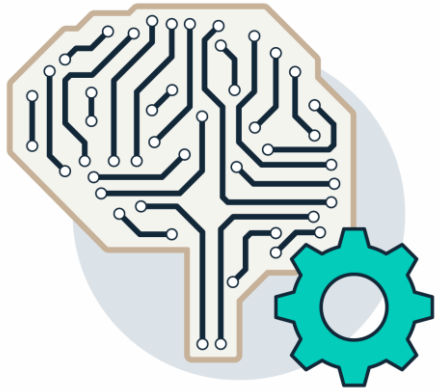
ShineBlue

VS



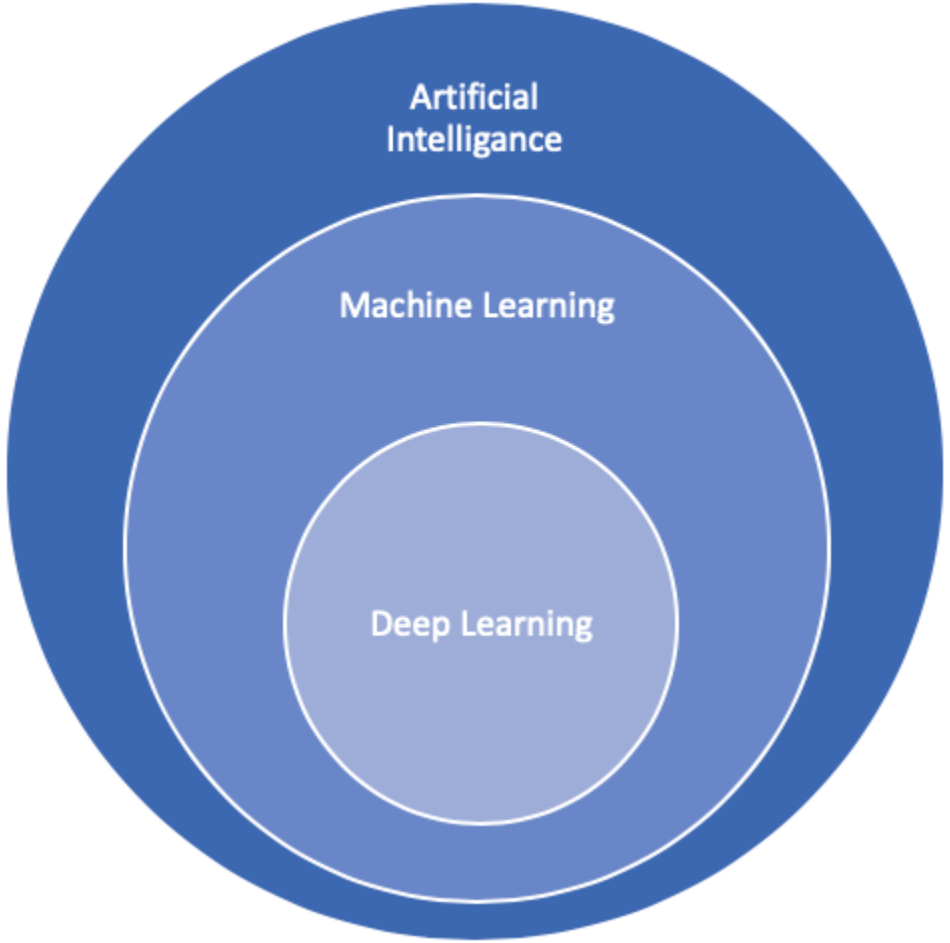
VS

Artificial Intelligence

Machine Learning

Deep Learning

AI vs ML vs DL



Machine Learning is a subset of Artificial Intelligence

Deep Learning is a subset of Machine Learning

What is Artificial Intelligence?

Artificial Intelligence is a branch of Computer Science that is concerned with building smart & intelligent Machines

Non – intelligent machines

Intelligent machines

Google Assistant

Machine Learning

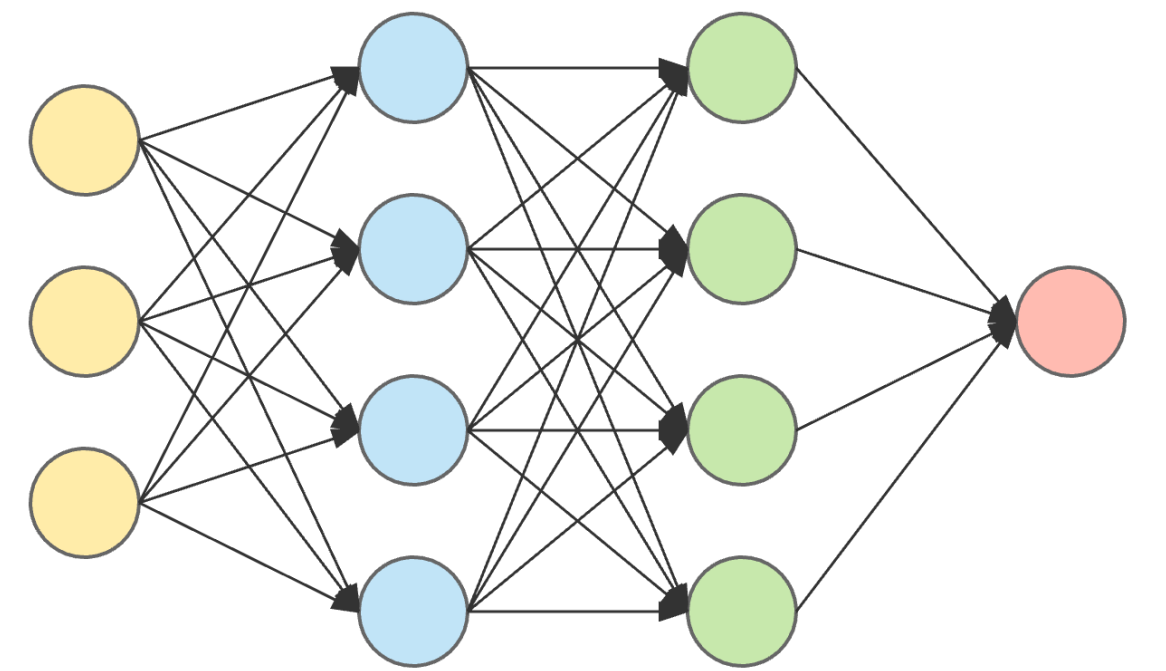
Machine Learning is a technique to implement AI that can learn from the data by themselves without being explicitly programmed.

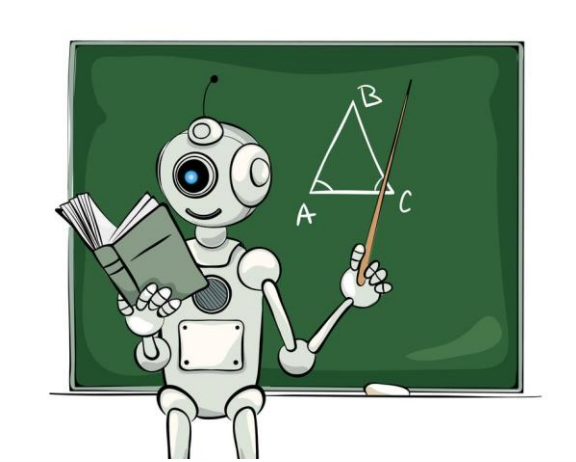
Iron Man

Captain America

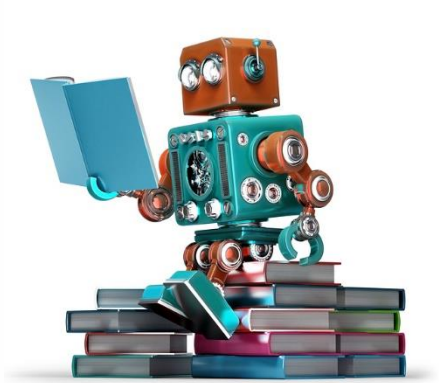
Deep Learning

Deep Learning is a subfield of Machine Learning that uses Artificial Neural Networks to learn from the data.

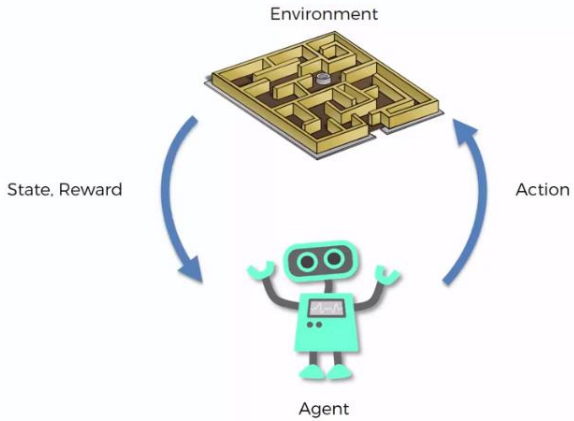




VS



VS



Supervised Learning

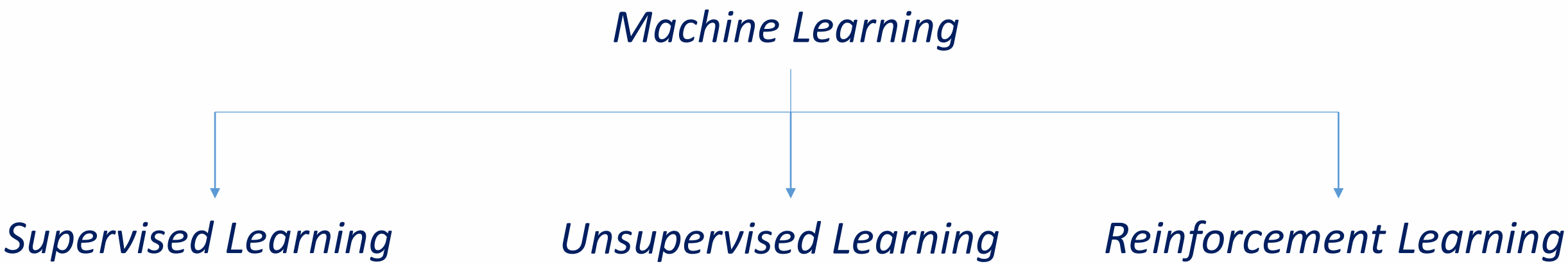
Unsupervised Learning

Reinforcement Learning

Machine Learning

Machine Learning is a technique to implement AI that can learn from the data by themselves without being explicitly programmed.

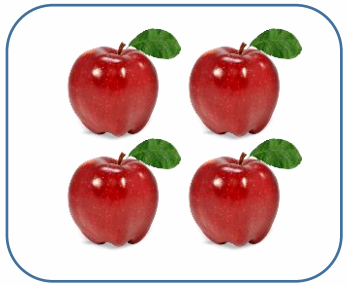
Types of Machine Learning

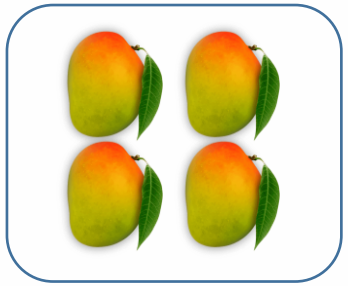


Supervised Learning

In Supervised Learning, the Machine Learning algorithm learns from ***Labelled Data***

Apples





Mangoes

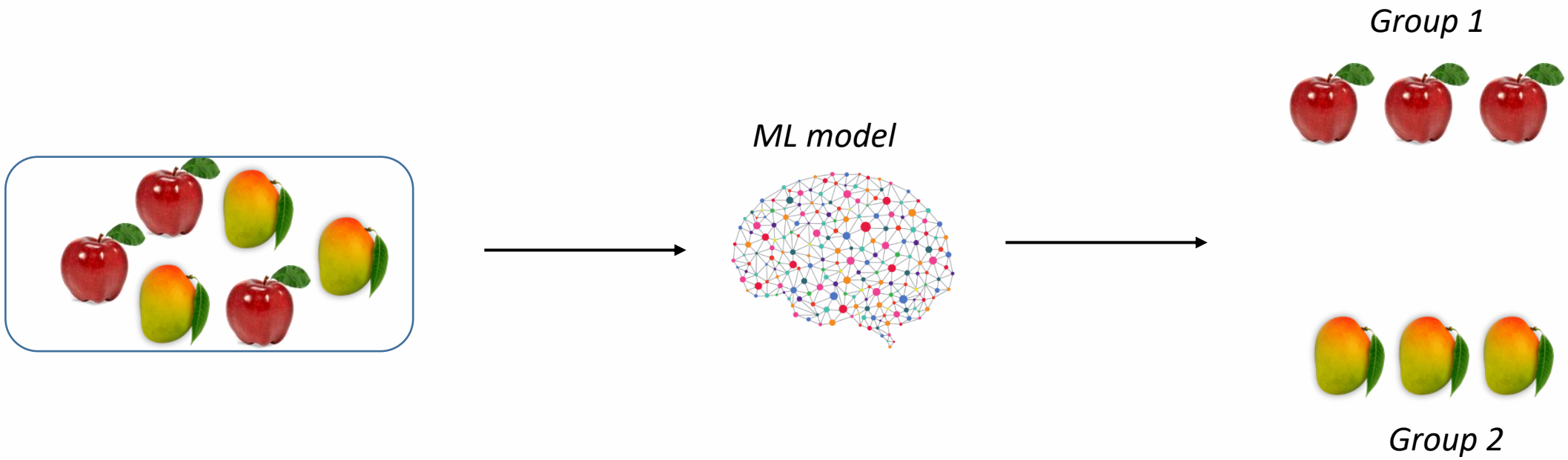
ML model

Apple

Unknown Image

Unsupervised Learning

In Unsupervised Learning, the Machine Learning algorithm learns from ***Unlabelled Data***

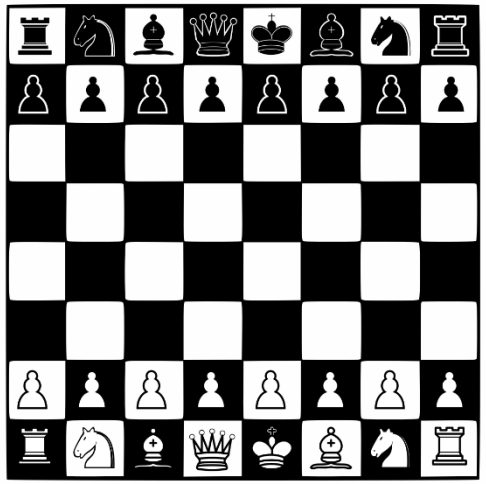


Reinforcement Learning

Reinforcement Learning is an area of Machine Learning concerned with how intelligent agents take actions in an environment to maximize its rewards.

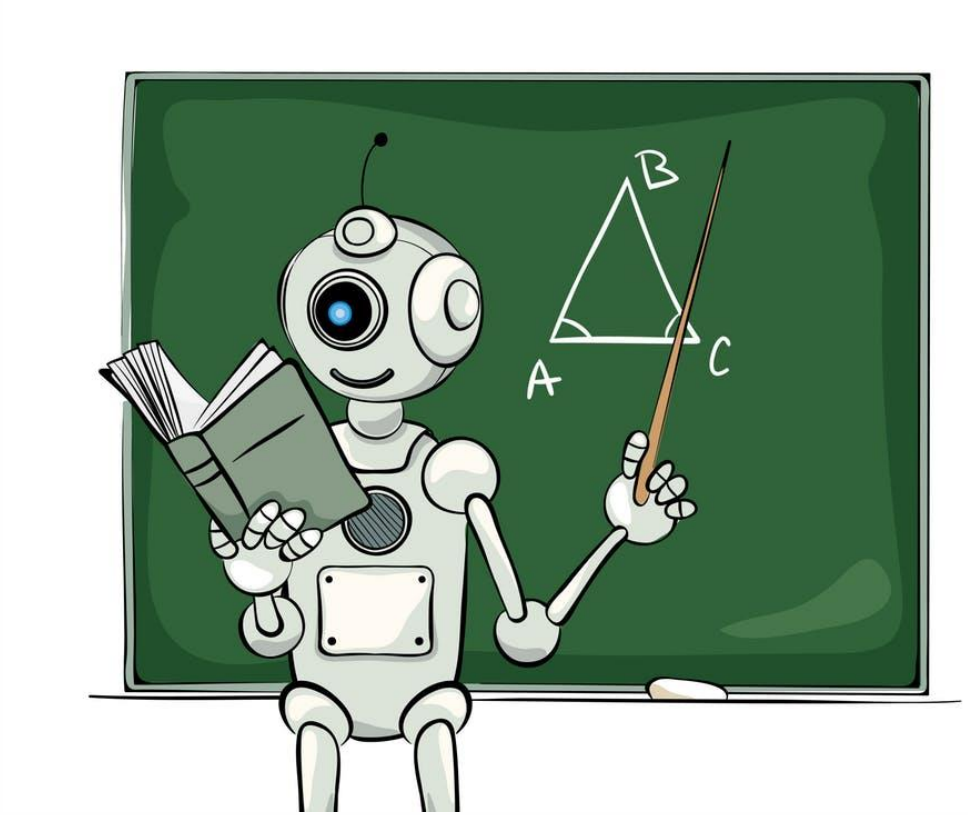
1. *Environment*
2. *Agent*
3. *Action*
4. *Reward*

Agent



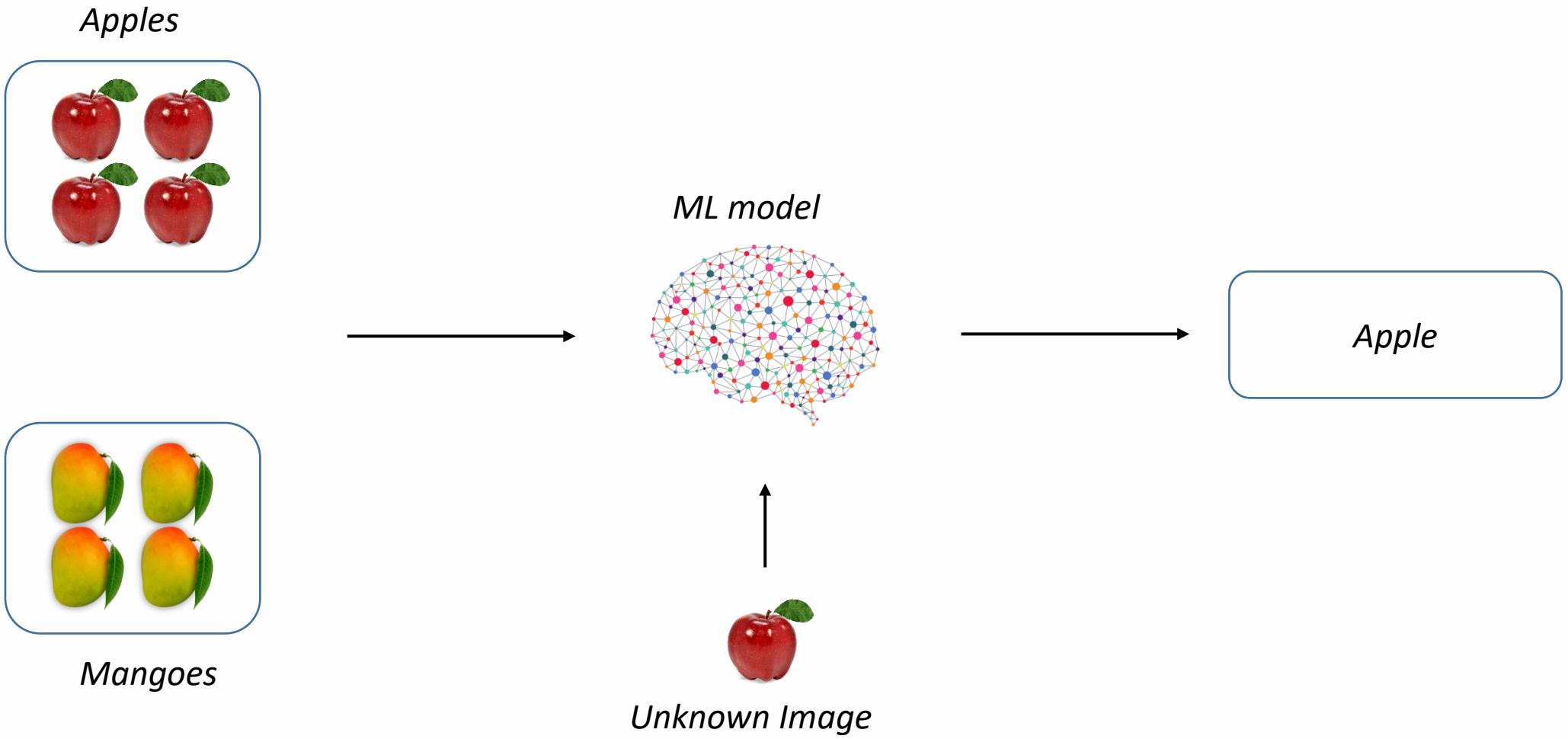
Environment

Supervised Learning: Classification & Regression



Supervised Learning

In Supervised Learning, the Machine Learning algorithm learns from ***Labelled Data***



Types of Supervised Learning

Classification

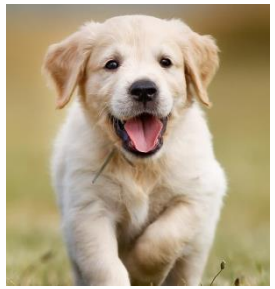
Classification is about predicting a class or discrete values
Eg: Male or Female; True or False

Regression

Regression is about predicting a quantity or continuous values
Eg: Salary; age; Price.

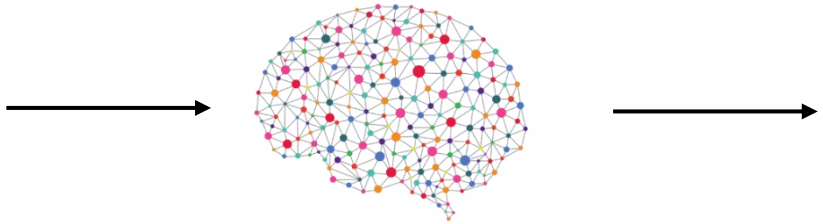
Types of Supervised Learning

Classification:



Dog

Cat

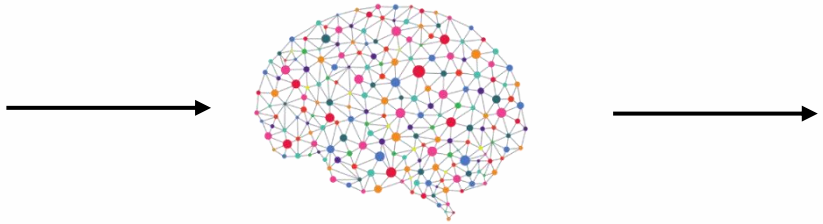


(Dog or Cat)

Regression:

Temperature

Rainfall in cm



Rainfall in cm

Algorithms

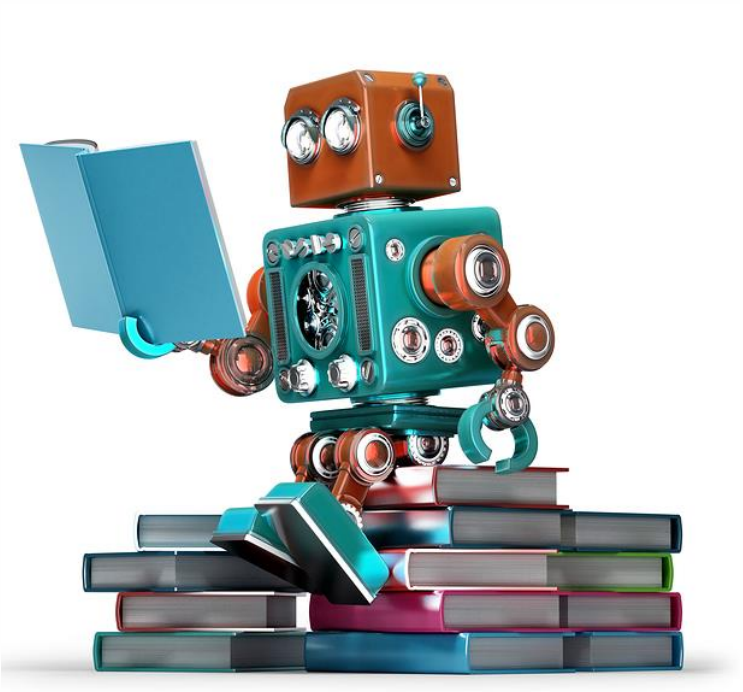
Classification:

1. *Decision Tree Classification*
2. *Random Forest Classification*
3. *K-nearest Neighbor*
4. *Logistic Regression*

Regression:

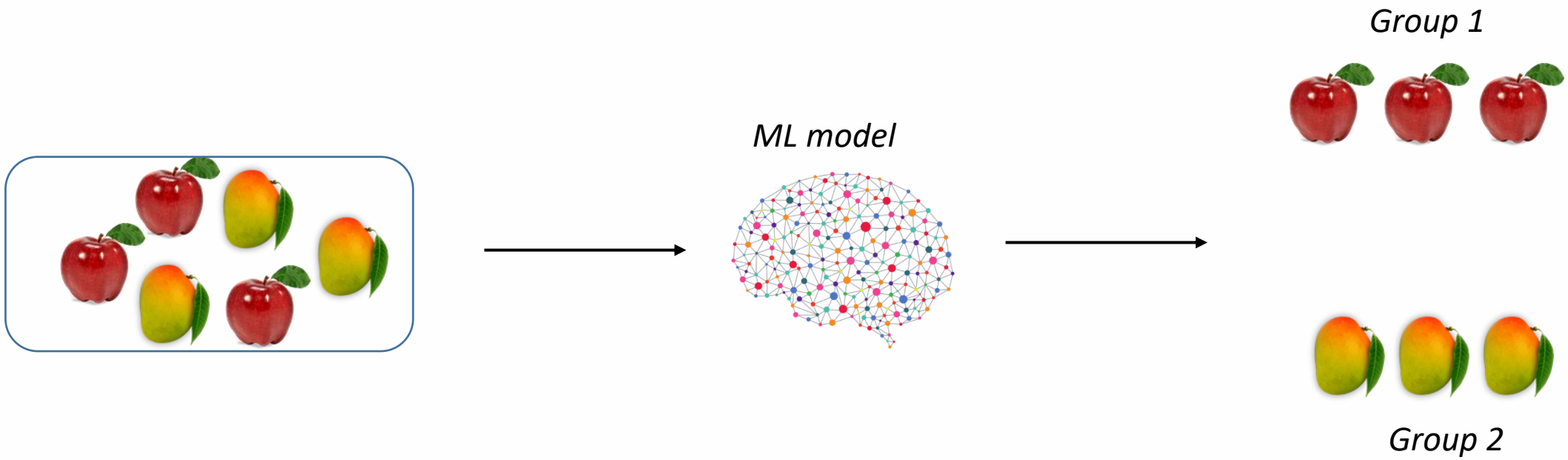
1. *Polynomial Regression*
2. *Support Vector Machines Regressor*

Unsupervised Learning: Clustering & Association



Unsupervised Learning

In Unsupervised Learning, the Machine Learning algorithm learns from ***Unlabelled Data***



Types of Unsupervised Learning

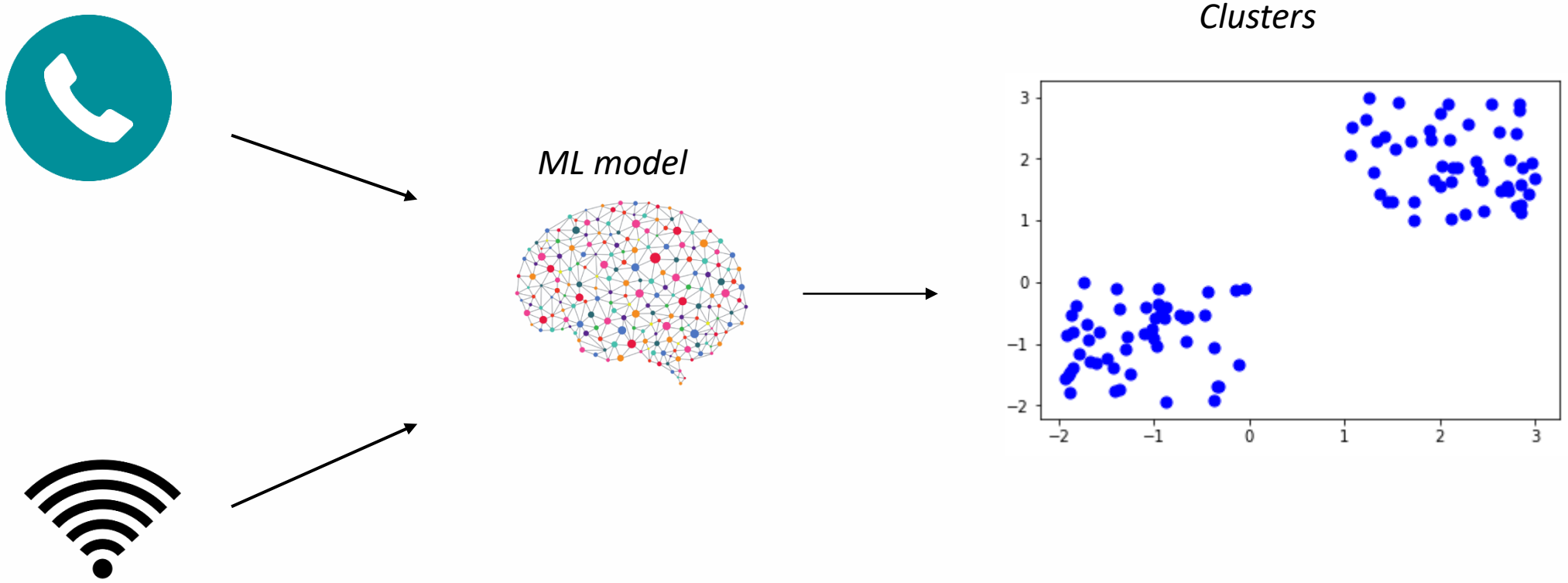
Clustering

Clustering is an unsupervised task which involves grouping the similar data points.

Association

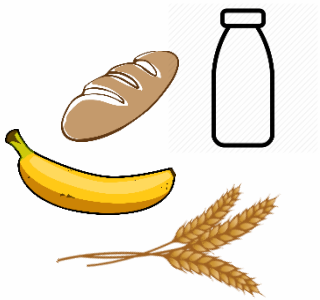
Association is an unsupervised task that is used to find important relationship between data points

Clustering



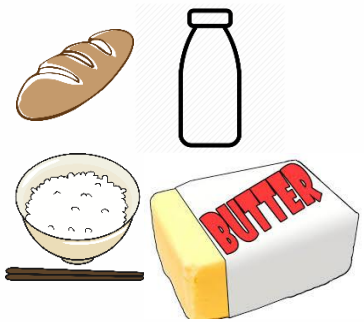
Association

Customer 1



- *Bread*
- *Milk*
- *Fruits*
- *wheat*

Customer 2



- *Bread*
- *Milk*
- *Rice*
- *Butter*

Customer 3

Now, when customer 3 goes and buys bread, it is highly likely that he will also buy milk.

Unsupervised Learning Algorithms

1. *K-Means Clustering*
2. *Hierarchical Clustering*
3. *Principal Component Analysis (PCA)*
4. *Apriori*
5. *Eclat*

ML Project Work Flow

Data

Data pre processing

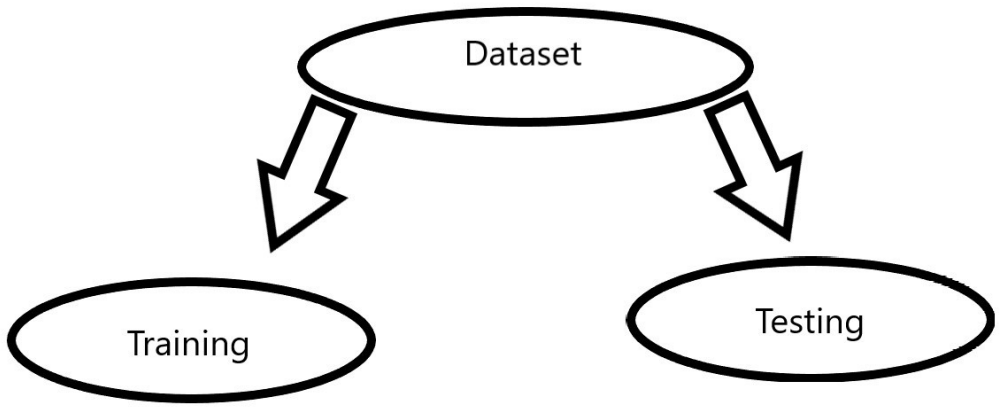
Data Analysis

Train Test split

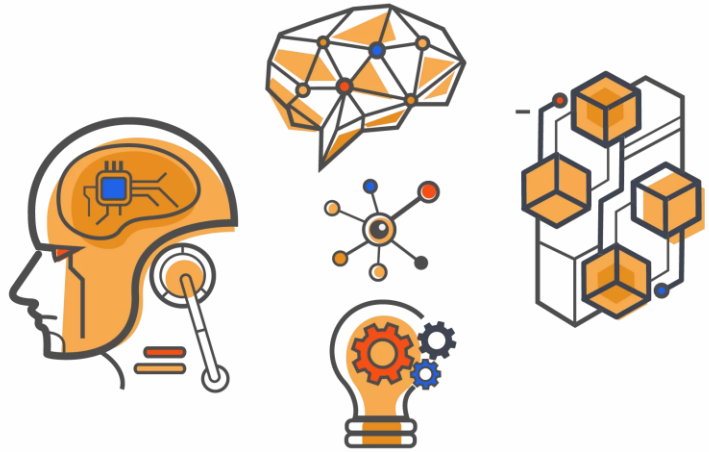
Machine Learning Model

Evaluation

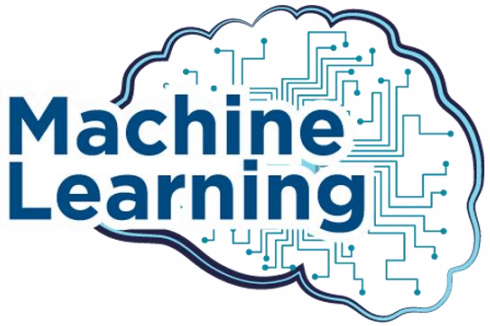
Training & Testing Data



What is a Machine Learning Model?



Machine Learning

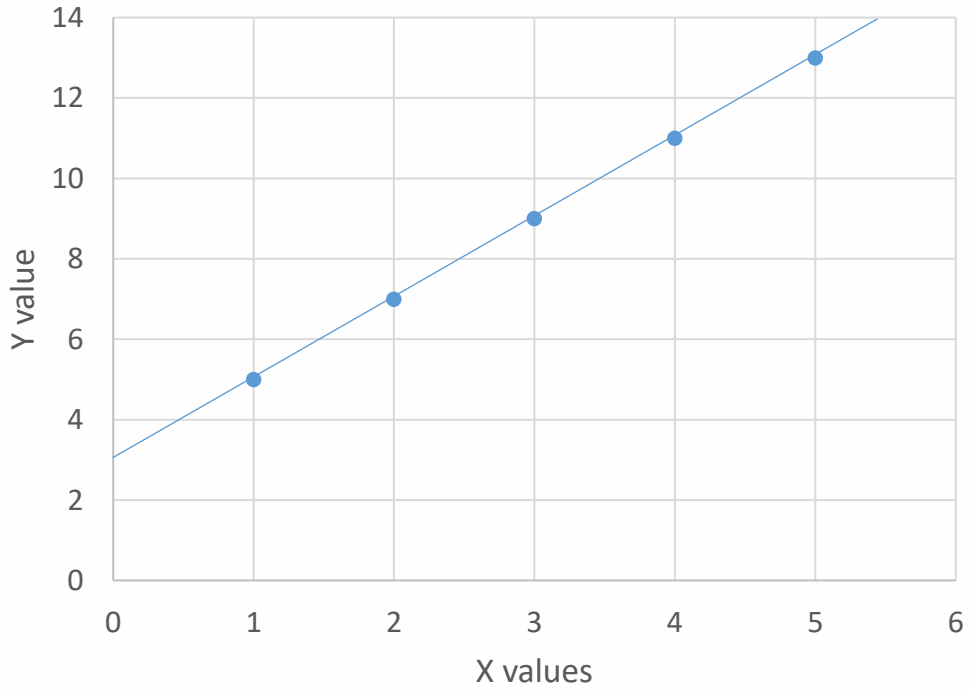


Data

Machine Learning Models

Machine Learning Model

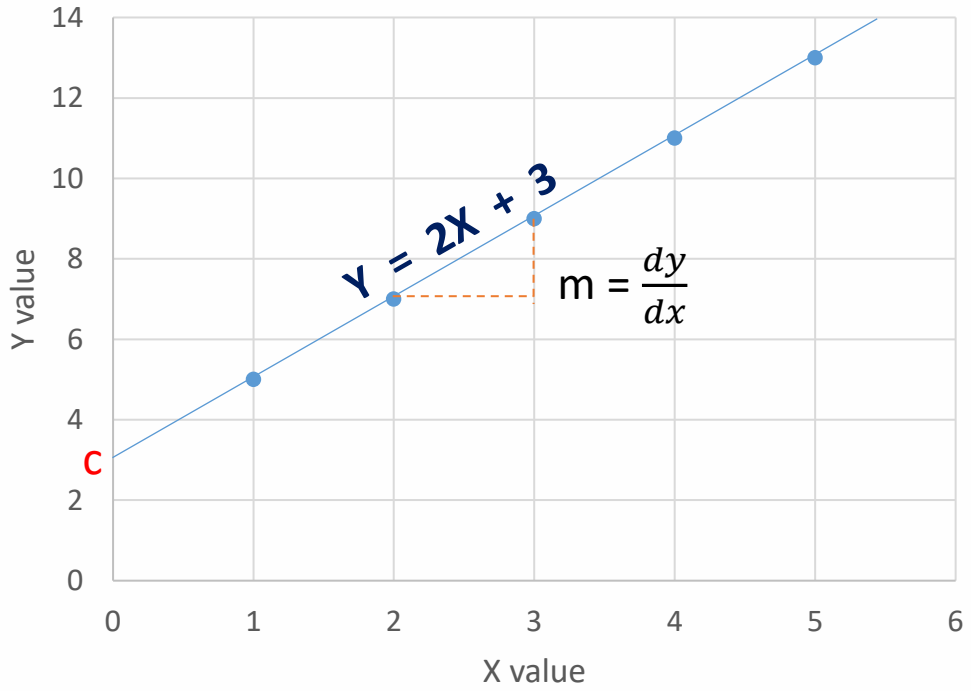
X	1	2	3	4	5
Y	5	7	9	11	13



$Y = mX + c$

X --> X value
Y --> Y value
m --> Slope
c --> Intercept

Machine Learning Model



Inference: The above Line equation is a function that relates X and Y.
For a given value of X, we can find the corresponding value of Y

Equation of a Straight Line : **Y = mX + c**

Find the values of m and c:

Point P1 (2,7)
Point P2 (3,9)

Slope, m = $\frac{y_2 - y_1}{x_2 - x_1} = \frac{9 - 7}{3 - 2} = 2$

m = 2

Intercept, c:

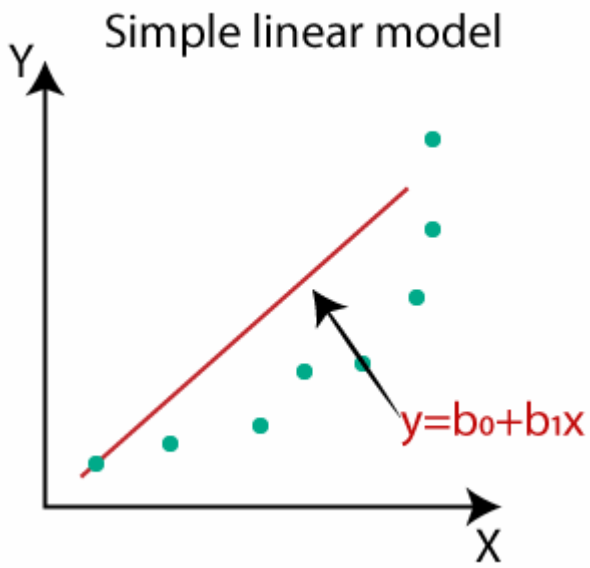
Point (4,11)

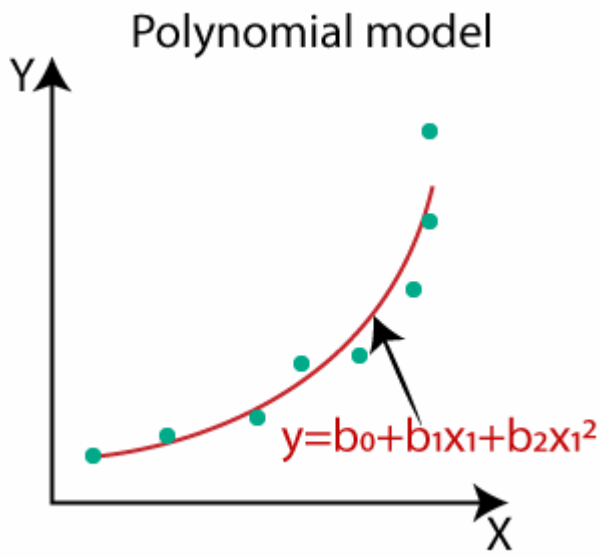
Y = 2X + c

11 = 2(4) + c

c = 3

Machine Learning Model





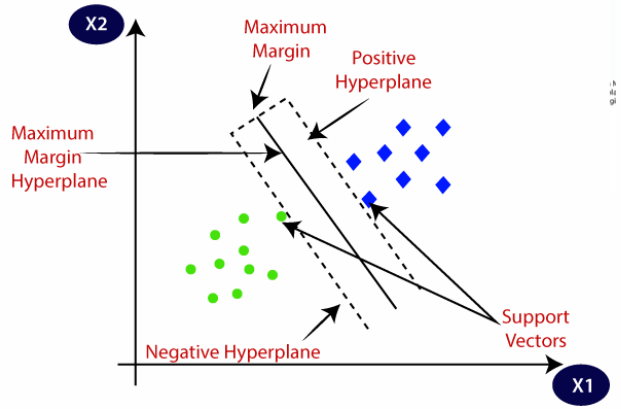
We cannot have a linear relationship between the variables all the time.

Machine Learning Model

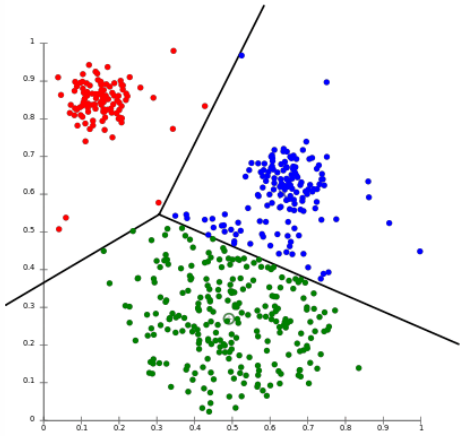
A **Machine Learning Model** is a function that tries to find the relationship between the Features and the Target variable.

It tries to find the pattern in the data, understand the data and trains on the data. Based on this learning, a Machine Learning Model makes Predictions and recognize patterns.

Logistic Regression



Support Vector Machine

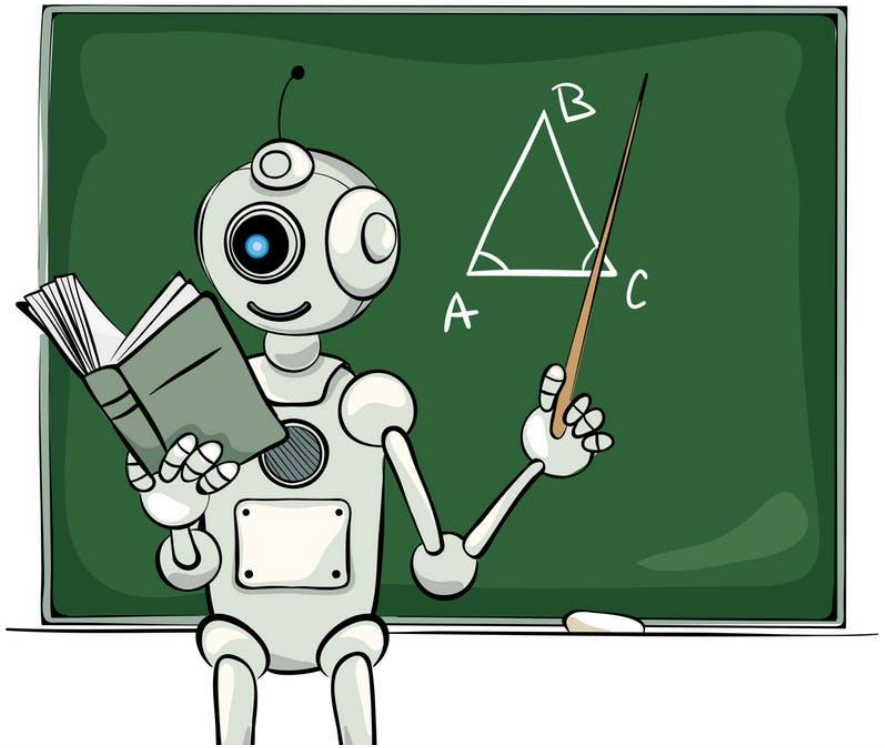


K-Means Clustering

Topics covered in this module:

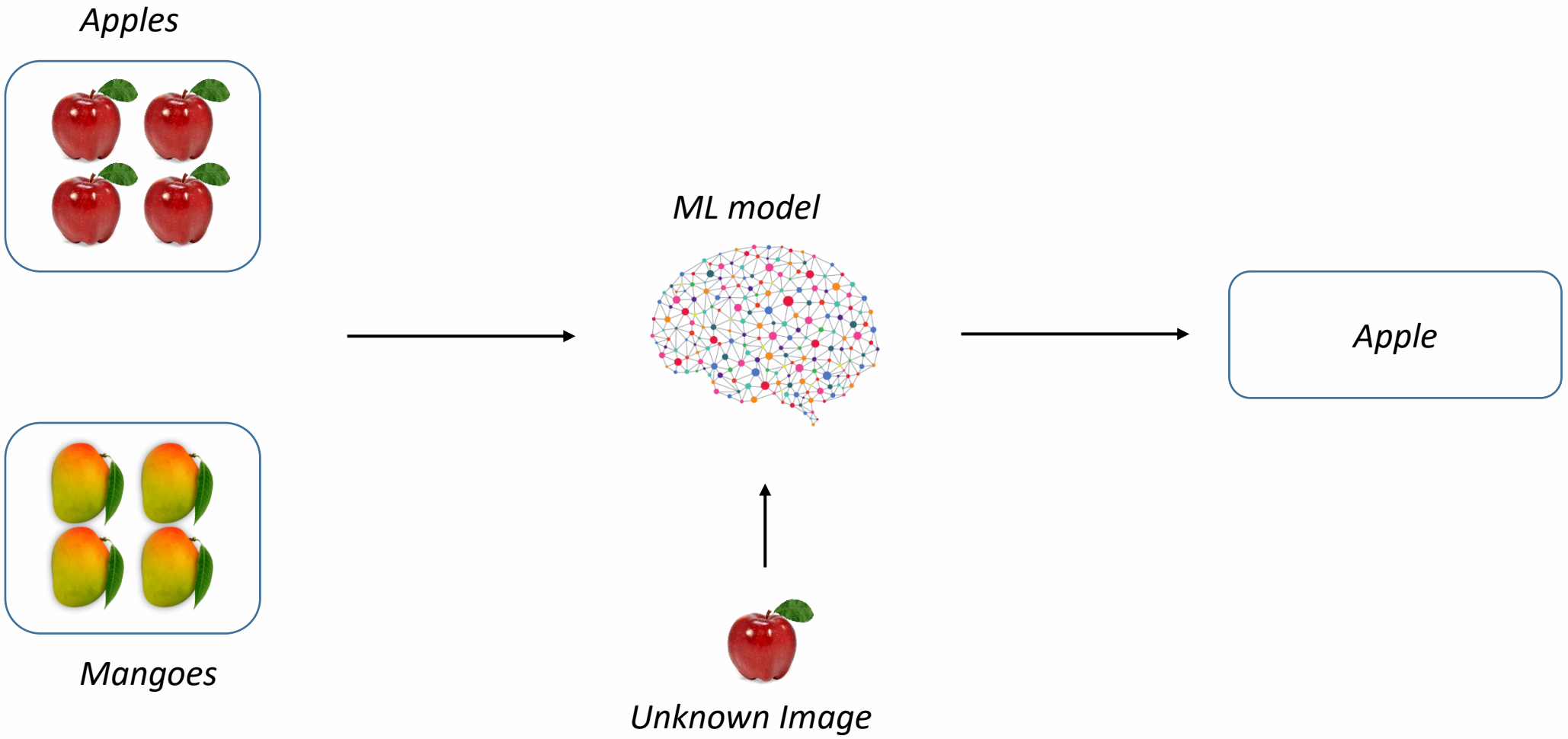
1. What is a Machine Learning Model?
2. Supervised ML Models
3. Unsupervised ML Models
4. Model Selection
5. Overfitting
6. Underfitting
7. Model Optimization
8. Loss Function
9. Model Evaluation

Supervised Learning Models



Supervised Learning

In Supervised Learning, the Machine Learning algorithm learns from ***Labelled Data***



Types of Supervised Learning

Classification

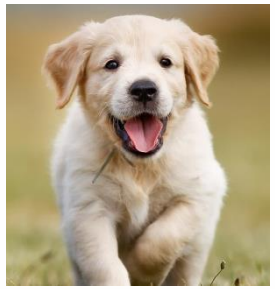
Classification is about predicting a class or discrete values
Eg: Male or Female; True or False

Regression

Regression is about predicting a quantity or continuous values
Eg: Salary; age; Price.

