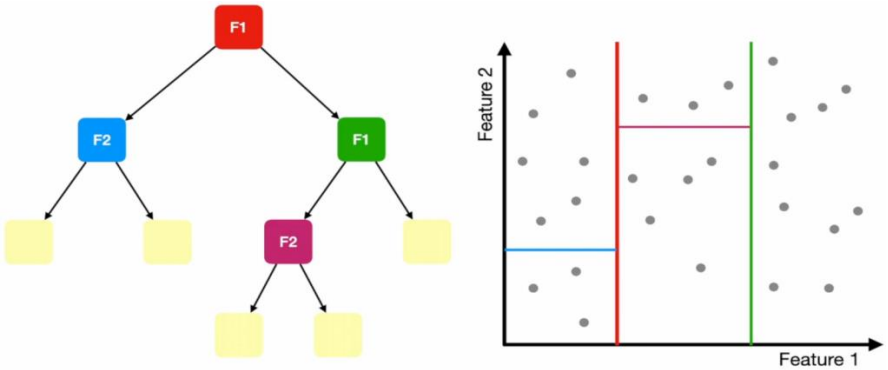
1. ***Ridge Regression***
2. **Lasso Regression (L1 Regression)**



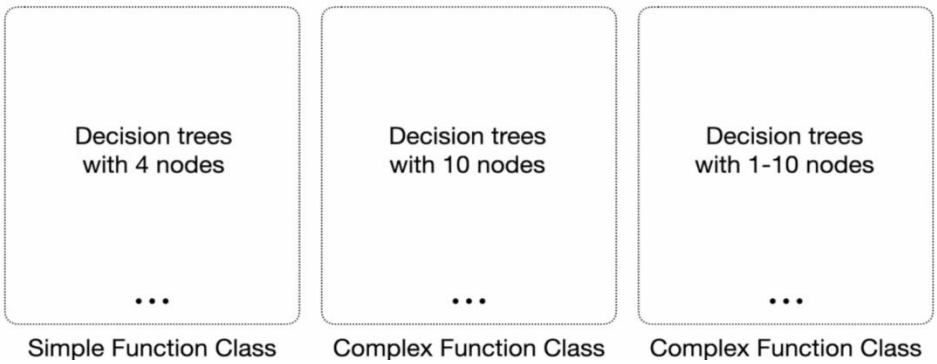


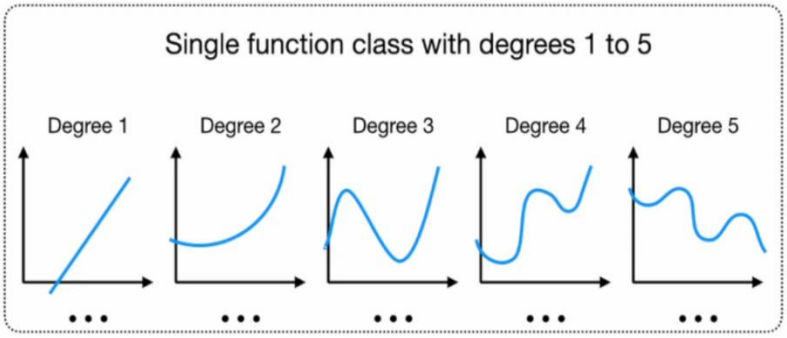
Complexity of $F = \max_{f \in F} \{\text{Complexity of } f\}$





Complexity of $F = \max_{f \in F} \{\text{Complexity of } f\}$





$$\min_f \{L(f) + C(f)\}$$

Training loss of f

Term that measures complexity of f

Regularization

Degree	Training Loss	Additional term = Degree	Objective Function value
1	4.2	1	5.2
2	2.5	2	4.5
3	1.2	3	4.2
4	0.4	4	4.4
5	0.1	5	5.1

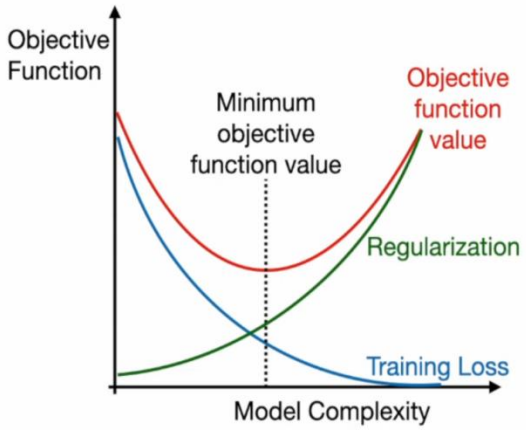
High training loss

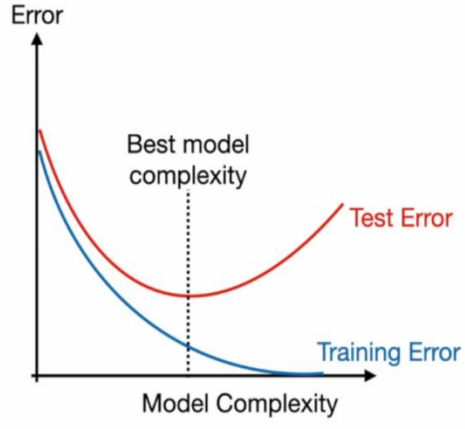
Training loss and degree are both not very large
=> Solution of the optimization problem

High degree

Degree	Training Loss	Additional term = Degree	Objective Function value
1	4.2	1	5.2
2	2.5	2	4.5
3	1.2	3	4.2
4	0.4	4	4.4
5	0.1	5	5.1

Increase in penalty = 1
Decrease in loss = 1.7
=> Net gain = 0.7





Confusion Matrix

Confusion matrix or error matrix is a specific table that is used to measure the performance of an algorithm.

To summarize the performance of a classification algorithm.

It is mostly used in supervised learning;
in unsupervised learning, it's called the matching matrix.

Two parameters:
•Actual
•Predicted

Predicted

27 (total)	Yes	No	
Actual Yes	12 (TP)	3 (FN)	15
No	2 (FP)	10 (TN)	12
	14	**13**	

Confusion Matrix

Predicted

45 (total)	Cat	Dog	Fish	
Actual Cat	12	3	2	17
Dog	2	10	1	13
Fish	0	1	14	15
	14	14	17	

Confusion Matrix

True positives are those cases which correctly get classified as True and are True.
False positives are those cases which wrongly get classified as True but are False.
False negatives are those cases which wrongly get classified as False but are True.

$$\text{Accuracy} = \frac{TP + TN}{\text{Total}} = \frac{12 + 10}{27} = 0.81$$

$$\text{Error Rate} = (1 - \text{Accuracy}) = 1 - .81 = 0.19$$

$$\text{Error Rate} = \frac{FP + FN}{\text{Total}} = \frac{2 + 3}{27} = 0.19$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{12}{14} = 0.85$$

(Predicted Yes)

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{12}{15} = 0.80$$

(actual yes)