Supervised Learning

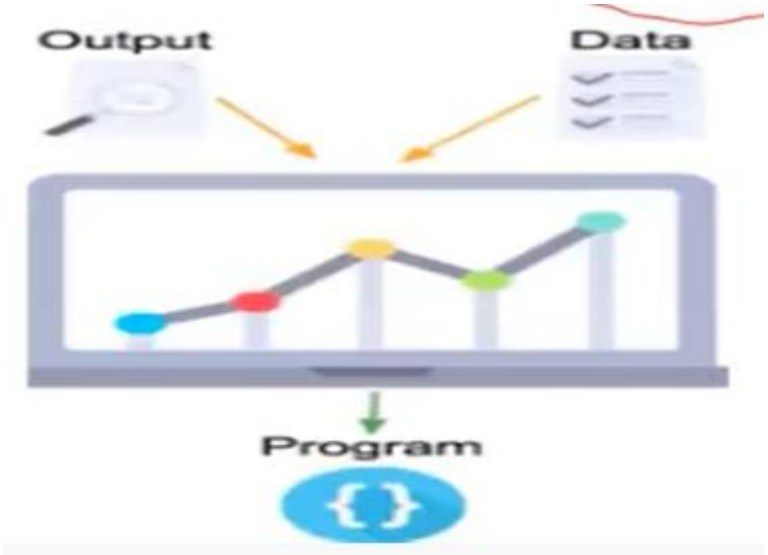
- Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output.

- The labelled data means some input data is already tagged with the correct output.

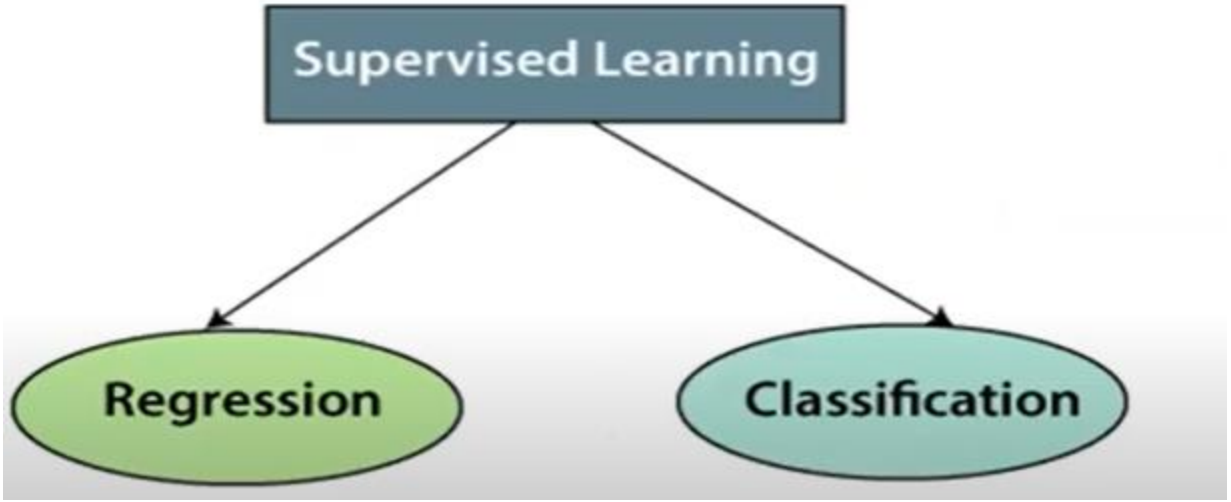
- In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly.