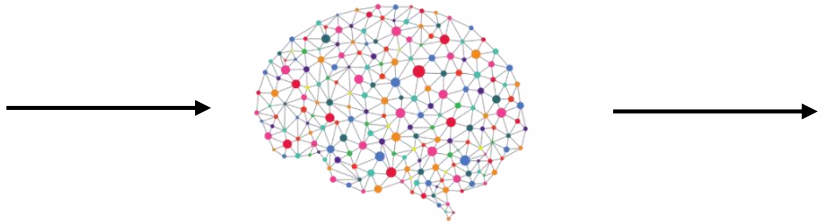
Types of Supervised Learning

Classification:



Dog

Cat

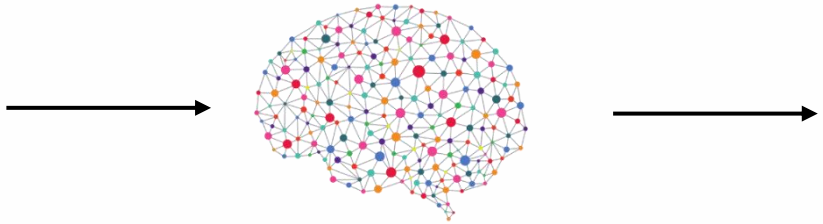


(Dog or Cat)

Regression:

Temperature

Rainfall in cm



Rainfall in cm

Supervised Learning Models

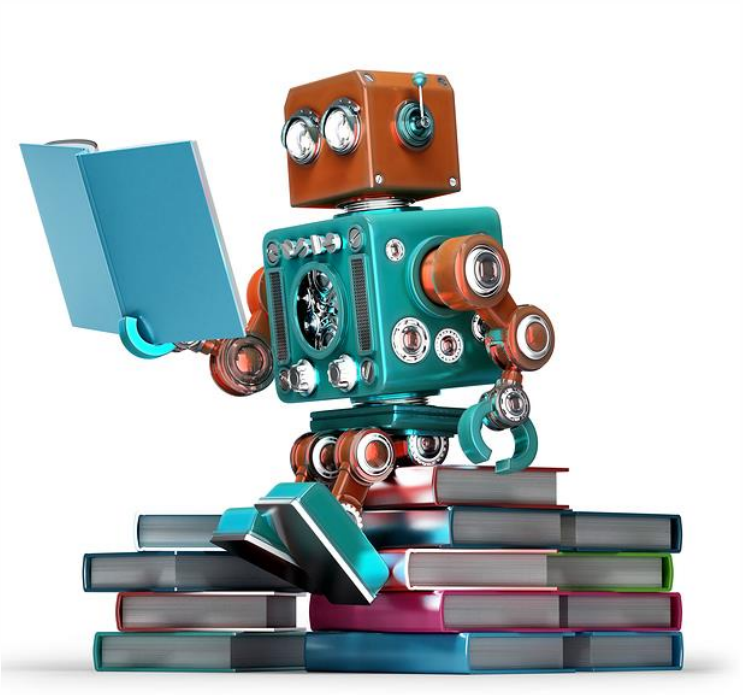
Classification:

1. *Logistic Regression*
2. *Support Vector Machine Classifier*
3. *Decision Tree*
4. *K-Nearest Neighbors*
5. *Random Forest*
6. *Naïve Bayes Classifier*

Regression:

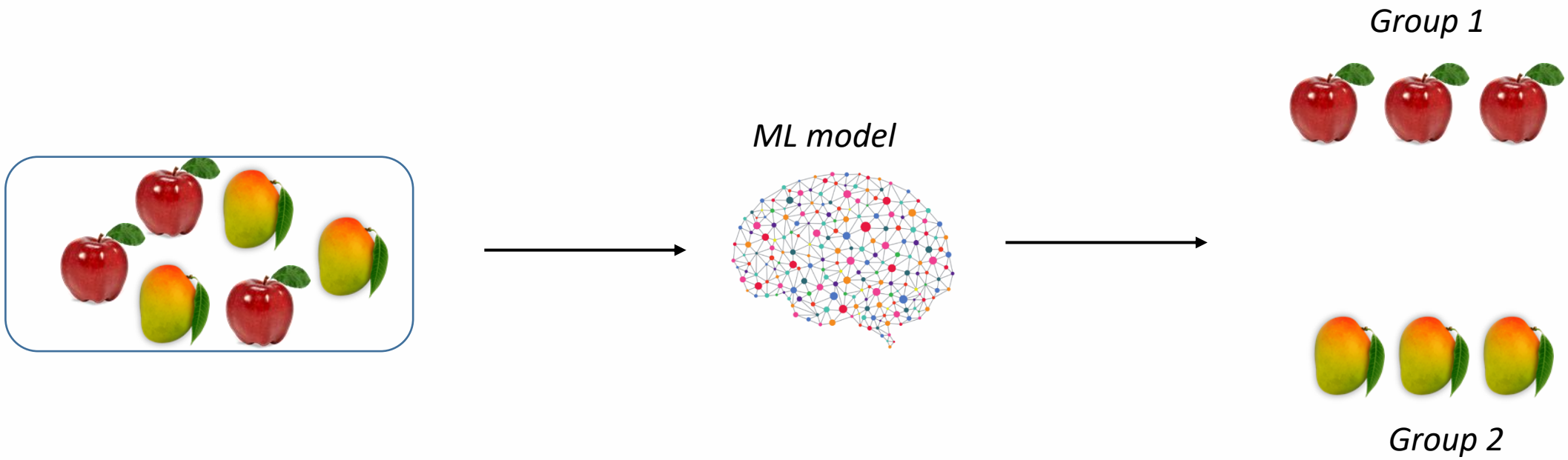
1. *Linear Regression*
2. *Lasso Regression*
3. *Polynomial Regression*
4. *Support Vector Machine Regressor*
5. *Random Forest Regressor*
6. *Bayesian Linear Regressor*

Unsupervised Learning Models



Unsupervised Learning

In Unsupervised Learning, the Machine Learning algorithm learns from ***Unlabelled Data***



Types of Unsupervised Learning

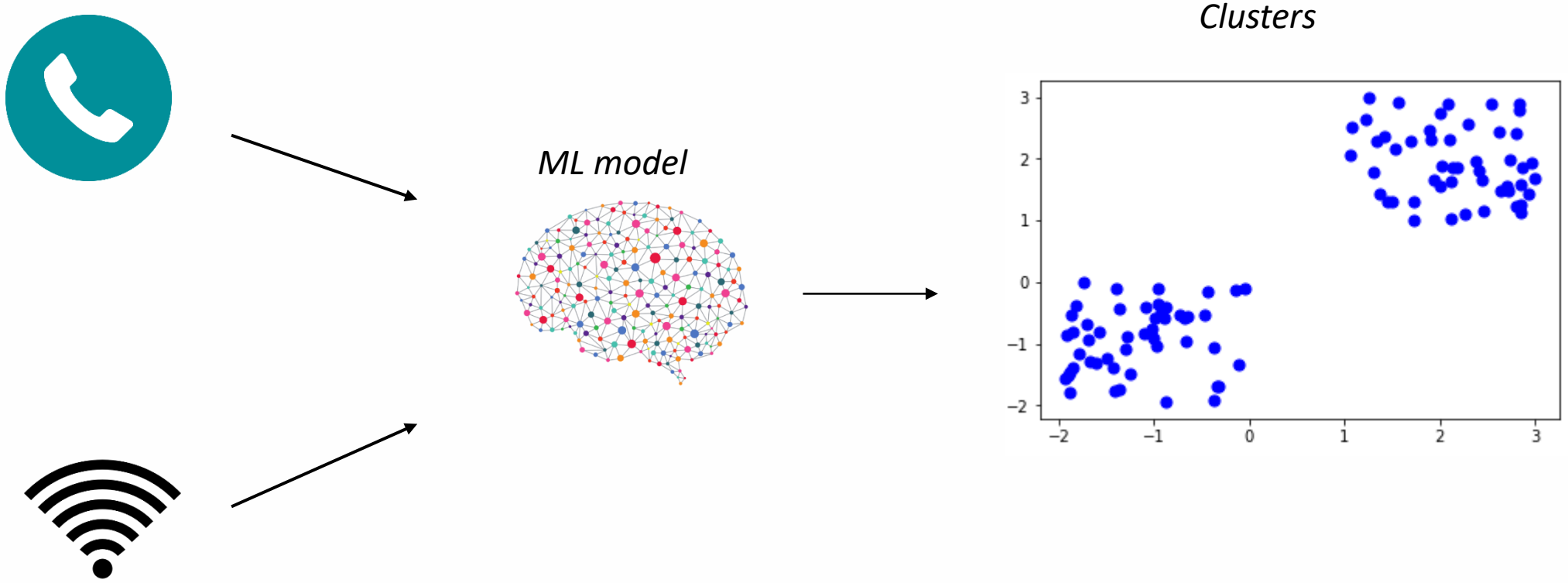
Clustering

Clustering is an unsupervised task which involves grouping the similar data points.

Association

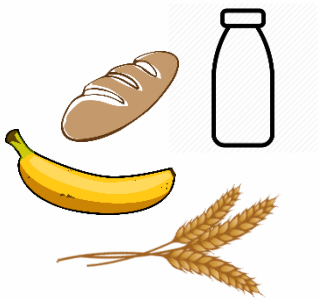
Association is an unsupervised task that is used to find important relationship between data points

Clustering



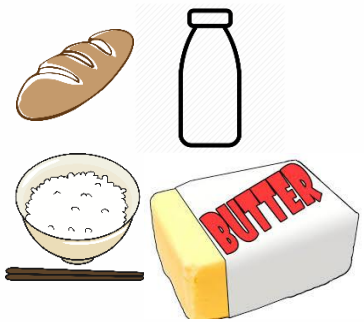
Association

Customer 1



- *Bread*
- *Milk*
- *Fruits*
- *wheat*

Customer 2



- *Bread*
- *Milk*
- *Rice*
- *Butter*

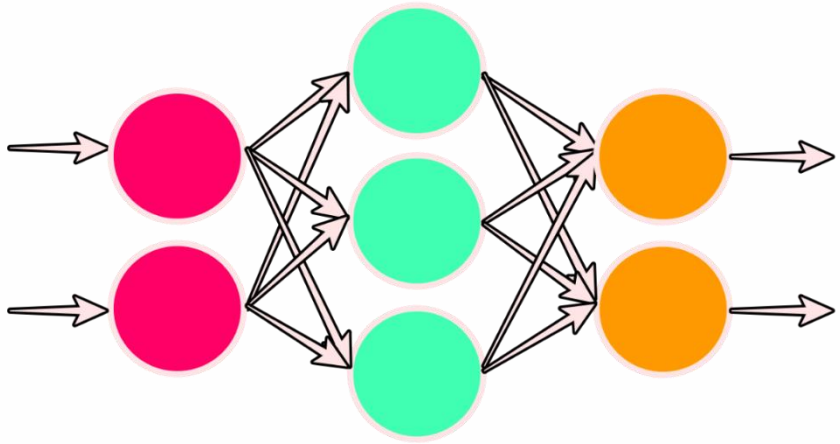
Customer 3

Now, when customer 3 goes and buys bread, it is highly likely that he will also buy milk.

Unsupervised Learning Models

1. *K-Means Clustering*
2. *Hierarchical Clustering*
3. *Principal Component Analysis (PCA)*
4. *Apriori*
5. *Eclat*

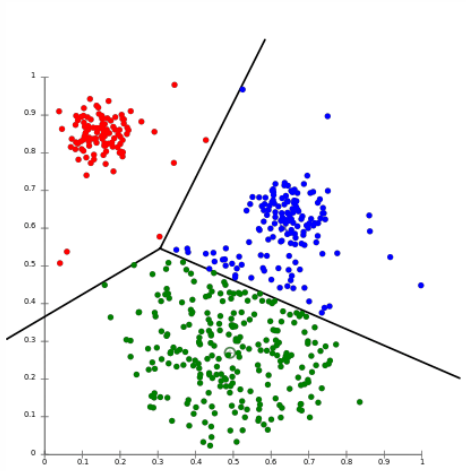
How to choose the right Machine Learning Model? (Model Selection)



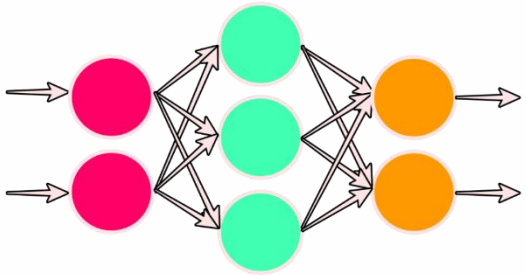
Model Selection

Model Selection in Machine Learning is the process of choosing the best suited model for a particular problem. Selecting a model depends on various factors such as the dataset, task, nature of the model, etc.

Logistic Regression



K-Means Clustering



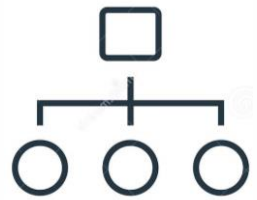
Neural Network

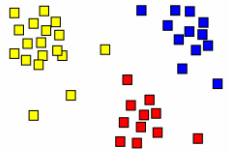
Model Selection

Models can be selected based on :

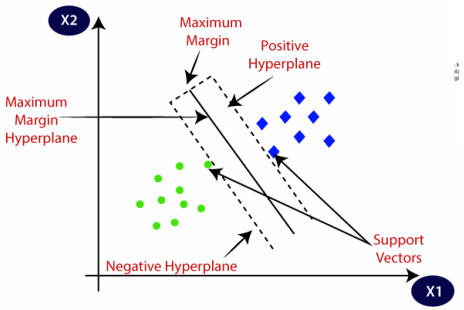
1. Type of Data available:
   a. Images & Videos – CNN
   b. Text data or Speech data – RNN
   c. Numerical data – SVM, Logistic Regression, Decision trees, etc.

2. Based on the task we need to carry out:
   a. Classification tasks – SVM, Logistic Regression, Decision trees, etc.
   b. Regression tasks – Linear regression, Random Forest, Polynomial regression, etc.
   c. Clustering tasks – K-Means Clustering, Hierarchical Clustering





Cross Validation

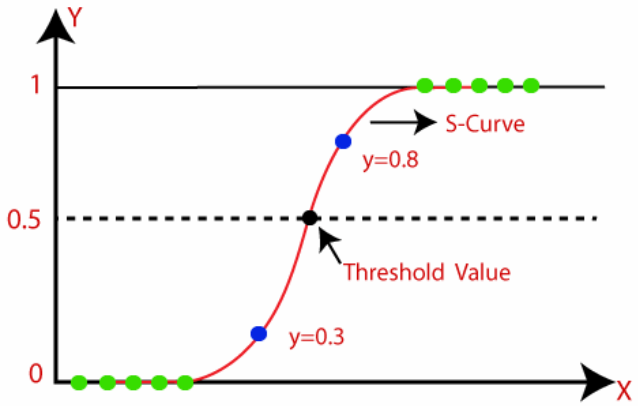


Support Vector Machine

	Dataset					*Accuracy*
Iteration 1	Train	Train	Train	Train	Test	88%
Iteration 2	Train	Train	Train	Test	Train	83%
Iteration 3	Train	Train	Test	Train	Train	86%
Iteration 4	Train	Test	Train	Train	Train	81%
Iteration 5	Test	Train	Train	Train	Train	84%

$$\text{Mean Accuracy} = \frac{88 + 83 + 86 + 81 + 84}{5} = 84.4\ \%$$

Cross Validation



Logistic Regression

	Dataset					*Accuracy*
Iteration 1	Train	Train	Train	Train	Test	90%
Iteration 2	Train	Train	Train	Test	Train	88%
Iteration 3	Train	Train	Test	Train	Train	86%
Iteration 4	Train	Test	Train	Train	Train	91%
Iteration 5	Test	Train	Train	Train	Train	85%

$$\text{Mean Accuracy} = \frac{90 + 88 + 86 + 91 + 85}{5} = 88\ \%$$

Cross Validation

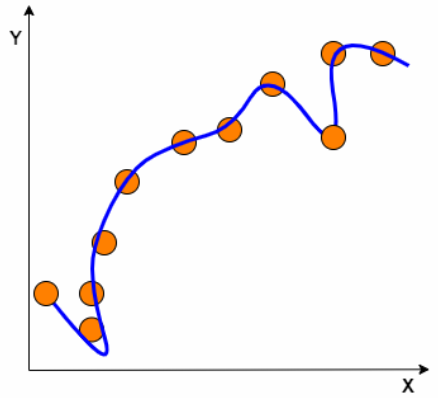
✓ *Accuracy score for SVM = 84.4 %*

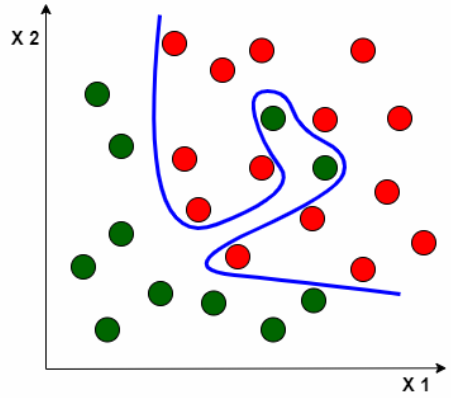
✓ *Accuracy score for Logistic Regression = 88 %*

Cross Validation Implementation:

```
>>> from sklearn import datasets, linear_model
>>> from sklearn.model_selection import cross_val_score
>>> diabetes = datasets.load_diabetes()
>>> X = diabetes.data[:150]
>>> y = diabetes.target[:150]
>>> lasso = linear_model.Lasso()
>>> print(cross_val_score(lasso, X, y, cv=3))
[0.33150734 0.08022311 0.03531764]
```

Overfitting in Machine Learning





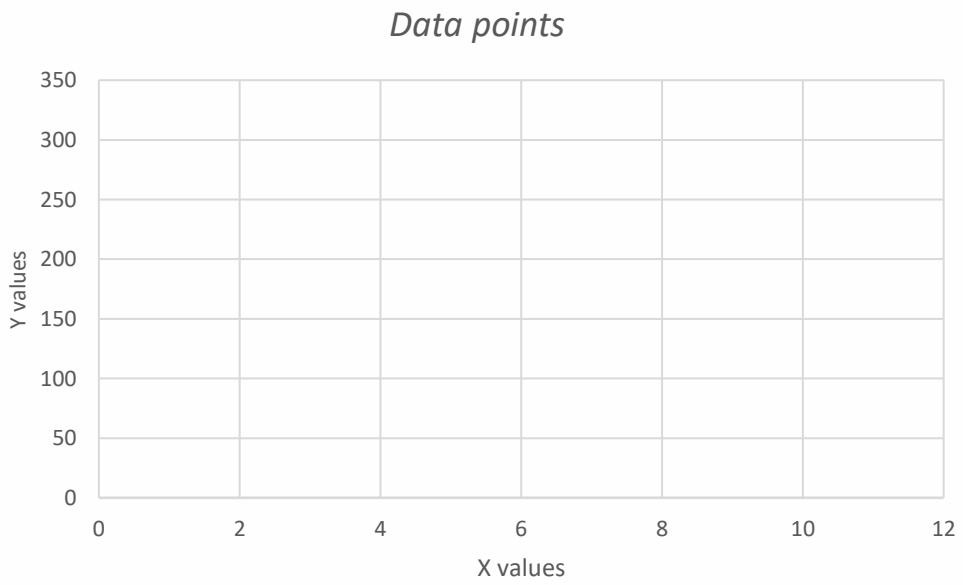
Overfitting

Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training dataset to the extent that it negatively impacts the performance of the model.

Sign that the model has Overfitted : High Training data Accuracy & very low Test data Accuracy

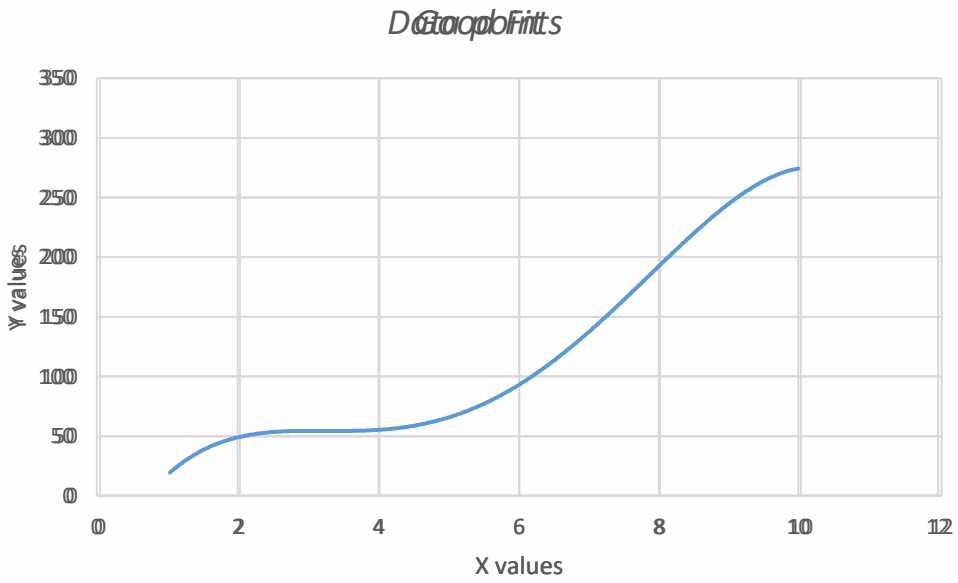
Overfitting

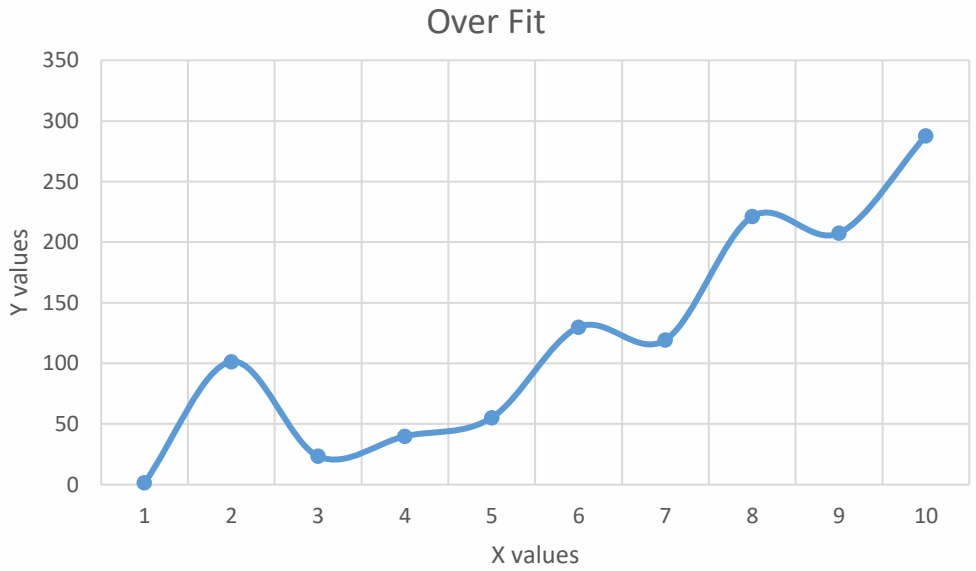
X	1	2	3	4	5	6	7	8	9	10
Y	1.38	101.41	23.34	39.89	55.23	129.91	119.33	221.09	207.43	287.80



Overfitting

X	1	2	3	4	5	6	7	8	9	10
Y	1.38	101.41	23.34	39.89	55.23	129.91	119.33	221.09	207.43	287.80





Overfitting

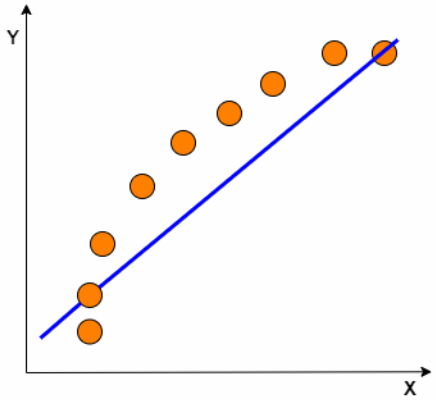
Causes for Overfitting:

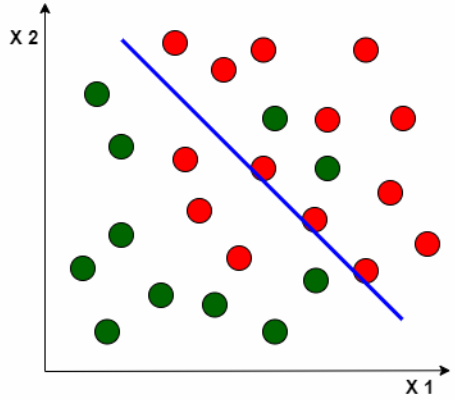
1. Less Data
2. Increased Complexity of the model
3. More number of layers in Neural Network

Preventing Overfitting by:

1. Using more data
2. Reduce the number of layers in the Neural network
3. Early Stopping
4. Bias – Variance Tradeoff
5. Use Dropouts

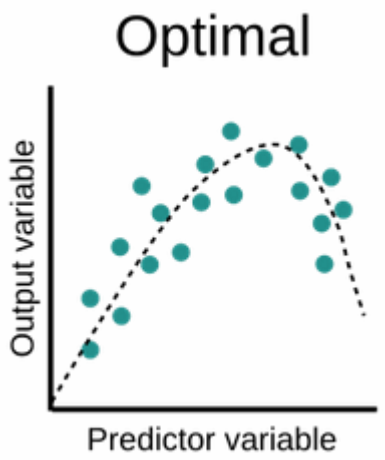
Underfitting in Machine Learning

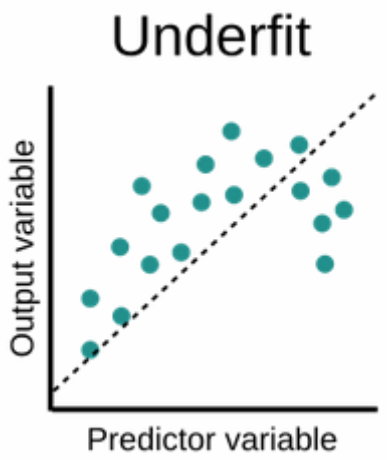




Underfitting

Underfitting happens when the model **does not learn enough** from the data. Underfitting occurs when a machine learning model cannot capture the underlying trend of the data

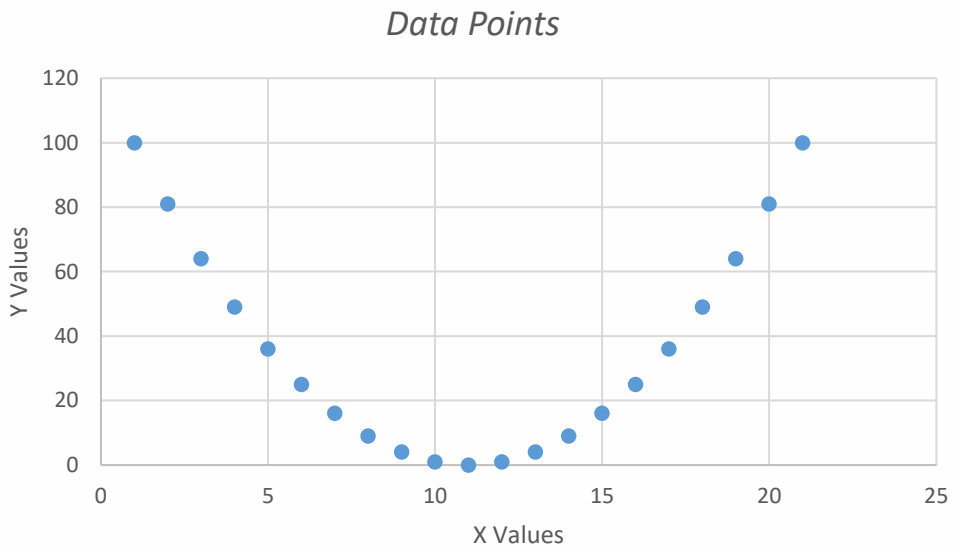




Sign that the model has Underfitted : Very Low Training data Accuracy

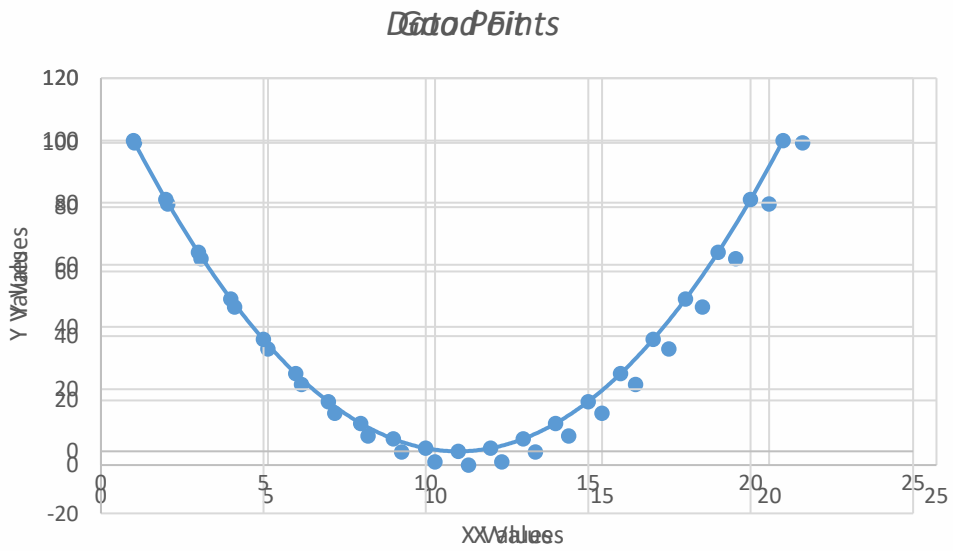
Underfitting

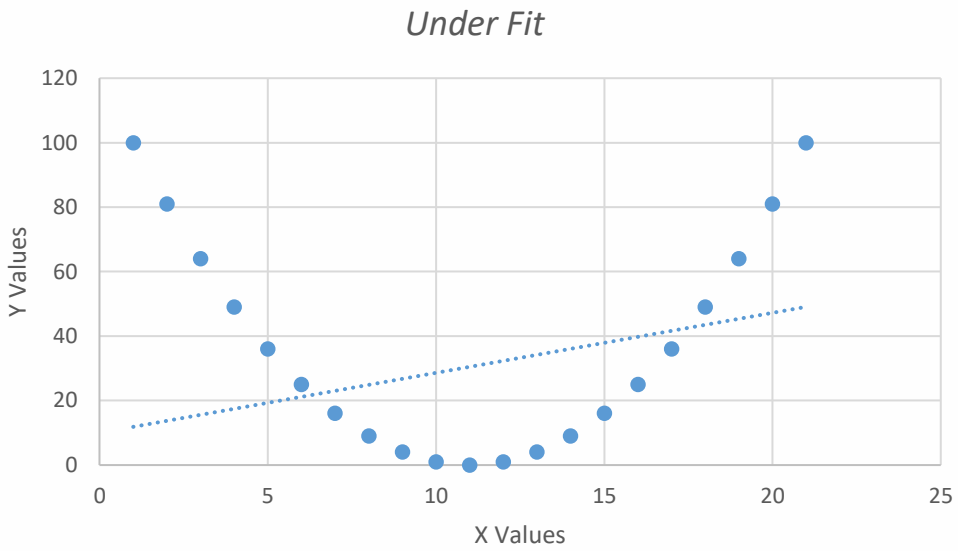
X	-10	-9	-8	-7	.....	0	.....	7	8	9	10
Y	100	81	64	49	.....	0	.....	49	64	81	100



Underfitting

X	-10	-9	-8	-7	.....	0	.....	7	8	9	10
Y	100	81	64	49	.....	0	.....	49	64	81	100





Underfitting

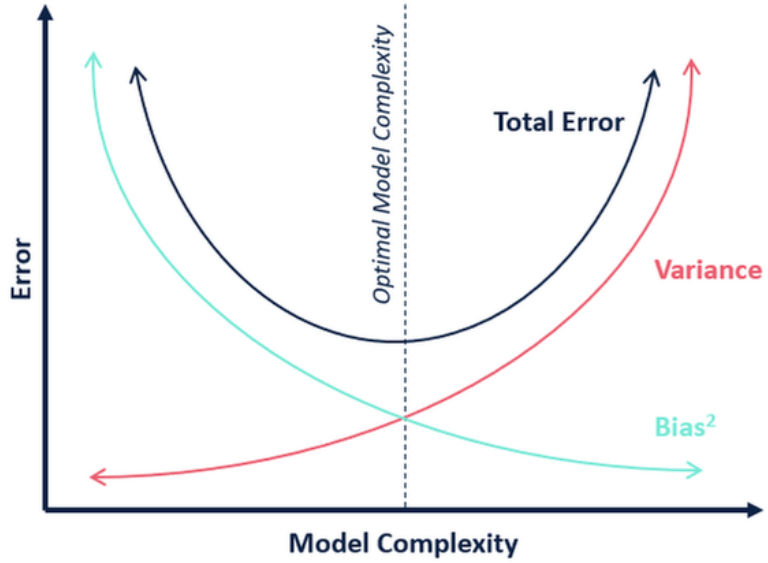
Causes for Underfitting:

1. Choosing a wrong model
2. Less complexity of the model
3. Less variance but high bias

Prevent Underfitting by:

1. Choosing the correct model appropriate for the problem
2. Increasing the complexity of the model
3. More number of parameters to the model
4. Bias – Variance Tradeoff

Bias – Variance Tradeoff in Machine Learning



Loss Function in Machine Learning

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Loss Function

Loss function measures how far an estimated value is from its true value.

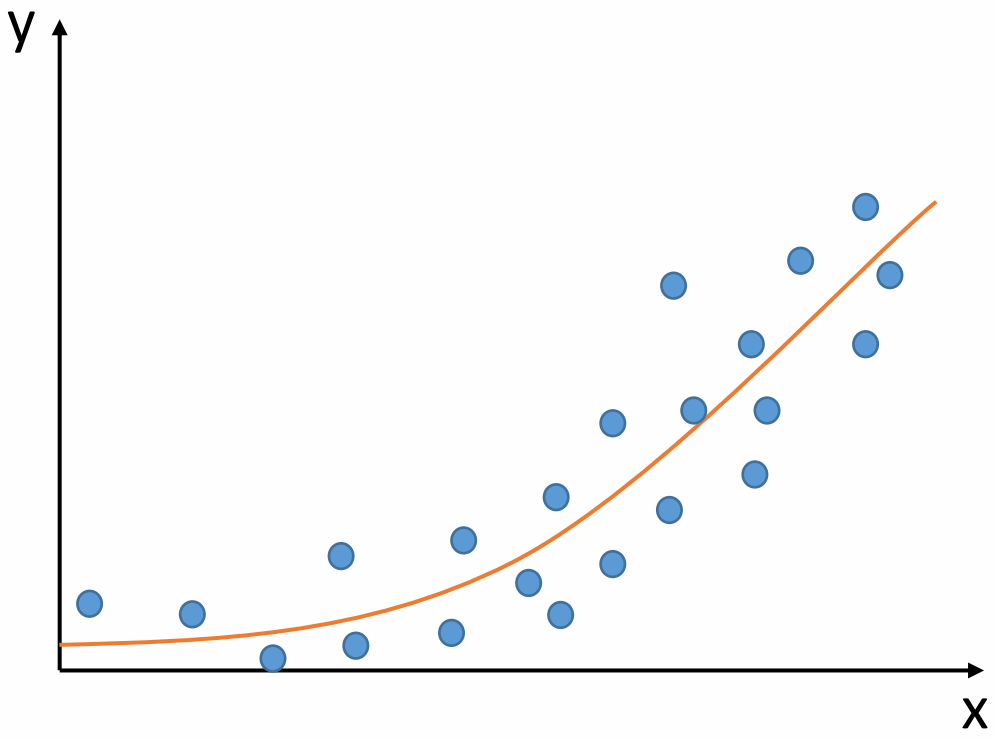
It is helpful to determine which model performs better & which parameters are better.

$$\textbf{\textit{Loss}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Types of Loss Function:

- ❖ Cross Entropy Loss
- ❖ Squared Error Loss
- ❖ KL Divergence

Loss Function



$$y = 0.0000003x^3 + 0.0002x^2 + 0.010x + 0.025$$

Degree 3 Polynomial

Loss Function

$y_1 = 0.0000015x^3 + 0.0042x^2 + 0.020x + 0.035$

$y_2 = 0.0000023x^3 + 0.0001x^2 + 0.015x + 0.020$

$y_3 = 0.0000045x^3 + 0.0003x^2 + 0.040x + 0.028$

$y = 0.0000003x^3 + 0.0002x^2 + 0.010x + 0.025$

x	y	y_1	y_2	y_3
0.30	0.35	0.38	0.39	0.41
0.45	0.48	0.45	0.47	0.56
0.50	0.55	0.59	0.58	0.63
0.55	0.63	0.65	0.69	0.70
0.66	0.72	0.75	0.78	0.78

Loss Function

x	y	y_1	y_2	y_3
0.30	0.35	0.38	0.39	0.41
0.45	0.48	0.45	0.47	0.56
0.50	0.55	0.59	0.58	0.63
0.55	0.63	0.65	0.69	0.70
0.66	0.72	0.75	0.78	0.78

$$\textit{Loss} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$Loss_1 = [\ (0.35-0.38)^2 + (0.48-0.45)^2 + (0.55-0.59)^2 + (0.63-0.65)^2 + (0.72-0.75)^2\ ] / 5$

$Loss_1 = 0.173$

Low Loss value → High Accuracy

Loss Function

Loss function measures how far an estimated value is from its true value.

It is helpful to determine which model performs better & which parameters are better.

$$\textbf{\textit{Loss}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

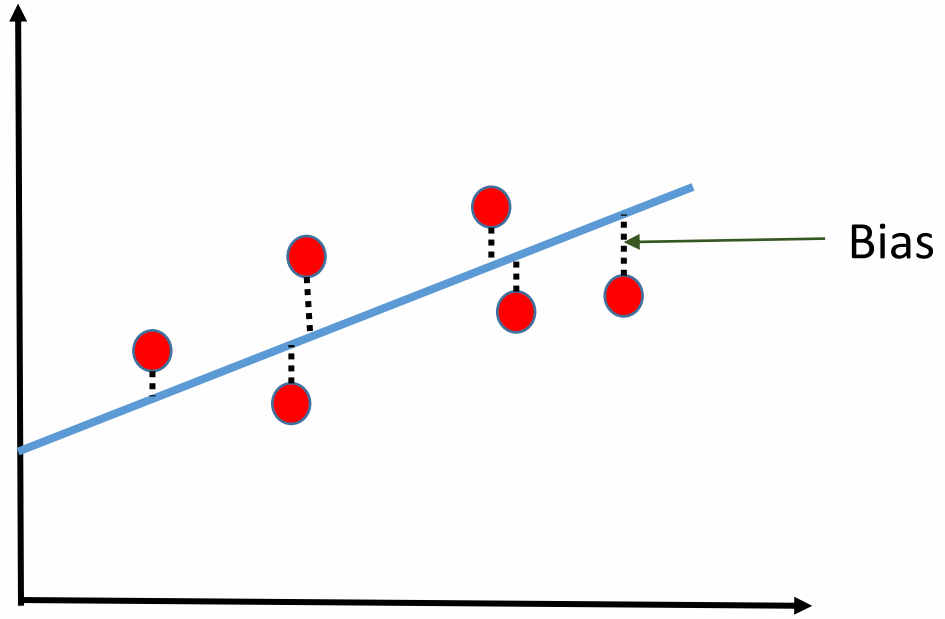
Types of Loss Function:

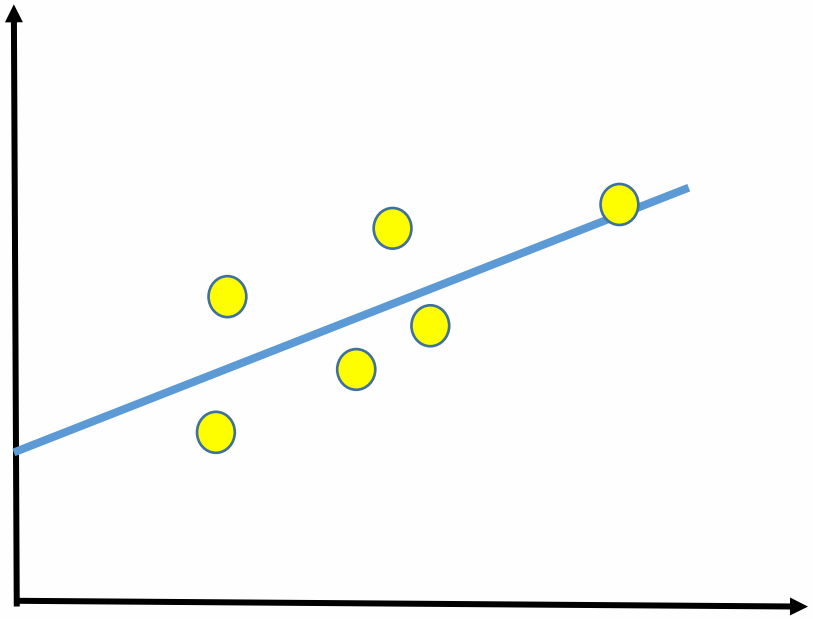
- ❖ Cross Entropy Loss
- ❖ Squared Error Loss
- ❖ KL Divergence

Bias – Variance Tradeoff

Bias :

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict.

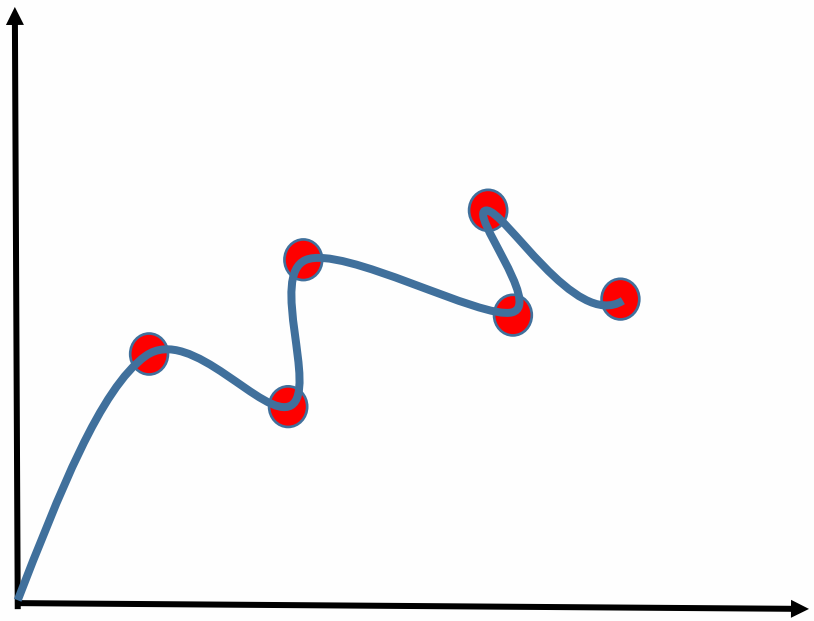




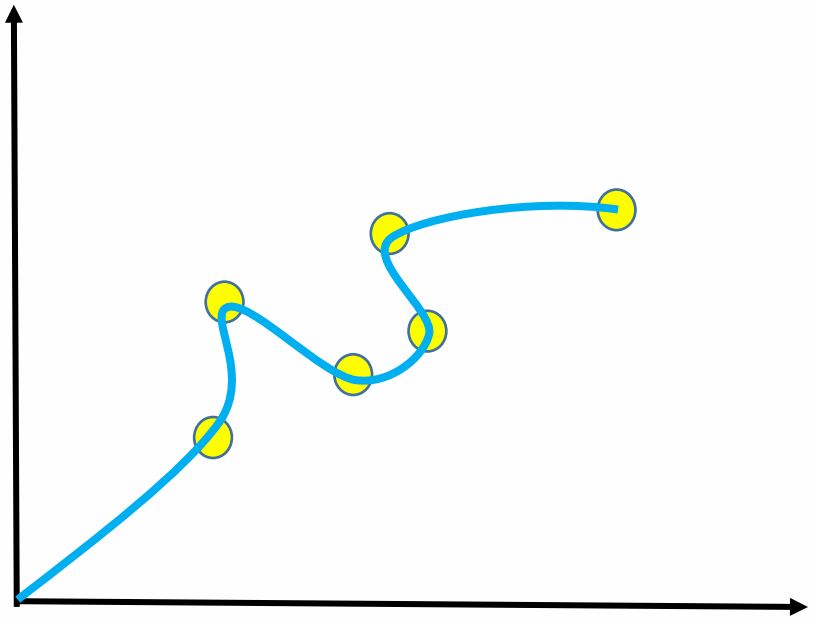
Bias – Variance Tradeoff

Variance :

Variance is the amount that the estimate of the target function will change if different training data was used.

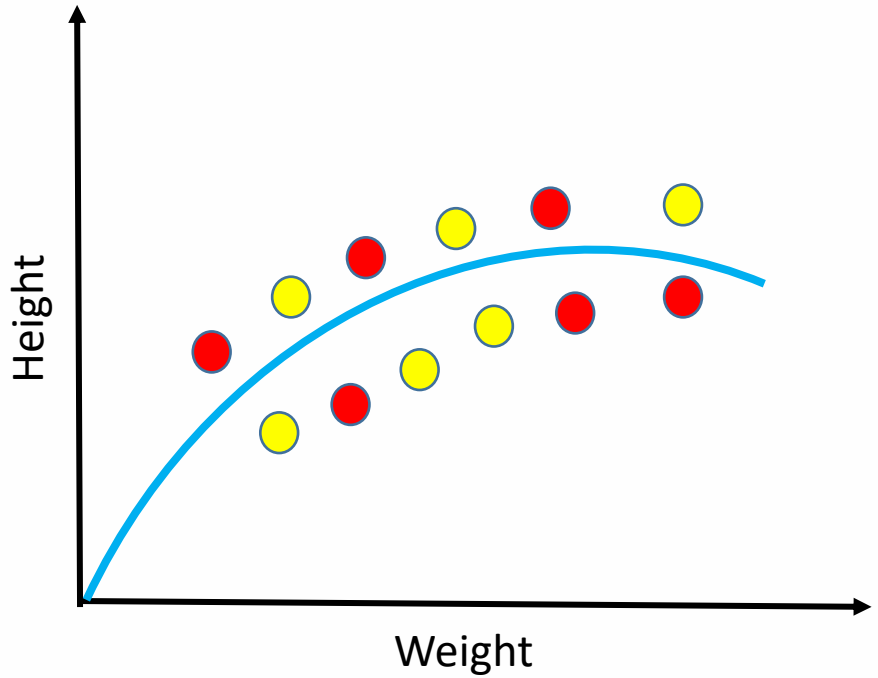


Bias = 0



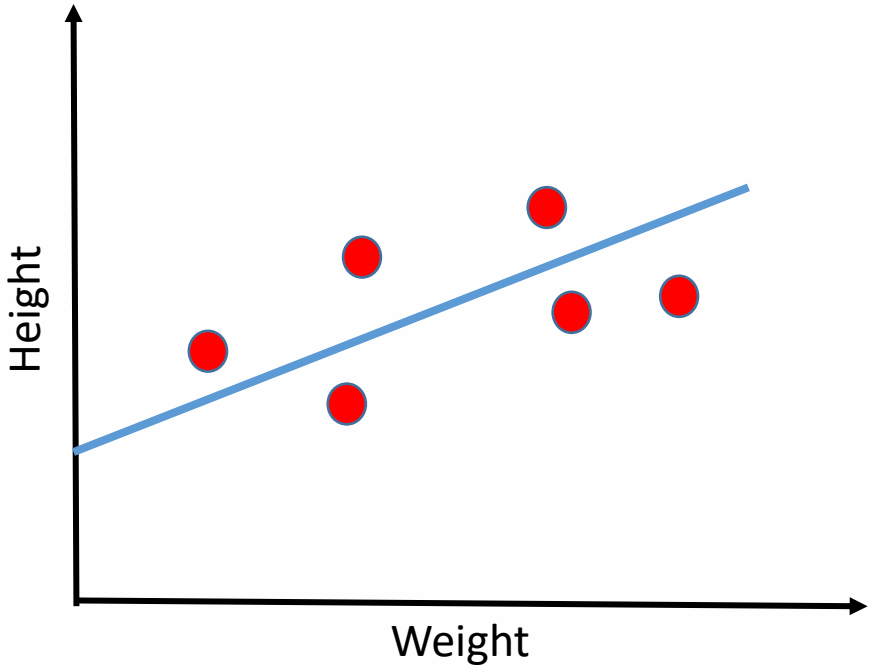
Bias – Variance Tradeoff

Problem statement: *Identify an appropriate model to predict the Height of a person, When their weight is given.*

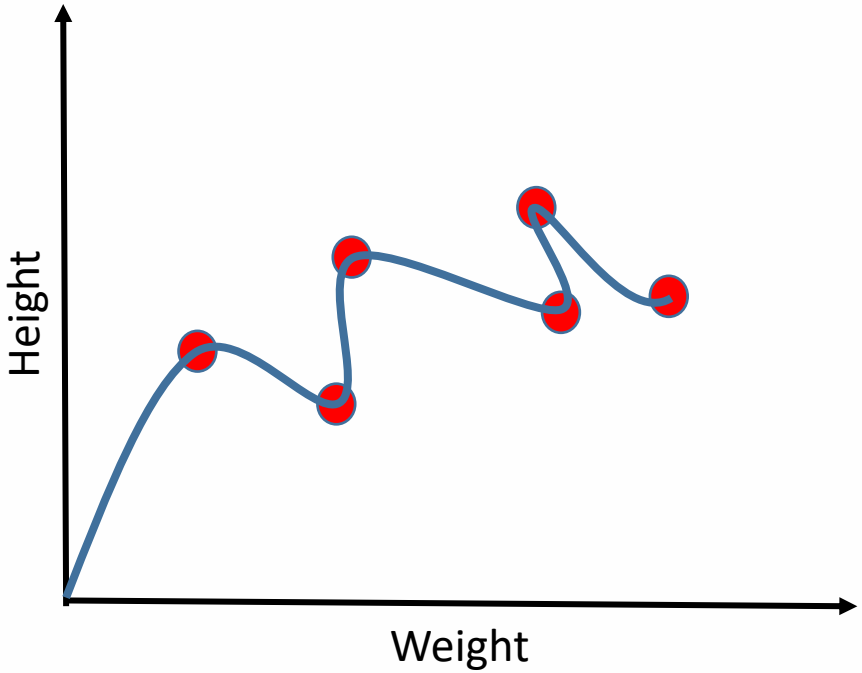


Underfitting & Overfitting

(Plot on training data)



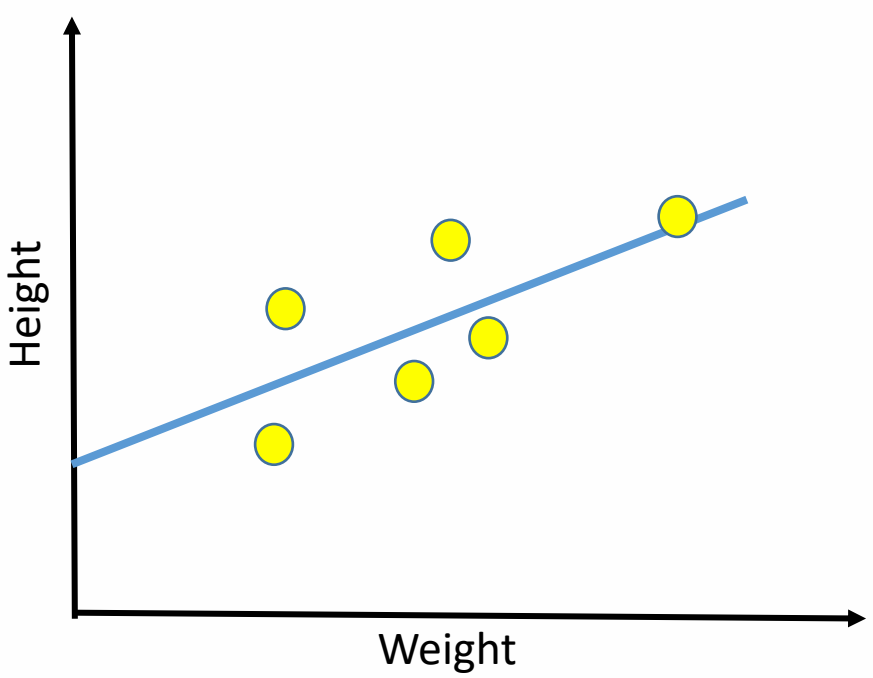
(i) Underfitting



(ii) Overfitting

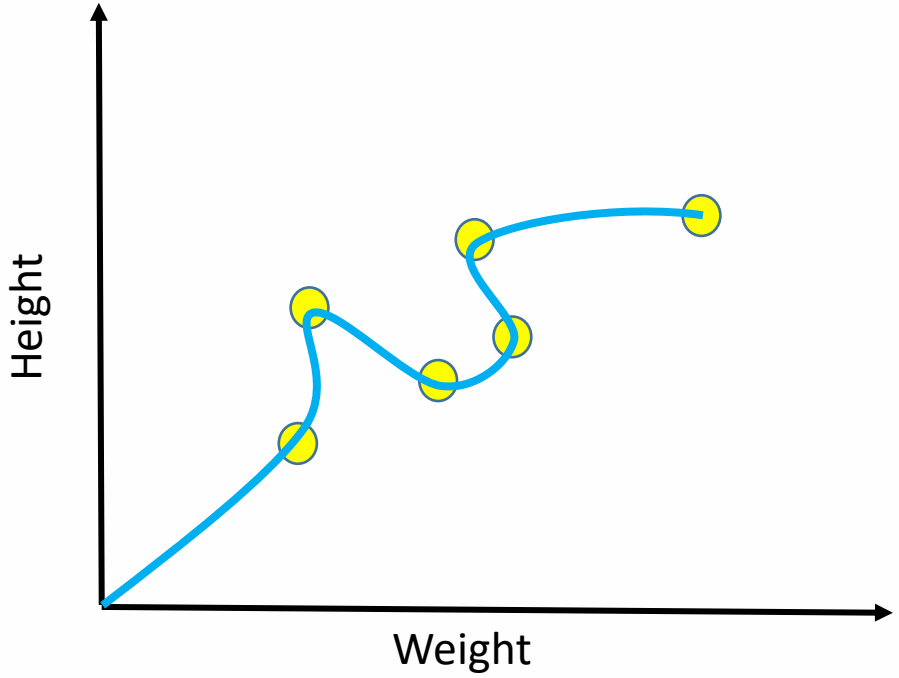
Bias – Variance Tradeoff

(Testing with different data)



(i) Underfitting

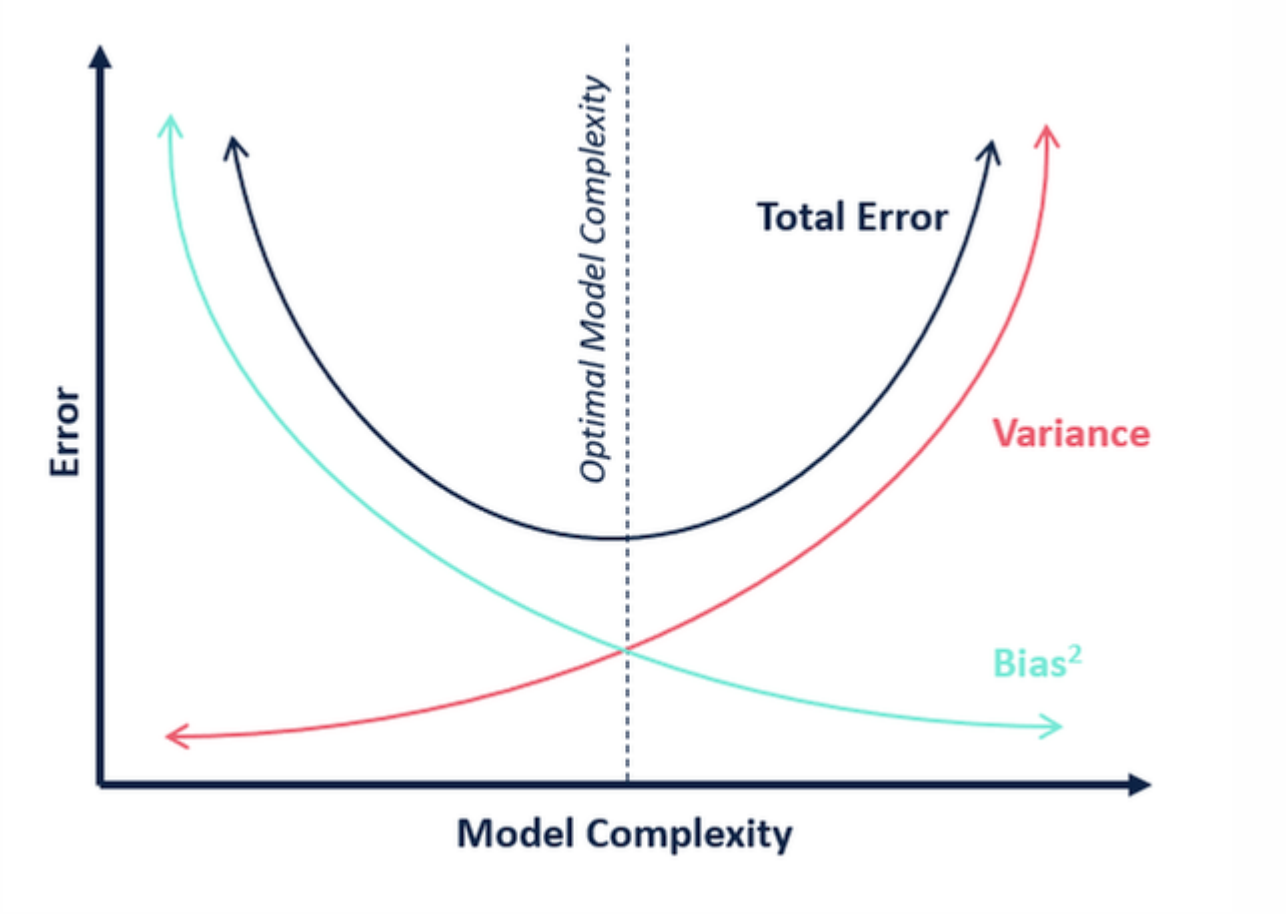
Inference:
a. *High Bias*
b. *Low Variance*



(ii) Overfitting

Inference:
a. *Low Bias*
b. *High Variance*

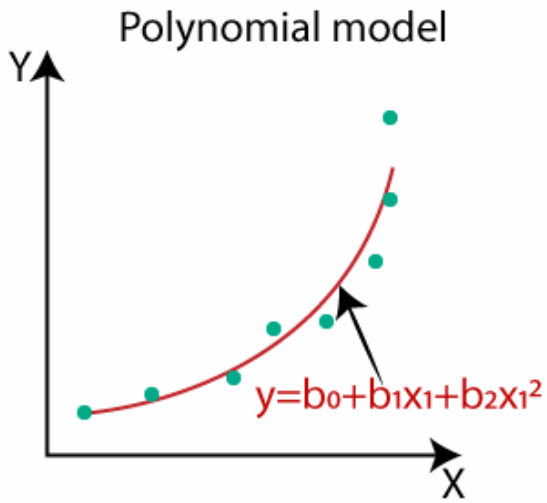
Bias – Variance Tradeoff



Bias – Variance Tradeoff

Techniques to have better Bias – Variance Tradeoff :

1. *Good Model Selection*
2. *Regularization*
3. *Dimensionality Reduction*
4. *Ensemble methods*



Model Evaluation in Machine Learning

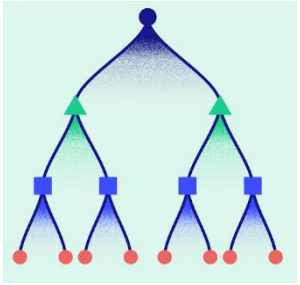
Work Flow of a ML Project

Data

Data pre processing

Data Analysis

Train Test split



XGBoost Regressor

Evaluation

Types of Supervised Learning

Supervised Learning

Classification

Classification is about predicting a class or discrete values
Eg: Male or Female; True or False

Evaluation metric for Classification: **Accuracy score**

Regression

Regression is about predicting a quantity or continuous values
Eg: Salary; age; Price.

Evaluation metric for Regression: **Mean Absolute Error**

Accuracy Score

In Classification, **Accuracy Score** is the ratio of **number of correct predictions** to the **total number of input data points**.

$$\text{Accuracy Score} = \frac{\text{Number of correct predictions}}{\text{Total Number of data points}} \times 100\ \%$$

Number of correct predictions = 128

Total Number of data points = 150

Accuracy Score = 85.3 %

```
from sklearn.metrics import accuracy_score
```

Mean Squared Error

Mean Squared Error measures the average of the squares of the errors, that is, the average squared difference between the estimated values and the actual value.

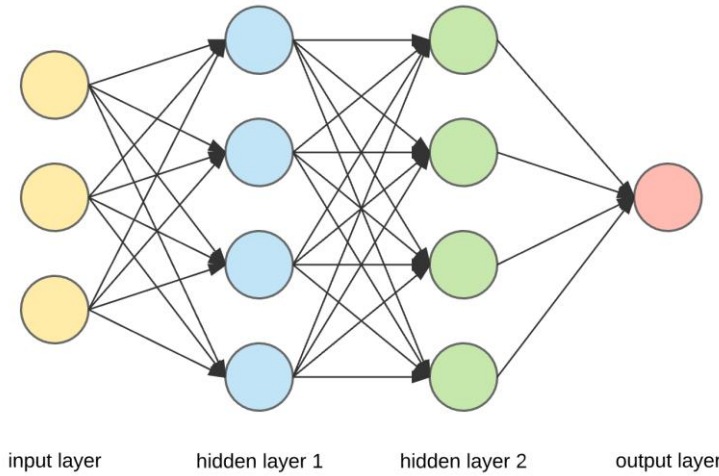
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Actual Value (Y_i = 140 mg/dL)

Predicted Value (Y_i = 160 mg/dL)

```
from sklearn.metrics import mean_squared_error
```

Model Parameters & Hyperparameters



Types of Parameters

Model Parameters

These are the parameters of the model that can be determined by training with training data. These can be considered as internal Parameters.

- **Weights**
- **Bias**

Y = w*X + b

Hyperparameters

Hyperparameters are parameters whose values control the learning process. These are adjustable parameters used to obtain an optimal model. External Parameters.

- **Learning rate**
- **Number of Epochs**

Model Parameters

Weights: Weight decides how much influence the input will have on the output.

Applicant's Details

Name	Degree	College	C	C++	Python	Height	Weight	No. of Backlogs
A	B.E	ABC college	✓	✗	✓	165	72	1
B	M.E	XYZ College	✓	✓	✗	168	80	0
C	M.C.A	State College	✓	✗	✗	175	67	0
D	B.E	ZYX College	✓	✓	✓	168	70	2

✗ ✓ ✓ ✓ ✓ ✓ ✗ ✗ ✓

Model Parameters

Weights:

Weight decides how much influence the input will have on the output.

$$Y = w*X + b$$

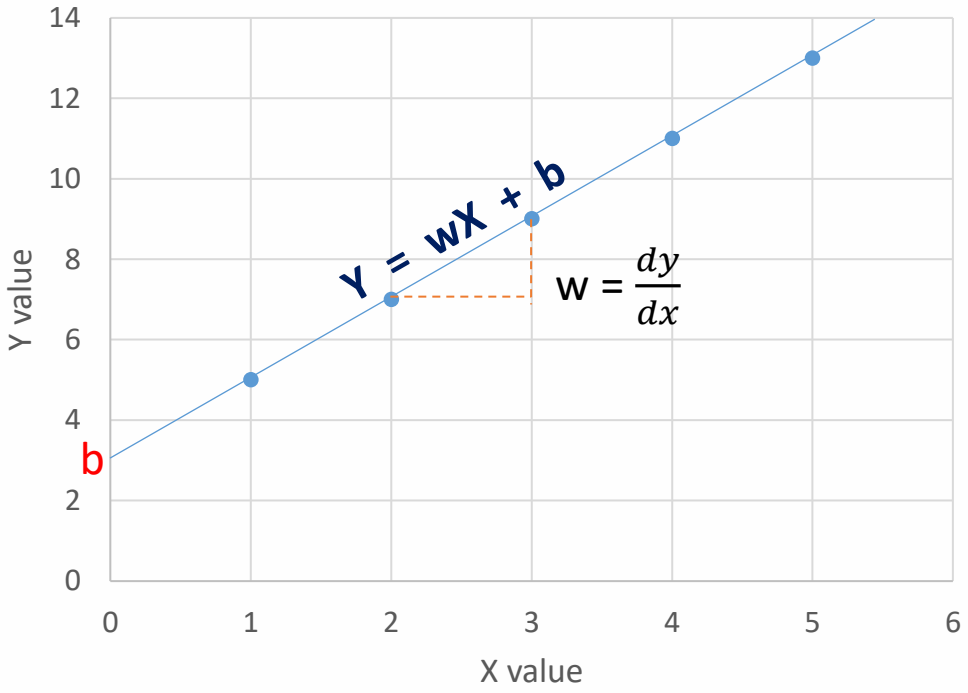
$$Y = w_1*X_1 + w_2*X_2 + w_3*X_3 + b$$

X – feature or input variable
Y – Target or output variable
w – weight
b – bias

Bias:

Bias is the offset value given to the model. Bias is used to shift the model in a particular direction. It is similar to a Y-intercept. ‘b’ is equal to ‘Y’ when all the feature values are zero.

Linear Regression



Y = wX + b

X --> X value
Y --> Y value
w --> weight
b --> bias

Bias:

Bias is the offset value given to the model. Bias is used to shift the model in a particular direction. It is similar to a Y-intercept. ‘b’ is equal to ‘Y’ when all the feature values are zero.

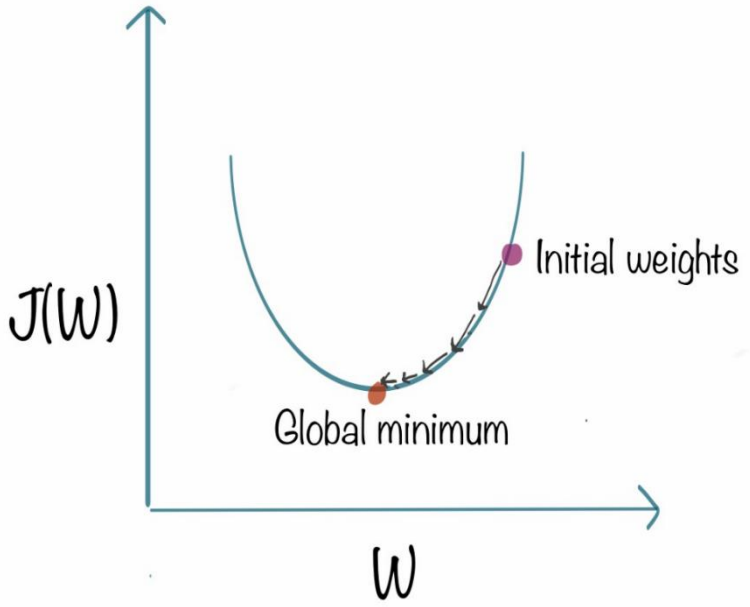
Hyperparameters

Learning Rate:

The **Learning Rate** is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function.

Number of Epochs:

Number of Epochs represents the number of times the model iterates over the entire dataset.



Gradient Descent

Types of Parameters

Model Parameters

These are the parameters of the model that can be determined by training with training data. These can be considered as internal Parameters.

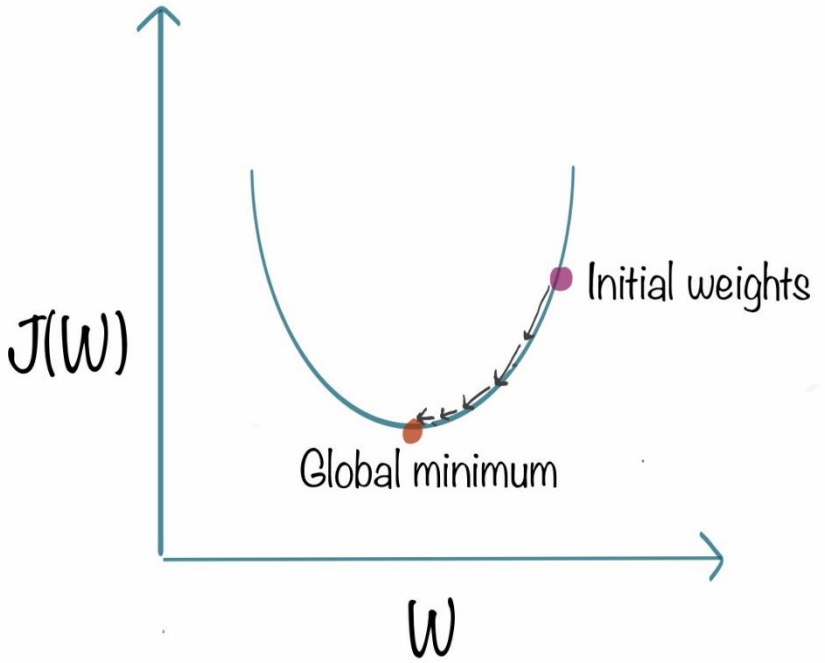
- **Weights**
- **Bias**

Hyperparameters

Hyperparameters are parameters whose values control the learning process. These are adjustable parameters used to obtain an optimal model. External Parameters.

- **Learning rate**
- **Number of Epochs**

Gradient Descent in Machine Learning



Model Parameters

Weights:

Weight decides how much influence the input will have on the output.

$$Y = w*X + b$$

$$Y = w_1*X_1 + w_2*X_2 + w_3*X_3 + b$$

X – feature or input variable
Y – Target or output variable
w – weight
b – bias

Bias:

Bias is the offset value given to the model. Bias is used to shift the model in a particular direction. It is similar to a Y-intercept. ‘b’ is equal to ‘Y’ when all the feature values are zero.

Hyperparameters

Learning Rate:

The **Learning Rate** is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function.

Number of Epochs:

Number of Epochs represents the number of times the model iterates over the entire dataset.

Loss Function

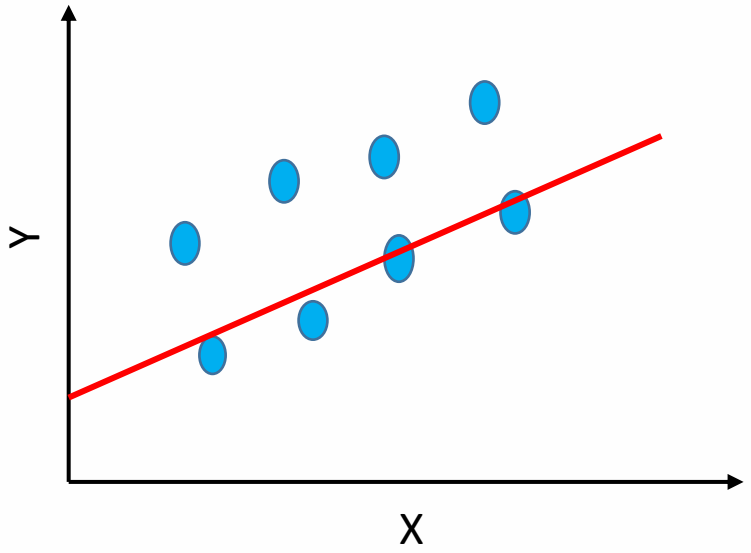
Loss function measures how far an estimated value is from its true value.

It is helpful to determine which model performs better & which parameters are better.

$$\textbf{\textit{Loss}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Model Optimization

Optimization refers to determining best parameters for a model, such that the loss function of the model decreases, as a result of which the model can predict more accurately.

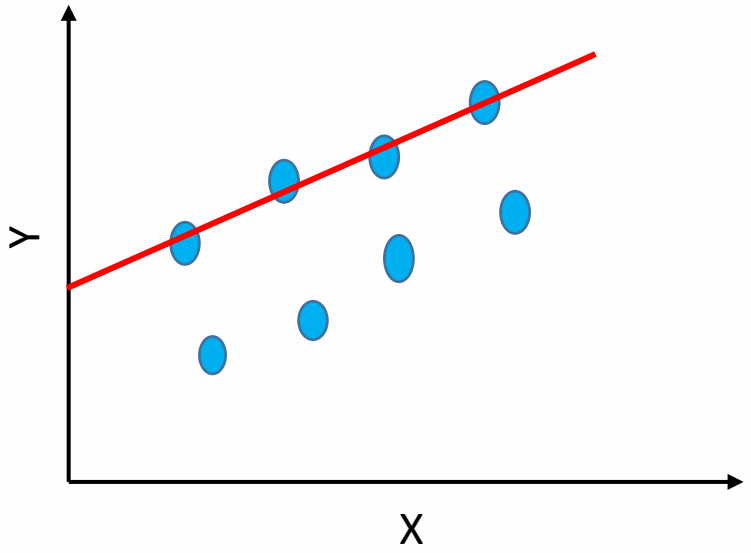


$Y = w_1X + b_1$

(w_1 & b_1 are the parameters of the line)

Model Optimization

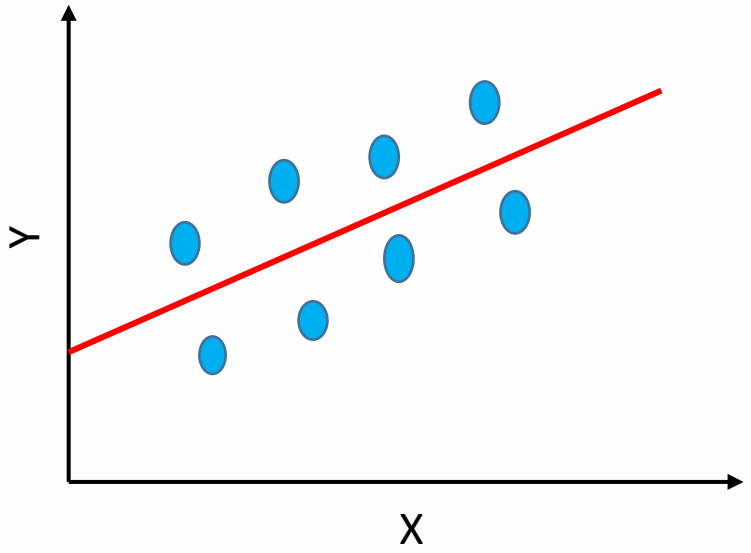
Optimization refers to determining best parameters for a model, such that the loss function of the model decreases, as a result of which the model can predict more accurately.



$$Y = w_2X + b_2$$

Model Optimization

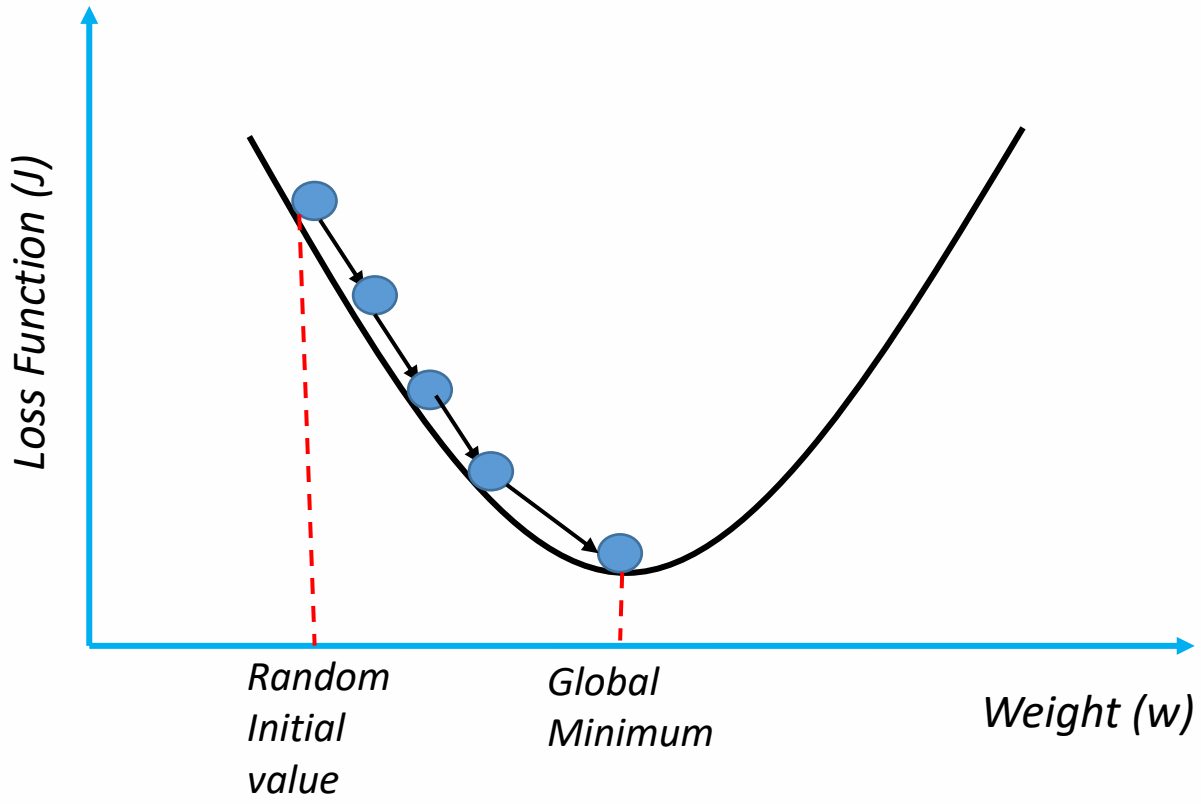
Optimization refers to determining best parameters for a model, such that the loss function of the model decreases, as a result of which the model can predict more accurately.



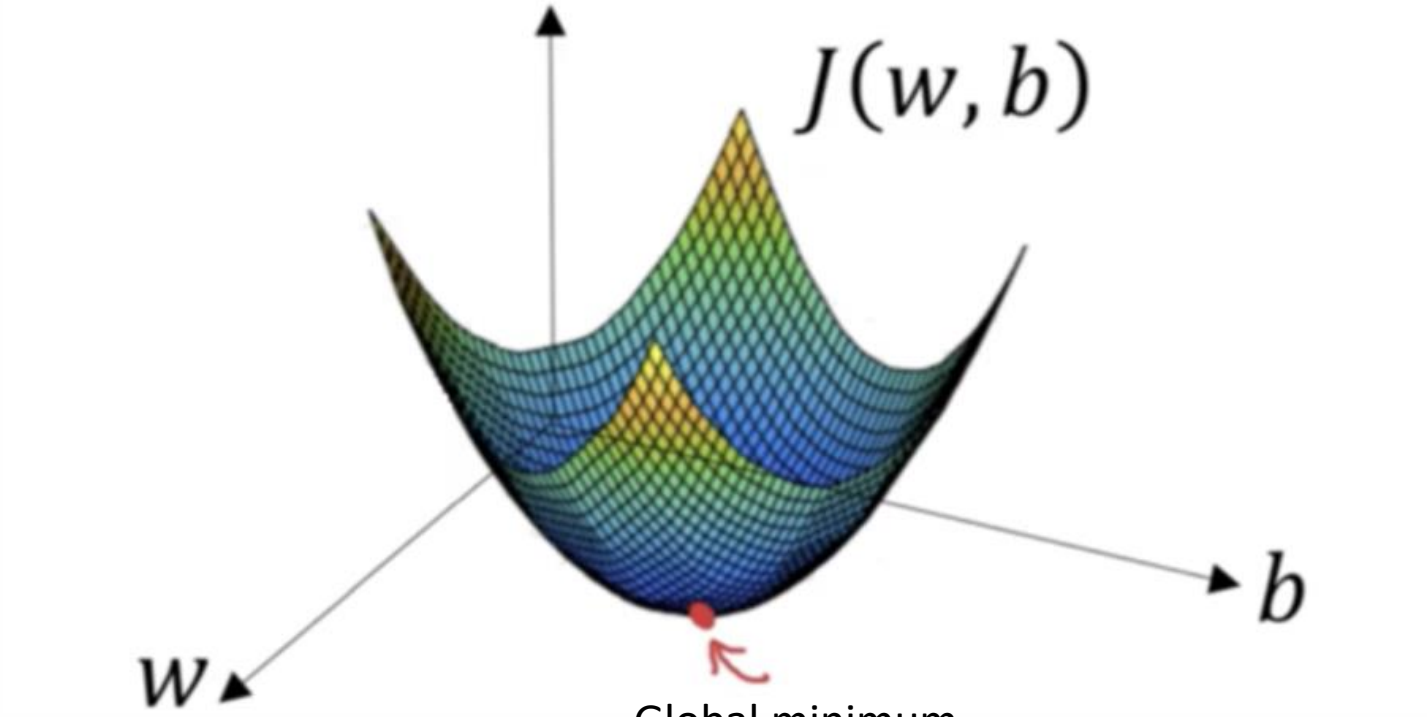
$Y = w_3X + b_3$

Hence, w_3 & b_3 are the best parameters

Gradient Descent



Gradient Descent in 3 Dimension



Global minimum

Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the loss function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$w_2 = w_1 - L*dw$$

$$b_2 = b_1 - L*db$$

$\frac{dy}{dx}$

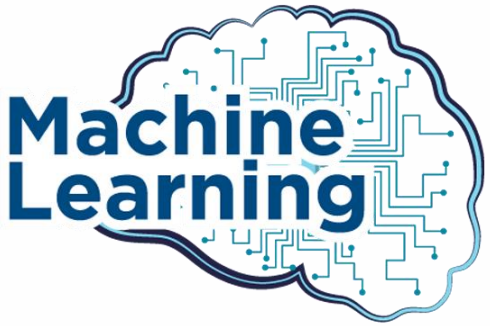
w --> weight
b --> bias
L --> Learning Rate

dw --> Partial Derivative of loss function with respect to m ~~m~~ w

db --> Partial Derivative of loss function with respect to c ~~c~~ b

Linear Regression
- intuition

Machine Learning



Data

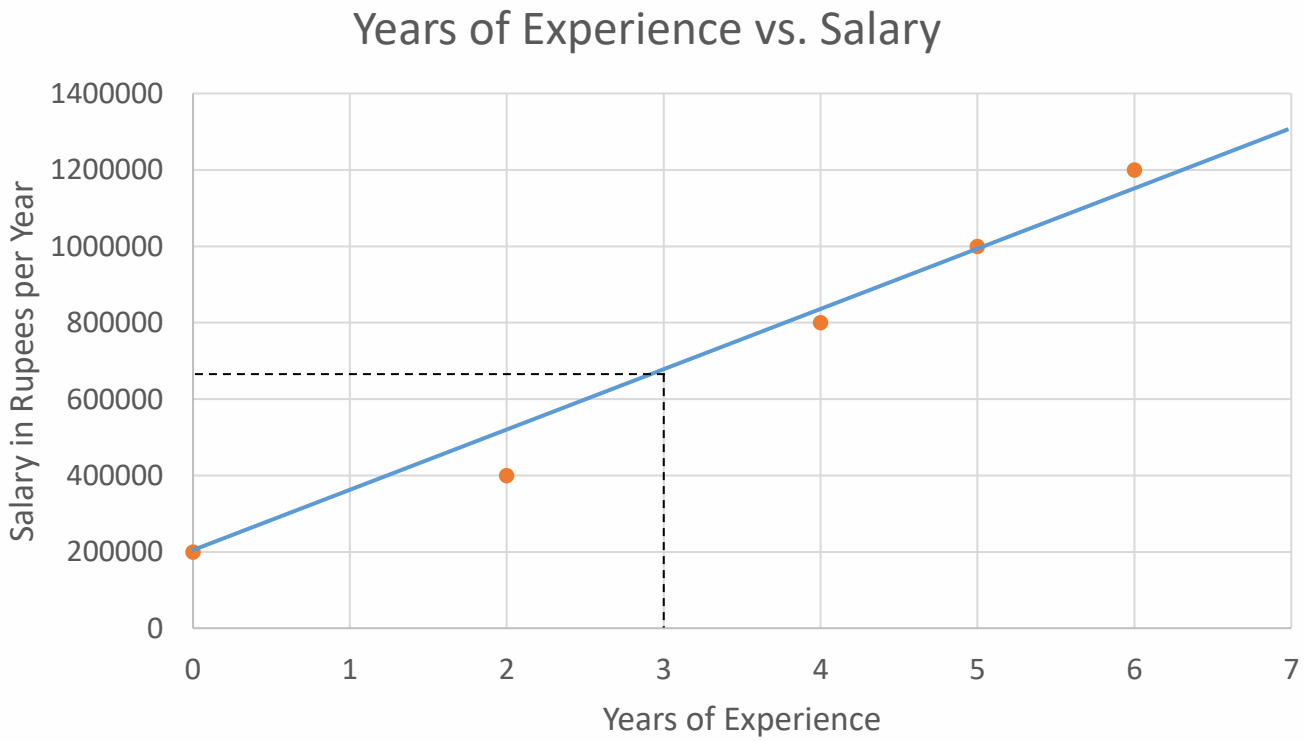
Machine Learning model

Linear Regression

Experience in Years	0	2	4	5	6
Salary	2,00,000	4,00,000	8,00,000	10,00,000	12,00,000

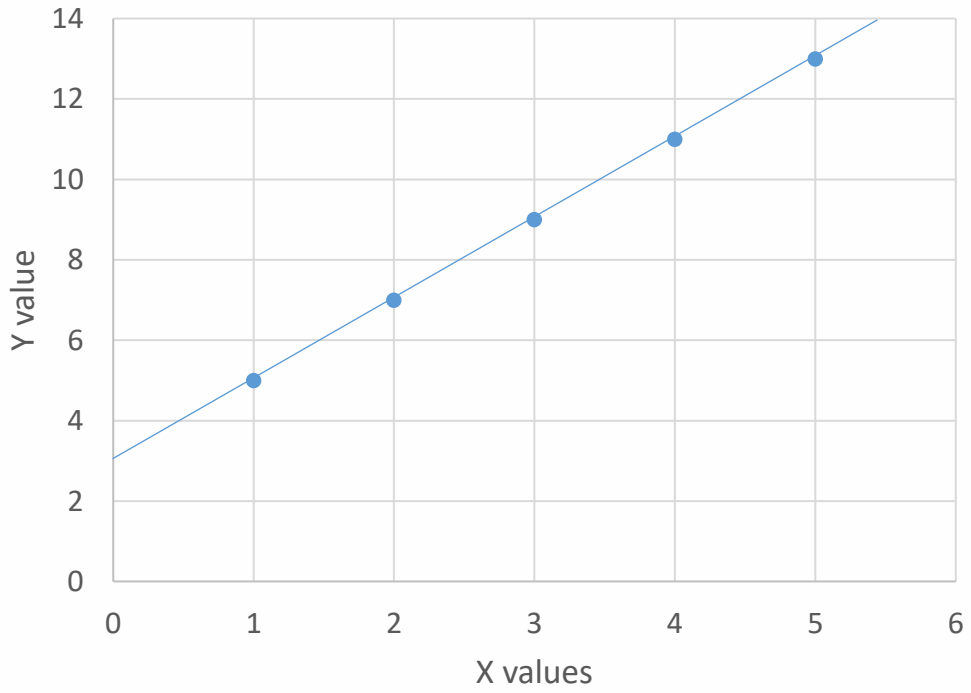
What would be the **salary** of a person with **3 years of Experience?**

~ ₹ 650000 per Year



Linear Regression

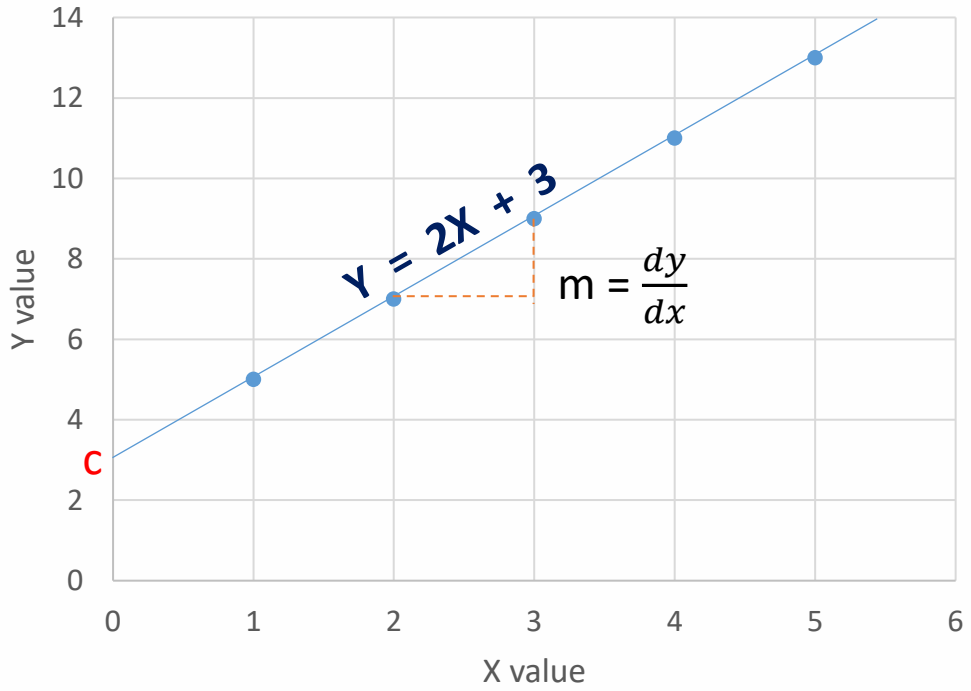
X	1	2	3	4	5
Y	5	7	9	11	13



$Y = mX + c$

X --> X value
Y --> Y value
m --> Slope
c --> Intercept

Linear Regression



Inference: The above Line equation is a function that relates X and Y.
For a given value of X, we can find the corresponding value of Y

Equation of a Straight Line : **Y = mX + c**

Find the values of m and c:

Point P1 (2,7)
Point P2 (3,9)

Slope, m = $\frac{y_2 - y_1}{x_2 - x_1} = \frac{9 - 7}{3 - 2} = 2$

m = 2

Intercept, c:

Point (4,11)

Y = 2X + c

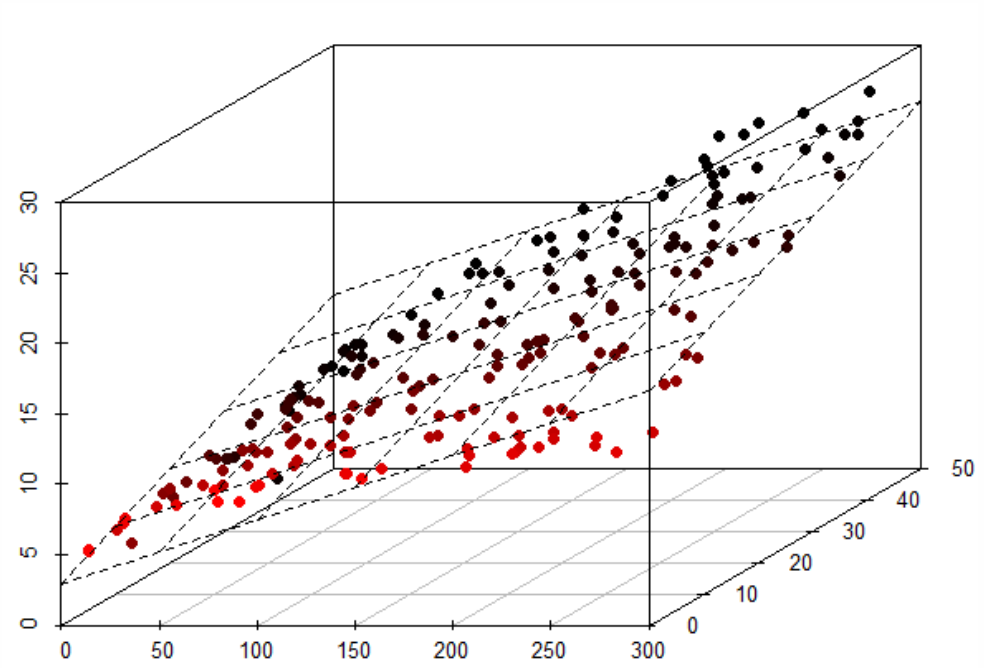
11 = 2(4) + c

c = 3

What if there are more than 2 Variables?

Multiple Linear Regression

Multiple linear regression is a model for predicting the value of one dependent variable based on two or more independent variables.



Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

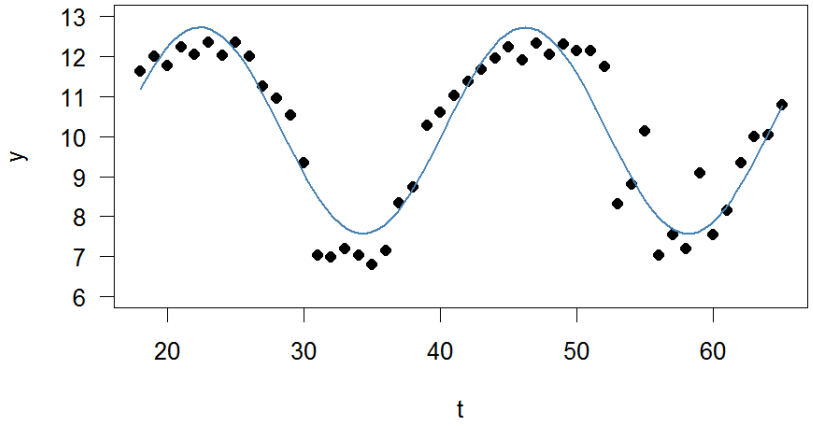
Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

Linear Regression

Advantages:

1. Very simple to implement
2. Performs well on data with linear relationship

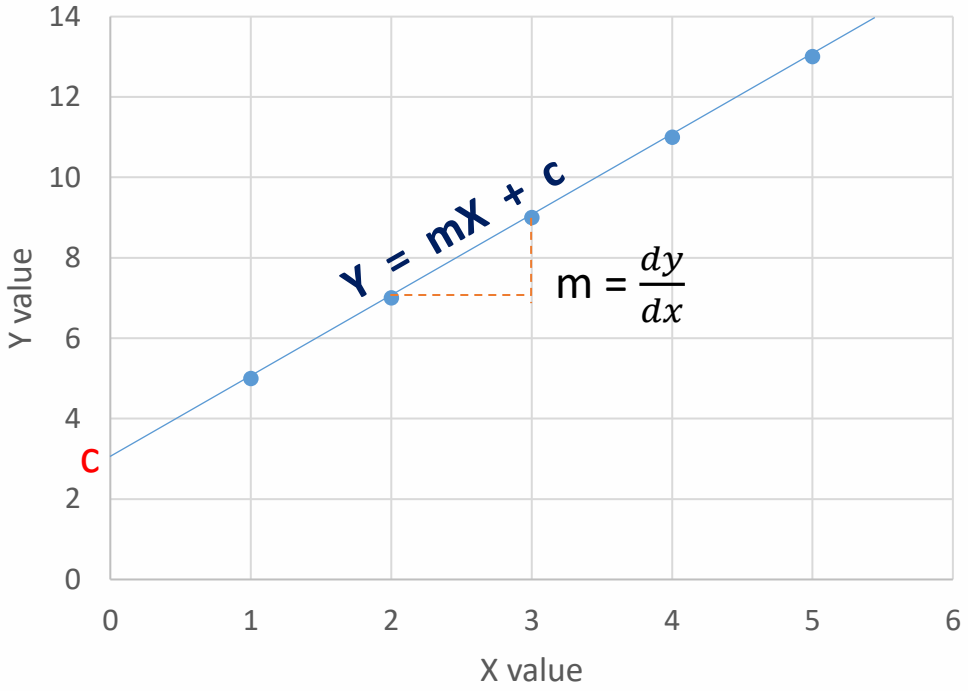


Disadvantages:

1. Not suitable for data having non-linear relationship
2. Underfitting issue
3. Sensitive to Outliers

Linear Regression - Mathematical Understanding

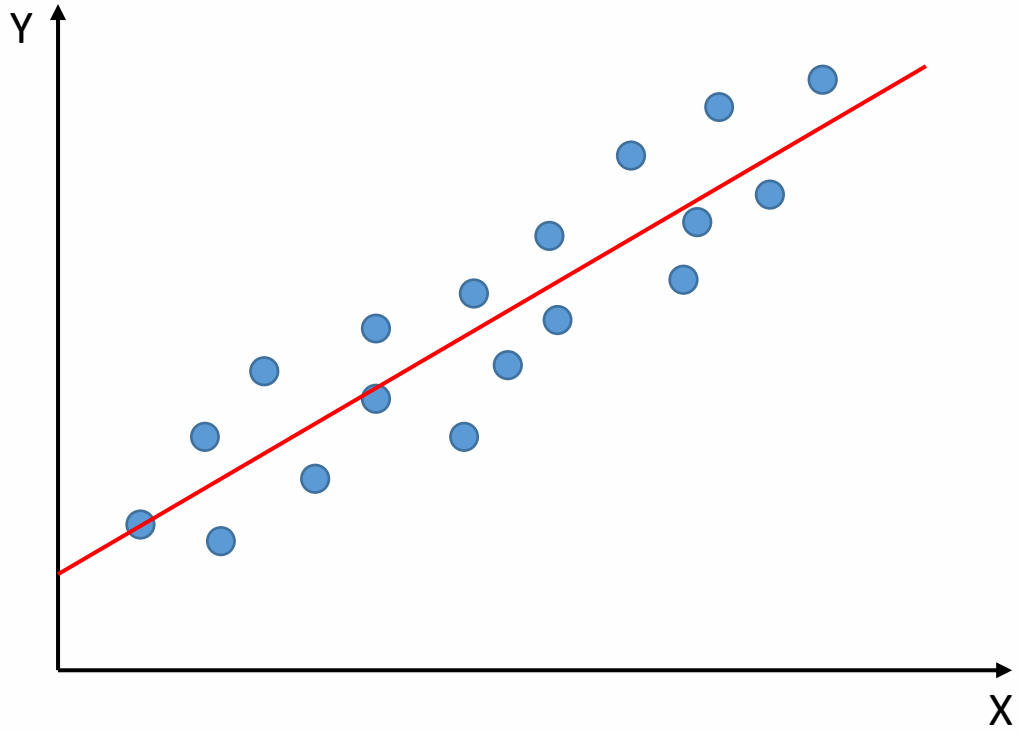
Linear Regression



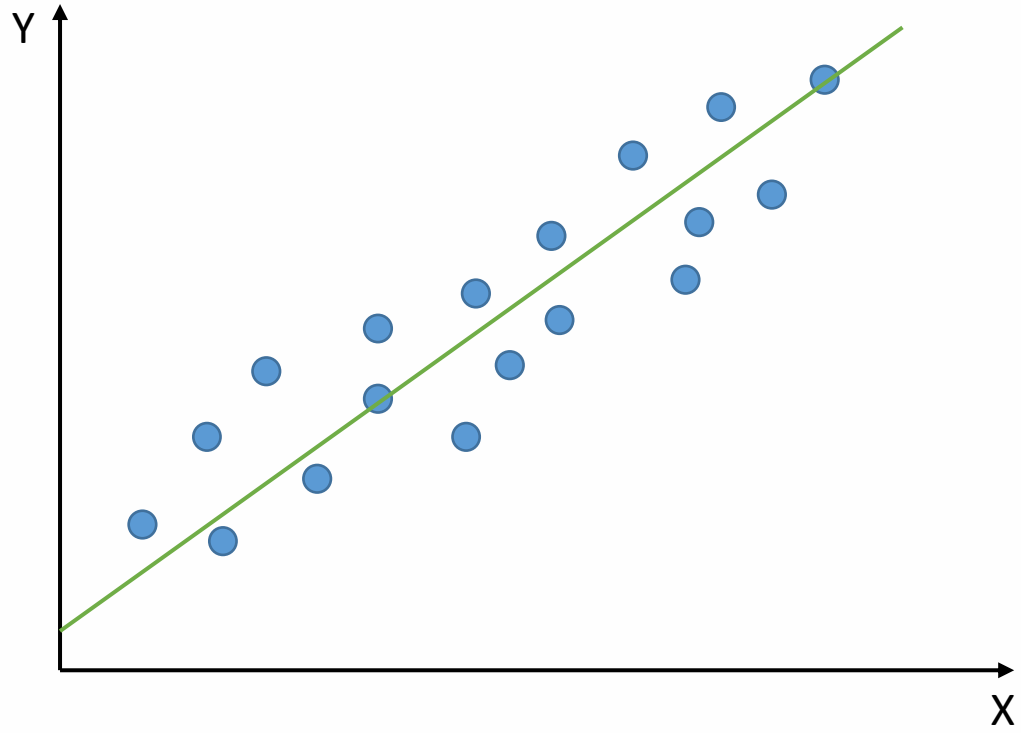
Y = mX + c

X --> X value
Y --> Y value
m --> Slope
c --> Intercept

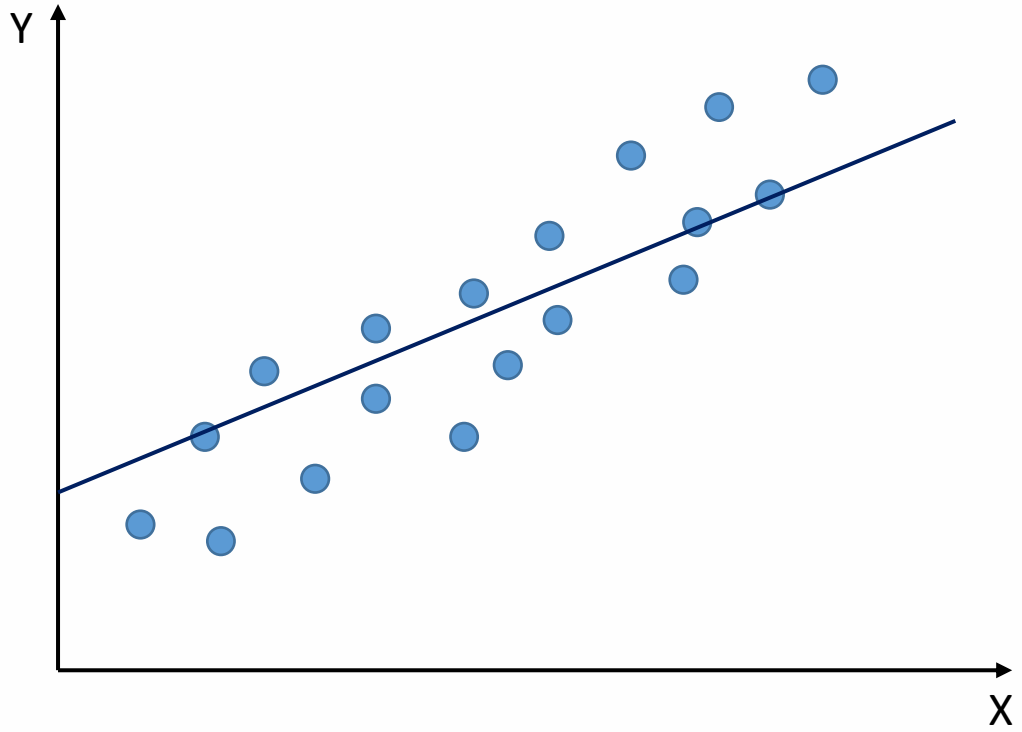
Linear Regression



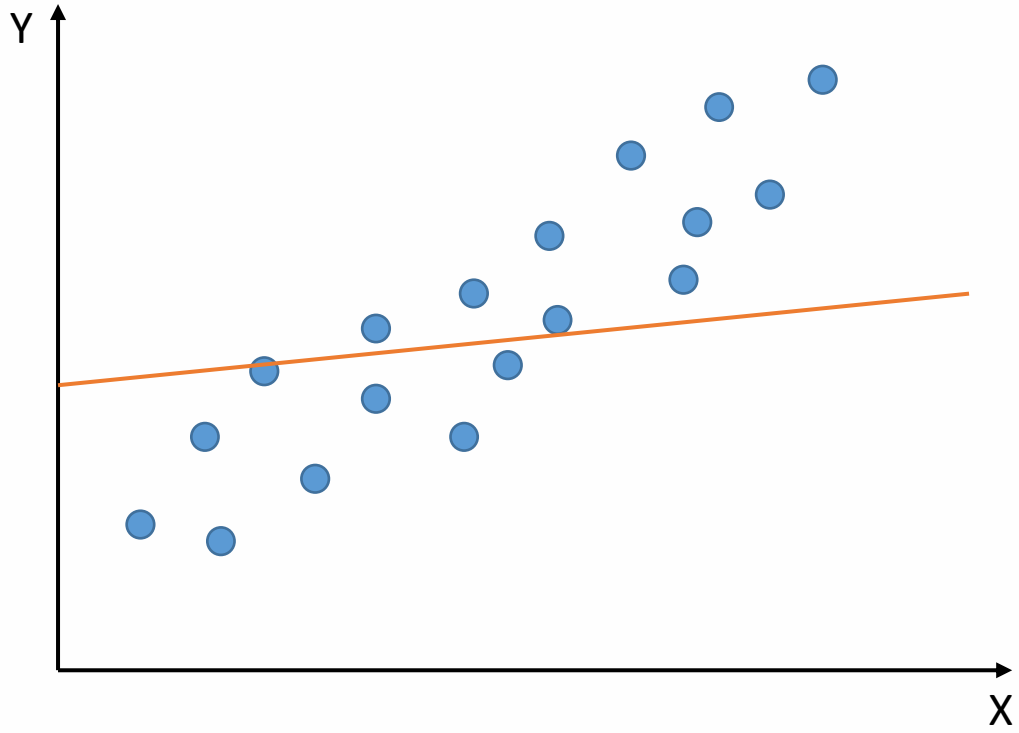
Linear Regression



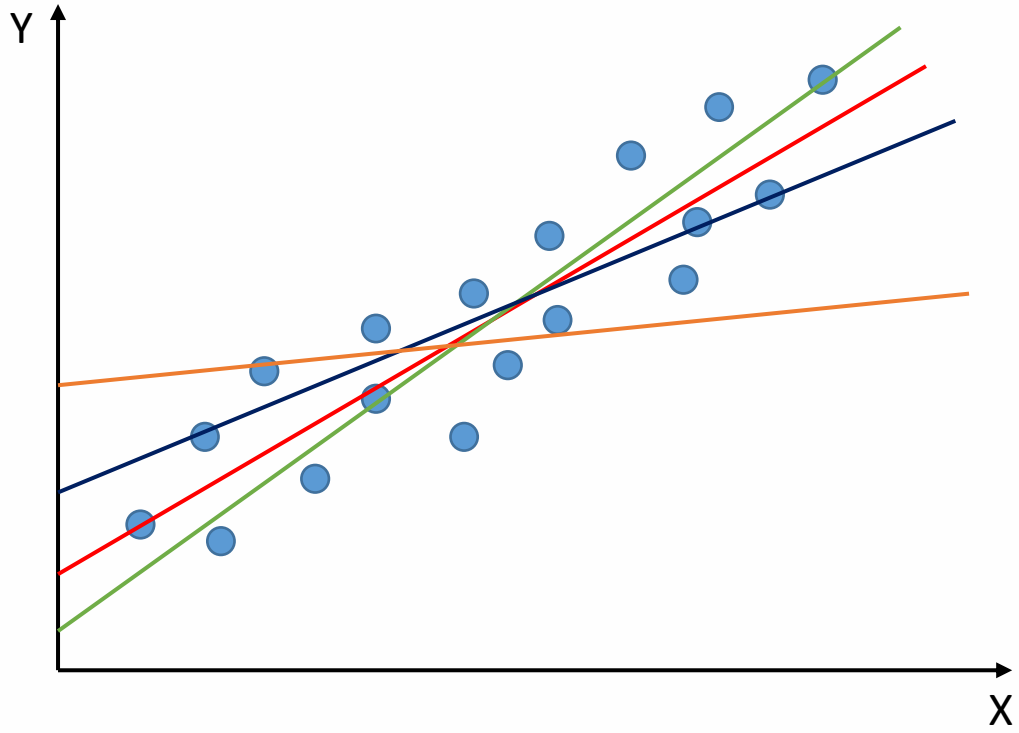
Linear Regression



Linear Regression



Linear Regression



Loss Function

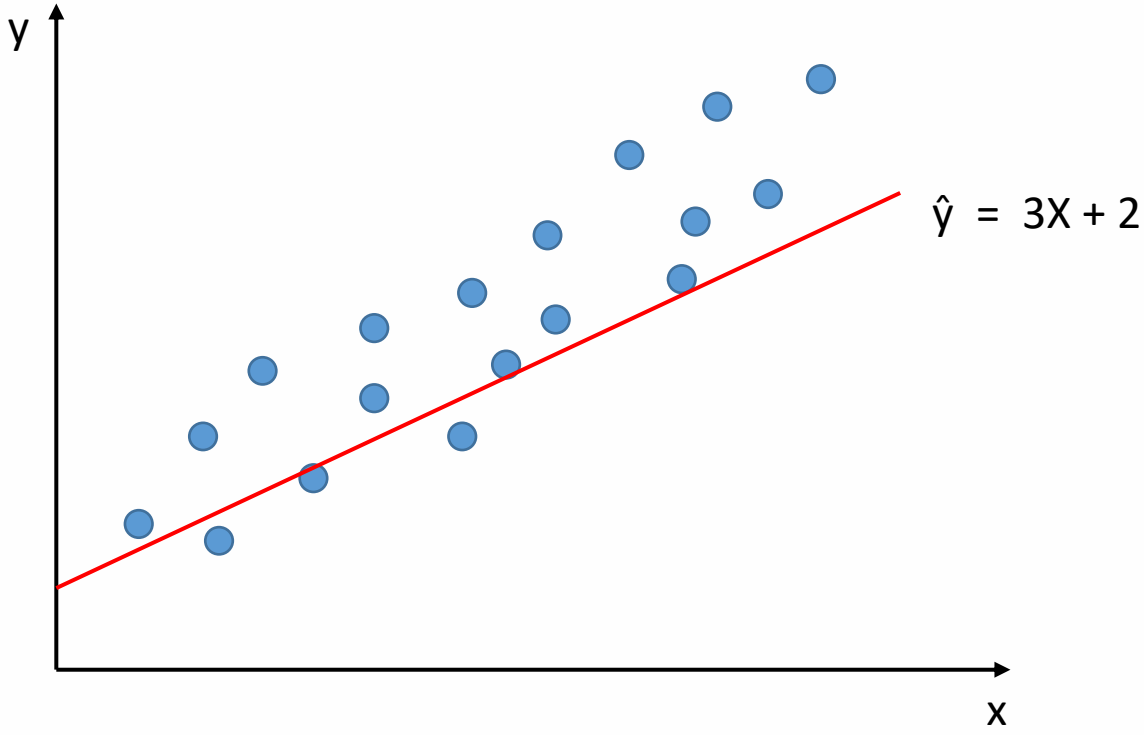
Loss function measures how far an estimated value is from its true value.