Supervised Learning

- Supervised learning is a process of providing input data as well as correct output data to the machine learning model.
- The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y).**



Types of Supervised Learning



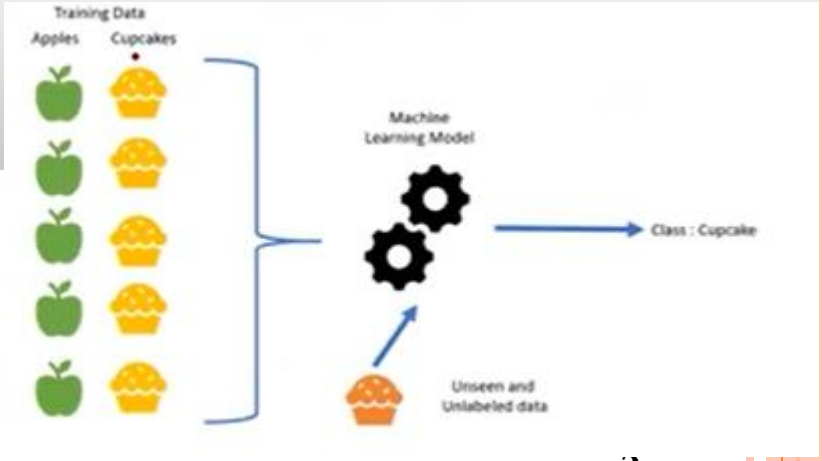
Regression

- Regression algorithms are algorithms are used when the output variable is continuous variables, such as Weather forecasting, Market Trends, etc.

  - Linear Regression
  - Regression Trees
  - Non-Linear Regression
  - Bayesian Linear Regression
  - Polynomial Regression, etc.

Classification

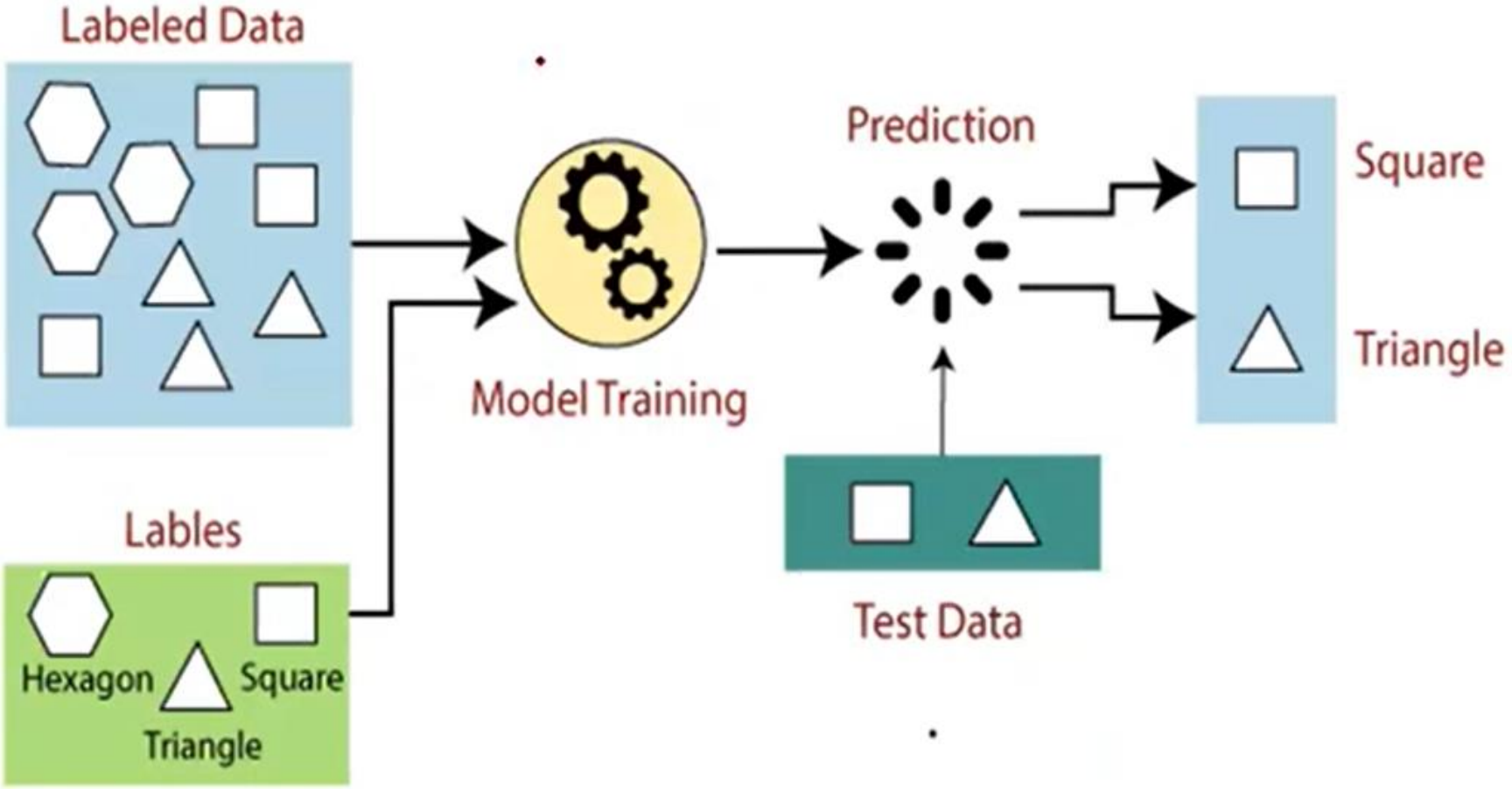
- Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.
  - Naïve Bayes
  - Random Forest
  - Decision Trees
  - Logistic Regression
  - Support vector Machines etc