It is helpful to determine which model performs better & which parameters are better.

$$\textbf{\textit{Loss}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Loss Function

Randomly assigned Parameters: m = 3; c = 2



x	y	ŷ
2	10	8
3	14	11
4	18	14
5	22	17
6	26	20

Loss Function

x	y	ŷ
2	10	8
3	14	11
4	18	14
5	22	17
6	26	20

$$\textbf{Loss} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

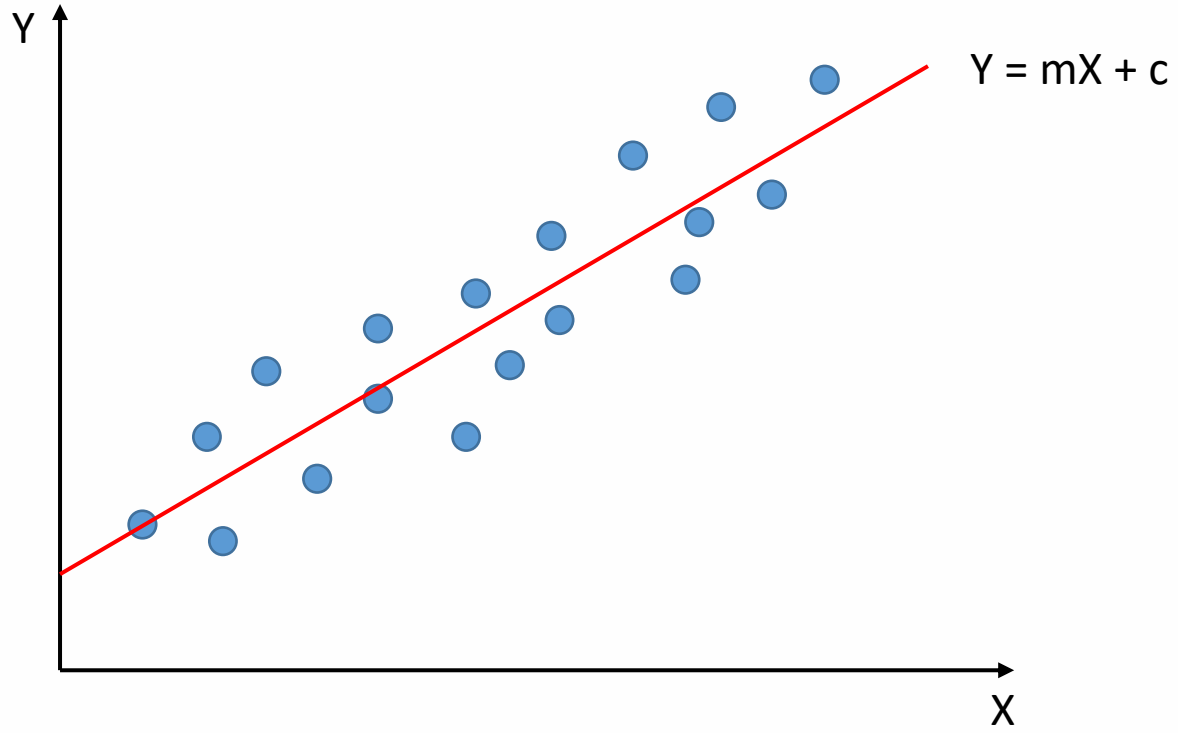
Loss = [$(10 - 8)^2 + (14 - 11)^2 + (18 - 14)^2 + (22 - 17)^2 + (26 - 20)^2$] / 5

Loss = [4 + 9 + 16 + 25 + 36] / 5

Loss = 18

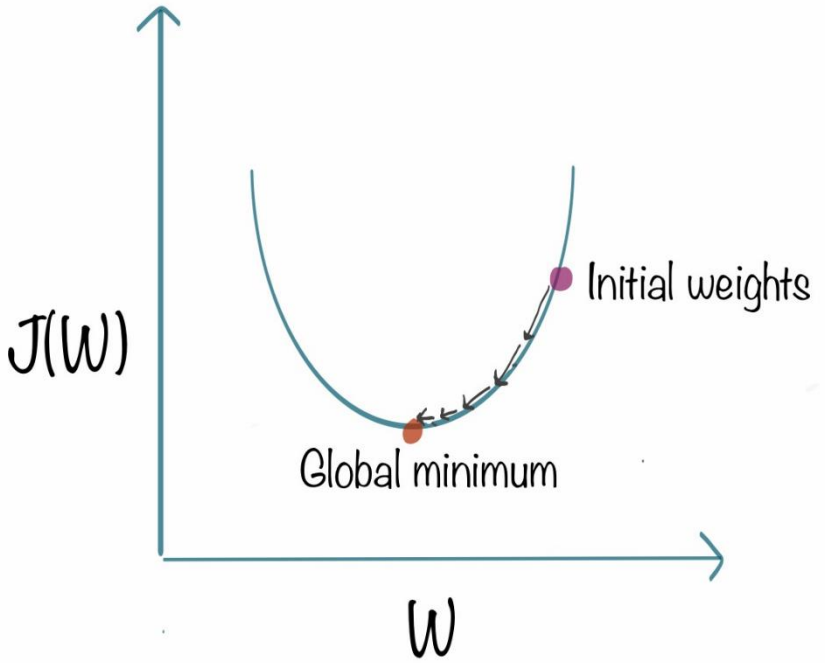
Low Loss value → High Accuracy

Linear Regression



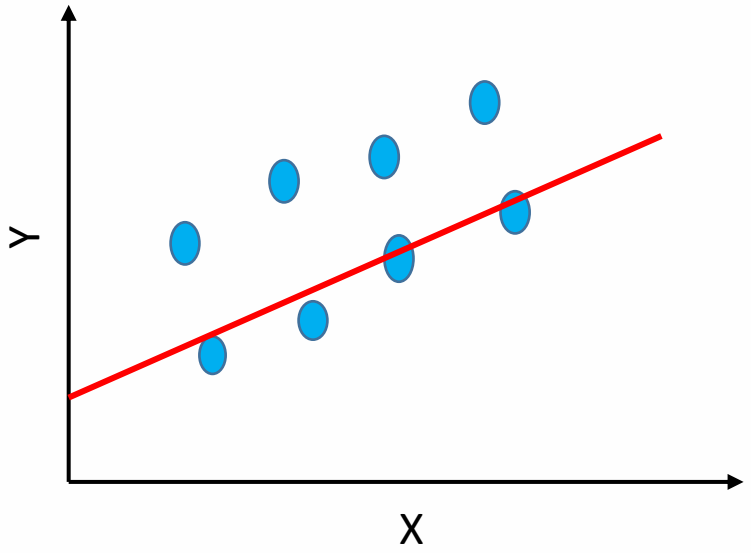
Best Fit

Gradient Descent for Linear Regression



Model Optimization

Optimization refers to determining best parameters for a model, such that the loss function of the model decreases, as a result of which the model can predict more accurately.

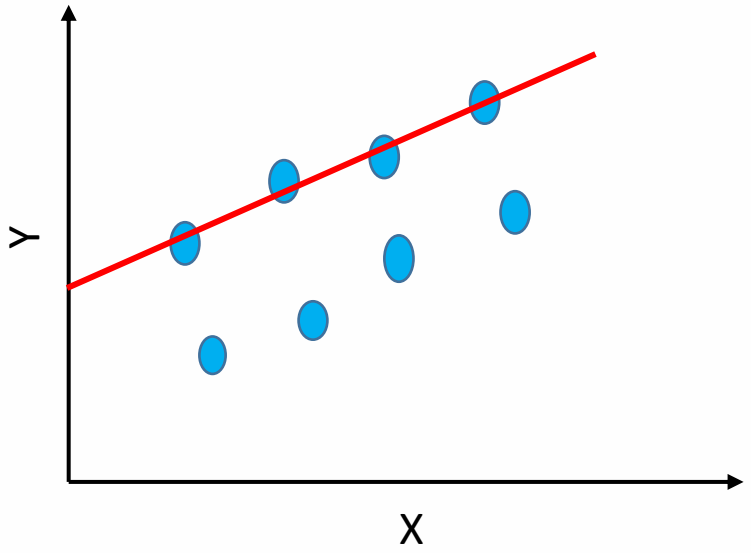


$Y = m_1X + C_1$

(m_1 & C_1 are the parameters of the line)

Model Optimization

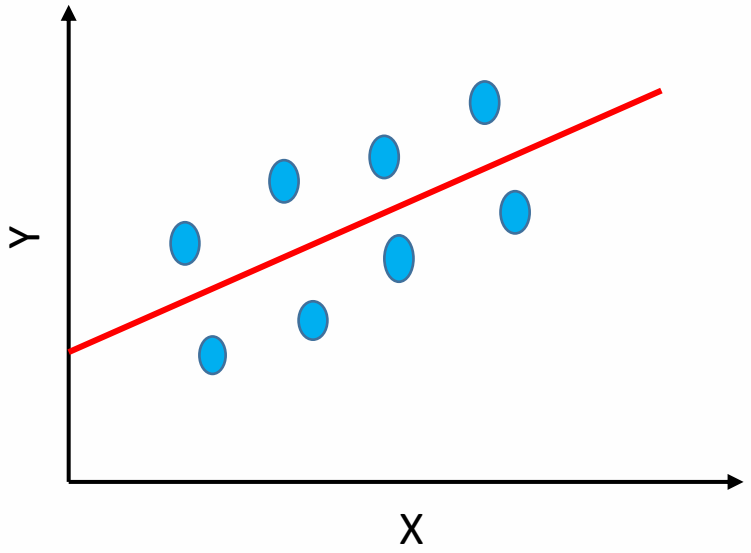
Optimization refers to determining best parameters for a model, such that the loss function of the model decreases, as a result of which the model can predict more accurately.



$$Y = m_2X + C_2$$

Model Optimization

Optimization refers to determining best parameters for a model, such that the loss function of the model decreases, as a result of which the model can predict more accurately.



$Y = m_3X + C_3$

Hence, m_3 & C_3 are the best parameters

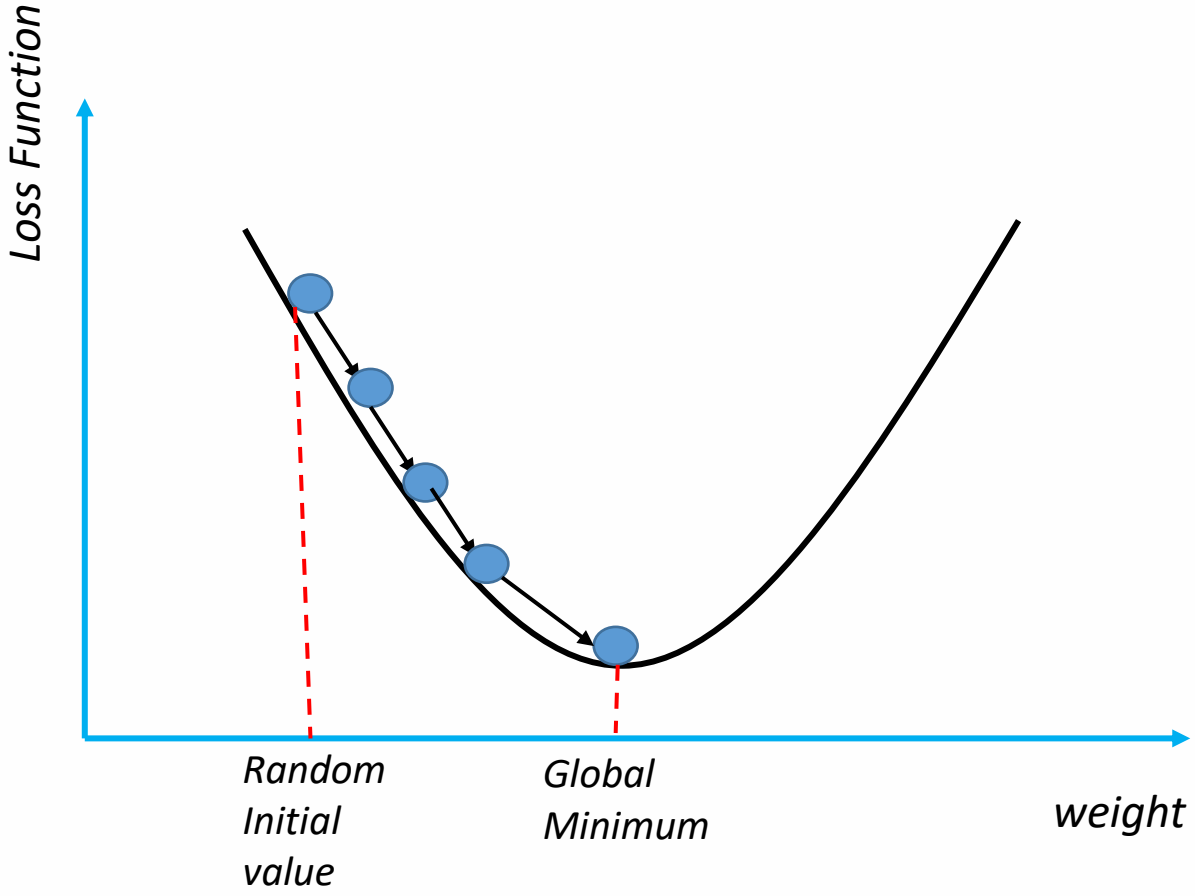
Loss Function

Loss function measures how far an estimated value is from its true value.

It is helpful to determine which model performs better & which parameters are better.

$$\textbf{\textit{Loss}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Gradient Descent



Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the loss function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$m = m - LD_m$$

$$c = c - LDc$$

m --> slope
c --> intercept
L --> Learning Rate

D_m --> Partial Derivative of loss function with respect to m

D_c --> Partial Derivative of loss function with respect to c

Gradient Descent

$$D_m = \frac{\partial(Cost\ Function)}{\partial m} = \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=0}^{n}\left(y_i - y_{i\,pred}\right)^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}\left(y_i - (mx_i + c)\right)^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}\left(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic\right)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n} x_i\left(y_i - (mx_i + c)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n} x_i\left(y_i - y_{i\,pred}\right)$$

$$D_c = \frac{\partial(Cost\ Function)}{\partial c} = \frac{\partial}{\partial c}\left(\frac{1}{n}\sum_{i=0}^{n}\left(y_i - y_{i\,pred}\right)^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}\left(y_i - (mx_i + c)\right)^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}\left(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic\right)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}\left(y_i - (mx_i + c)\right)$$

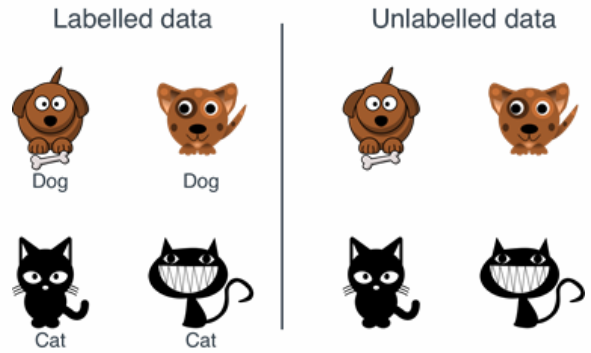
$$\frac{-2}{n}\sum_{i=0}^{n}\left(y_i - y_{i\,pred}\right)$$

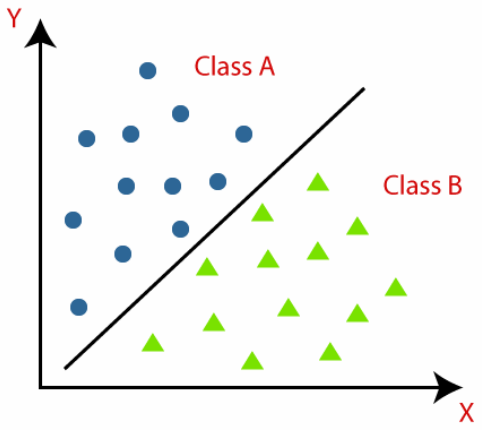
Logistic Regression - intuition

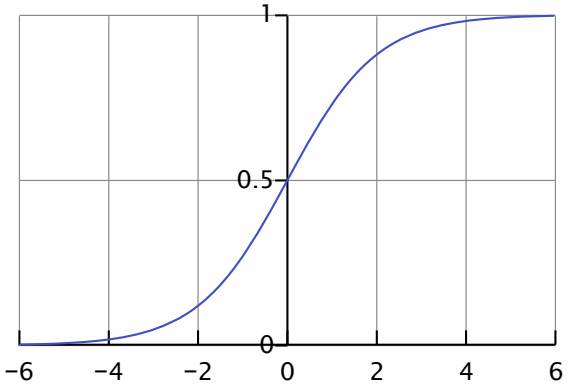
Logistic Regression

About Logistic Regression:

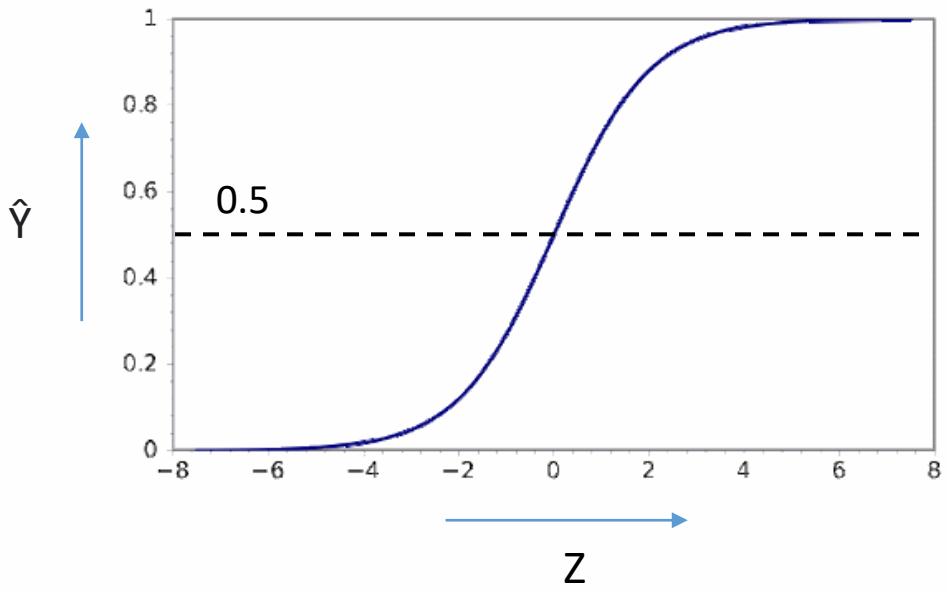
1. Supervised Learning Model
2. Classification model
3. Best for Binary Classification Problem
4. Uses Sigmoid function







Logistic Regression



$$\hat{Y} = \frac{1}{1 + e^{-Z}}$$

$$Z = w.X + b$$

Sigmoid Function

Ŷ - Probability that (y = 1)

Ŷ = P(Y=1 | X)

X - input features

w – weights
(number of weights is equal to the number of input features in a dataset)

b - bias

Ŷ = σ (Z)

Logistic Regression

Advantages:

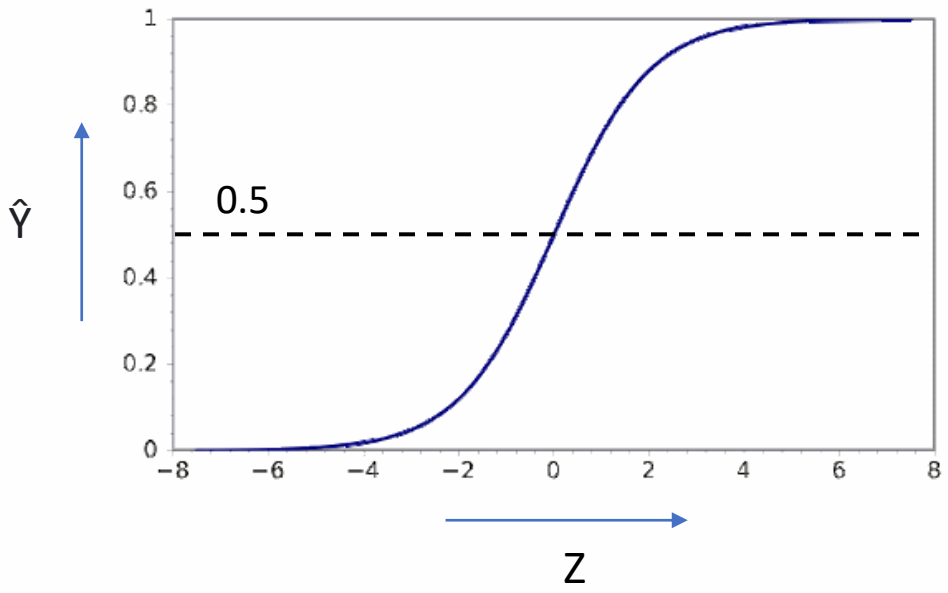
1. Easy to implement
2. Performs well on data with linear relationship
3. Less prone to over-fitting for low dimensional dataset

Disadvantages:

1. High dimensional dataset causes over-fitting
2. Difficult to capture complex relationships in a dataset
3. Sensitive to Outliers
4. Needs a larger dataset

Math behind Logistic Regression

Logistic Regression



$$\hat{Y} = \frac{1}{1 + e^{-Z}}$$

$$Z = w.X + b$$

Sigmoid Function

Ŷ - Probability that (y = 1)

Ŷ = P(Y=1 | X)

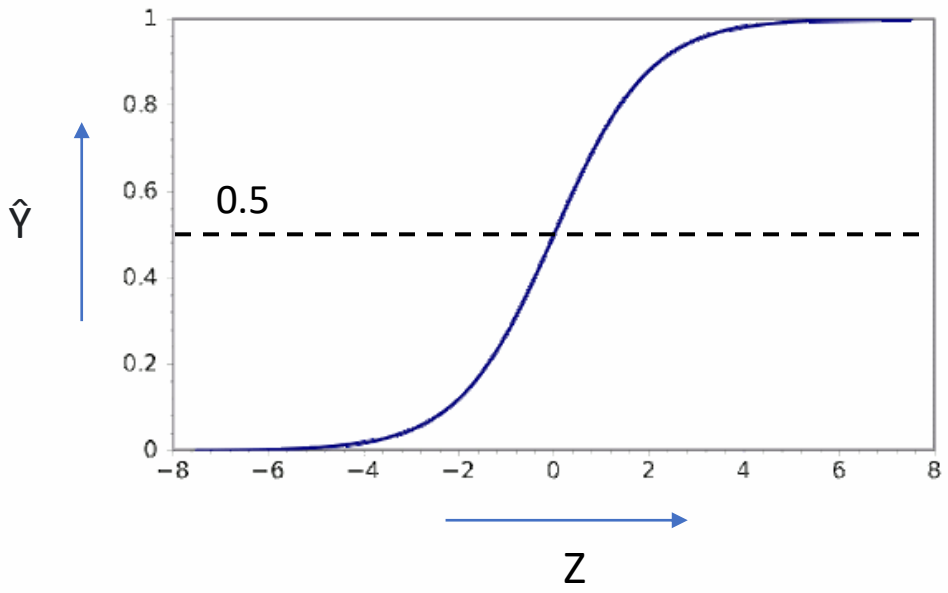
X - input features

w – weights
(number of weights is equal to the number of input features in a dataset)

b - bias

Ŷ = σ (Z)

Logistic Regression



$$\hat{Y} = \frac{1}{1 + e^{-Z}}$$

Sigmoid Function

Z = 5X + 10

Ŷ - Probability that (y = 1)

Ŷ = P(Y=1 | X)

X - input features

w – weights
(number of weights is equal to the number of input features in a dataset)

b - bias

Ŷ = σ (Z)

Logistic Regression

X	-9	-8	0	8	9
$\hat{Y}$					

$$Z = 5X + 10 \qquad \hat{Y} = \frac{1}{1+e^{-Z}}$$

X = -9

Z = 5(-9) + 10

Z = -35

$$\hat{Y} = \frac{1}{1+e^{35}}$$

$\hat{Y} = 0$

X = -8

Z = 5(-8) + 10

Z = -30

$$\hat{Y} = \frac{1}{1+e^{30}}$$

$\hat{Y} = 0$

X = 0

Z = 5(0) + 10

Z = 10

$$\hat{Y} = \frac{1}{1+e^{-10}}$$

$\hat{Y} = 1$

X = 8

Z = 5(8) + 10

Z = 50

$$\hat{Y} = \frac{1}{1+e^{-50}}$$

$\hat{Y} = 1$

X = 9

Z = 5(9) + 10

Z = 55

$$\hat{Y} = \frac{1}{1+e^{-55}}$$

$\hat{Y} = 1$

Logistic Regression

X	-9	-8	0	8	9
$\hat{Y}$	0	0	1	1	1

$$Z = 5X + 10 \qquad \hat{Y} = \frac{1}{1+e^{-Z}}$$

X = -9

Z = 5(-9) + 10

Z = -35

$$\hat{Y} = \frac{1}{1+e^{35}}$$

$\hat{Y} = 0$

X = -8

Z = 5(-8) + 10

Z = -30

$$\hat{Y} = \frac{1}{1+e^{30}}$$

$\hat{Y} = 0$

X = 0

Z = 5(0) + 10

Z = 10

$$\hat{Y} = \frac{1}{1+e^{-10}}$$

$\hat{Y} = 1$

X = 8

Z = 5(8) + 10

Z = 50

$$\hat{Y} = \frac{1}{1+e^{-50}}$$

$\hat{Y} = 1$

X = 9

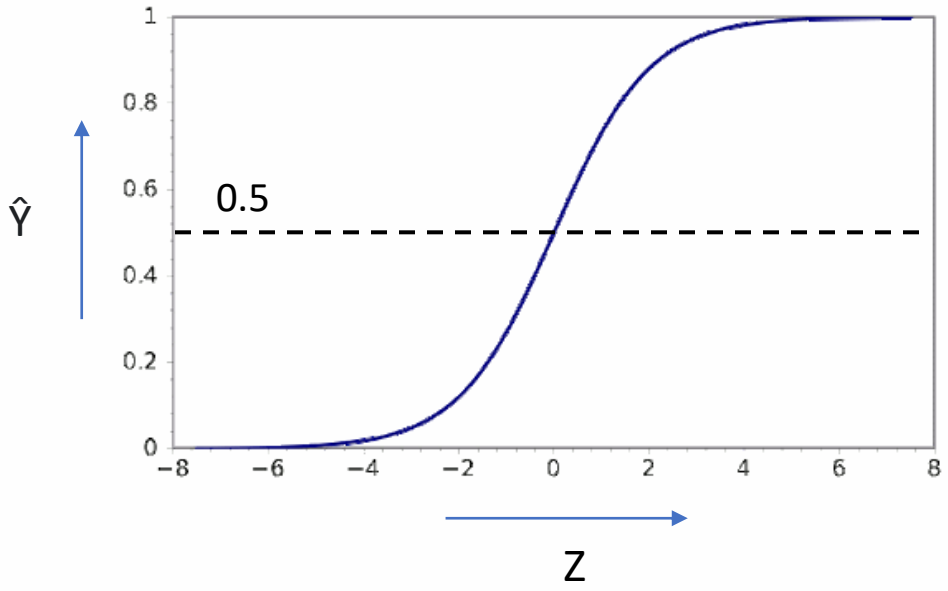
Z = 5(9) + 10

Z = 55

$$\hat{Y} = \frac{1}{1+e^{-55}}$$

$\hat{Y} = 1$

Logistic Regression



$$\hat{Y} = \frac{1}{1 + e^{-Z}}$$

$$Z = w.X + b$$

Sigmoid Function

Inference:

If Z value is a large positive number, $\hat{Y} = \frac{1}{1 + 0}$

$\hat{Y} = 1$

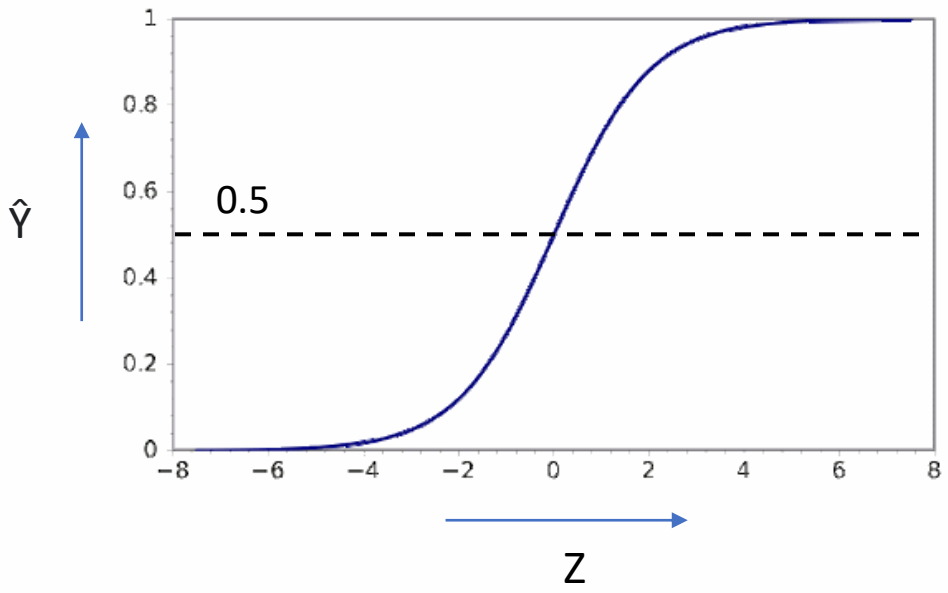
If Z value is a large negative number, $\hat{Y} = \frac{1}{1 + (large\ positive\ number)}$

$\hat{Y} = 0$

Loss Function & Cost Function for Logistic Regression

J (w,b)

Logistic Regression



$$\hat{Y} = \frac{1}{1 + e^{-Z}}$$

$$Z = w.X + b$$

Sigmoid Function

Ŷ - Probability that (y = 1)

Ŷ = P(Y=1 | X)

X - input features

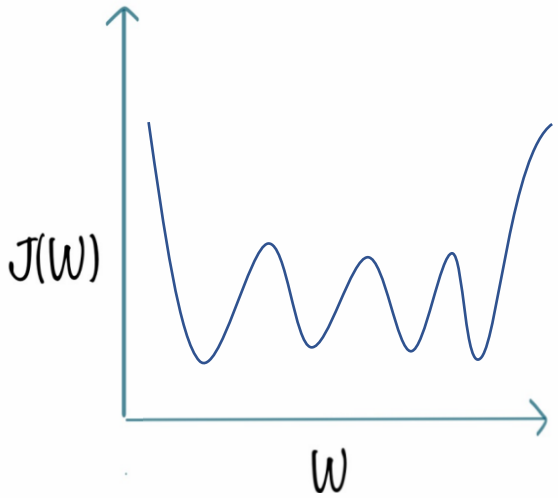
w – weights
(number of weights is equal to the number of input features in a dataset)

b - bias

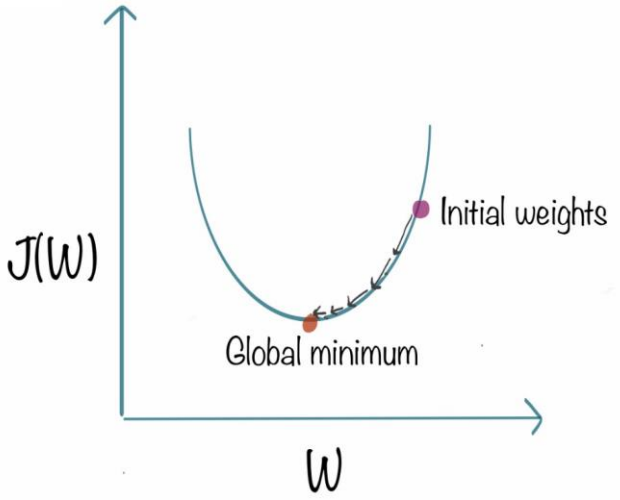
Loss Function

Loss function measures how far an estimated value is from its true value.

$$\textbf{\textit{Loss}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$



Gradient Descent
With Local minima



Gradient Descent
With Global minima

Loss Function for Logistic Regression

Binary Cross Entropy Loss Function (or) Log Loss :

$$L(y, \hat{y}) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

When $y = 1$, $\Rightarrow$ $L(1, \hat{y}) = -(1 \log \hat{y} + (1 - 1) \log(1 - \hat{y}))$ $\Rightarrow$ $L(1, \hat{y}) = -\log \hat{y}$

We always want a smaller Loss Function value, hence, $\hat{y}$ should be very large,
so that $(-\log \hat{y})$ will be a large negative number.

When $y = 0$, $\Rightarrow$ $L(0, \hat{y}) = -(0 \log \hat{y} + (1 - 0) \log(1 - \hat{y}))$ $\Rightarrow$ $L(0, \hat{y}) = -\log(1 - \hat{y})$

We always want a smaller Loss Function value, hence, $\hat{y}$ should be very small,
so that $-\log(1 - \hat{y})$ will be a large negative number.

Cost Function for Logistic Regression

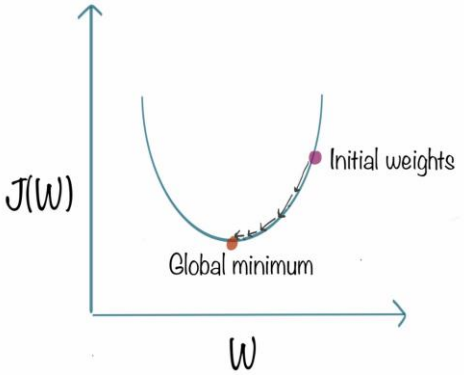
Loss function (L) mainly applies for a single training set as compared to the cost function (J) which deals with a penalty for a number of training sets or the complete batch.

$$L(y, \hat{y}) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} (L(y^{(i)}, \hat{y}^{(i)})) = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

(‘m’ denotes the number of data points in the training set)

Gradient Descent for Logistic Regression



Logistic Regression

About Logistic Regression:

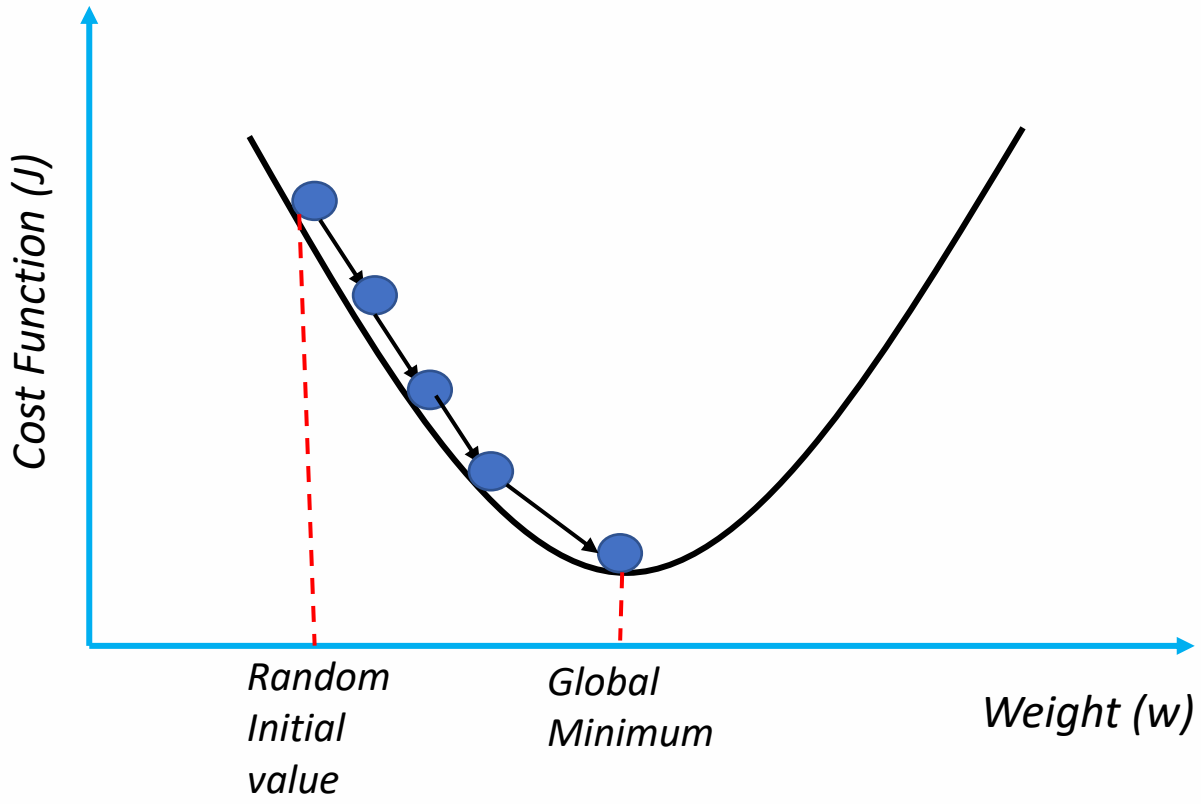
1. Supervised Learning Model
2. Classification model
3. Best for Binary Classification Problem
4. Uses Sigmoid function
5. Binary Cross Entropy Loss Function (or) Log Loss

$$\hat{Y} = \frac{1}{1 + e^{-Z}} \qquad Z = w.X + b$$

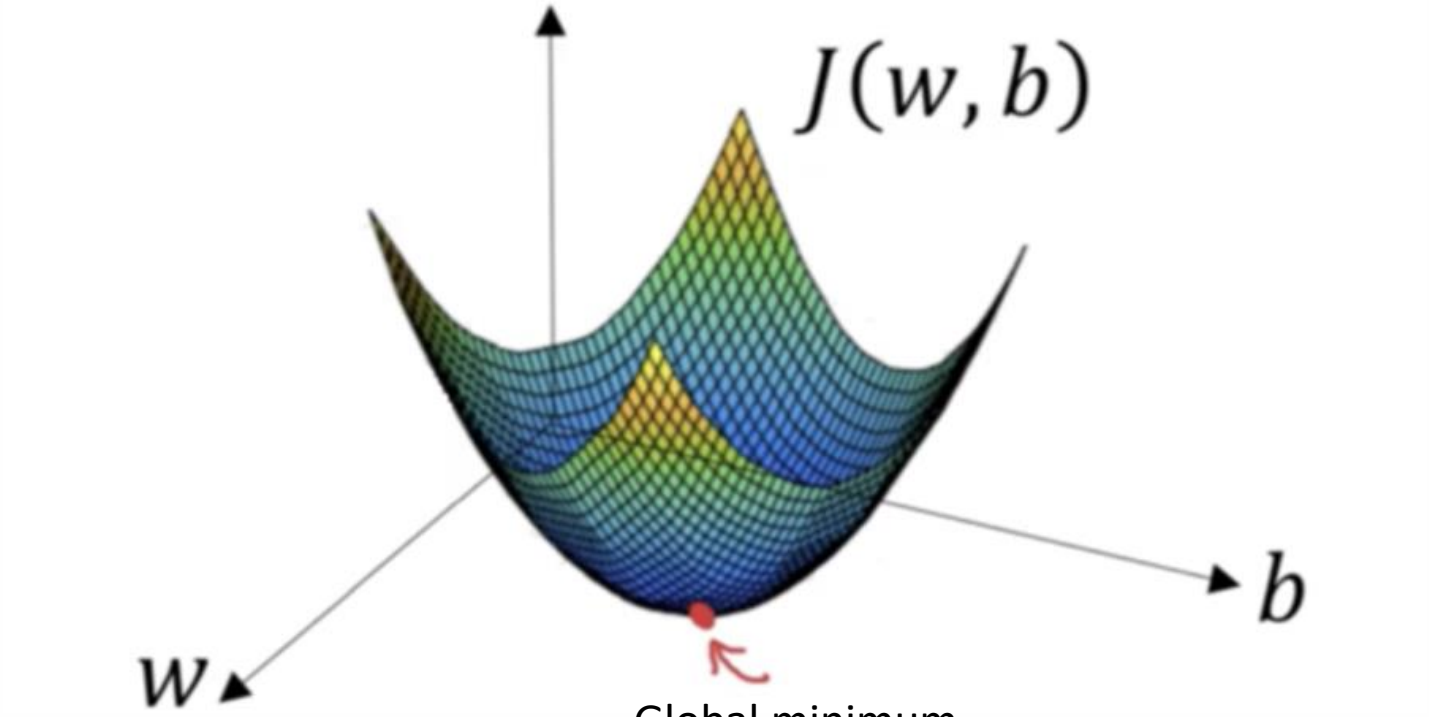
Sigmoid Function

$$J(w, b) = \frac{1}{m} \Sigma(L(y^{(i)}, \hat{y}^{(i)})) = -\frac{1}{m} \Sigma(y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)}))$$

Gradient Descent



Gradient Descent in 3 Dimension



Global minimum

Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$w_2 = w_1 - L*dw$$

$$b_2 = b_1 - L*db$$

w --> weight
b --> bias
L --> Learning Rate

dw --> Partial Derivative of cost function with respect to w

db --> Partial Derivative of cost function with respect to b

$$dw = \frac{1}{m} * (\hat{Y} - Y).X$$

$$db = \frac{1}{m} * (\hat{Y} - Y)$$

Logistic Regression

Logistic Regression model:

- Sigmoid Function

$$\hat{Y} = \frac{1}{1 + e^{-Z}} \qquad Z = w.X + b$$

- Updating weights through Gradient Descent

$$w_2 = w_1 - L*dw$$

$$b_2 = b_1 - L*db$$

- Derivatives

$$dw = \frac{1}{m} * (\hat{Y} - Y).X$$

$$db = \frac{1}{m} * (\hat{Y} - Y)$$

Building Logistic Regression model from Scratch in Python

Logistic Regression

About Logistic Regression:

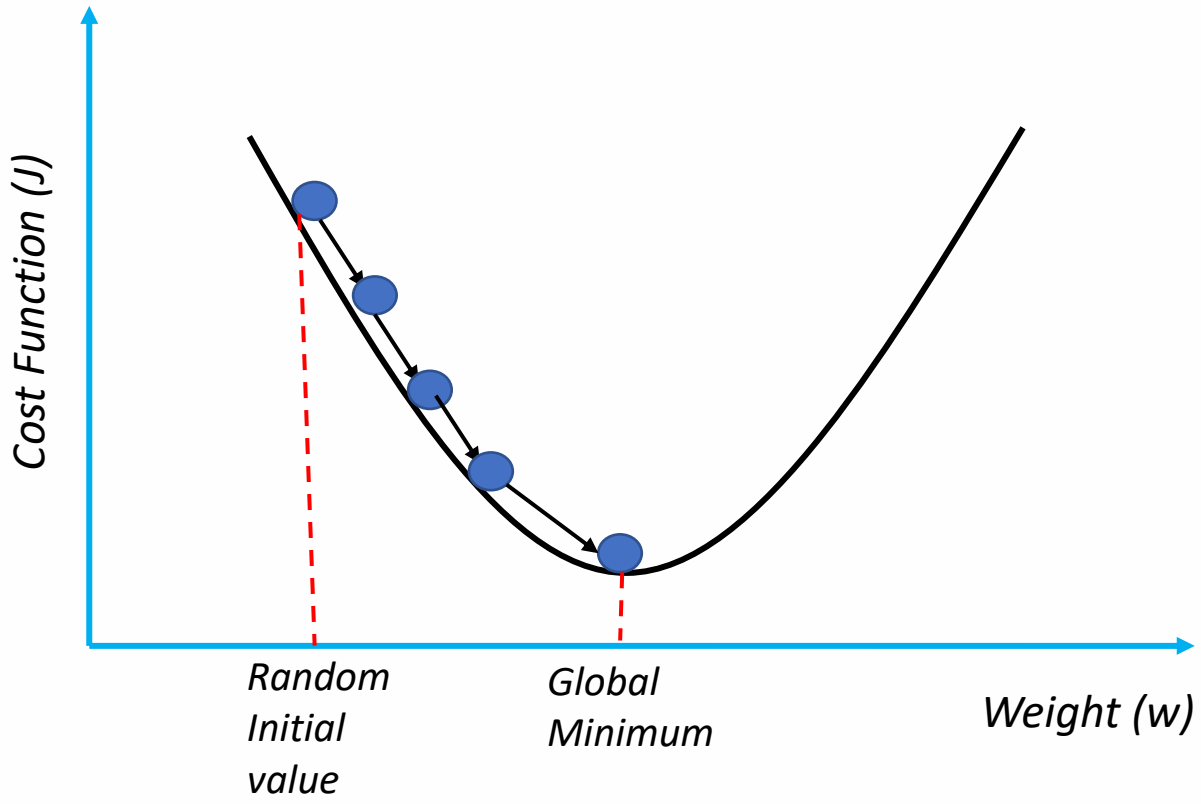
1. Supervised Learning Model
2. Classification model
3. Best for Binary Classification Problem
4. Uses Sigmoid function
5. Binary Cross Entropy Loss Function (or) Log Loss

$$\hat{Y} = \frac{1}{1 + e^{-Z}} \qquad Z = w.X + b$$

Sigmoid Function

$$J(w, b) = \frac{1}{m} \Sigma(L(y^{(i)}, \hat{y}^{(i)})) = -\frac{1}{m} \Sigma(y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)}))$$

Gradient Descent



Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$w_2 = w_1 - L*dw$$

$$b_2 = b_1 - L*db$$

w --> weight
b --> bias
L --> Learning Rate

dw --> Partial Derivative of cost function with respect to w

db --> Partial Derivative of cost function with respect to b

$$dw = \frac{1}{m} * (\hat{Y} - Y).X$$

$$db = \frac{1}{m} * (\hat{Y} - Y)$$

Logistic Regression

Logistic Regression model:

- Sigmoid Function

$$\hat{Y} = \frac{1}{1 + e^{-Z}} \qquad Z = w.X + b$$

- Updating weights through Gradient Descent

$$w_2 = w_1 - L*dw$$

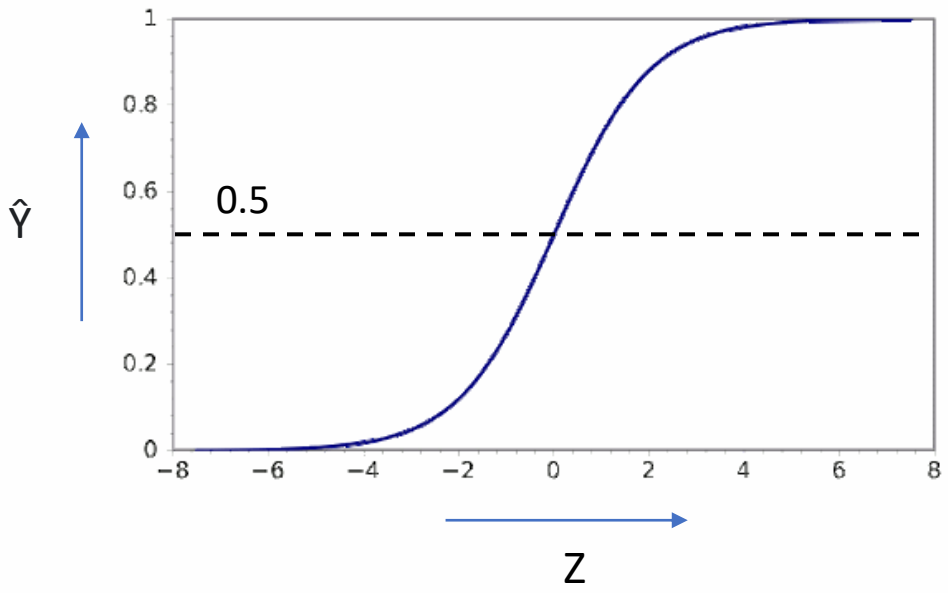
$$b_2 = b_1 - L*db$$

- Derivatives

$$dw = \frac{1}{m} * (\hat{Y} - Y).X$$

$$db = \frac{1}{m} * (\hat{Y} - Y)$$

Logistic Regression



$$\hat{Y} = \frac{1}{1 + e^{-Z}}$$

$$Z = w.X + b$$

Sigmoid Function

Ŷ - Probability that (y = 1)

Ŷ = P(Y=1 | X)

X - input features

w – weights
(number of weights is equal to the number of input features in a dataset)

b - bias

Multiplying 2 Matrices

Rule : The number of columns in the First matrix should be equal to the number of rows in the Second Matrix

The resultant matrix will have the same number of rows as the first matrix & the same number of columns as the Second Matrix

$$\begin{bmatrix} 2 & 3 \\ 10 & 5 \end{bmatrix} \quad x \quad \begin{bmatrix} 10 & 5 \\ 20 & 4 \end{bmatrix}$$

2 x 2 2 x 2

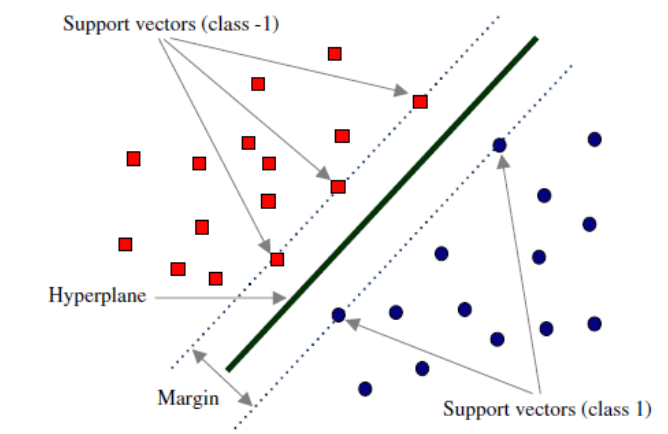
Can be multiplied.
Resultant matrix will have the shape 2 x 2

$$\begin{bmatrix} 2 & 1 \\ 4 & 2 \\ 6 & 3 \end{bmatrix} \quad x \quad \begin{bmatrix} 5 & 2 \\ 3 & 6 \\ 2 & 5 \end{bmatrix}$$

3 x 2 3 x 2

Cannot be multiplied.

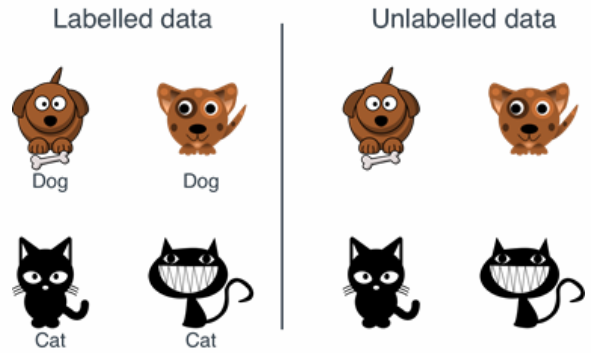
Support Vector Machine (SVM) Classifier - intuition

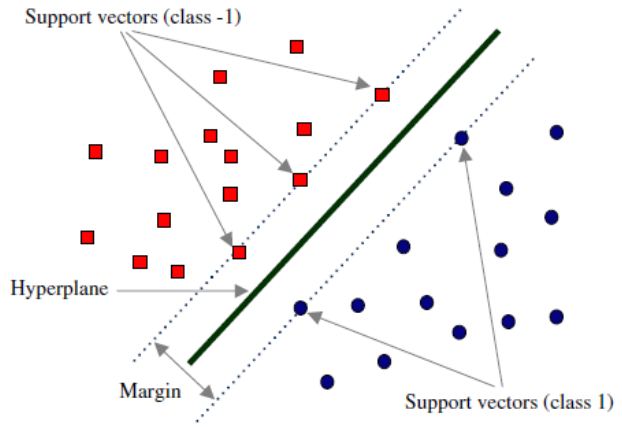


Support Vector Machine

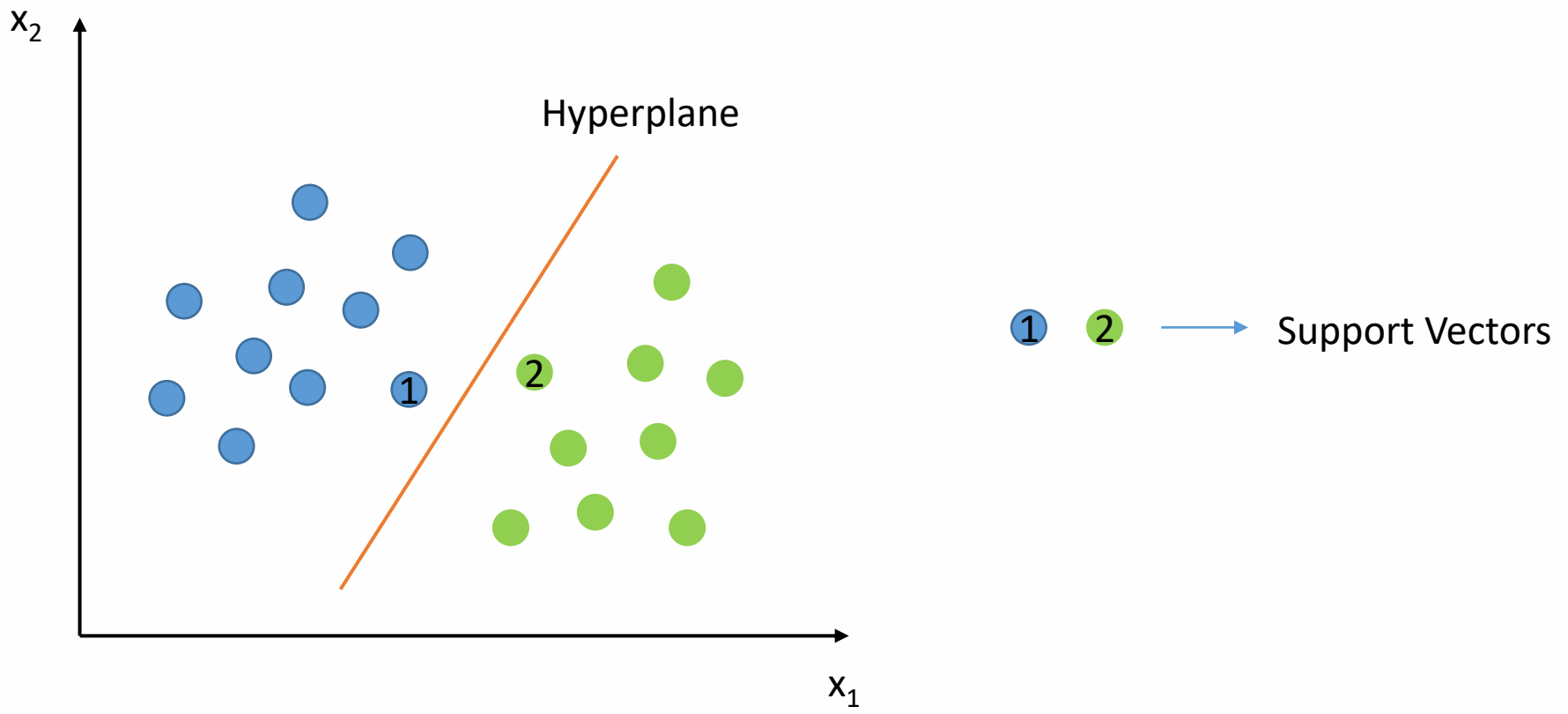
About Support Vector Machine model:

1. Supervised Learning Model
2. Both Classification & Regression
3. Hyperplane
4. Support Vectors

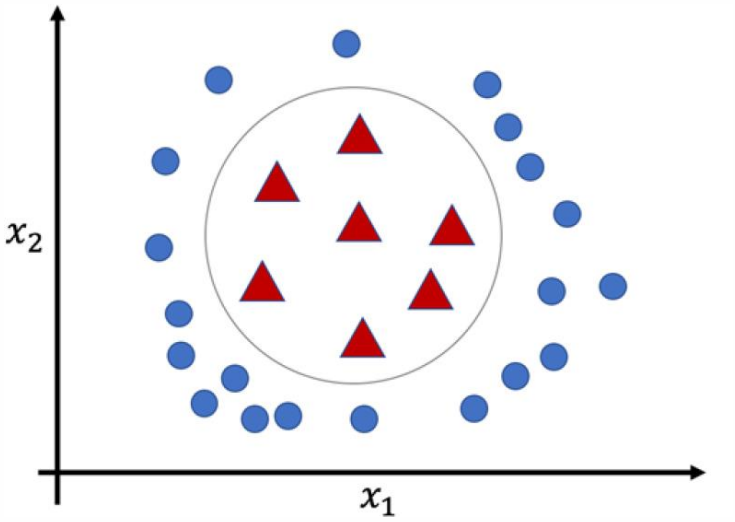




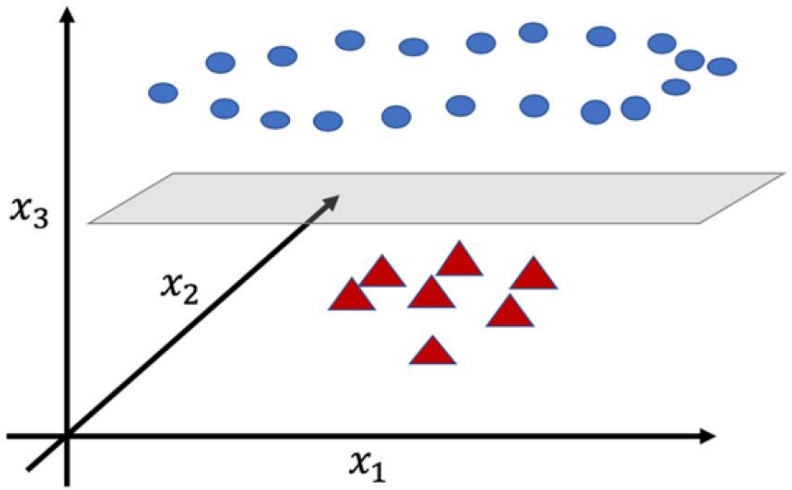
Support Vector Machine Classifier



Support Vector Machine Classifier



SVM in 2 dimensions



SVM in 3 dimensions

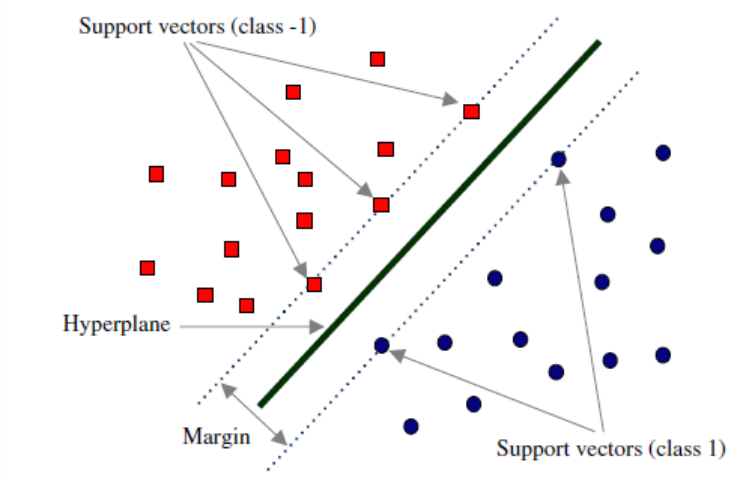
Support Vector Machine Classifier

Hyperplane:

Hyperplane is a line (in 2d space) or a plane that separate the data points into 2 classes.

Support Vectors:

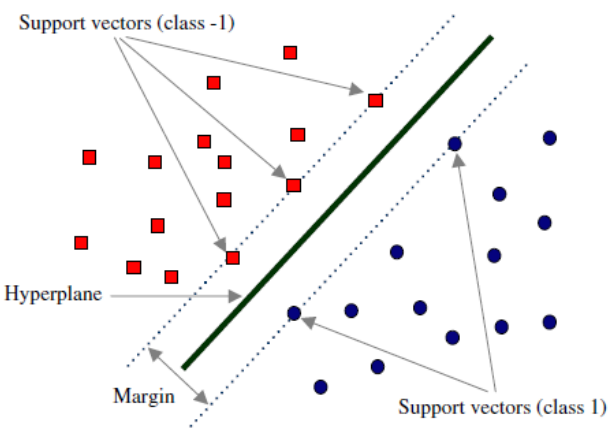
Support Vectors are the data points which lie nearest to the hyperplane. If theses data points changes, the position of the hyperplane changes.



Support Vector Machine Classifier

Advantages:

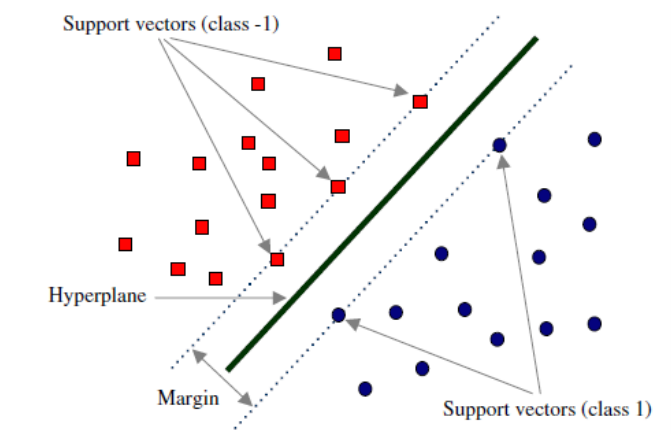
1. Works well with smaller datasets
2. Works efficiently when there is a clear margin of separation
3. Works well with high dimensional data



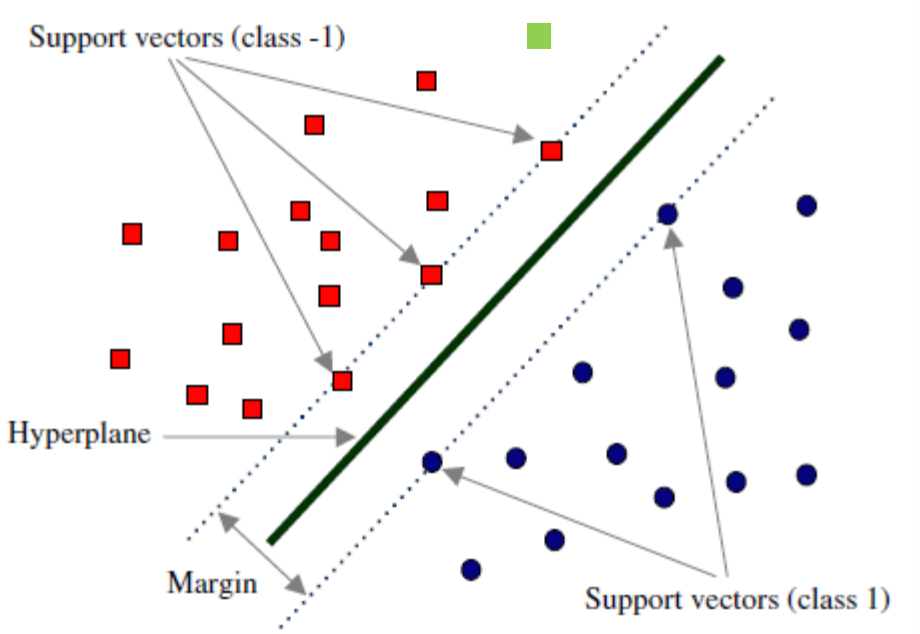
Disadvantages:

1. Not suitable for large datasets as the training time is higher
2. Not suitable for noisier datasets with overlapping classes

Math behind Support Vector Machine (SVM) Classifier

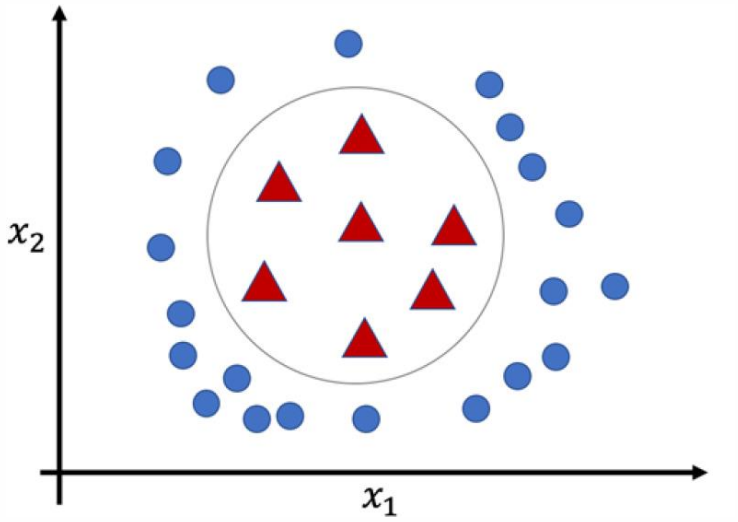


Support Vector Machine Classifier



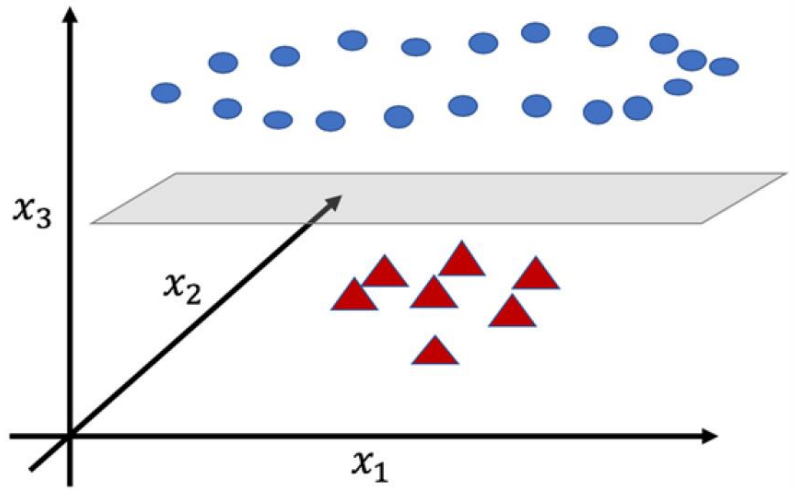
- Hyperplane
- Support Vectors
- Margin
- Linearly separable data

Support Vector Machine Classifier



SVM in 2 dimensions

Kernel



SVM in 3 dimensions

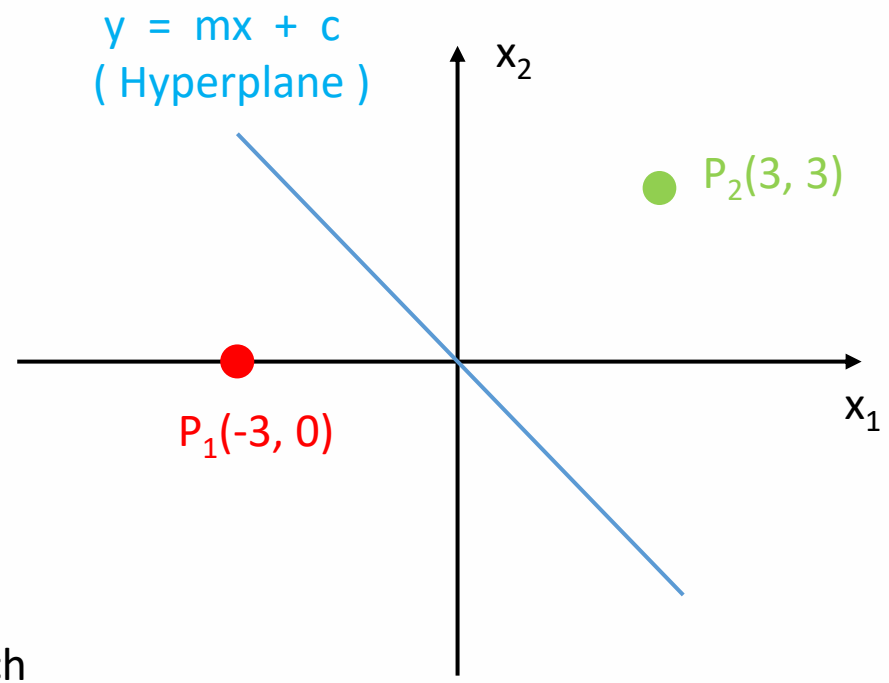
Support Vector Machine Classifier

$P_1(-3, 0)$

$w^T x = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} -3 & 0 \end{bmatrix}$

$w^T x = 3$

(Positive)



$P_2(3, 3)$

$w^T x = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} 3 & 3 \end{bmatrix}$

$w^T x = -3$

(Negative)

Inference: For all the points which lie in the left side of the hyperplane, $w^T x$ value will be Positive

Let slope, m = - 1

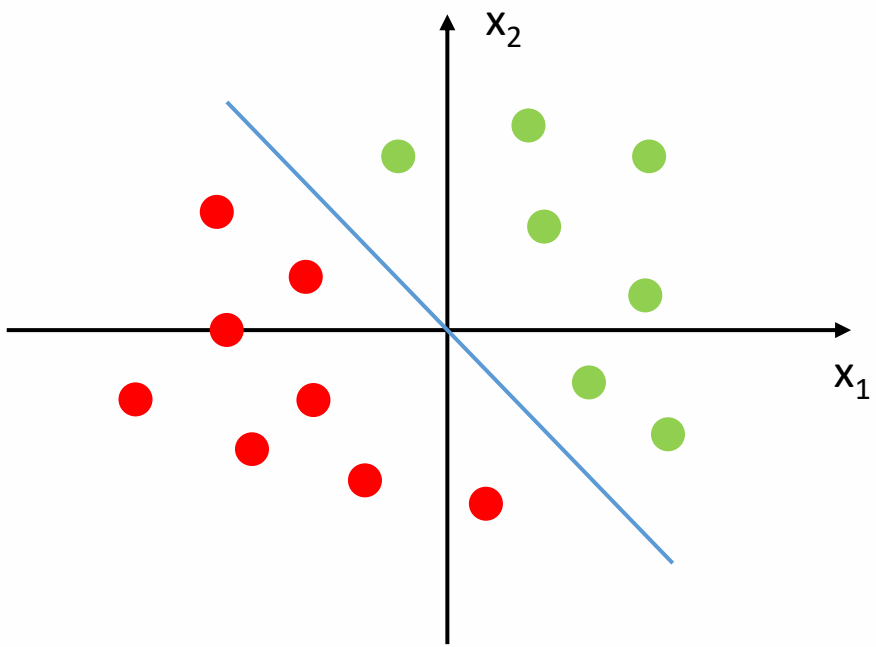
Intercept, c = 0

w --> parameters of the line

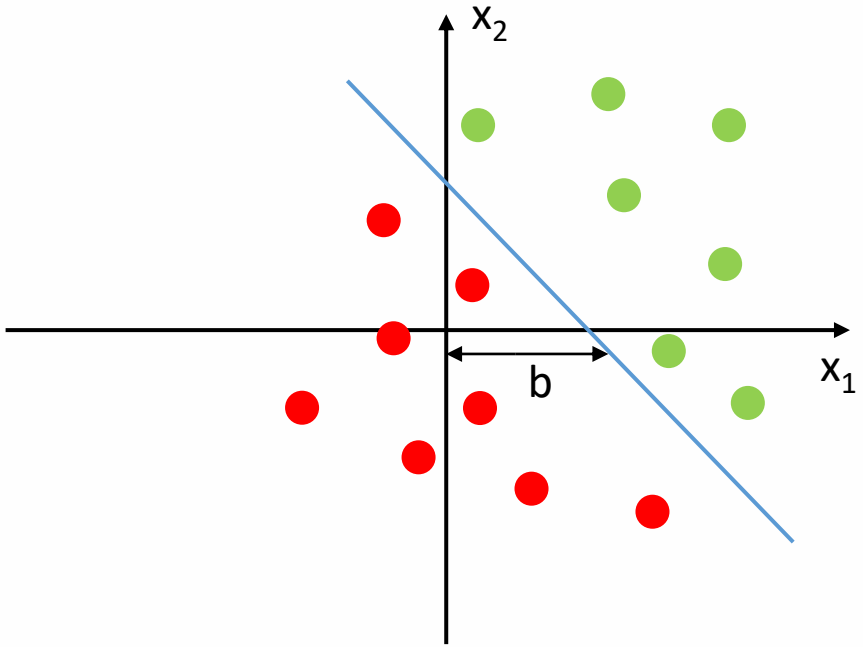
(m, c) = (-1, 0)

Inference: For all the points which lie in the right side of the hyperplane, $w^T x$ value will be Negative

Support Vector Machine Classifier



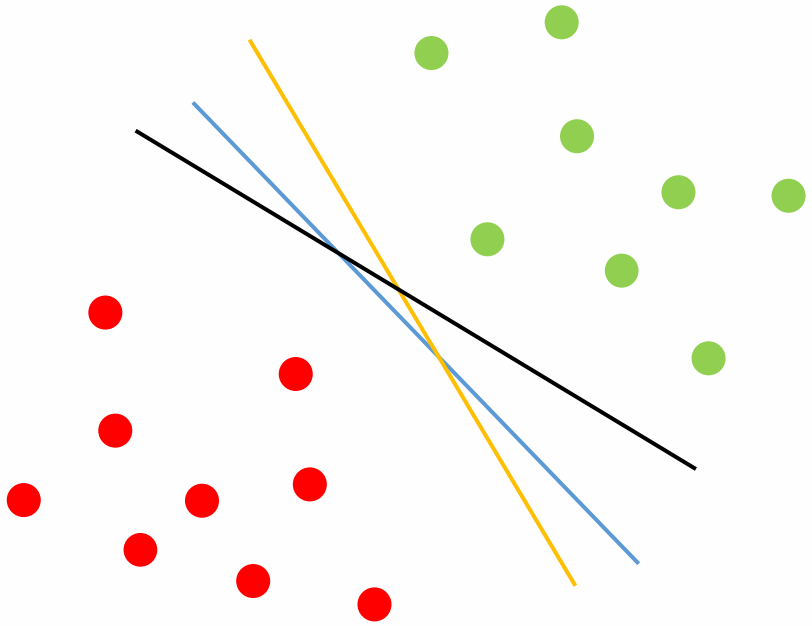
$w^T x = \text{Label}$



$w^T x + b = \text{Label}$

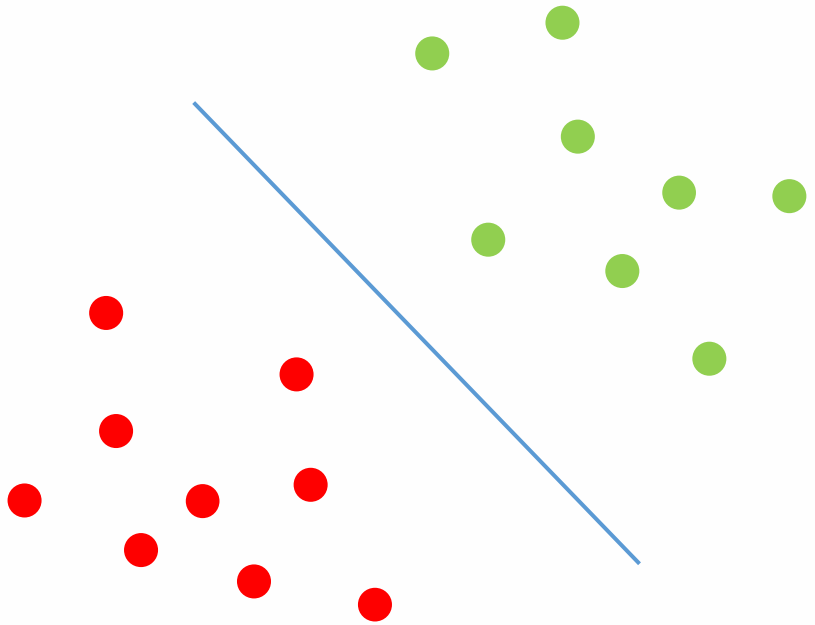
Support Vector Machine Classifier

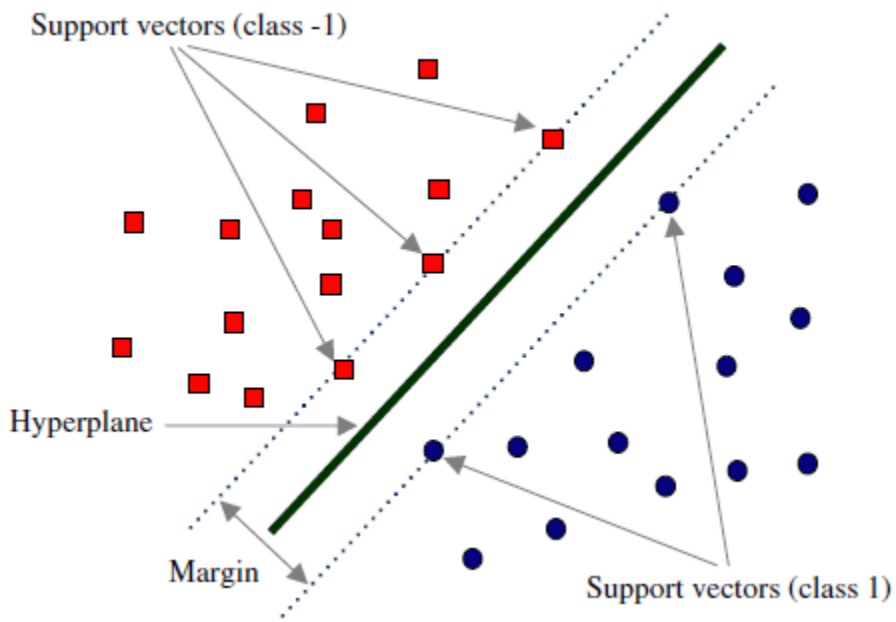
Which is the best Hyperplane?



Support Vector Machine Classifier

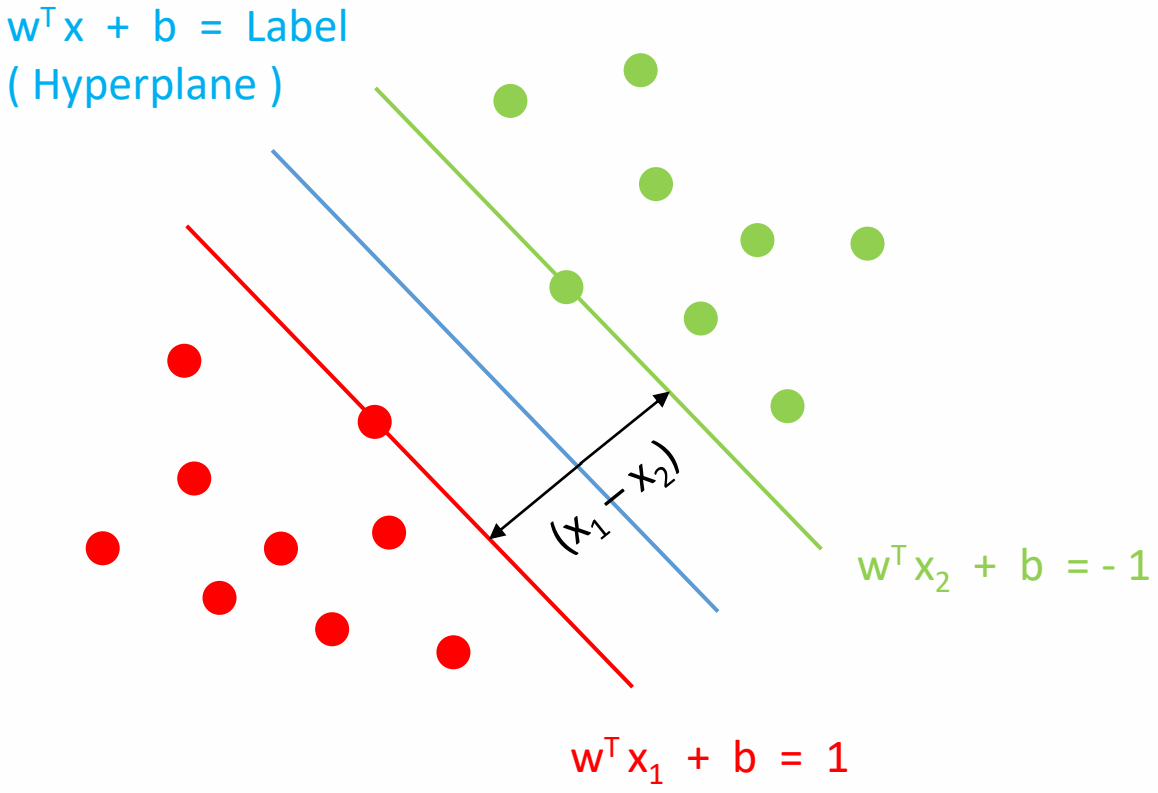
Which is the best Hyperplane?





Support Vector Machine Classifier

Optimization for Maximum margin:



$$w^T x_1 + b = 1$$

$$(-)\quad w^T x_2 + b = -1$$

$$w^T (x_1 - x_2) = 2$$

Divide by $||\ w\ ||$

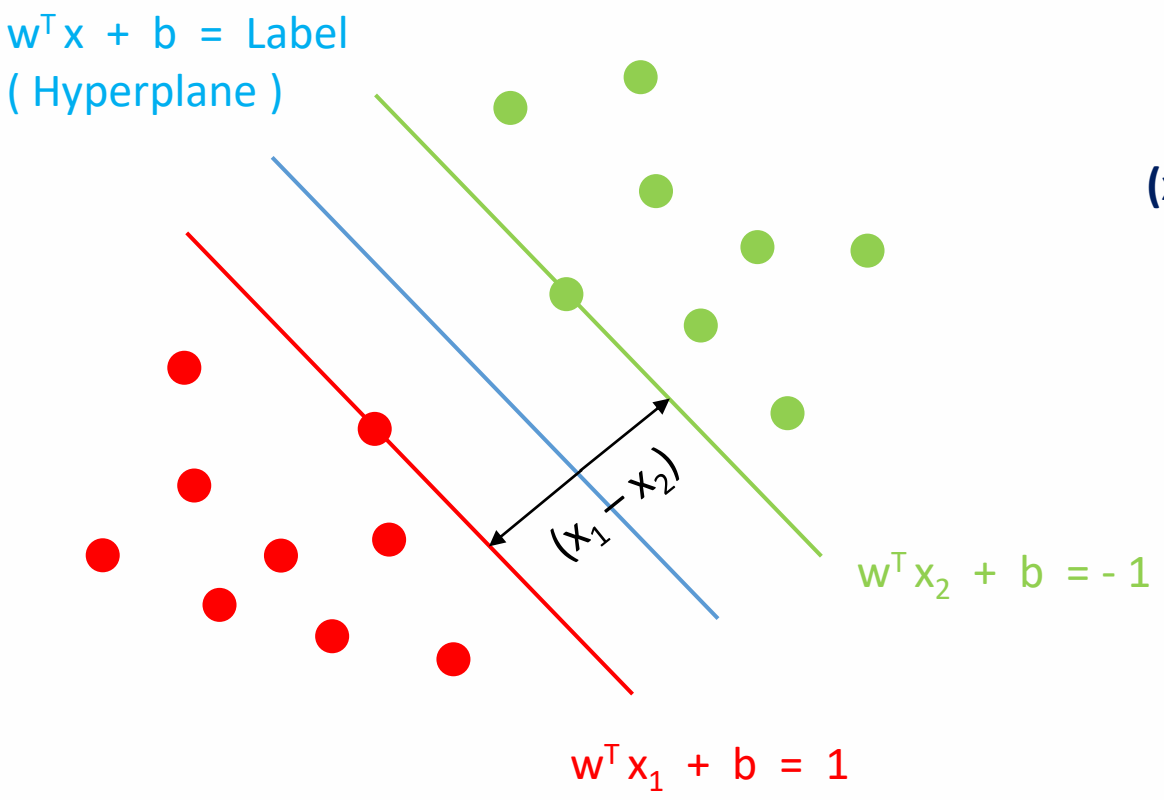
(magnitude of the vector)

$$\frac{w^T}{||\ w\ ||}(x_1 - x_2) = \frac{2}{||\ w\ ||}$$

$$\mathbf{(x_1 - x_2) = \frac{2}{||\ w\ ||}}$$ (margin)

Support Vector Machine Classifier

Optimization for Maximum margin:



$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases} \quad \text{(Label)}$$

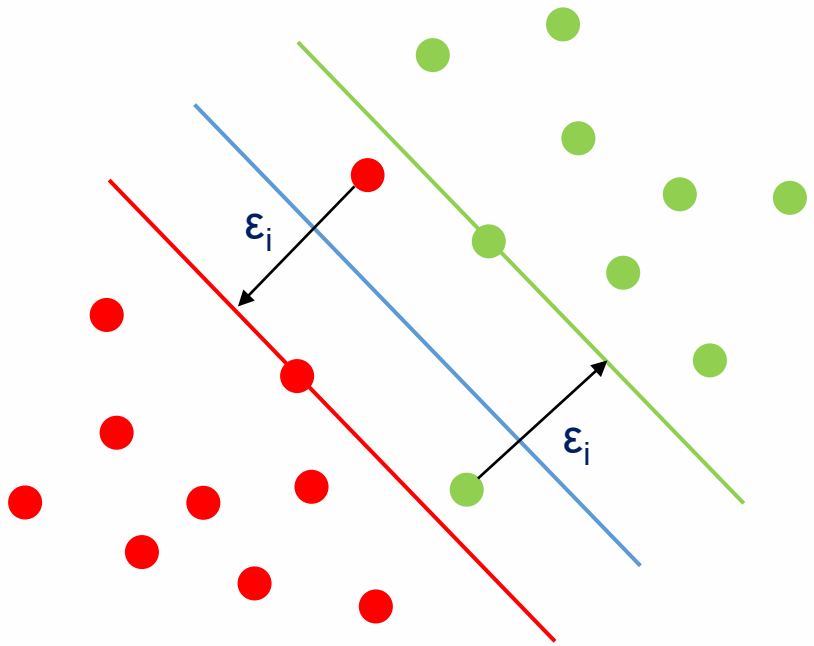
$$(x_1 - x_2) = \frac{2}{||w||} \quad \text{(margin)}$$

$$\max \left(\frac{2}{||w||} \right) \quad \text{Such that,}$$

$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

Support Vector Machine Classifier

Maximum margin without overfitting:



$$\max \left(\frac{2}{||w||} \right) \quad \text{Such that,}$$

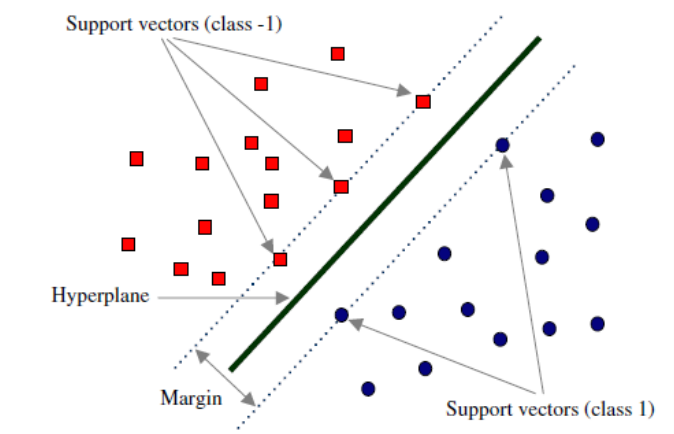
$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

$$\min \left(\frac{||w||}{2} \right) + c * \Sigma \varepsilon_i$$

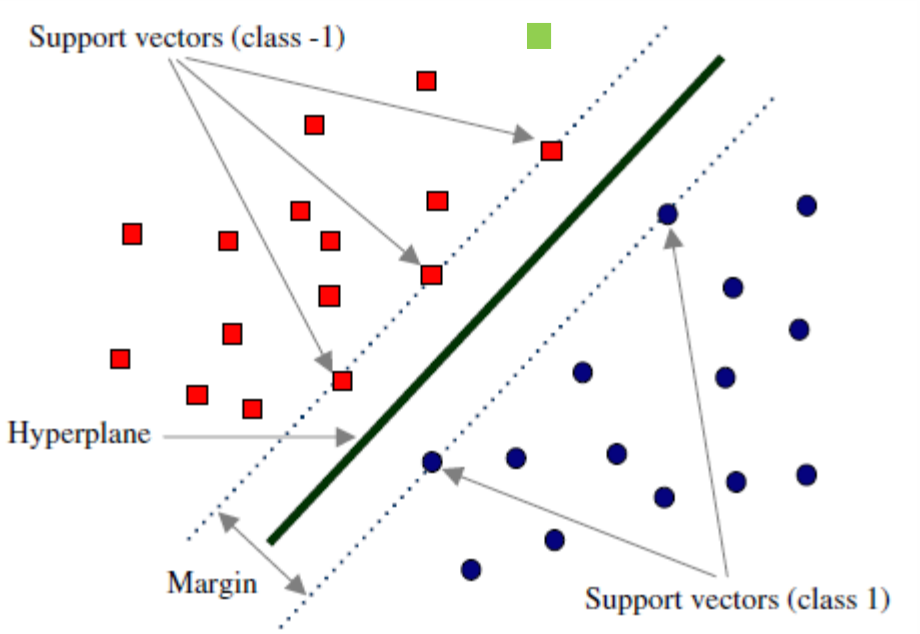
c --> Number of errors

ε_i --> Error magnitude

Support Vector Machine (SVM) - Kernels



Support Vector Machine Classifier

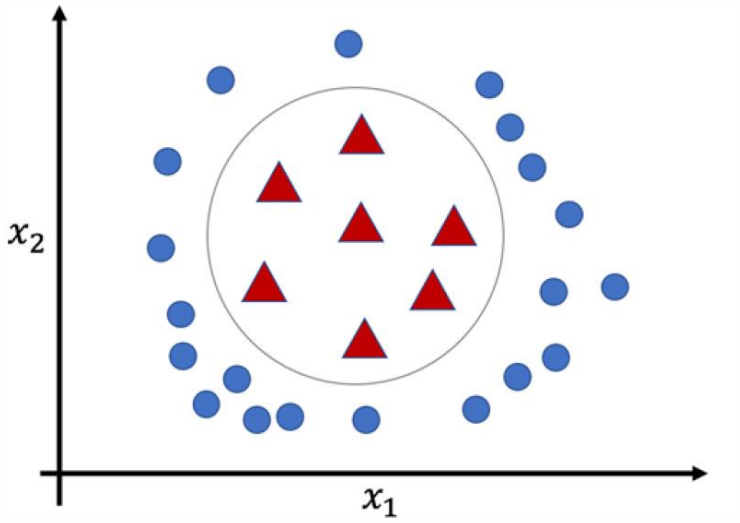


- Hyperplane
- Support Vectors
- Margin
- Linearly separable data

SVM Kernel

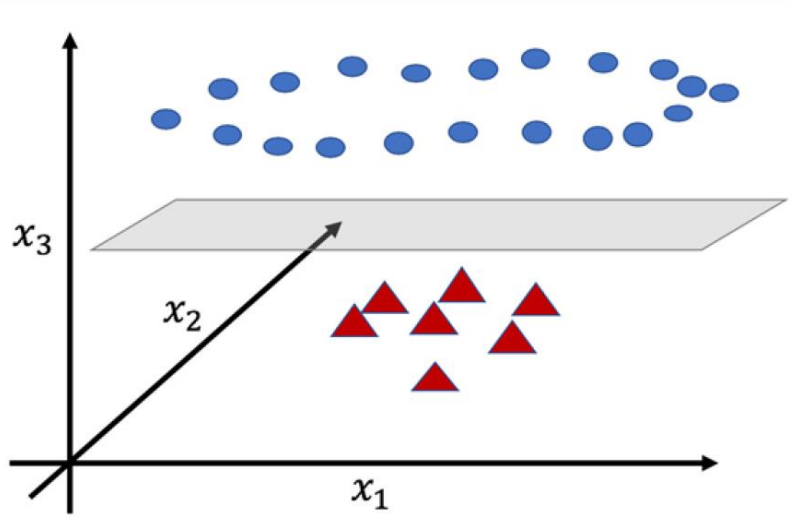
SVM Kernel :

Kernel Function generally transforms the training set of data so that a non-linear decision surface can be transformed to a linear equation in a higher number of dimension spaces.
It returns the inner product between two points in a standard feature dimension.



SVM in 2 dimensions

Kernel

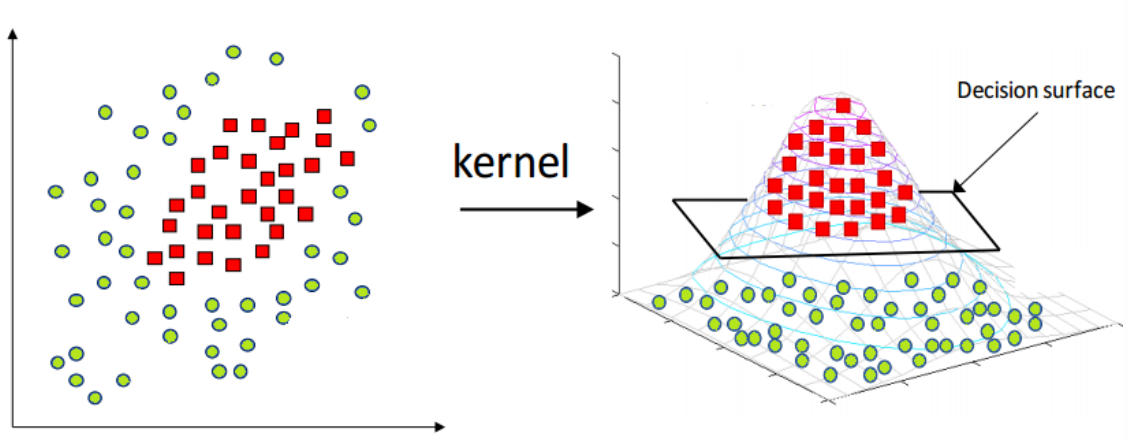


SVM in 3 dimensions

SVM Kernels

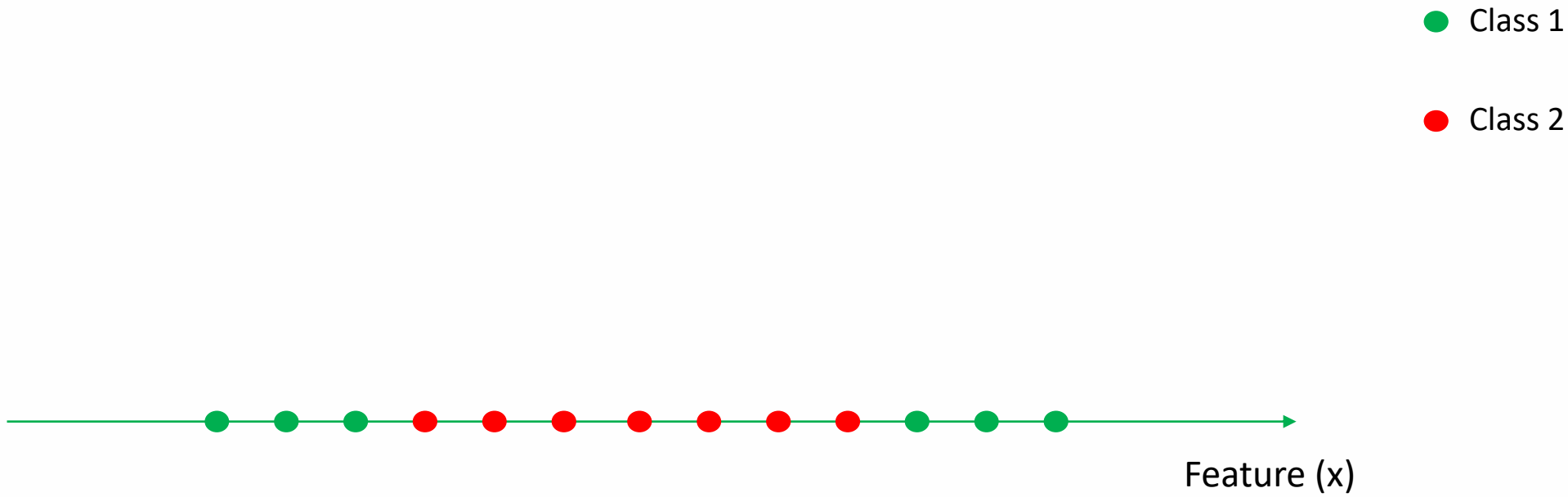
Types of SVM Kernels :

1. Linear
2. Polynomial
3. Radial Basis Function (rbf)
4. Sigmoid



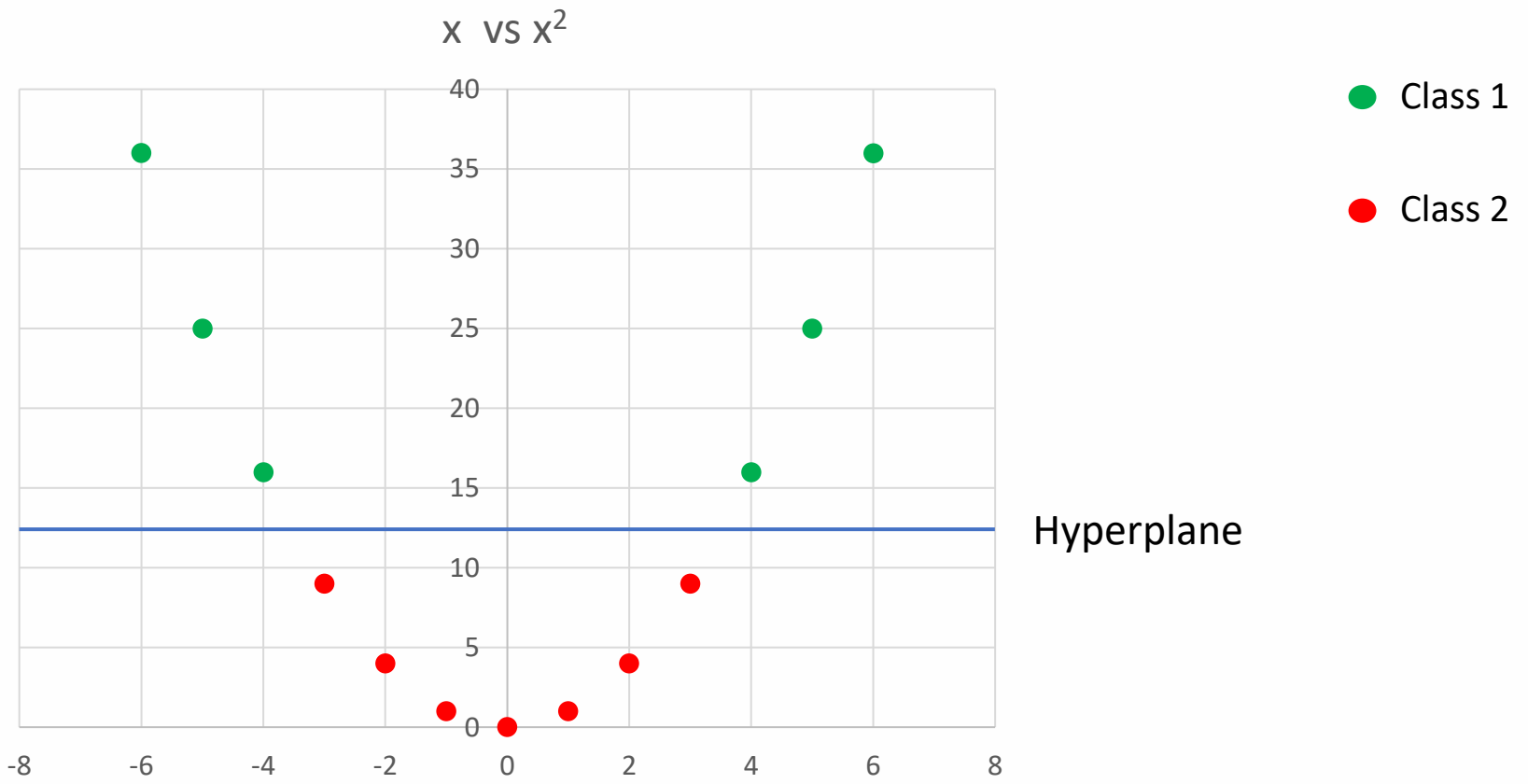
SVM Kernels

Feature (x)	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6



SVM Kernels

Feature (x)	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
x^2	36	25	16	9	4	1	0	1	4	9	16	25	36



Types of SVM Kernels

1. Linear Kernel :

$$K(x_1, x_2) = x_1^T x_2$$

2. Polynomial Kernel:

$$K(x_1, x_2) = (x_1^T x_2 + r)^d$$

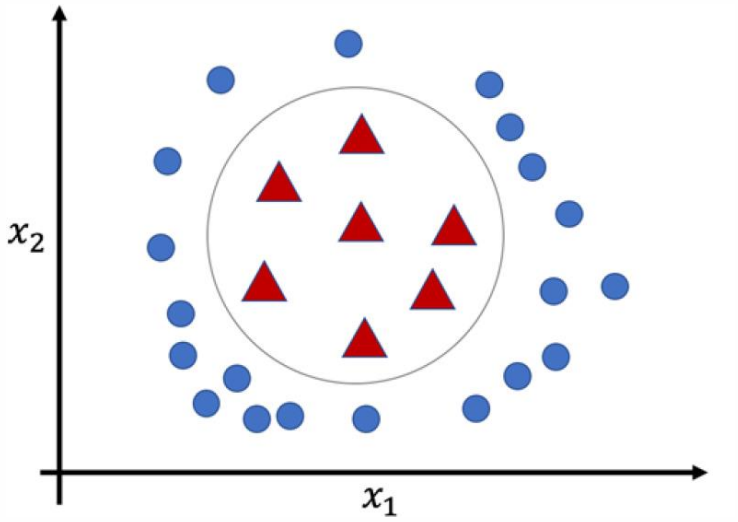
3. Radial Basis Function (rbf) Kernel :

$$K(x_1, x_2) = \exp(-\gamma . ||x_1 - x_2||^2)$$

4. Sigmoid Kernel :

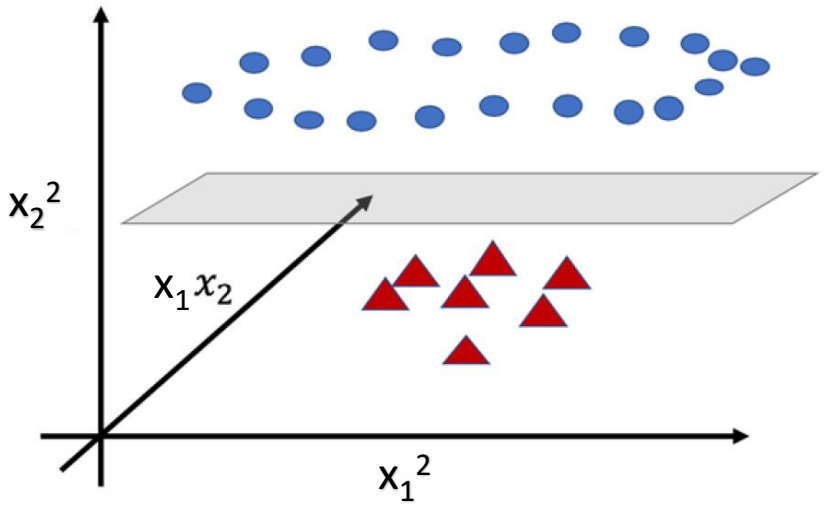
$$K(x_1, x_2) = \tanh(\gamma . x_1^T x_2 + r)$$

Support Vector Machine Classifier



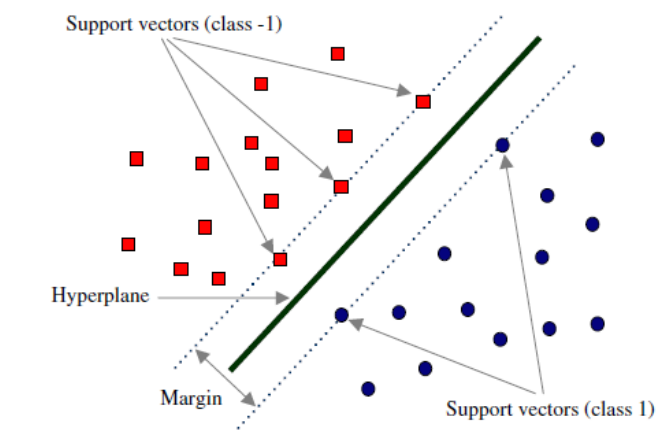
SVM in 2 dimensions

Kernel

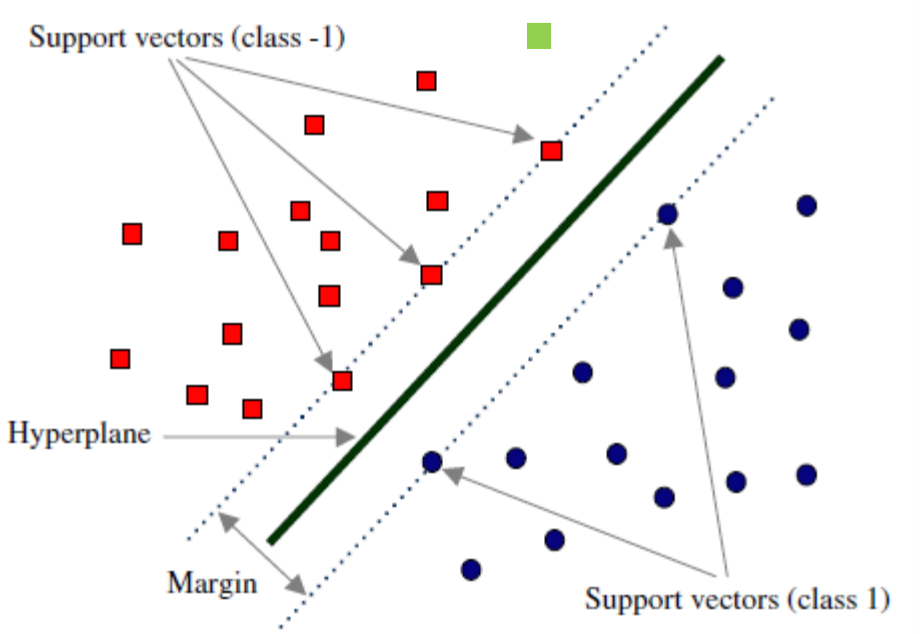


SVM in 3 dimensions

Loss Function for Support Vector Machine Classifier

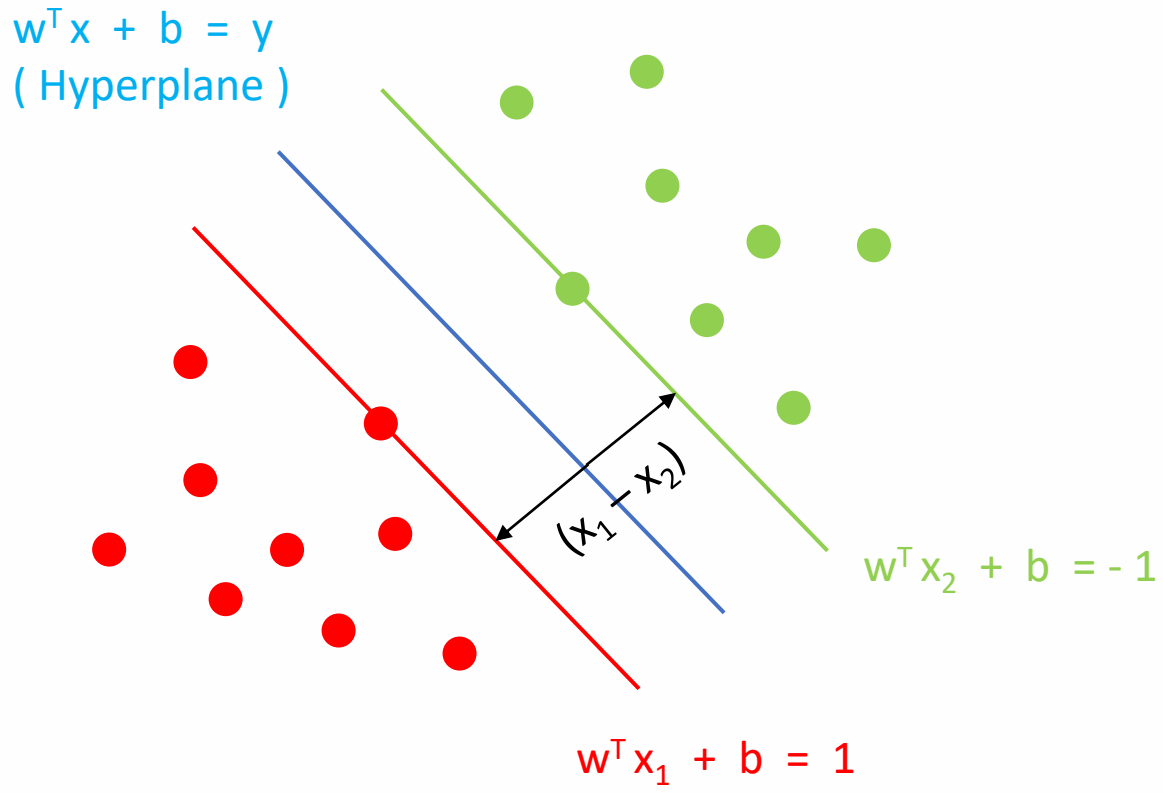


Support Vector Machine Classifier



- Hyperplane
- Support Vectors
- Margin
- Linearly separable data

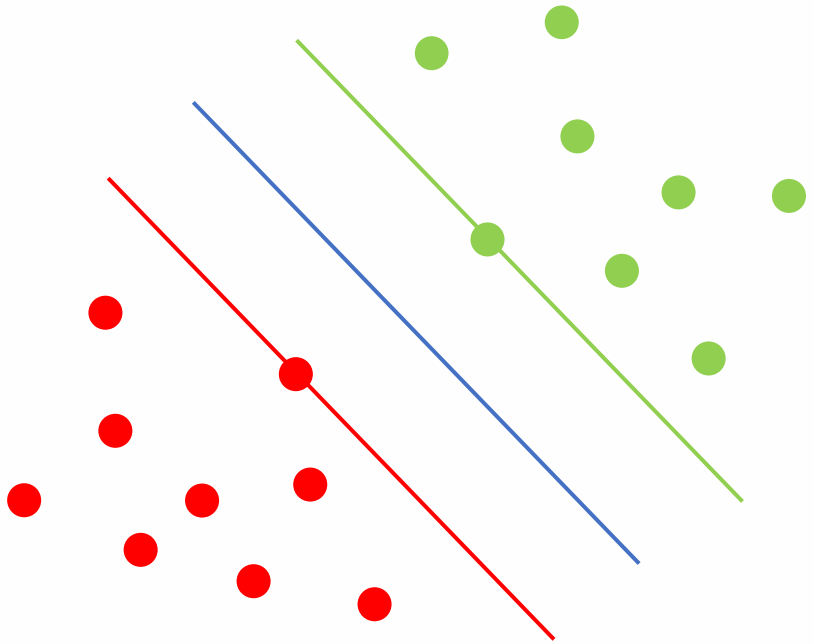
Support Vector Machine Classifier



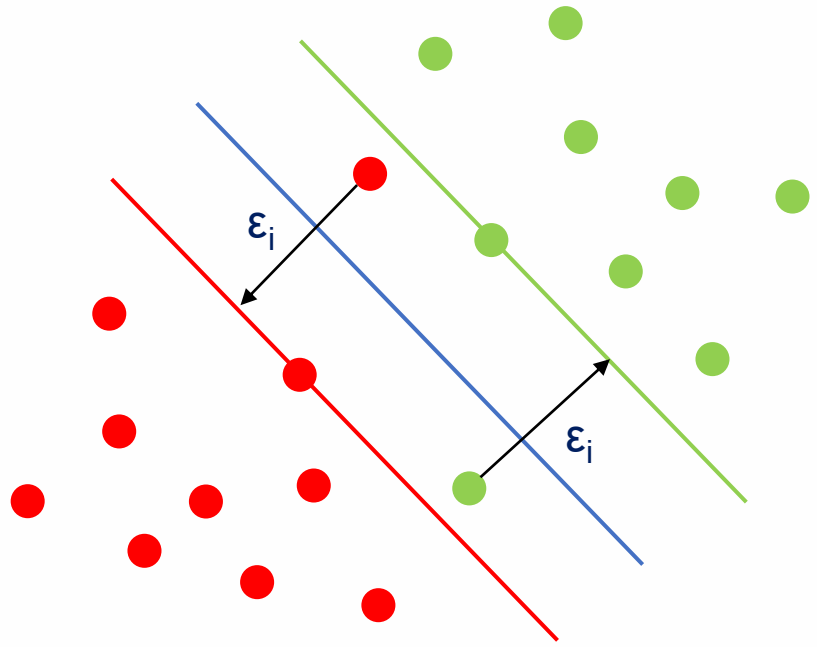
$$\max \left(\frac{2}{||w||} \right) \quad \text{(margin)}$$

$$\hat{y}_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

Support Vector Machine Classifier



Hard Margin



Soft Margin

Loss Function

Loss function measures how far an estimated value is from its true value.