How Supervised Learning Algorithm works

- First determine the type of problem
- Collect the labelled data.
- Split the dataset into **training (80%)** and **testing (20%).**
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

How Supervised Learning Algorithm works



Evaluate classification and regression predictions

- Classification predictions can be evaluated using accuracy, whereas regression predictions cannot.

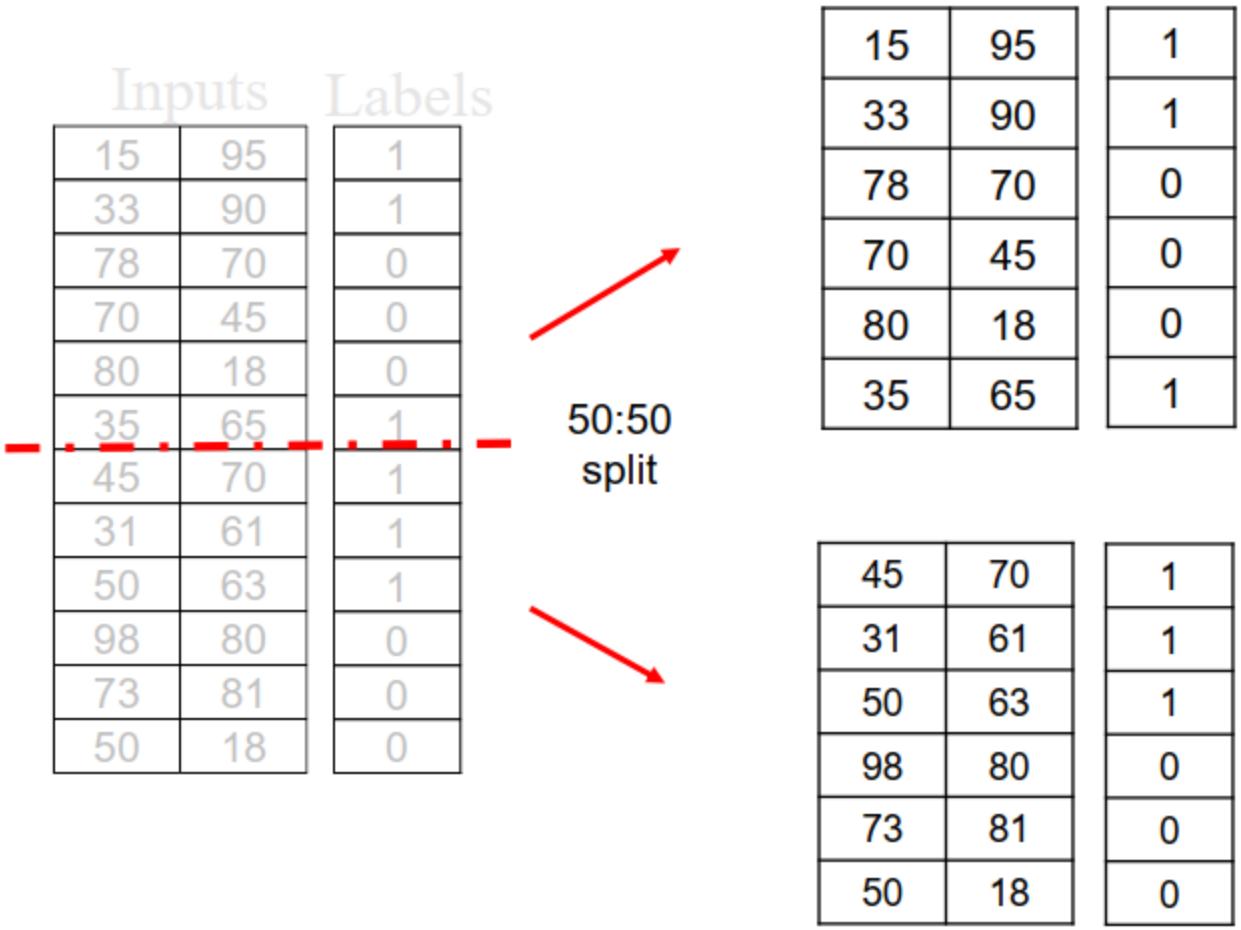
- Regression predictions can be evaluated using root mean squared error, whereas classification predictions cannot.