It is helpful to determine which model performs better & which parameters are better.

$$\textbf{\textit{Loss}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

For Support Vector Machine Classifier "Hinge Loss" is used as the Loss Function.

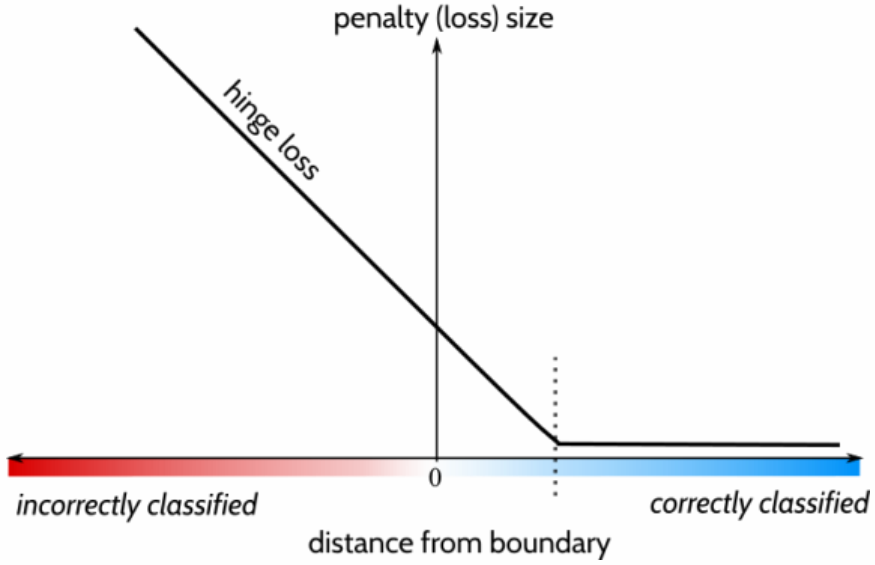
Hinge Loss

Hinge Loss is one of the types of Loss Function, mainly used for maximum margin classification models.

Hinge Loss incorporates a margin or distance from the classification boundary into the loss calculation. Even if new observations are classified correctly, they can incur a penalty if the margin from the decision boundary is not large enough.

$$L = \max(0, 1 - y_i (w^T x_i + b))$$

0 - for correct classification
1 - for wrong classification



Hinge Loss

Misclassification :

$y_i = 1 \quad \hat{y}_i = -1$

$L = (1 - (1)(-1)$

$L = (1 + 1)$

$L = 2$ (High loss Value)

$y_i = -1 \quad \hat{y}_i = 1$

$L = (1 - (-1)(1)$

$L = (1 + 1)$

$L = 2$ (High loss Value)

$L = \max (0, 1 - y_i (w^T x_i + b))$

0 - for correct classification
1 - for wrong classification

Correct classification :

$y_i = 1 \quad \hat{y}_i = 1$

$L = (0 - (1)(1)$

$L = (0 - 1)$

$L = -1$ (Low loss Value)

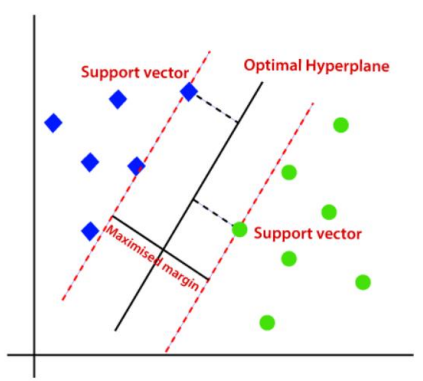
$y_i = -1 \quad \hat{y}_i = -1$

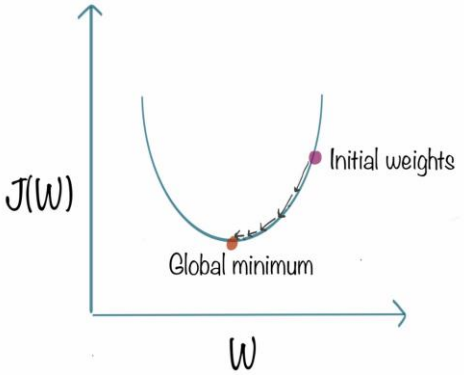
$L = (0 - (-1)(-1)$

$L = (0 - 1)$

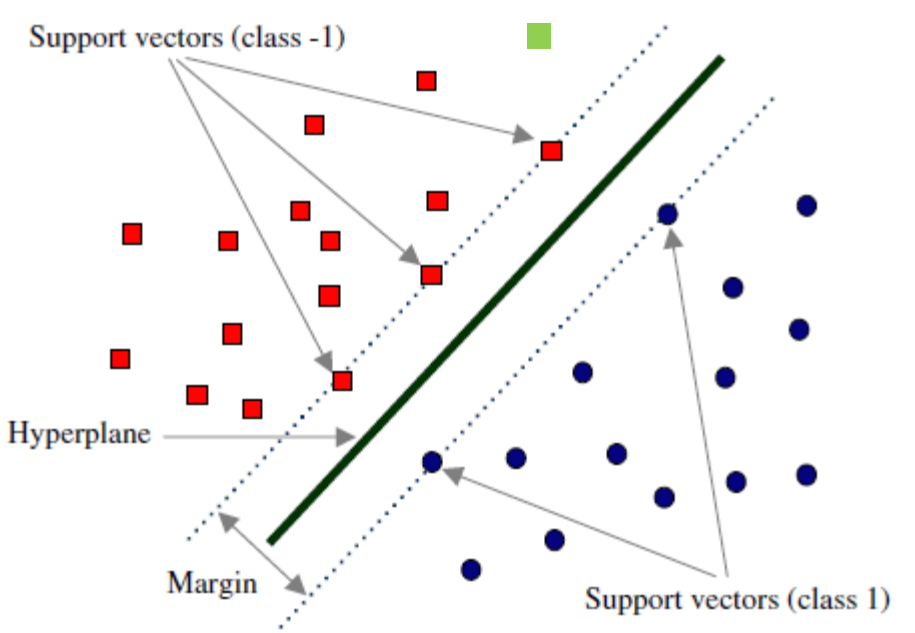
$L = -1$ (Low loss Value)

Gradient Descent for Support Vector Machine Classifier



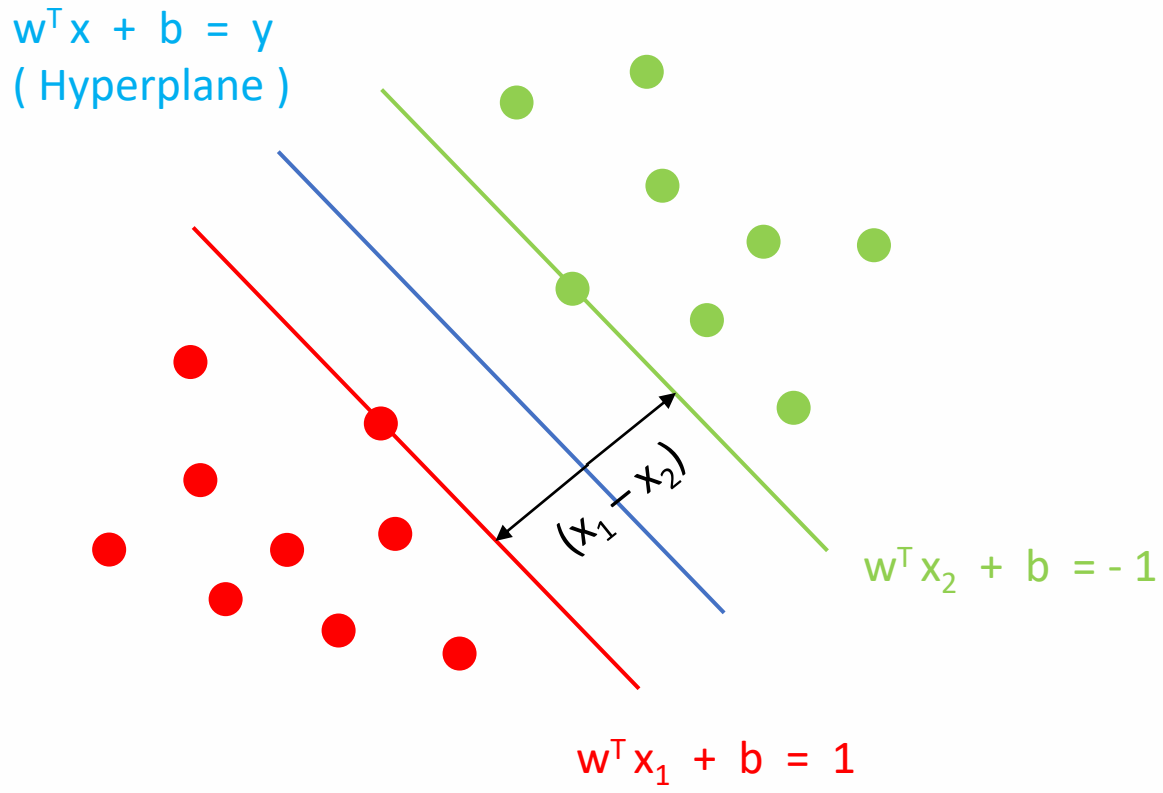


Support Vector Machine Classifier



- Hyperplane
- Support Vectors
- Margin

Support Vector Machine Classifier



$$\max \left(\frac{2}{||w||} \right) \quad \text{(margin)}$$

$$\hat{y}_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

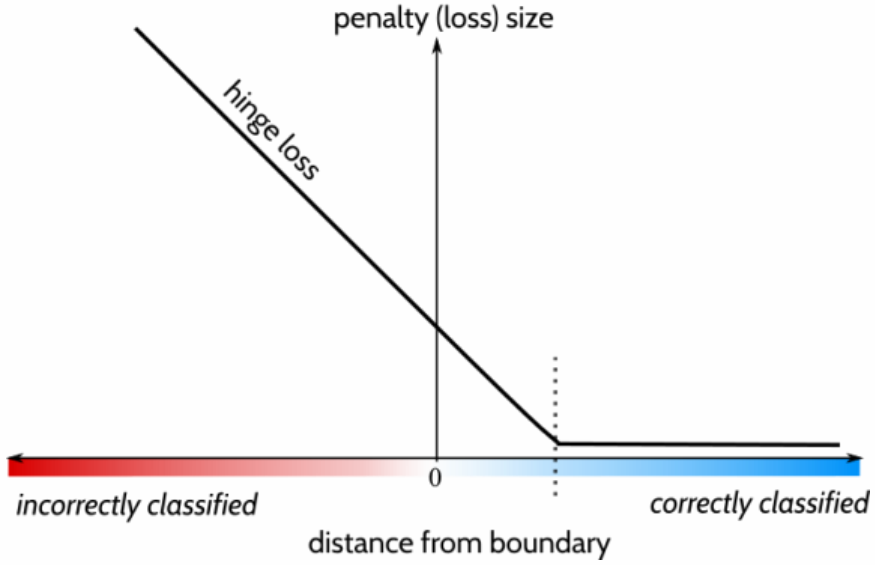
Hinge Loss

Hinge Loss is one of the types of Loss Function, mainly used for maximum margin classification models.

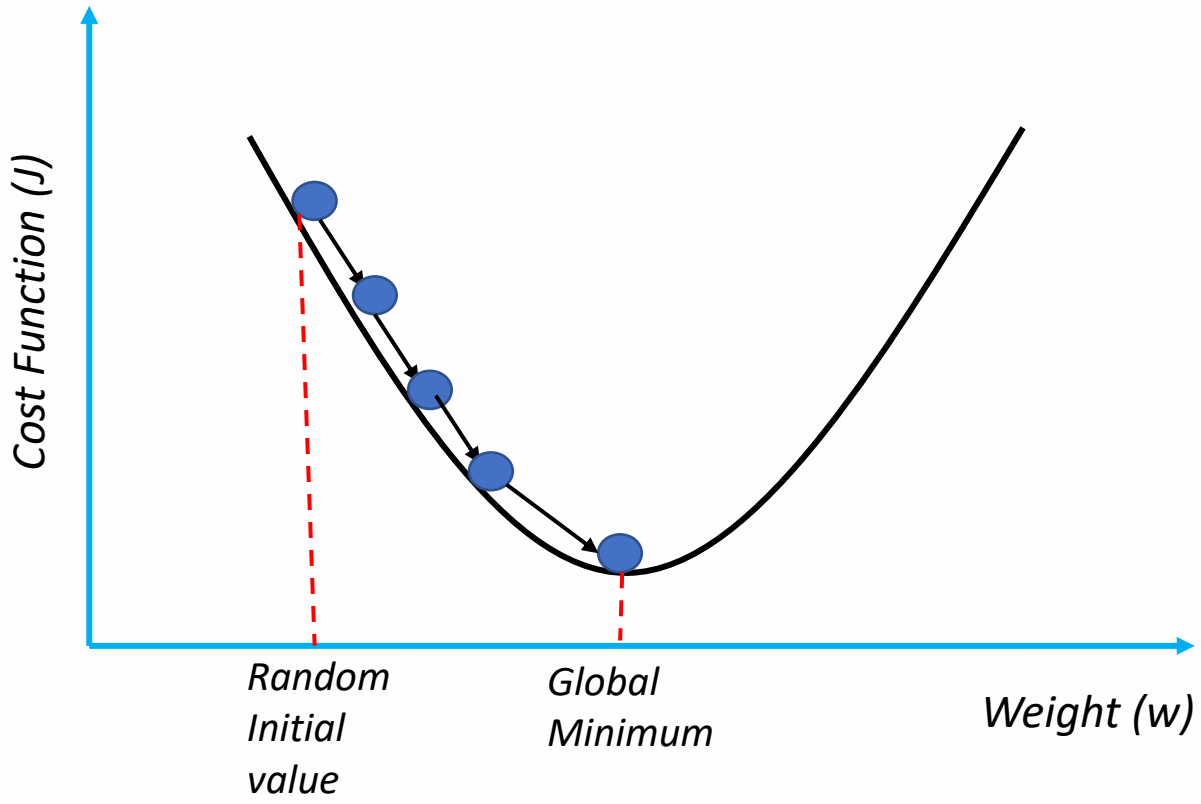
Hinge Loss incorporates a margin or distance from the classification boundary into the loss calculation. Even if new observations are classified correctly, they can incur a penalty if the margin from the decision boundary is not large enough.

$$L = \max(0, 1 - y_i (w^T x_i + b))$$

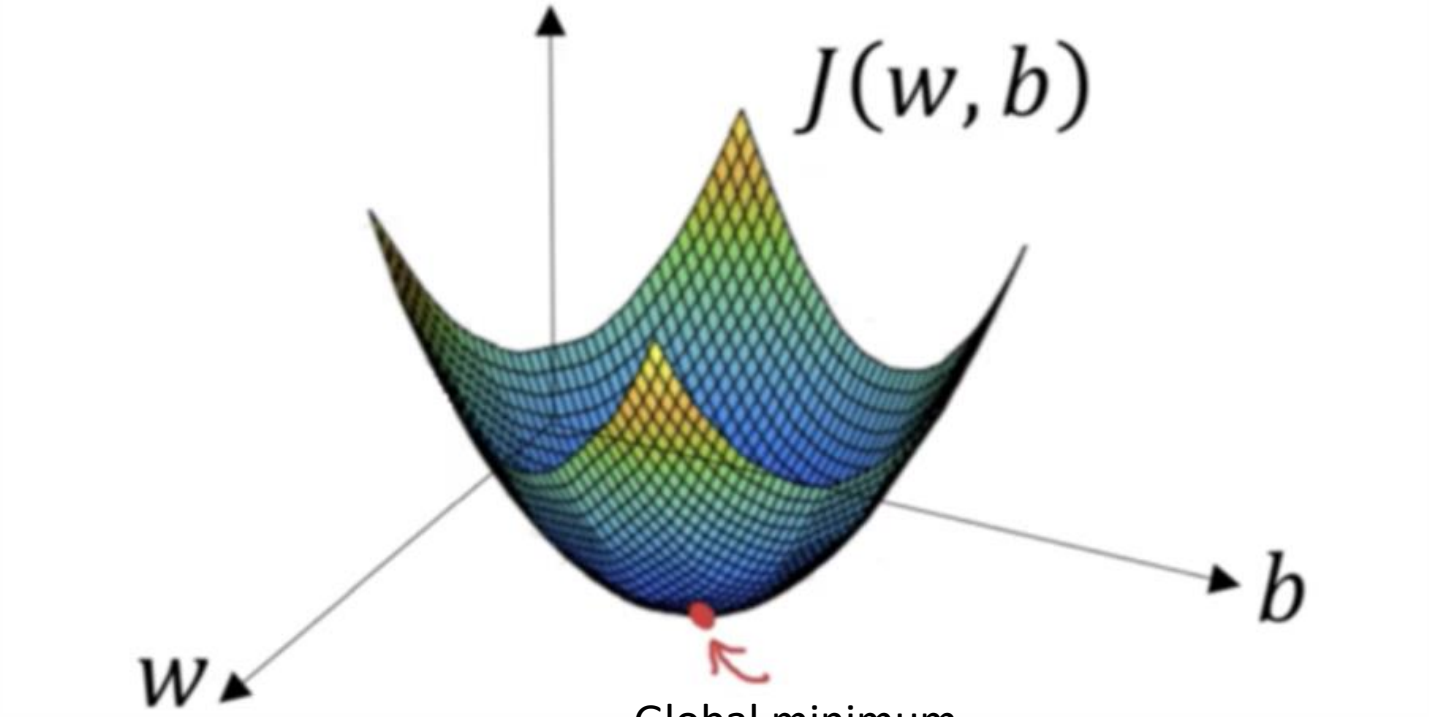
0 - for correct classification
1 - for wrong classification



Gradient Descent



Gradient Descent in 3 Dimension



Global minimum

Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

w --> weight
b --> bias
L --> Learning Rate

$\frac{dJ}{dw}$ --> Partial Derivative of cost function with respect to w

$\frac{dJ}{db}$ --> Partial Derivative of cost function with respect to b

Gradients for SVM Classifier

if (y_i . (w.x + b) ≥ 1) :

$$\frac{dJ}{dw} = 2\lambda w$$

$$\frac{dJ}{db} = 0$$

else (y_i . (w.x + b) < 1) :

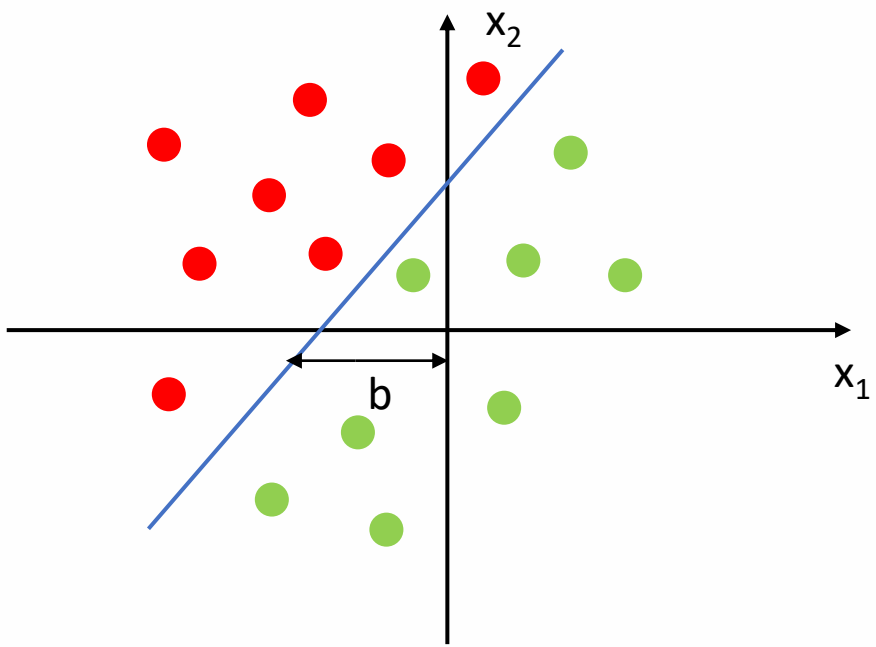
$$\frac{dJ}{dw} = 2\lambda w - y_i \cdot x_i$$

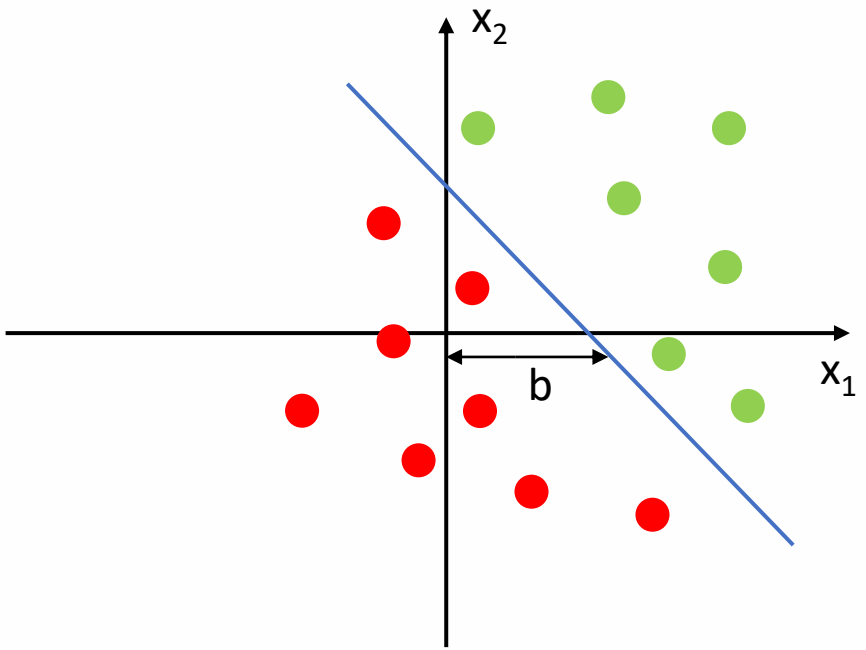
$$\frac{dJ}{db} = yi$$

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

Support Vector Machine Classifier





Gradients for SVM Classifier

if (y_i . ($w.x_i$ - b) ≥ 1) :

$$\frac{dJ}{dw} = 2\lambda w$$

$$\frac{dJ}{db} = 0$$

else (y_i . ($w.x_i$ - b) < 1) :

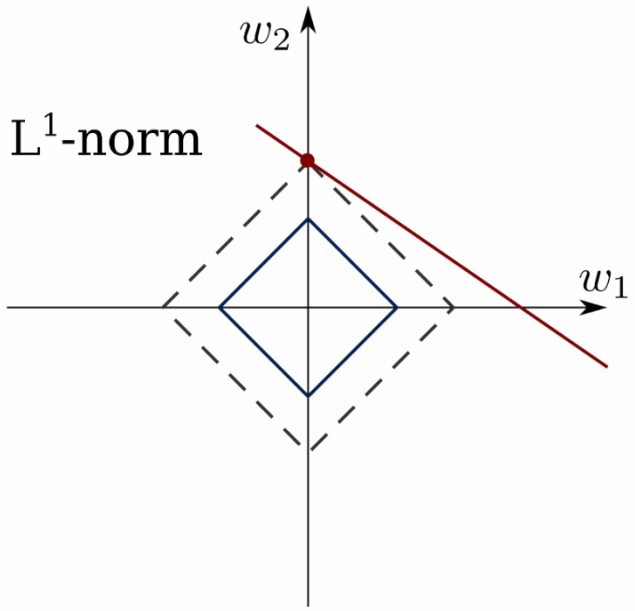
$$\frac{dJ}{dw} = 2\lambda w - y_i \, . \, x_i$$

$$\frac{dJ}{db} = yi$$

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

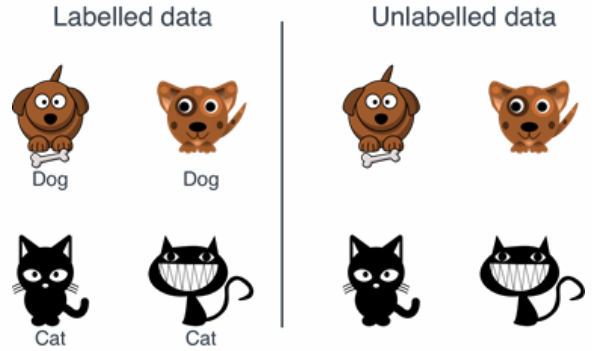
Lasso Regression - intuition

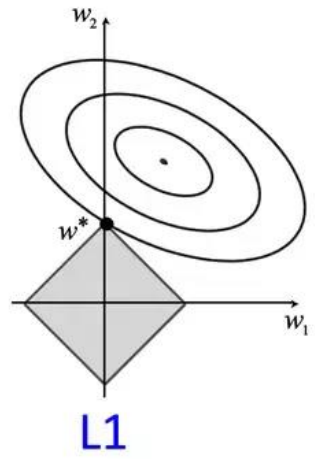


Lasso Regression

About Lasso Regression:

1. Supervised Learning Model
2. Regression model
3. **L**east **A**bsolute **S**hrinkage and **S**election **O**perator
4. Implements Regularization (L1) to avoid Overfitting



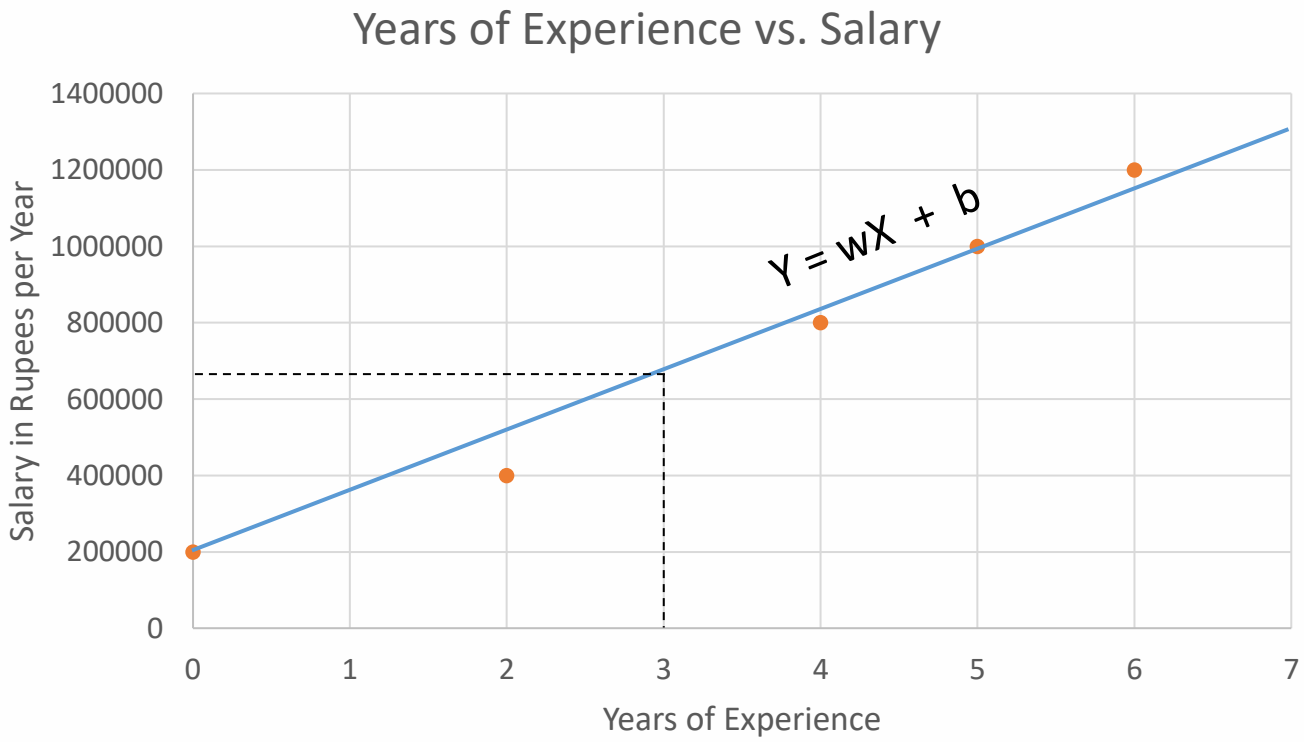


Linear Regression

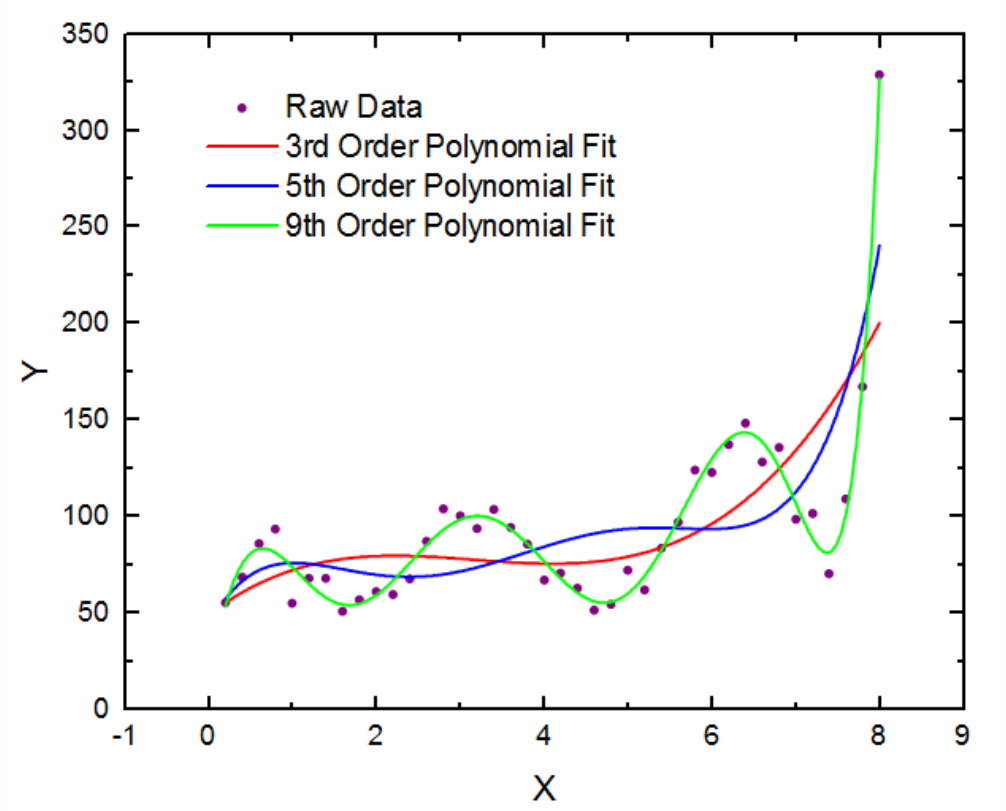
Experience in Years	0	2	4	5	6
Salary	2,00,000	4,00,000	8,00,000	10,00,000	12,00,000

What would be the **salary** of a person with **3 years of Experience?**

~ ₹ 650000 per Year



Polynomial Equations



1st order Polynomial equation : $y = ax + d$

2nd order Polynomial equation : $y = ax^2 + bx + d$

3rd order Polynomial equation : $y = ax^3 + bx^2 + cx + d$

y --> Dependent Variable

x --> Independent Variable

a, b, c --> coefficients

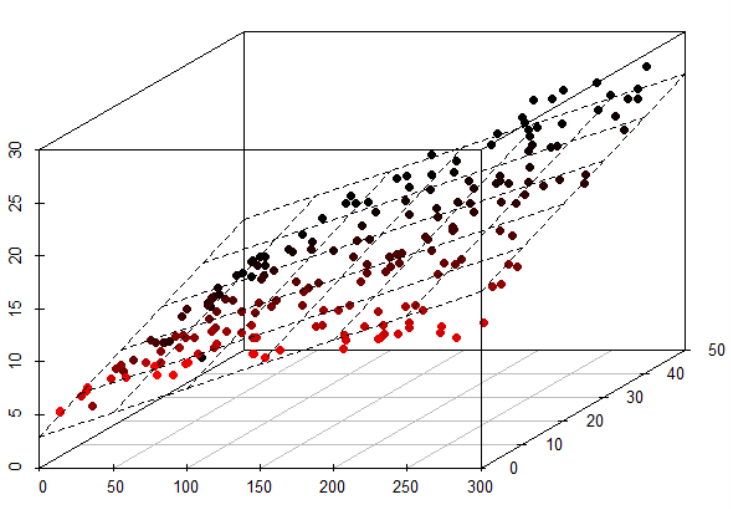
d --> constant term

Inference: As the complexity of the model increases,
It tends to Overfit with the data.

What if there are more than 2 Variables?

Multiple Linear Regression

Multiple linear regression is a model for predicting the value of one dependent variable based on two or more independent variables.



Simple Linear Regression

$$Y = w_1 X_1 + b$$

Multiple Linear Regression

$$Y = w_1 X_1 + w_2 X_2 + w_3 X_3 + b$$

Regularization

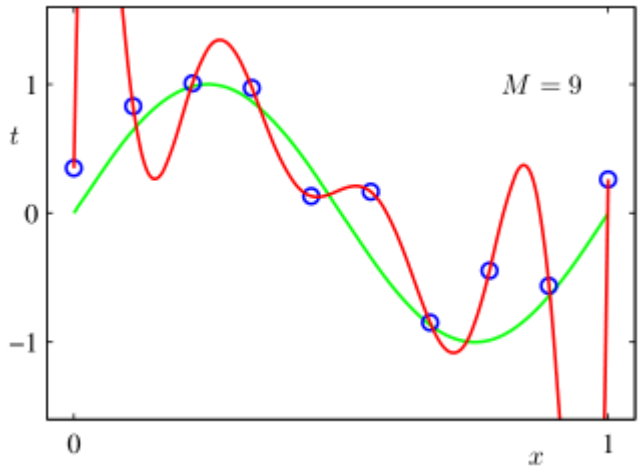
Regularization is used to reduce the overfitting of the model by adding a penalty term (λ) to the model. Lasso Regression uses L1 regularization technique.

The “penalty” term reduces the value of the coefficients or eliminate few coefficients, so that the model has fewer coefficients. As a result, overfitting can be avoided.

3rd order Polynomial equation : $y = ax^3 + bx^2 + cx + d$

This Process is called as Shrinkage.

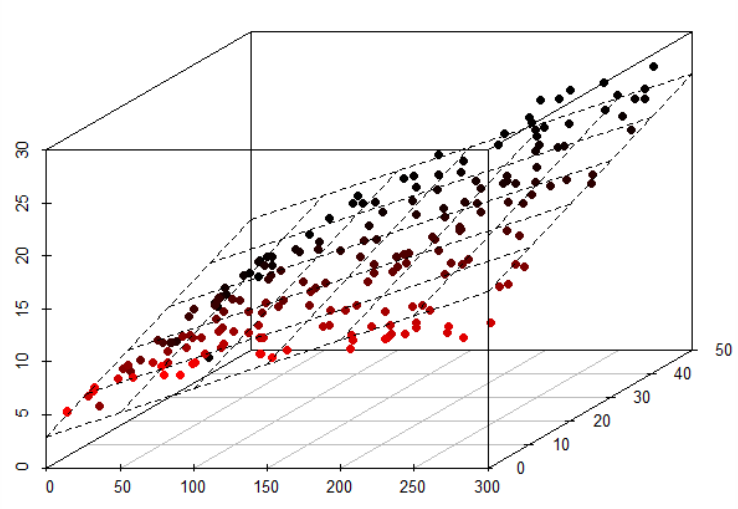
LASSO --> Least **A**bsolute **S**hrinkage and **S**election **O**perator



What if there are more than 2 Variables?

Multiple Linear Regression

Multiple linear regression is a model for predicting the value of one dependent variable based on two or more independent variables.



Simple Linear Regression

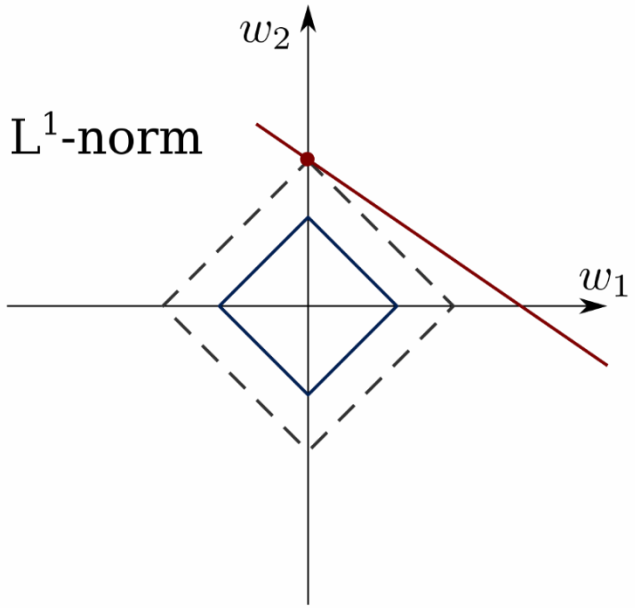
$$Y = w_1 X_1 + b$$

Multiple Linear Regression

$$Y = w_1 X_1 + w_2 X_2 + w_3 X_3 + b$$

Feature Selection

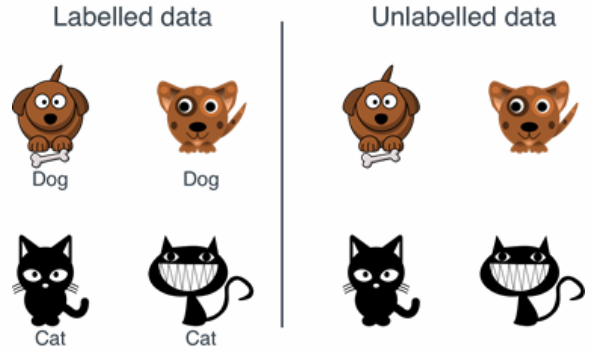
Math behind Lasso Regression

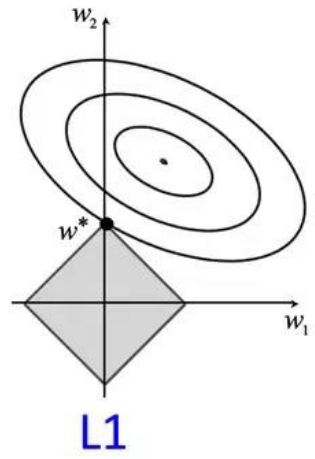


Lasso Regression

About Lasso Regression:

1. Supervised Learning Model
2. Regression model
3. **L**east **A**bsolute **S**hrinkage and **S**election **O**perator
4. Implements Regularization (L1) to avoid Overfitting





Regularization

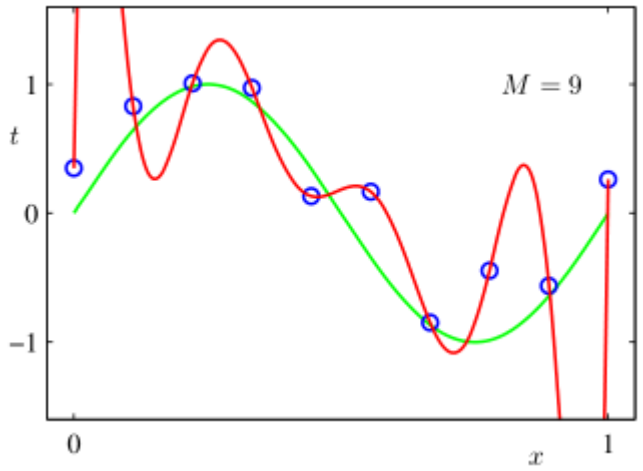
Regularization is used to reduce the overfitting of the model by adding a penalty term (λ) to the model. Lasso Regression uses L1 regularization technique.

The “penalty” term reduces the value of the coefficients or eliminate few coefficients, so that the model has fewer coefficients. As a result, overfitting can be avoided.

3rd order Polynomial equation : $y = ax^3 + bx^2 + cx + d$

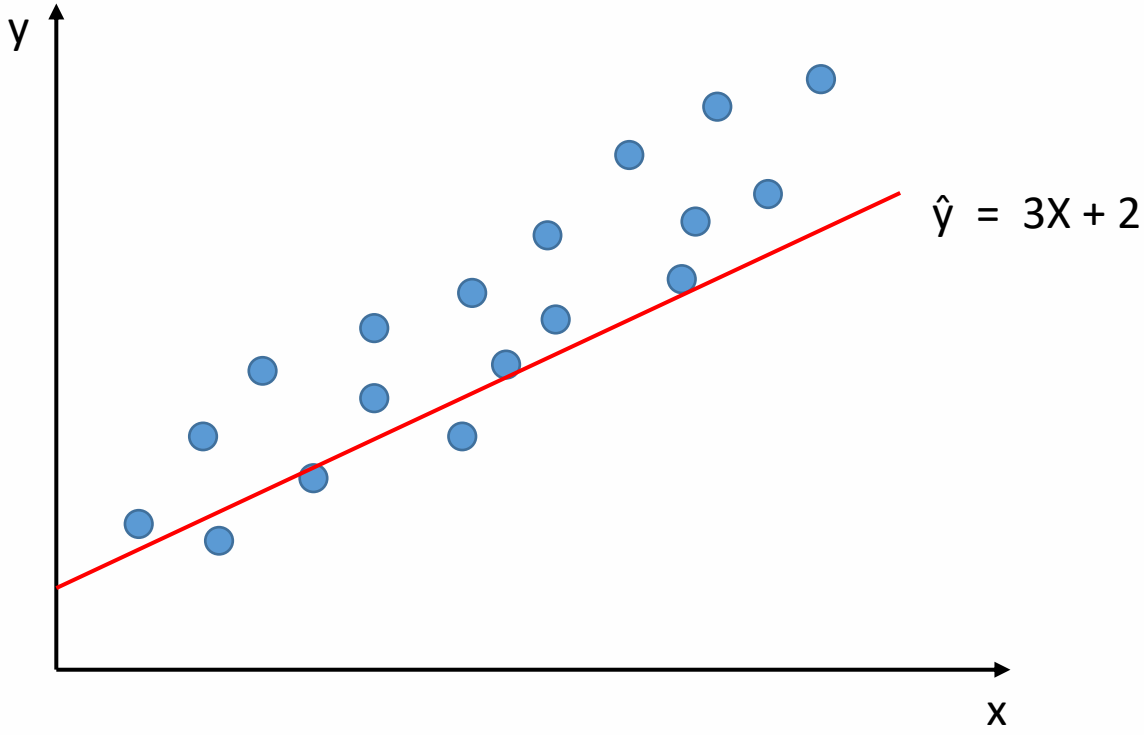
This Process is called as Shrinkage.

LASSO --> Least **A**bsolute **S**hrinkage and **S**election **O**perator



Linear Regression

Randomly assigned Parameters: w = 3; b = 2



x	y	ŷ
2	10	8
3	14	11
4	18	14
5	22	17
6	26	20

Cost Function

x	y	ŷ
2	10	8
3	14	11
4	18	14
5	22	17
6	26	20

$$\textit{Cost (J)} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Cost = [$(10 - 8)^2 + (14 - 11)^2 + (18 - 14)^2 + (22 - 17)^2 + (26 - 20)^2$] / 5

Cost = [4 + 9 + 16 + 25 + 36] / 5

Cost = 18

Low Cost value → High Accuracy

Lasso Regression

Cost Function for Lasso Regression :

$$J = \frac{1}{m}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j\right]$$

m --> Total number of Data Points

n --> Total number of input features

$y^{(i)}$ --> True Value

$\hat{y}^{(i)}$ --> Predicted Value

λ --> Penalty Term

w --> Parameter of the model

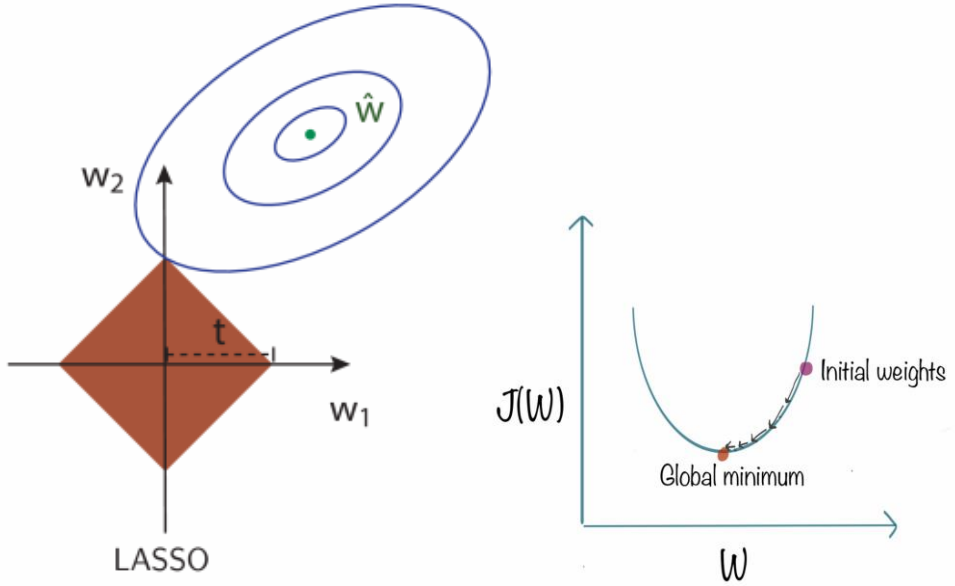
Boston House Price Dataset

The dataset used in this project comes from the UCI Machine Learning Repository. This data was collected in 1978 and each of the 506 entries represents aggregate information about 14 features of homes from various suburbs located in Boston.

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	price
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4

$$J = \frac{1}{m}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda\sum_{j=1}^{n} w_j\right]$$

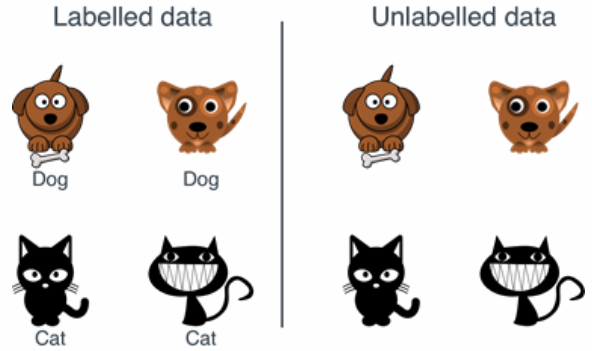
Gradient Descent for Lasso Regression

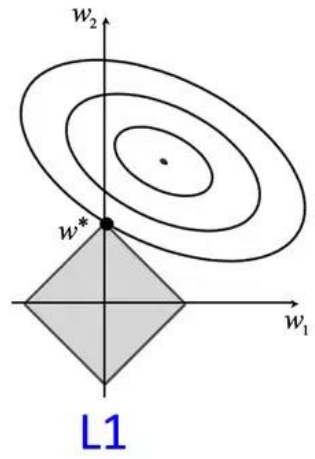


Lasso Regression

About Lasso Regression:

1. Supervised Learning Model
2. Regression model
3. **L**east **A**bsolute **S**hrinkage and **S**election **O**perator
4. Implements Regularization (L1) to avoid Overfitting





Regularization

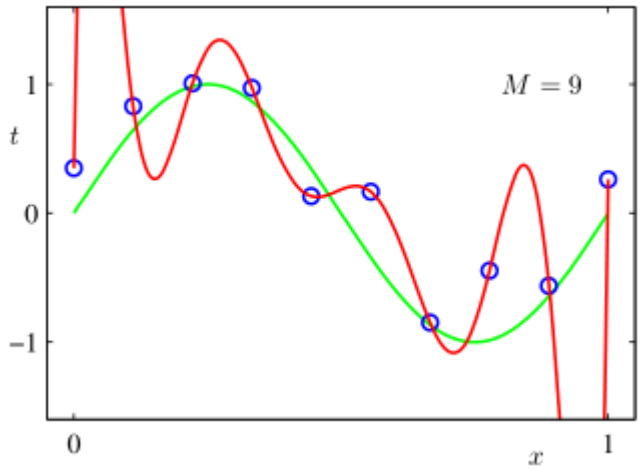
Regularization is used to reduce the overfitting of the model by adding a penalty term (λ) to the model. Lasso Regression uses L1 regularization technique.

The “penalty” term reduces the value of the coefficients or eliminate few coefficients, so that the model has fewer coefficients. As a result, overfitting can be avoided.

3rd order Polynomial equation : $y = ax^3 + bx^2 + cx + d$

This Process is called as Shrinkage.

LASSO --> Least **A**bsolute **S**hrinkage and **S**election **O**perator



Lasso Regression

Cost Function for Lasso Regression :

$$J = \frac{1}{m}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j\right]$$

m --> Total number of Data Points

n --> Total number of input features

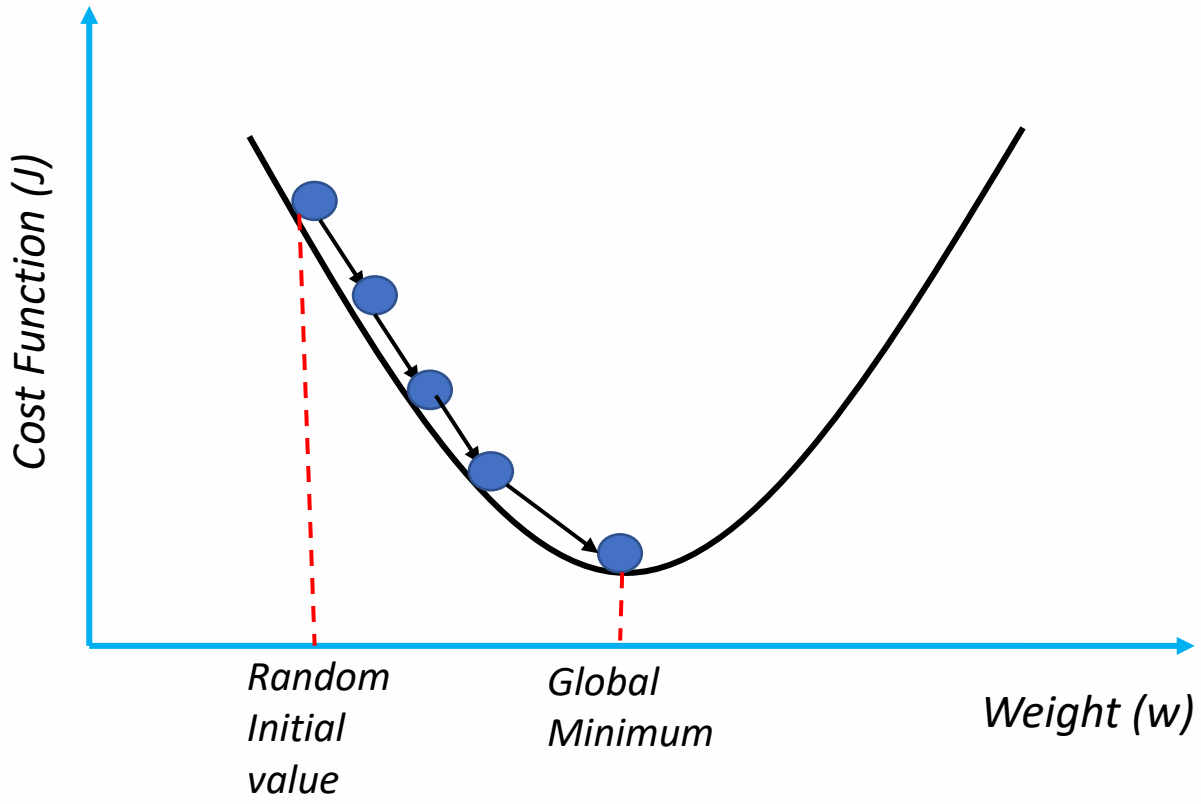
$y^{(i)}$ --> True Value

$\hat{y}^{(i)}$ --> Predicted Value

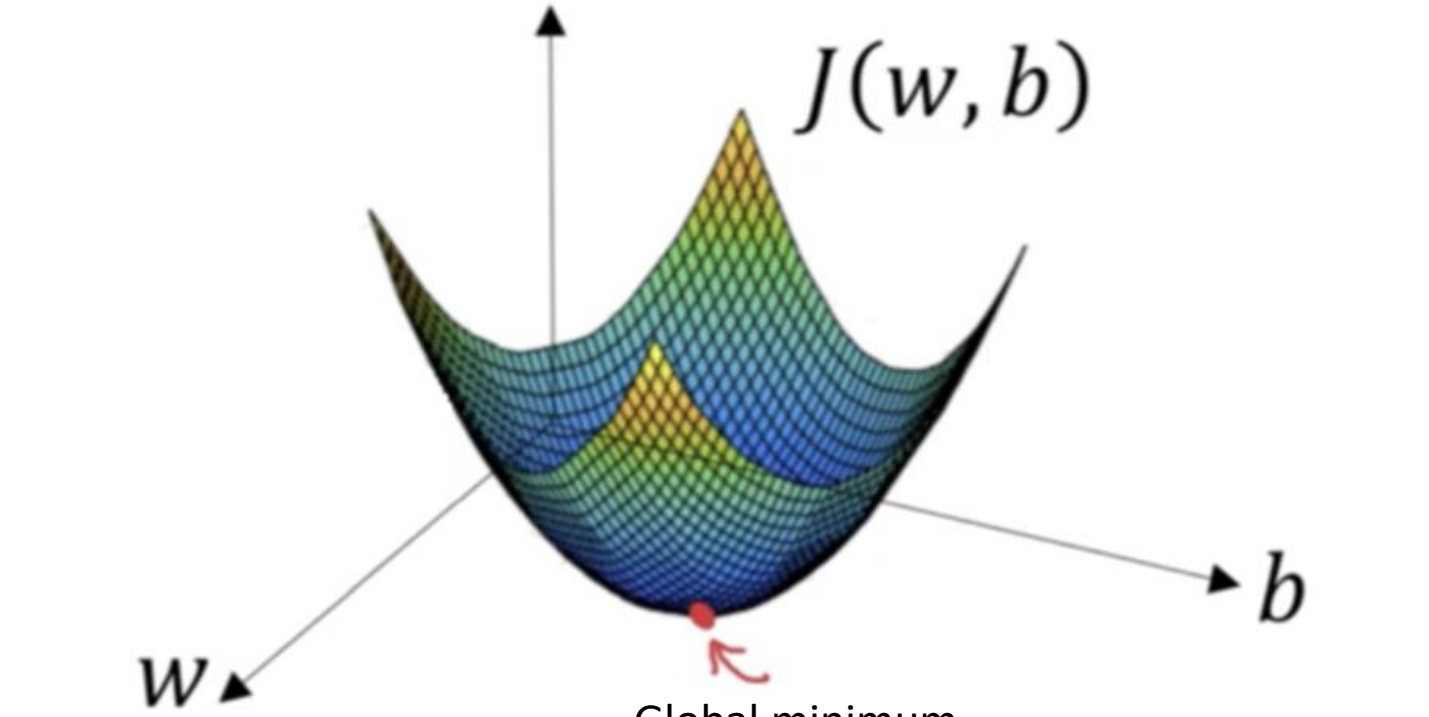
λ --> Penalty Term

w --> Parameter of the model

Gradient Descent



Gradient Descent in 3 Dimension



Global minimum

Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

w --> weight
b --> bias
L --> Learning Rate

$\frac{dJ}{dw}$ --> Partial Derivative of cost function with respect to w

$\frac{dJ}{db}$ --> Partial Derivative of cost function with respect to b

Gradients for Lasso Regularization

if $(w_j > 0)$:	*else* $(w_j \leq 0)$:
$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] + \lambda\right]$	$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] - \lambda\right]$

$$\frac{dJ}{db} = \frac{-2}{m}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)\right]$$

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

$$y = w.x + b$$

Gradients for Lasso Regularization

if $(w_j > 0)$:	*else* $(w_j \leq 0)$:
$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] + \lambda\right]$	$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] - \lambda\right]$

$$\frac{dJ}{db} = \frac{-2}{m}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)\right]$$

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

if (w_j > 0) :

$$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] + \lambda\right]$$

else ($w_j \leq 0$) :

$$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] - \lambda\right]$$

$$\frac{dJ}{db} = \frac{-2}{m}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)\right]$$

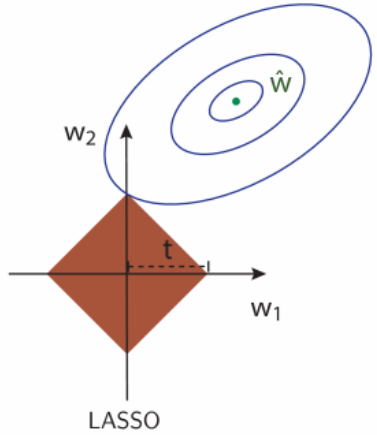
if (w_j > 0) :

$$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] + \lambda\right]$$

else ($w_j \leq 0$) :

$$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] - \lambda\right]$$

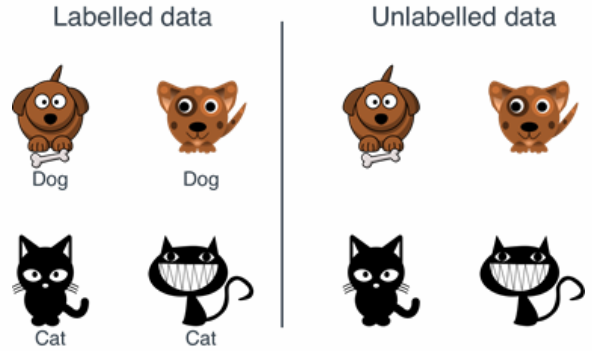
Building Lasso Regression from Scratch in Python

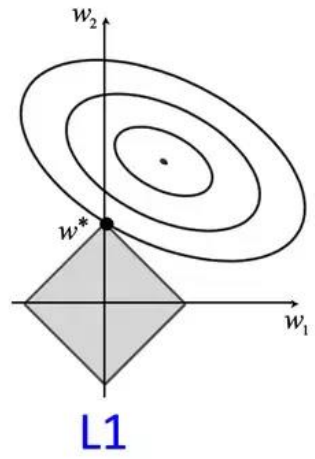


Lasso Regression

About Lasso Regression:

1. Supervised Learning Model
2. Regression model
3. **L**east **A**bsolute **S**hrinkage and **S**election **O**perator
4. Implements Regularization (L1) to avoid Overfitting





Regularization

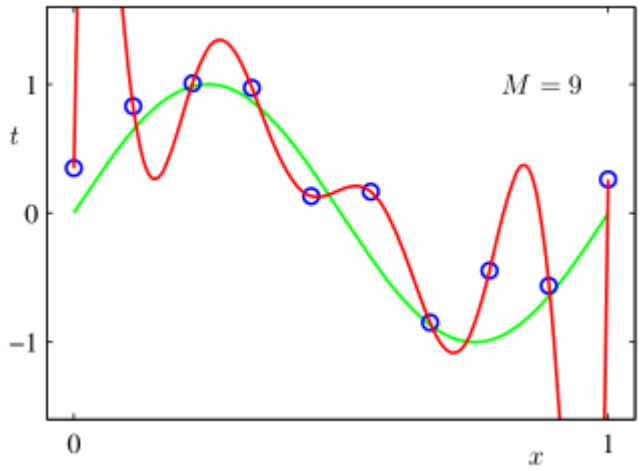
Regularization is used to reduce the overfitting of the model by adding a penalty term (λ) to the model. Lasso Regression uses L1 regularization technique.

The “penalty” term reduces the value of the coefficients or eliminate few coefficients, so that the model has fewer coefficients. As a result, overfitting can be avoided.

3rd order Polynomial equation : $y = ax^3 + bx^2 + cx + d$

This Process is called as Shrinkage.

LASSO --> Least **A**bsolute **S**hrinkage and **S**election **O**perator



Lasso Regression

Cost Function for Lasso Regression :

$$J = \frac{1}{m}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j\right]$$

m --> Total number of Data Points

n --> Total number of input features

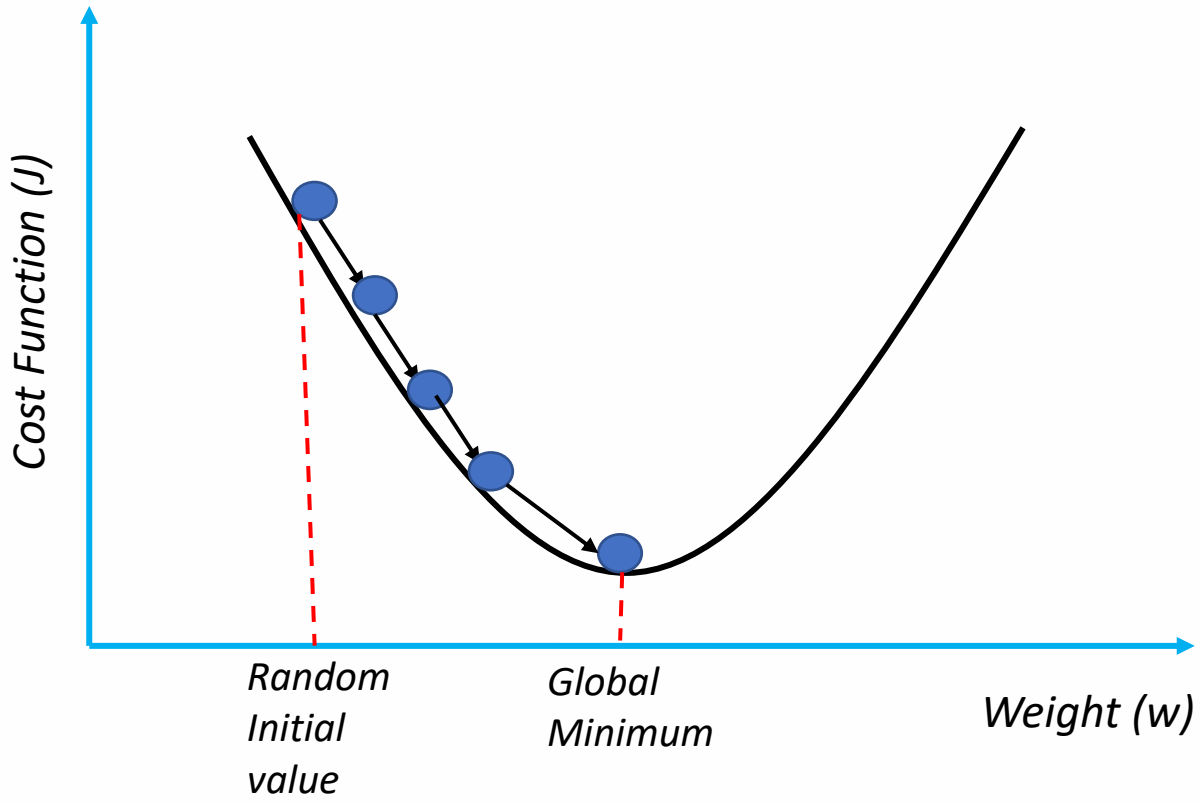
$y^{(i)}$ --> True Value

$\hat{y}^{(i)}$ --> Predicted Value

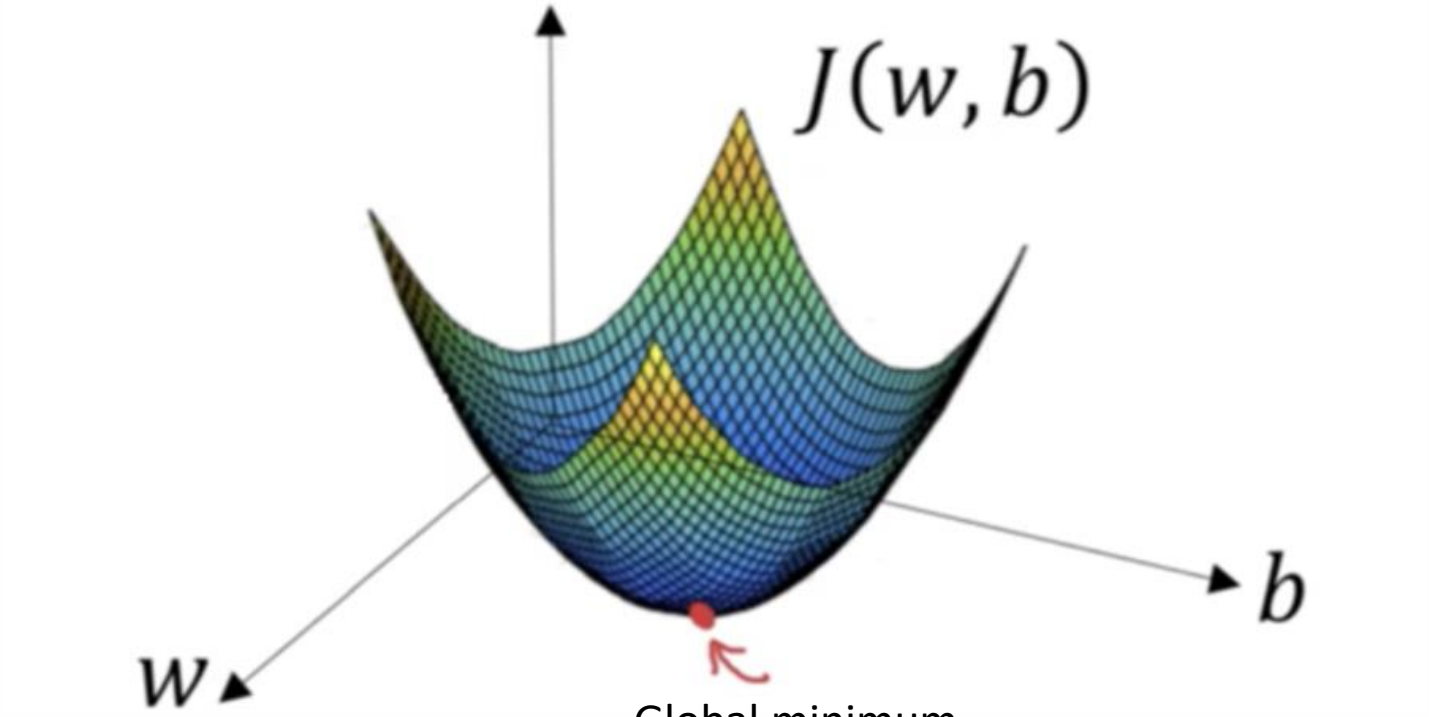
λ --> Penalty Term

w --> Parameter of the model

Gradient Descent



Gradient Descent in 3 Dimension



Global minimum

Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

w --> weight
b --> bias
L --> Learning Rate

$\frac{dJ}{dw}$ --> Partial Derivative of cost function with respect to w

$\frac{dJ}{db}$ --> Partial Derivative of cost function with respect to b

Gradients for Lasso Regularization

if $(w_j > 0)$:	*else* $(w_j \leq 0)$:
$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] + \lambda\right]$	$\frac{dJ}{dw} = \frac{-2}{m}\left[\left[\sum_{i=1}^{m} x_j \cdot\left(y^{(i)} - \hat{y}^{(i)}\right)\right] - \lambda\right]$

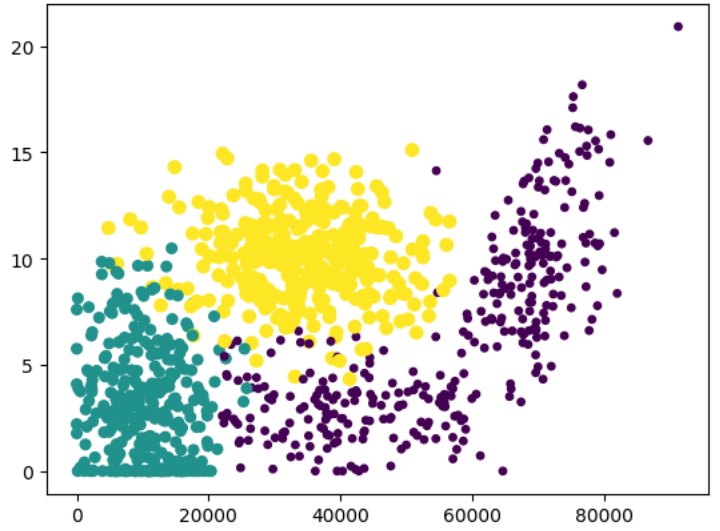
$$\frac{dJ}{db} = \frac{-2}{m}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)\right]$$

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

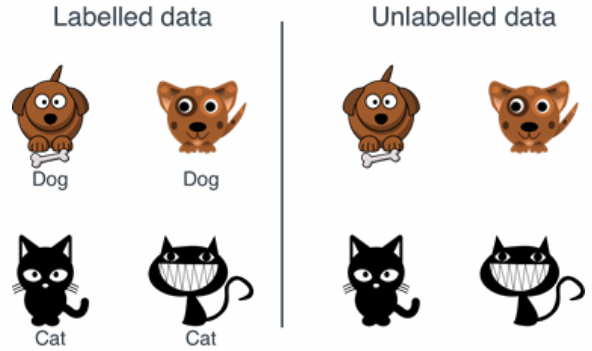
$$y = w.x + b$$

K-Nearest Neighbors (KNN) - intuition

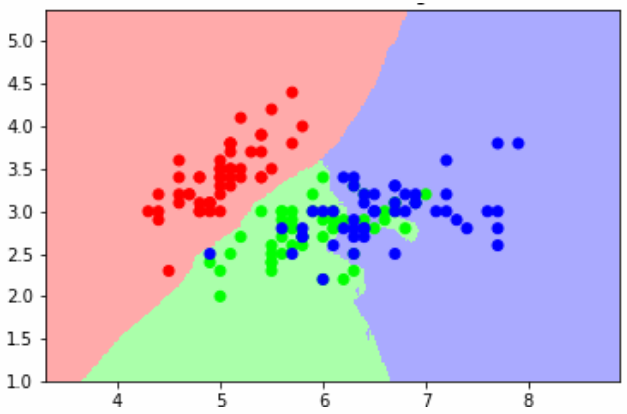


K-Nearest Neighbors

About K-Nearest Neighbors:

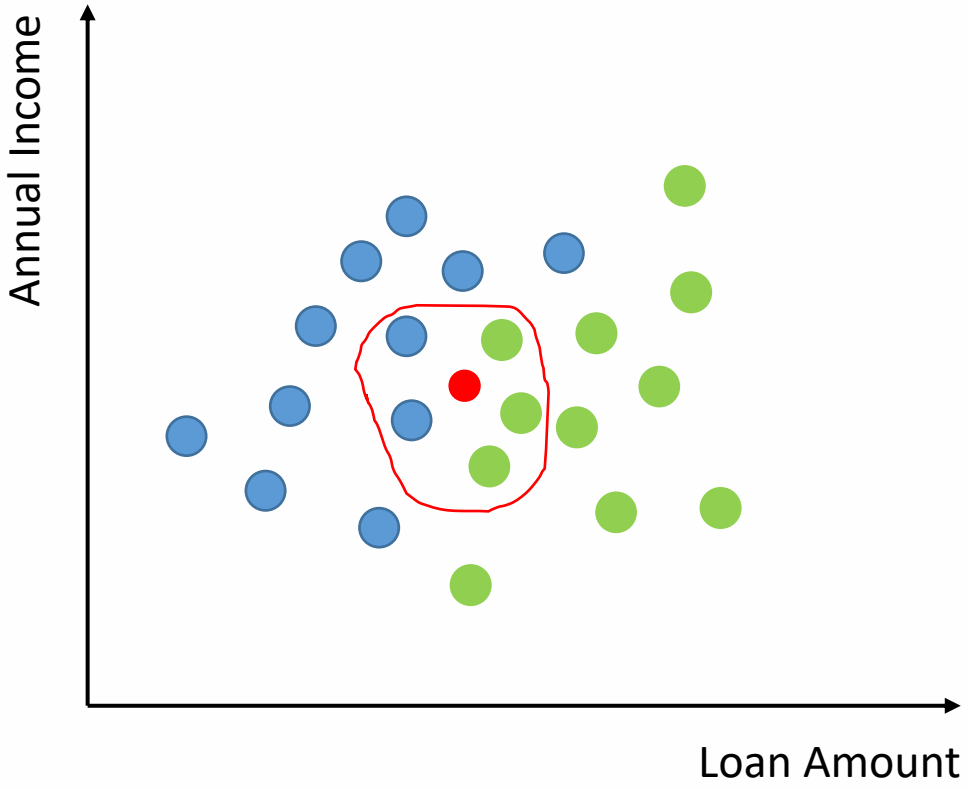


1. Supervised Learning Model
2. Used for both Classification & Regression
3. Can be used for non-linear data
4. K - Neighbors



K-Nearest Neighbors

Classification Problem:



K = 5

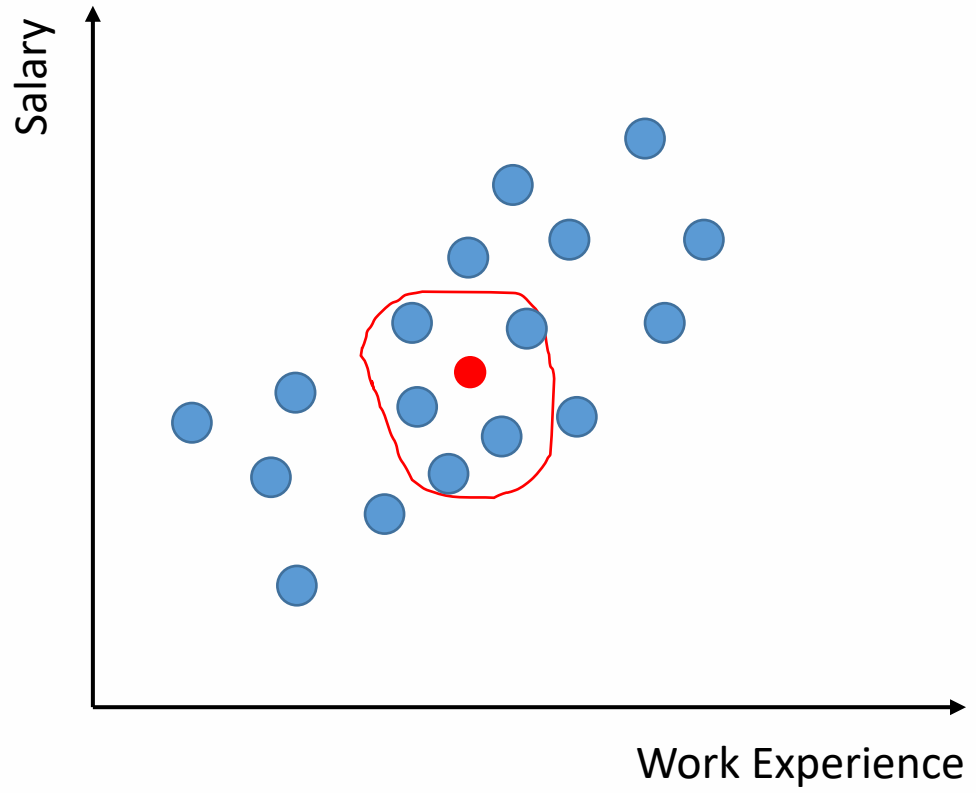
- Didn't repay on time
- Repaid on time
- May not repay the loan on time

To Measure the distance between the data points:

- Euclidean Distance
- Manhattan Distance

K-Nearest Neighbors

Regression Problem:



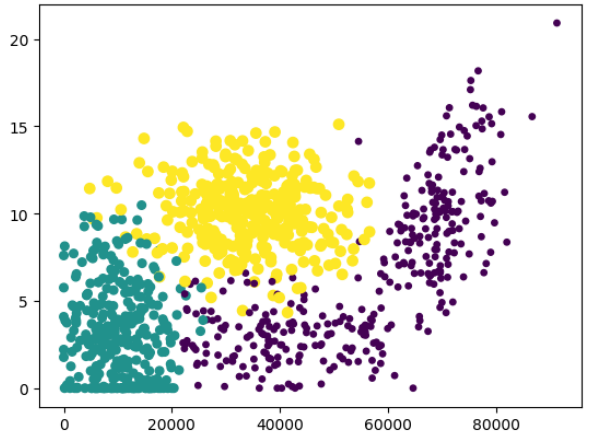
K = 5

Salary of the person can be calculated as the mean of 5 nearest neighbors.

K-Nearest Neighbors

Advantages:

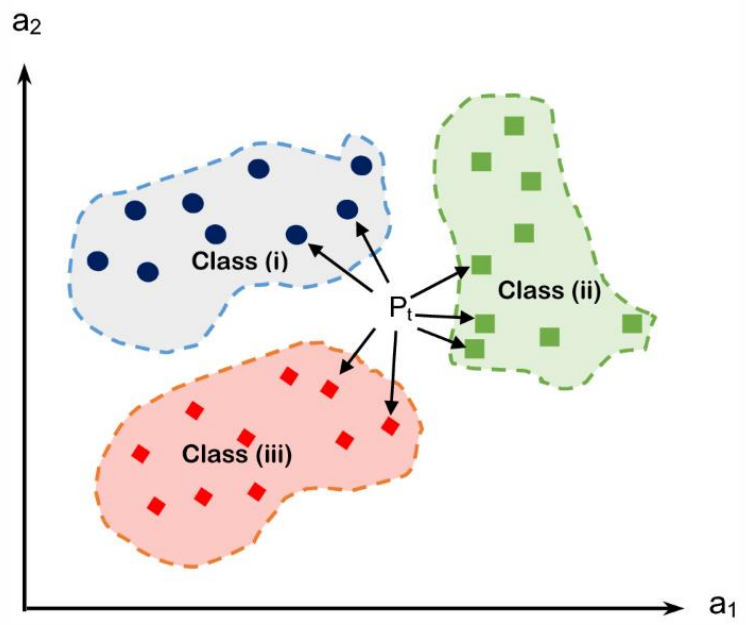
1. Works well with smaller datasets with less number of features
2. Can be used for both Classification & Regression
3. Easy to implement for Multi-class classification problems
4. Different distance criteria can be used
   (eg: Euclidean Distance, Manhattan Distance)



Disadvantages:

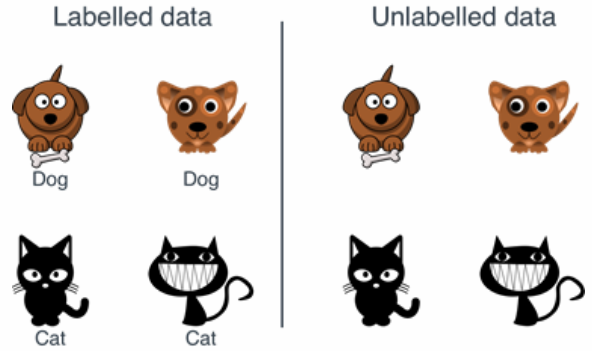
1. Choosing optimum “K” value
2. Less efficient with high dimensional data.
3. Doesn’t perform well on imbalanced dataset
4. Sensitive to Outliers

Math behind K-Nearest Neighbors (KNN) Classifier

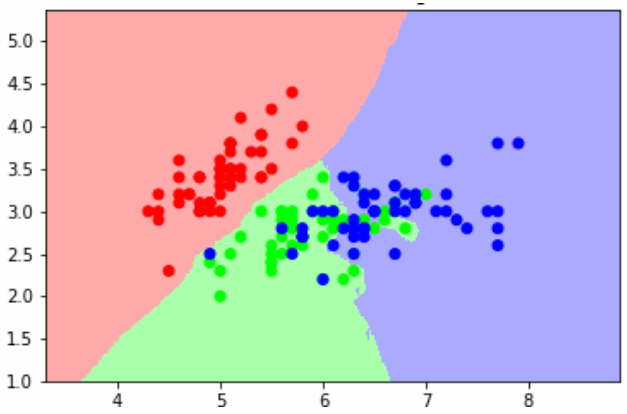


K-Nearest Neighbors

About K-Nearest Neighbors:

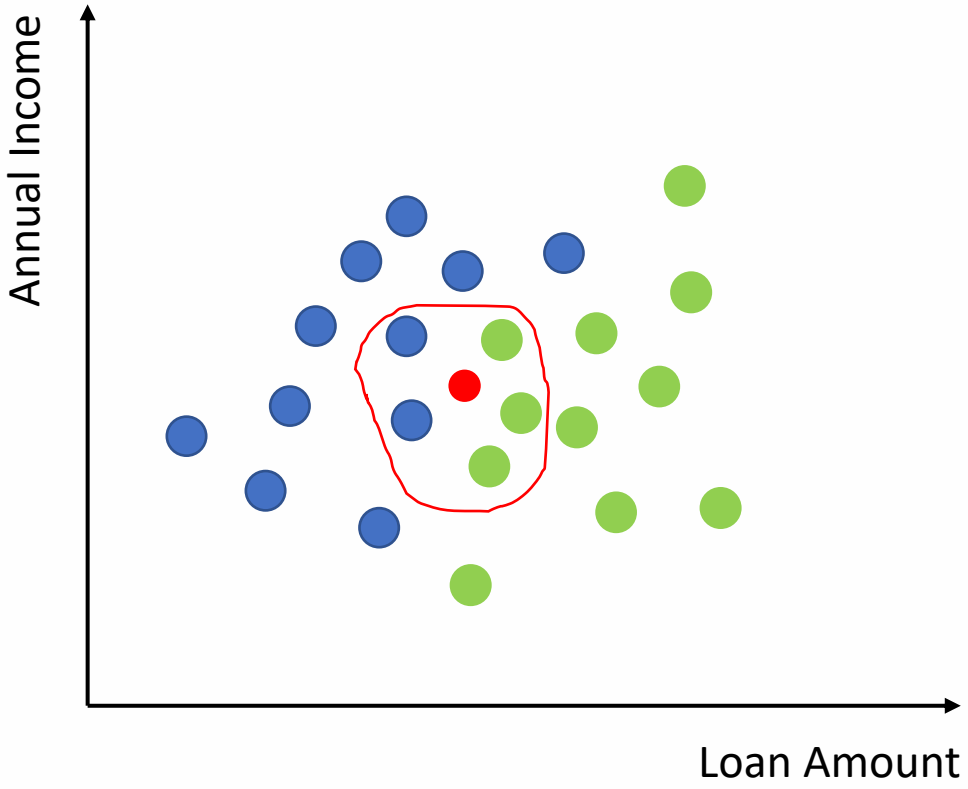


1. Supervised Learning Model
2. Used for both Classification & Regression
3. Can be used for non-linear data
4. K - Neighbors



K-Nearest Neighbors

Classification Problem:



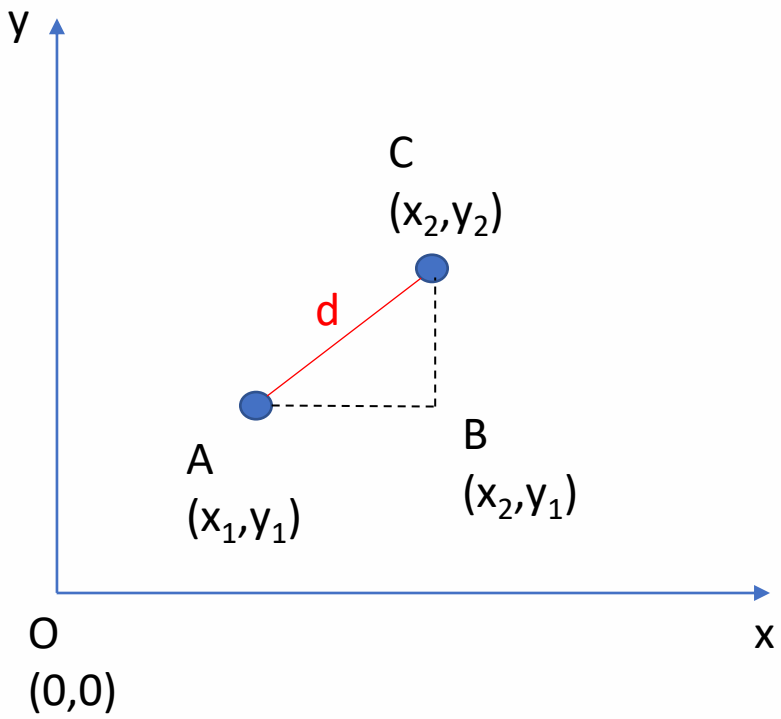
K = 5

- Didn't repay on time
- Repaid on time
- May not repay the loan on time

To Measure the distance between the data points:

- Euclidean Distance
- Manhattan Distance

Euclidean Distance



Pythagoras Theorem:

$$AC^2 = AB^2 + BC^2$$

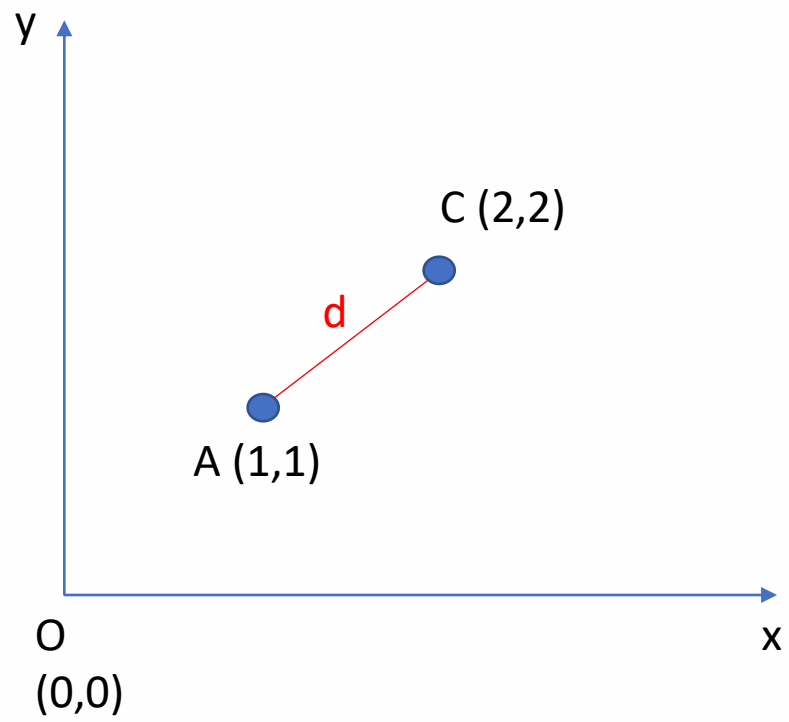
$$AC = \sqrt{AB^2 + BC^2}$$

$$AC = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

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

This distance “d” is called the Euclidean Distance.

Euclidean Distance



Euclidean Distance formula:

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

(x_1, y_1) = A (1,1)

(x_2, y_2) = B (2,2)

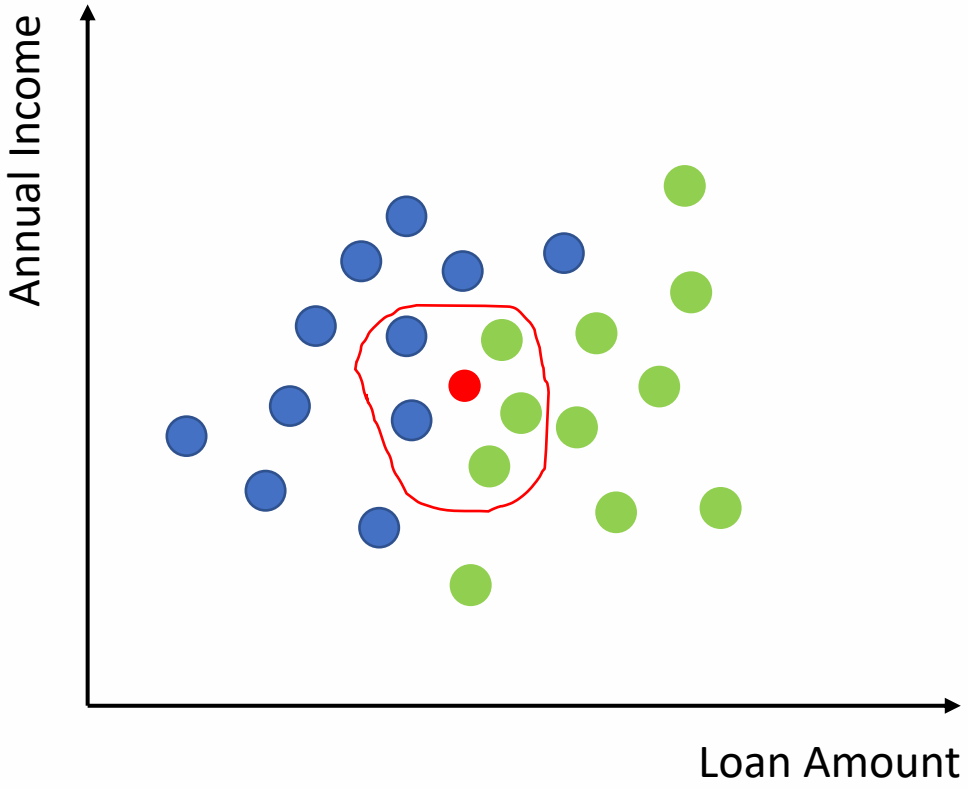
$$d = \sqrt{(2 - 1)^2 + (2 - 1)^2}$$

$$d = \sqrt{1 + 1}$$

$$d = \sqrt{2}$$

K-Nearest Neighbors

Classification Problem:



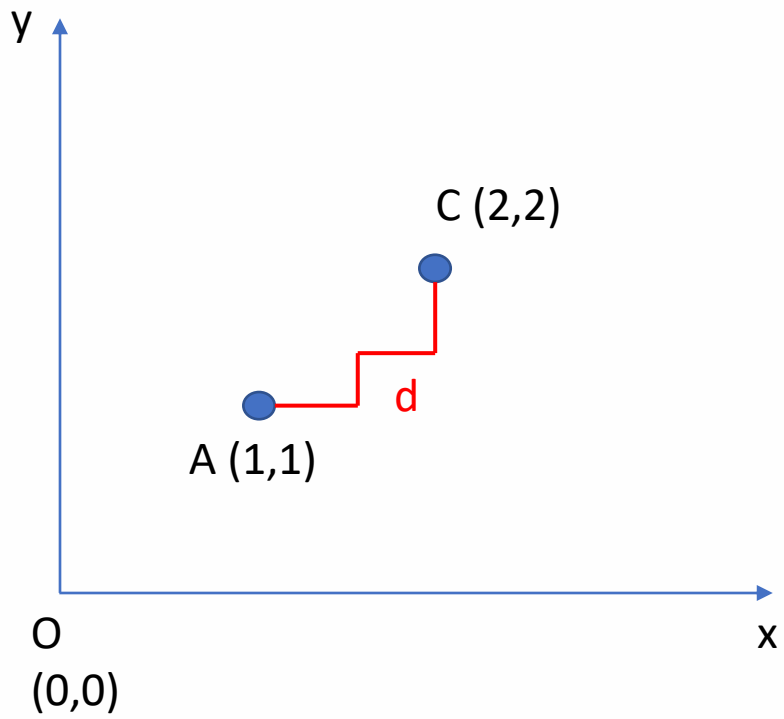
K = 5

- Didn't repay on time
- Repaid on time
- May not repay the loan on time

To Measure the distance between the data points:

- Euclidean Distance
- Manhattan Distance

Manhattan Distance



Manhattan Distance formula:

$$d = |x_1 - x_2| + |y_1 - y_2|$$

(x_1, y_1) = A (1,1)

(x_2, y_2) = B (2,2)

$$d = |1 - 2| + |1 - 2|$$

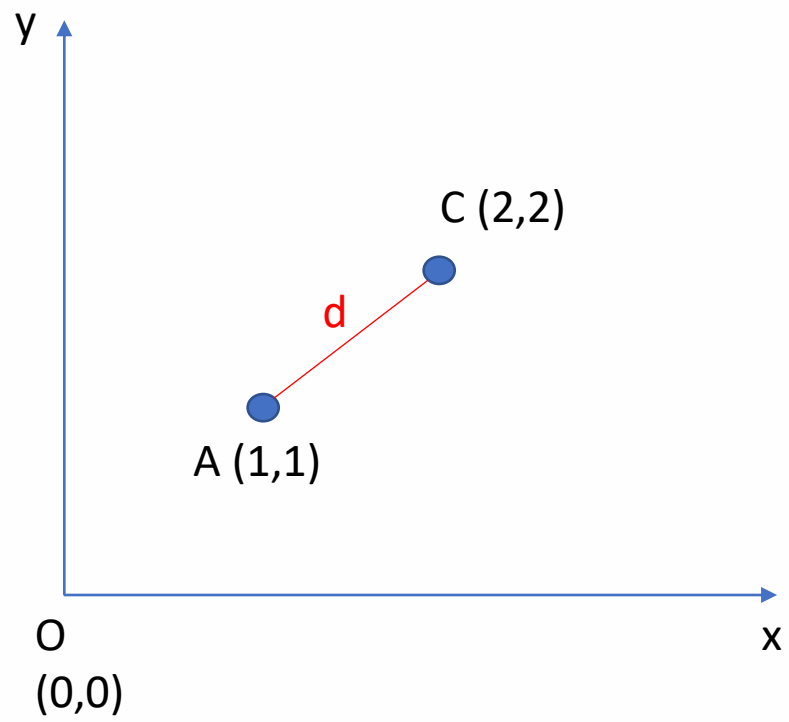
$$d = 1 + 1$$

$$d = 2$$

Manhattan distance is preferred over Euclidean distance when there is **high dimensionality** in the data.

Calculating Euclidean & Manhattan Distance in Python

Euclidean Distance



Euclidean Distance formula:

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

(x_1, y_1) = A (1,1)

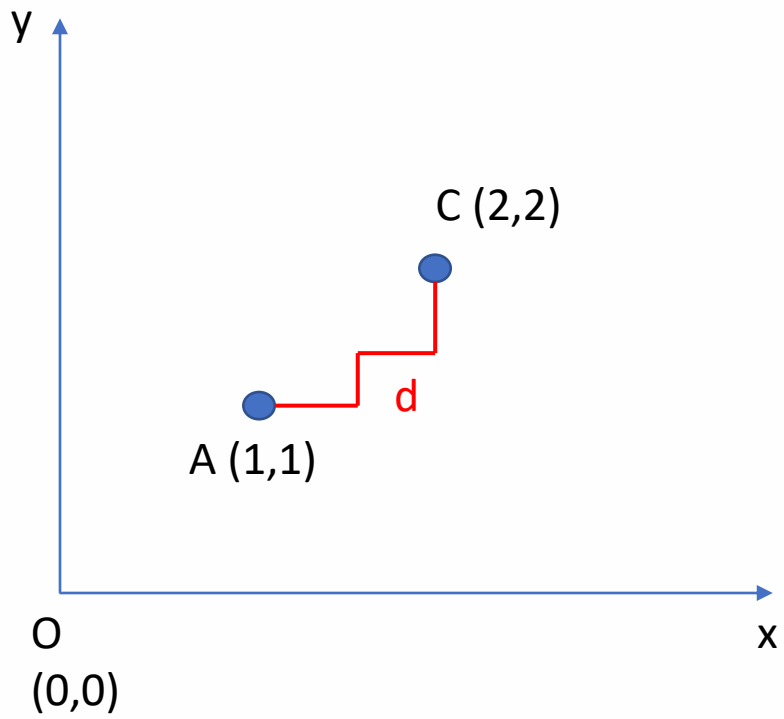
(x_2, y_2) = B (2,2)

$$d = \sqrt{(2 - 1)^2 + (2 - 1)^2}$$

$$d = \sqrt{1 + 1}$$

$$d = \sqrt{2}$$

Manhattan Distance



Manhattan Distance formula:

$$d = |x_1 - x_2| + |y_1 - y_2|$$

(x_1, y_1) = A (1,1)

(x_2, y_2) = B (2,2)

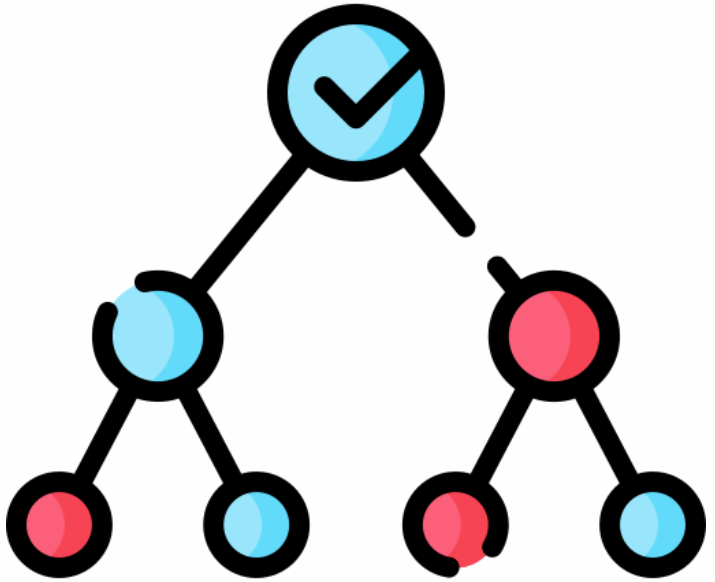
$$d = |1 - 2| + |1 - 2|$$

$$d = 1 + 1$$

$$d = 2$$

Manhattan distance is preferred over Euclidean distance when there is **high dimensionality** in the data.