Splitting of Training and test Data

- We need independent data sets to train, set parameters, and test performance
- Thus we will often divide a data set into three
  - Training set
  - Parameter selection set
  - Test set
- These **must** be independent
- Data set 2 is not always necessary

Dataset

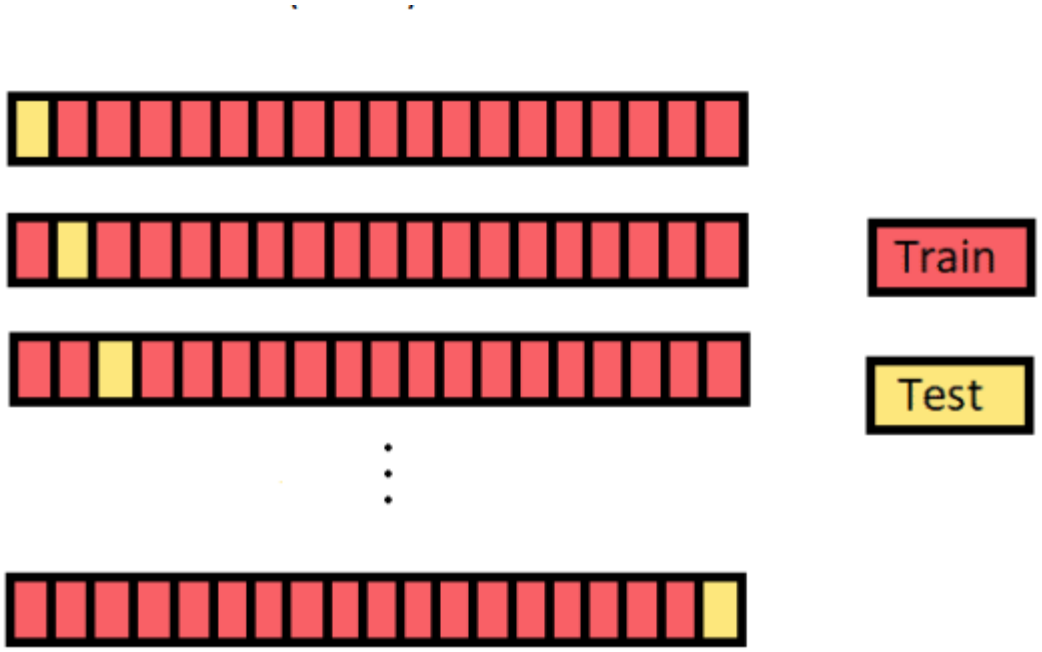
Inputs		Labels
15	95	1
33	90	1
78	70	0
70	45	0
80	18	0
35	65	1
45	70	1
31	61	1
50	63	1
98	80	0
73	81	0
50	18	0