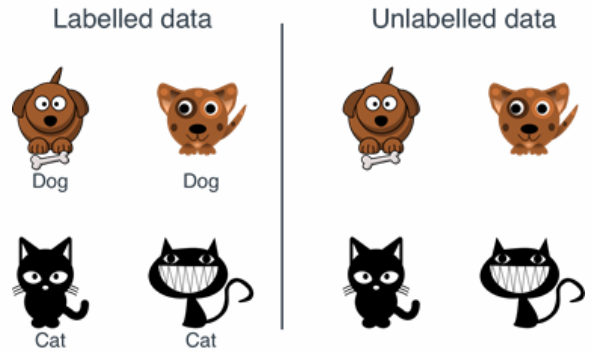
Decision Tree
- intuition

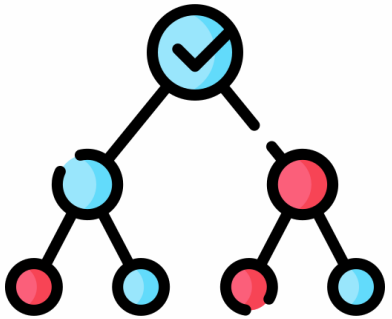


Decision Tree

About Decision Tree model:

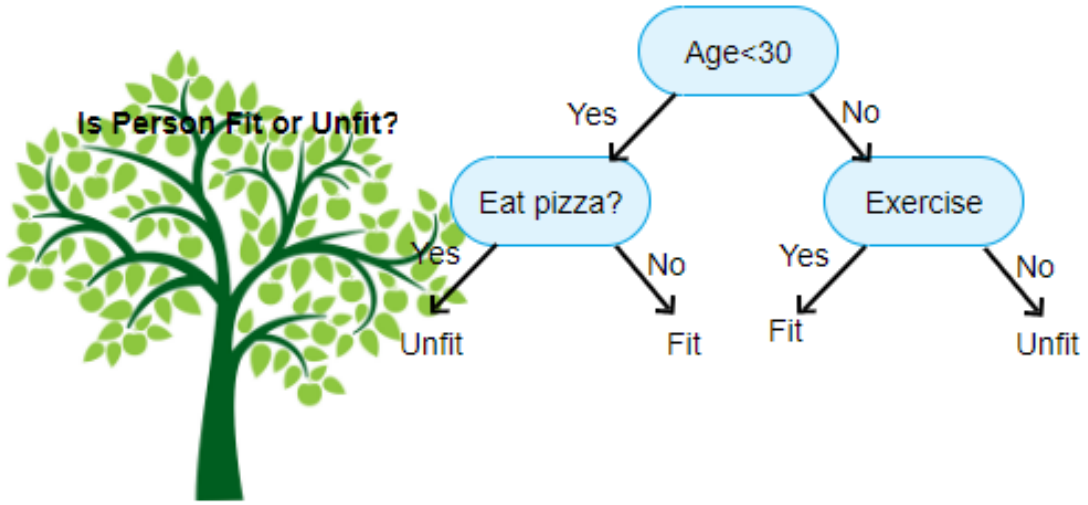
1. Supervised Learning Model
2. Used for both Classification & Regression
3. Builds Decision Nodes at each step
4. Basis of Tree-based models



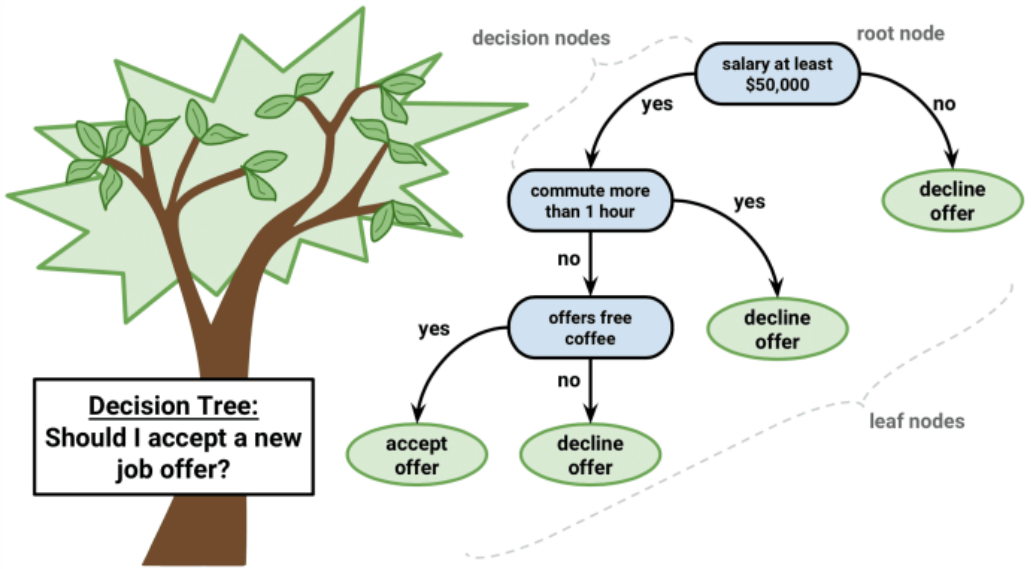


Decision Tree

Decision Tree – examples:

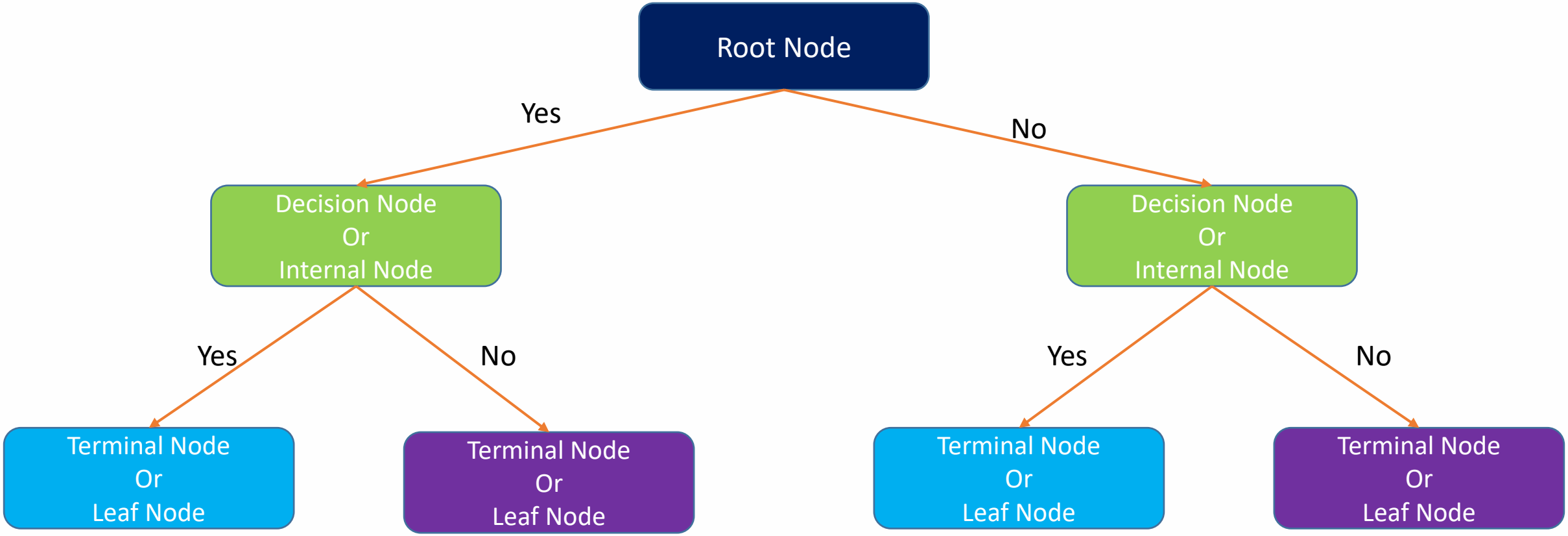


Picture credits: AI Time Journal



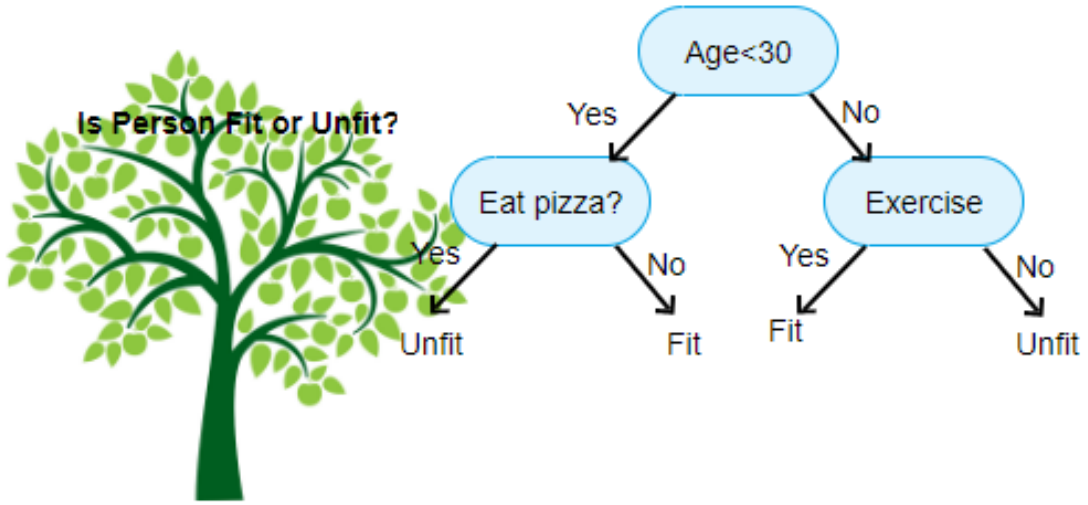
Picture credits:
https://towardsdatascience.com/decision-tree-hugging-b8851f853486

Decision Tree - Terminologies

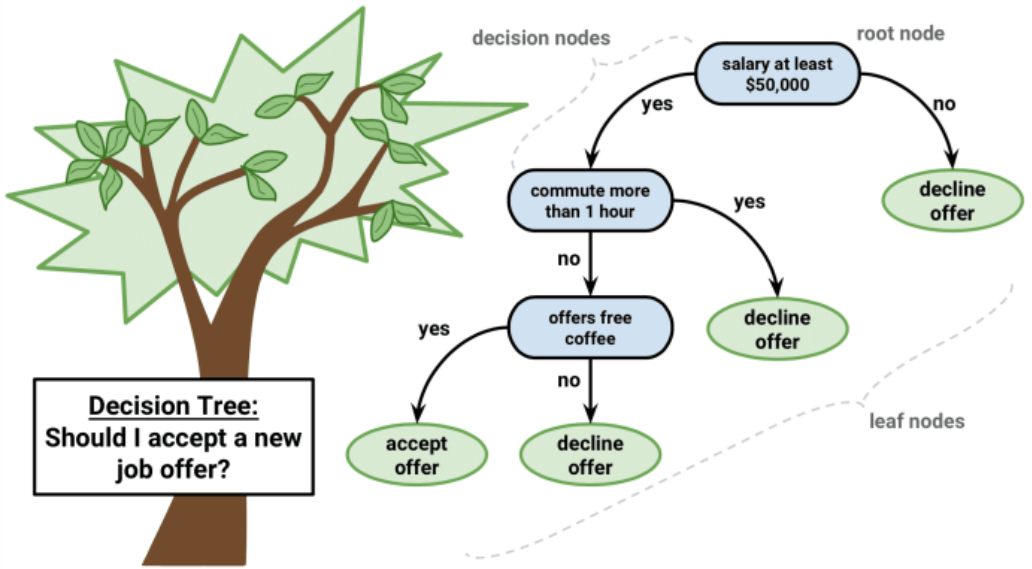


Decision Tree

Decision Tree – examples:



Picture credits: AI Time Journal

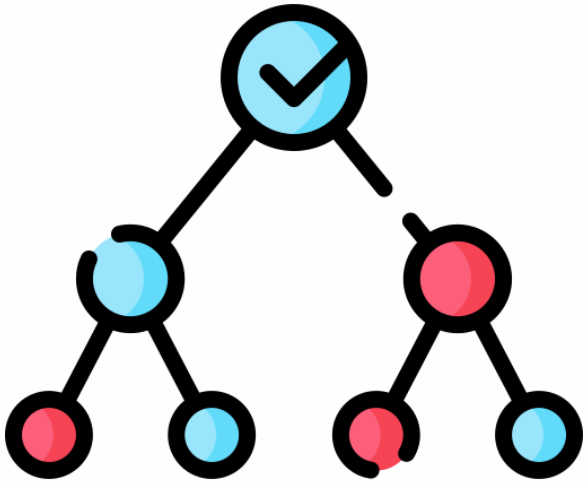


Picture credits:
https://towardsdatascience.com/decision-tree-hugging-b8851f853486

Decision Tree

Advantages:

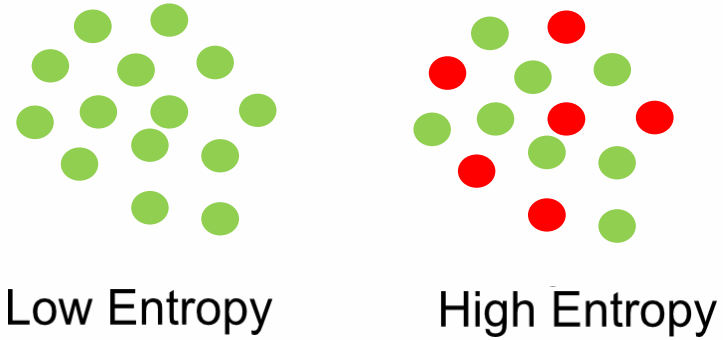
1. Can be used for both Classification & Regression
2. Easy to interpret
3. No need for normalization or scaling
4. Not sensitive to outliers



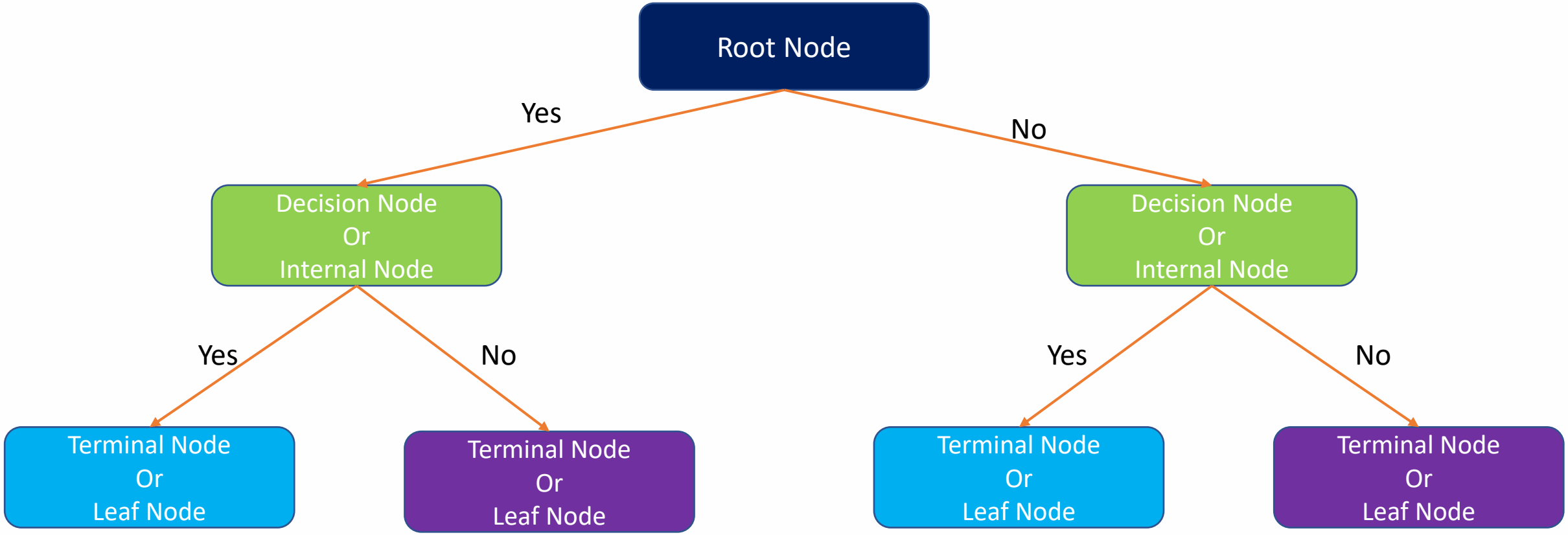
Disadvantages:

1. Overfitting issue
2. Small changes in the data alter the tree structure causing instability
3. Training time is relatively higher

Entropy, Information Gain & Gini Impurity



Decision Tree - Terminologies

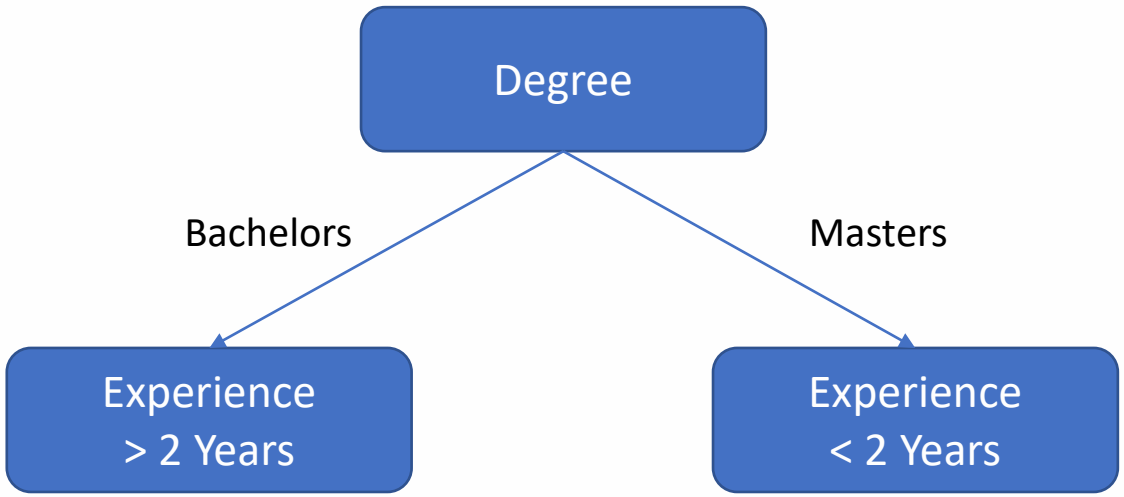


Decision Tree

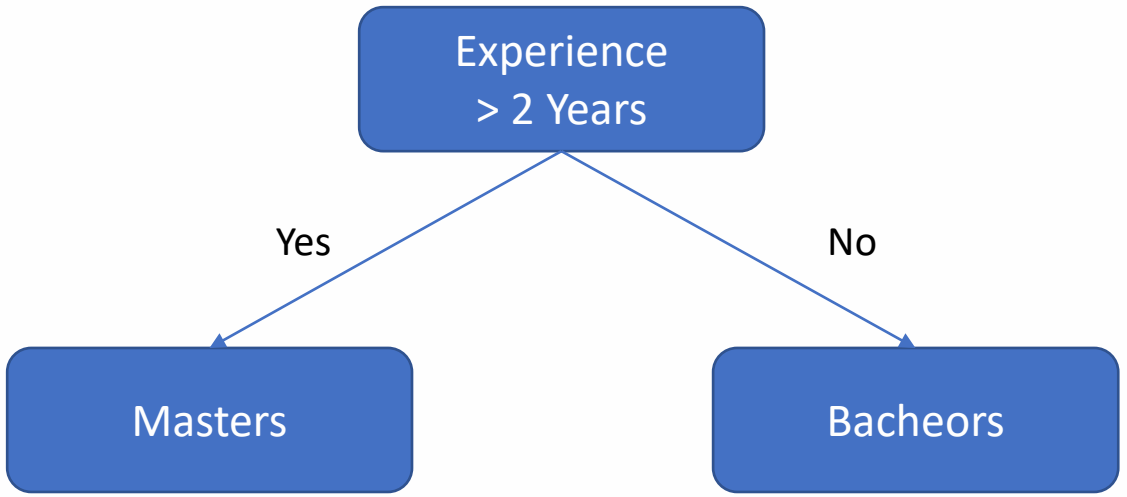
Problem Statement: Build a Decision Tree to determine whether a person will **get a Job or not** based on their **Degree** & **Years of Experience**.

Degree	Experience in Years	Placed / Not Placed
Masters	2	Placed
Bachelors	0	Not Placed
Masters	3	Placed
Masters	1	Not Placed
Bachelors	2	Placed
Masters	3	Placed
Bachelors	0	Not Placed
Bachelors	1	Not Placed

Decision Tree



Entropy: High
Information Gain: Low
Gini Impurity: High



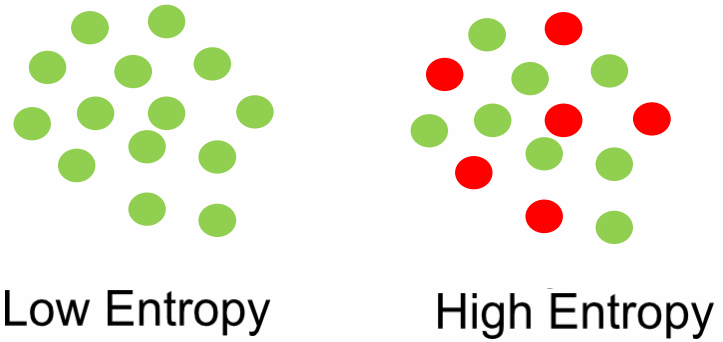
Entropy: Low
Information Gain: High
Gini Impurity: Low

Entropy

Entropy:

In Machine Learning, **Entropy** is the quantitative measure of the **randomness** of the information being processed.

A **high value of Entropy** means that the **randomness** in the system is **high** and thus making accurate predictions is tough.
A **low value of Entropy** means that the **randomness** in the system is **low** and thus making accurate predictions is easier.



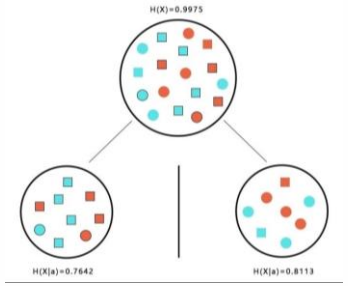
$$\textbf{Entropy} = \sum_{i=1}^{c} - p_i \log_2 p_i$$

c --> number of classes
p_i --> Probability of i^{th} class

Information Gain

Information Gain is the measure of how much information a feature provides about a class. Low entropy leads to increased Information Gain and high entropy leads to low Information Gain.

Information gain computes the difference between **entropy before split** and average entropy **after split** of the dataset based on a given feature.

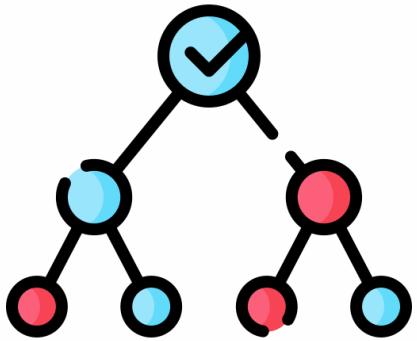


$$\textbf{Information gain (T, F)} = \mathbf{Entropy\ (T)} - \sum_{v \in F} \frac{|T_v|}{T} . \boldsymbol{Entropy\ (T)}$$

Gini Impurity

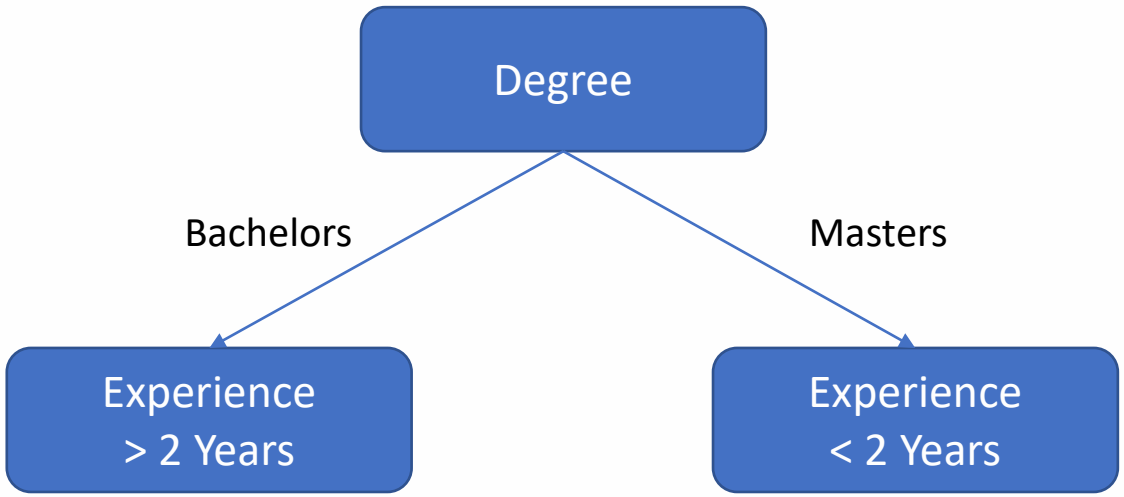
The split made in a Decision Tree is said to be pure if all the data points are accurately separated into different classes.

Gini Impurity measures the likelihood that a randomly selected data point would be incorrectly classified by a specific node.

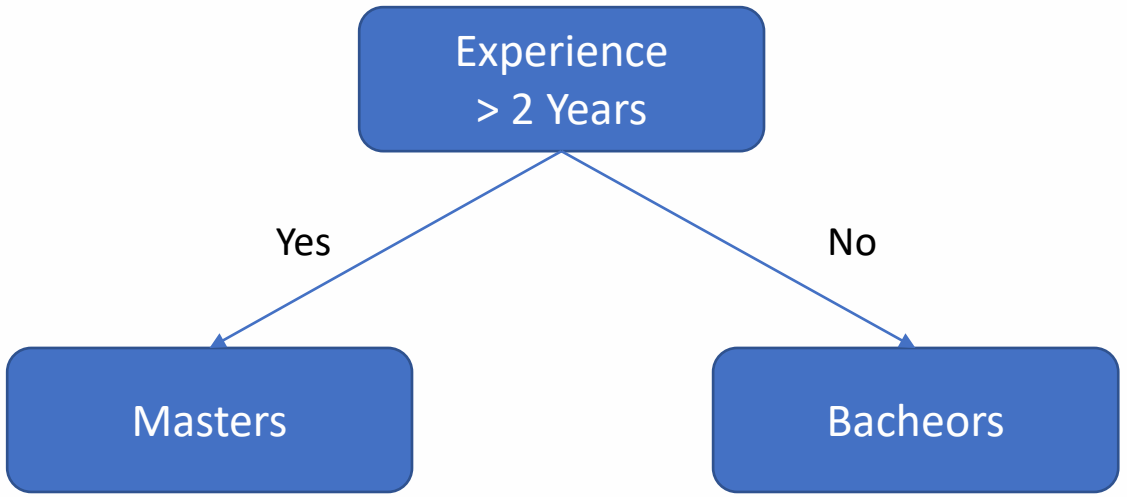


$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

Decision Tree



Entropy: High
Information Gain: Low
Gini Impurity: High



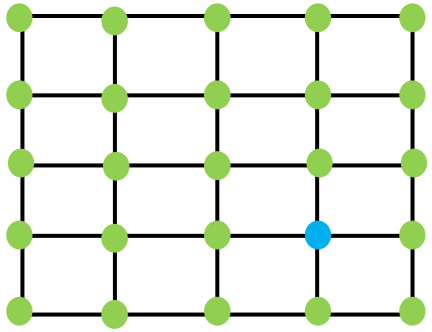
Entropy: Low
Information Gain: High
Gini Impurity: Low

Cross Validation, Hyperparameter Tuning, & Evaluation metrics

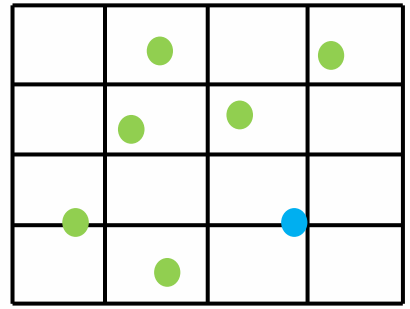
Module 8 - Outline

Cross Validation

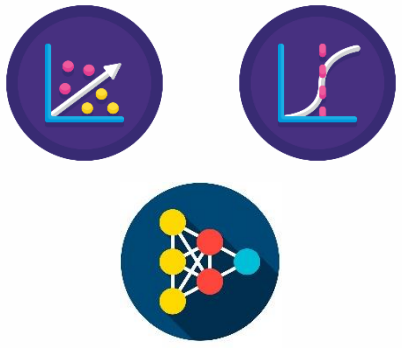
Hyperparameter Tuning



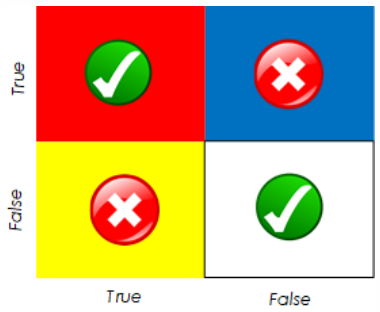
GridSearchCV



RandomizedSearchCV

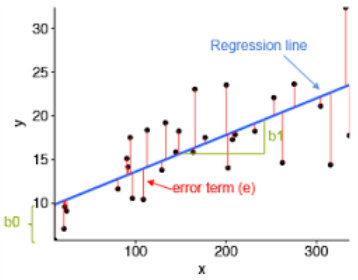


Model Selection



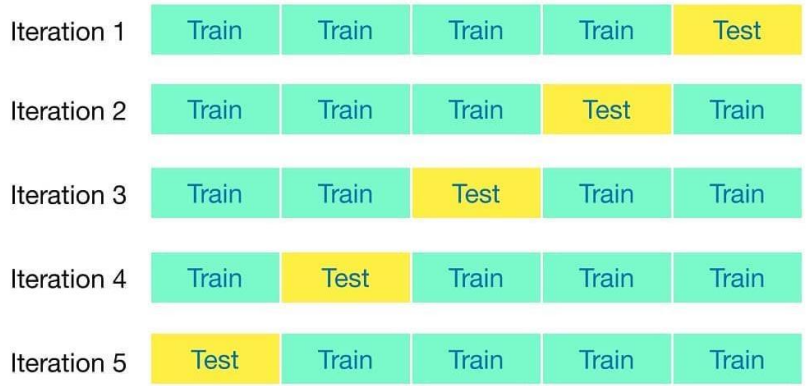
Accuracy & Confusion Matrix

Precision, Recall, F1 Score



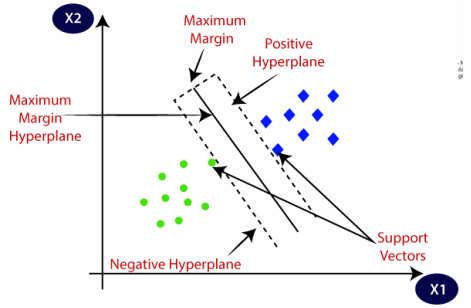
Metrics for Regression

K-Fold Cross-Validation

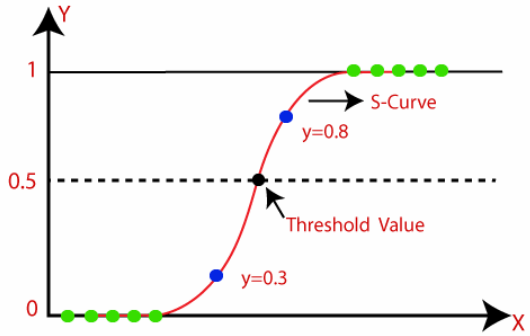


K-Fold Cross-Validation

In K-Fold Cross Validation, we split the dataset into **"K"** number of **folds** (subsets). One chunk of data is used as test data for evaluation & the remaining part of the data is used for training the model. Each time, a different chunk will be used as the test data.

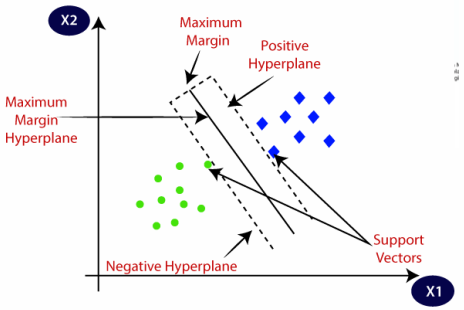


Support Vector Machine



Logistic Regression

K-Fold Cross-Validation



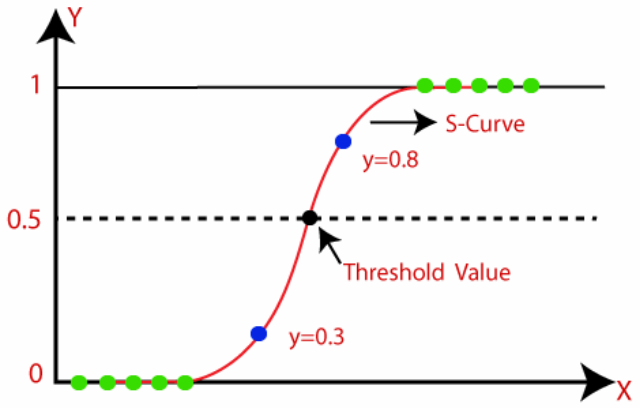
Support Vector Machine

K = 5

	Dataset					*Accuracy*
Iteration 1	Train	Train	Train	Train	Test	88%
Iteration 2	Train	Train	Train	Test	Train	83%
Iteration 3	Train	Train	Test	Train	Train	86%
Iteration 4	Train	Test	Train	Train	Train	81%
Iteration 5	Test	Train	Train	Train	Train	84%

$$\text{Mean Accuracy} = \frac{88 + 83 + 86 + 81 + 84}{5} = 84.4\ \%$$

K-Fold Cross-Validation



Logistic Regression

K = 5

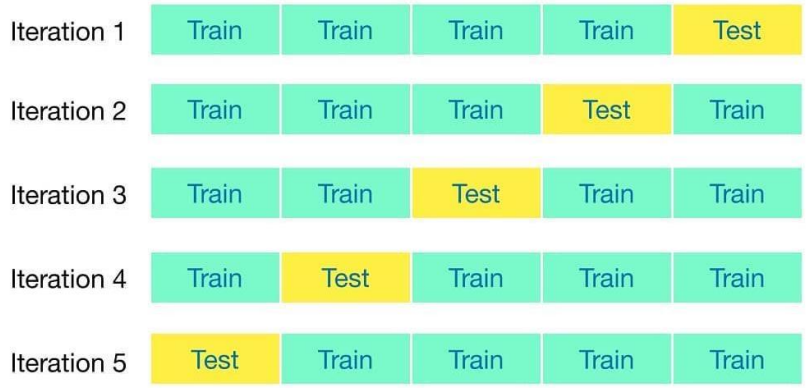
	Dataset					*Accuracy*
Iteration 1	Train	Train	Train	Train	Test	90%
Iteration 2	Train	Train	Train	Test	Train	88%
Iteration 3	Train	Train	Test	Train	Train	86%
Iteration 4	Train	Test	Train	Train	Train	91%
Iteration 5	Test	Train	Train	Train	Train	85%

$$\text{Mean Accuracy} = \frac{90 + 88 + 86 + 91 + 85}{5} = 88\ \%$$

K-Fold Cross-Validation

✓ *Accuracy score for SVM = 84.4 %*

✓ *Accuracy score for Logistic Regression = 88 %*

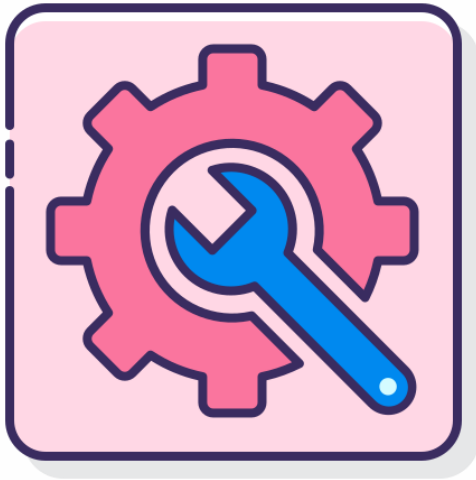


Advantages of using K-Fold Cross-validation:

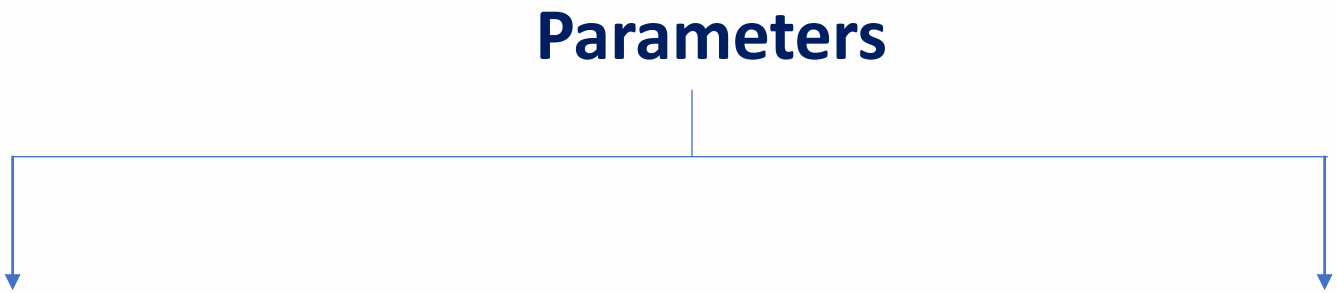
- Better alternative for train-test split when the dataset is small
- Better for multiclass classification problems
- More reliable
- Useful for Model Selection

Hyperparameter Tuning:

- GridSearchCV
- RandomizedSearchCV



Types of Parameters



Model Parameters

These are the parameters of the model that can be determined by training with training data. These can be considered as internal Parameters.

- **Weights**
- **Bias**

Y = w*X + b

Hyperparameters

Hyperparameters are parameters whose values control the learning process. These are adjustable parameters used to obtain an optimal model. External Parameters.

- **Learning rate**
- **Number of Epochs**
- **n_estimators**

Hyperparameter Tuning

Best Hyperparameters

Hyperparameter Tuning

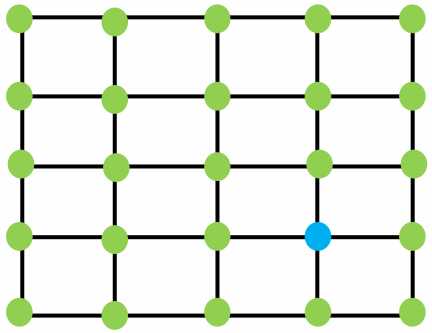
Best Model Parameters

Model Training

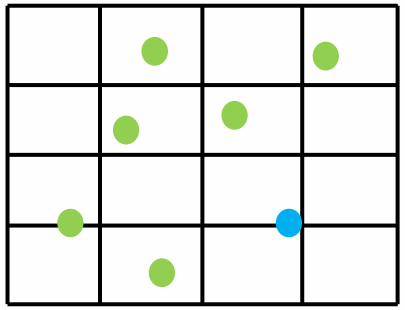
Hyperparameter Tuning

Hyperparameter Tuning refers to the process of choosing the optimum set of hyperparameters for a Machine Learning model. This process is also called **Hyperparameter Optimization.**

Hyperparameter Tuning Types:



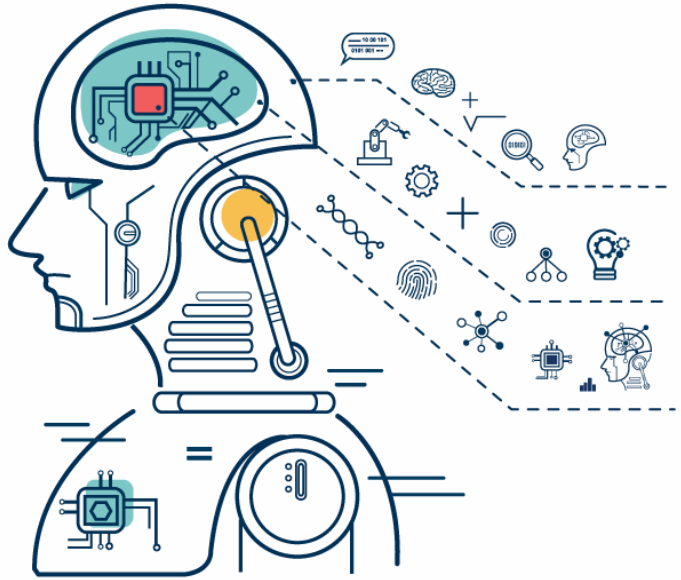
GridSearchCV



RandomizedSeaechCV

Support Vector Classifier:
C: [1,5,10]
kernel: (‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’)

Model Selection in Machine Learning

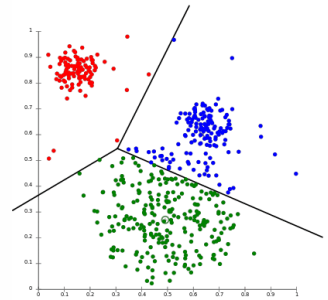


Model Selection

Model Selection in Machine Learning is the process of choosing the best suited model for a particular problem. Selecting a model depends on various factors such as the dataset, task, nature of the model, etc.

Two factors to be considered:

1. Logical Reason to select a model
2. Comparing the performance of the models

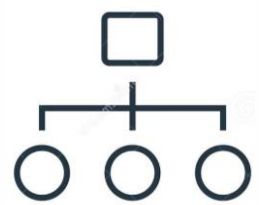


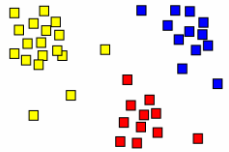
Model Selection

Models can be selected based on :

1. **Type of Data available:**
   a. Images & Videos – CNN
   b. Text data or Speech data – RNN
   c. Numerical data – SVM, Logistic Regression, Decision trees, etc.

2. **Based on the task we need to carry out:**
   a. Classification tasks – SVM, Logistic Regression, Decision trees, etc.
   b. Regression tasks – Linear regression, Random Forest, Polynomial regression, etc.
   c. Clustering tasks – K-Means Clustering, Hierarchical Clustering

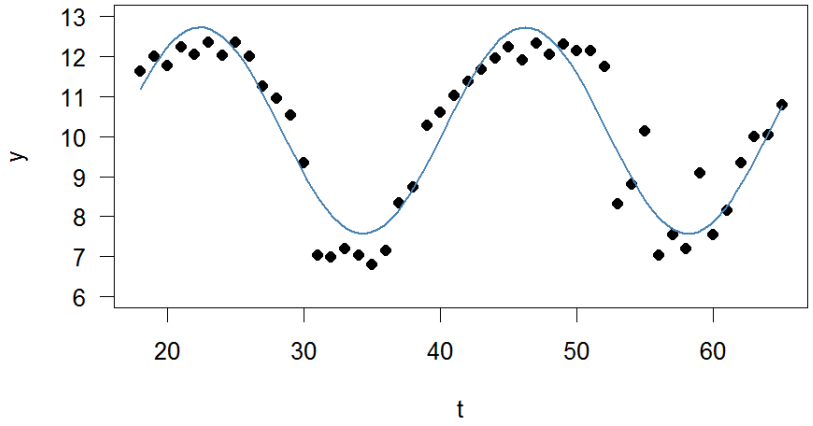




Linear Regression

Advantages:

1. Very simple to implement
2. Performs well on data with linear relationship



Disadvantages:

1. Not suitable for data having non-linear relationship
2. Underfitting issue
3. Sensitive to Outliers

Logistic Regression

Advantages:

1. Easy to implement
2. Performs well on data with linear relationship
3. Less prone to over-fitting for low dimensional dataset

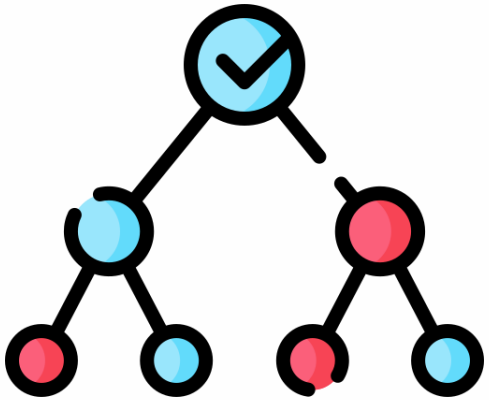
Disadvantages:

1. High dimensional dataset causes over-fitting
2. Difficult to capture complex relationships in a dataset
3. Sensitive to Outliers
4. Needs a larger dataset

Decision Tree

Advantages:

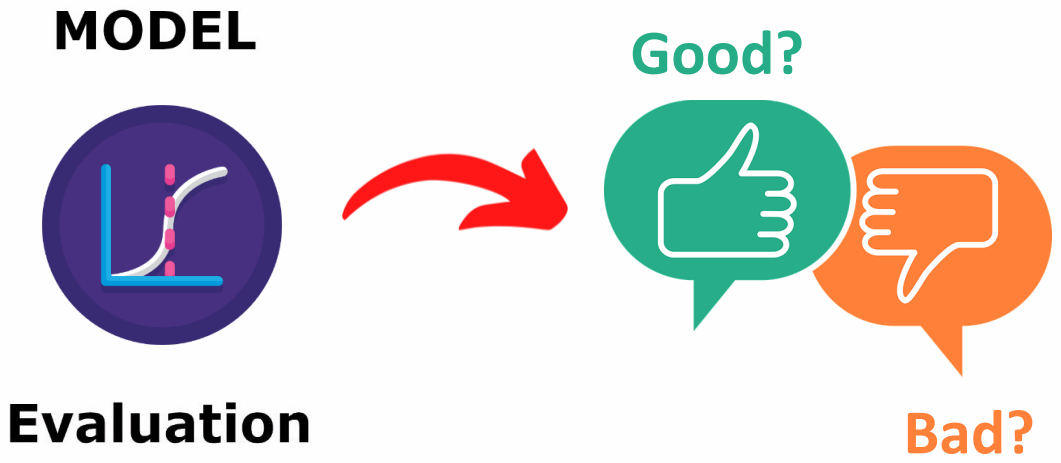
1. Can be used for both Classification & Regression
2. Easy to interpret
3. No need for normalization or scaling
4. Not sensitive to outliers



Disadvantages:

1. Overfitting issue
2. Small changes in the data alter the tree structure causing instability
3. Training time is relatively higher

Precision, Recall, & F1 Score



Accuracy Score

In Classification, **Accuracy Score** is the ratio of **number of correct predictions** to the **total number of input data points**.

$$\text{Accuracy Score} = \frac{\text{Number of correct predictions}}{\text{Total Number of data points}} \times 100\ \%$$

Number of correct predictions = 128

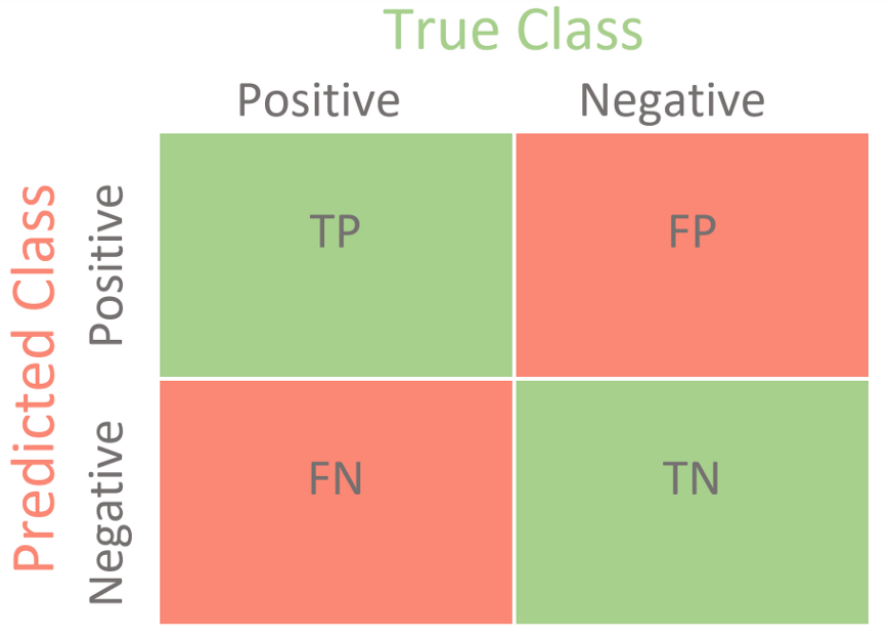
Total Number of data points = 150

Accuracy Score = 85.3 %

```
from sklearn.metrics import accuracy_score
```

Confusion Matrix

Confusion Matrix is a matrix used for evaluating the performance of a Classification Model. It gives more information than the accuracy score.



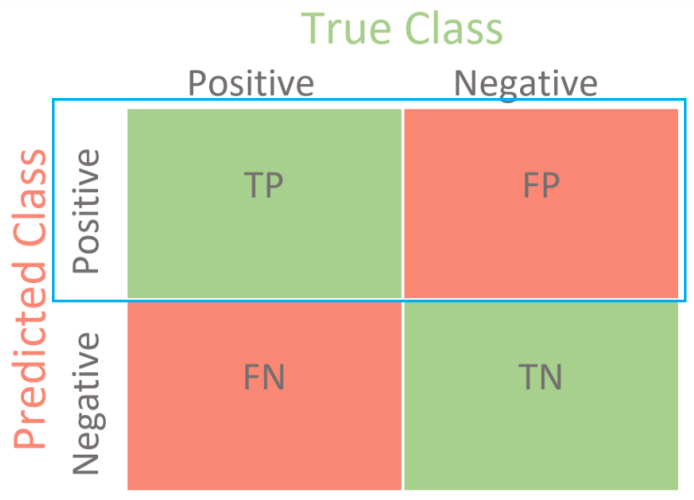
TP + TN = Correct Predictions

FP + FN = Wrong Predictions

sklearn.metrics.confusion_matrix

Precision

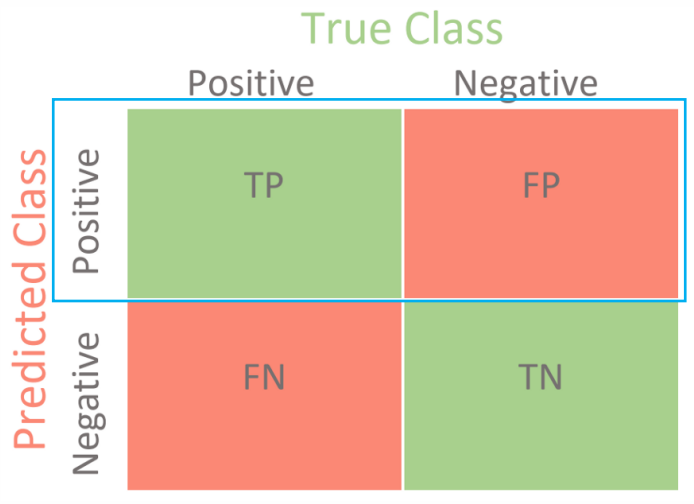
$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad \Rightarrow \quad \text{Precision} = \frac{\text{True Positive}}{\text{Total Predicted Positive}}$$



Precision is the ratio of number of **True Positive** to the **total number of Predicted Positive**. It measures, out of the total predicted positive, how many are actually positive.

Precision

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \textbf{False Positive}} \quad \Rightarrow \quad \text{Precision} = \frac{\text{True Positive}}{\text{Total Predicted Positive}}$$



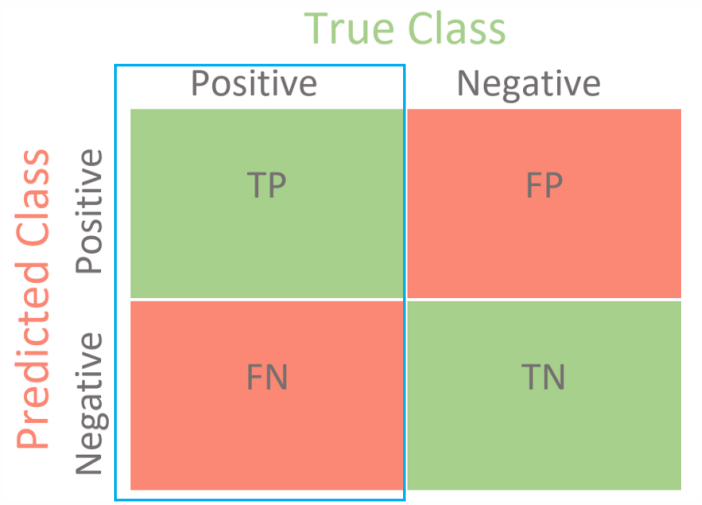
Precision is the ratio of number of **True Positive** to the **total number of Predicted Positive**. It measures, out of the total predicted positive, how many are actually positive.

Precision measures the error caused by **False Positives**. Hence it is a good evaluation metric when **False Positive** predictions are critical.

Example: Face Authentication

Recall

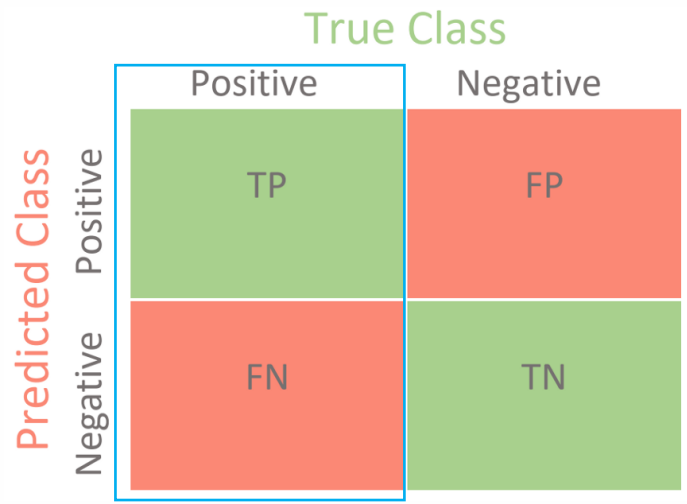
$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad \Rightarrow \quad \text{Recall} = \frac{\text{True Positive}}{\text{Total Actual Positive}}$$



Recall is the ratio of number of **True Positive** to the **total number of Actual Positive**. It measures, out of the total actual positive, how many are predicted as True Positive.

Recall

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \textbf{False Negative}} \quad \Rightarrow \quad \text{Recall} = \frac{\text{True Positive}}{\text{Total Actual Positive}}$$



Recall is the ratio of number of **True Positive** to the **total number of Actual Positive**. It measures, out of the total actual positive, how many are predicted as True Positive.

Recall measures the error caused by **False Negatives**. Hence it is a good evaluation metric when **False Negative** predictions are critical.

Example: Cancer Diagnosis

F1 Score

F1 Score is an important evaluation metric for binary classification that combines Precision & Recall. F1 Score is the **harmonic mean** of Precision & Recall.

This is a very useful metric when a dataset has imbalanced classes.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision, Recall & F1 Score

Example:

		Predicted	
		Positive	Negative
Actual	Positive	TP = 50	FN = 10
	Negative	FP = 5	TN = 20

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} = \frac{50}{50 + 5}$$

Precision = 0.91

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{50}{50 + 10}$$

Recall = 0.83

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.91 \times 0.83}{0.91 + 0.83}$$

F1 Score = 0.87