Inputs
Labels
15 95 1
33 90 1
78 70 0
70 45 0
80 18 0
35 65 1
45 70 1
31 61 1
50 63 1
98 80 0
73 81 0
50 18 0
50:50
split
15 95 1
33 90 1
78 70 0
70 45 0
80 18 0
35 65 1
45 70 1
31 61 1
50 63 1
98 80 0
73 81 0
50 18 0

Leave One Out Cross Validation

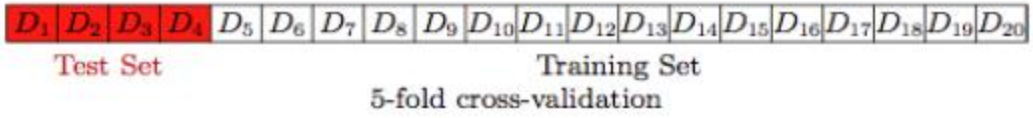
- In this approach, we reserve only one data point from the available dataset, and train the model on the rest of the data. This process iterates for each data point. This also has its own advantages and disadvantages. Let's look at them:
- We make use of all data points, hence the bias will be low
- We repeat the cross validation process n times (where n is number of data points) which results in a higher execution time
- This approach leads to higher variation in testing model effectiveness because we test against one data point. So, our estimation gets highly influenced by the data point. If the data point turns out to be an outlier, it can lead to a higher variation

LOOCV

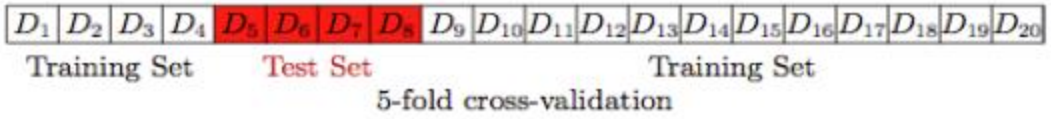


K-Fold Cross validation

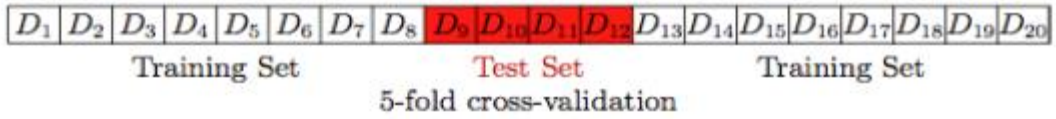
- Randomly split your entire dataset into k"folds"
- For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for *kth* fold
- Record the error you see on each of the predictions
- Repeat this until each of the k-folds has served as the test set
- The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the mode



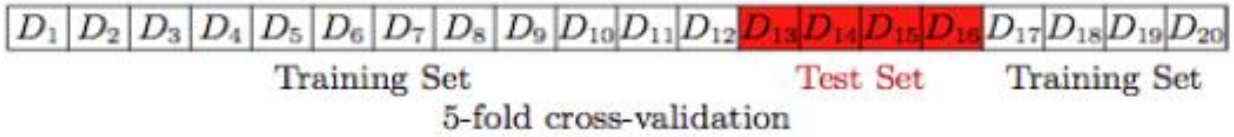
$E_g =$ 5.1



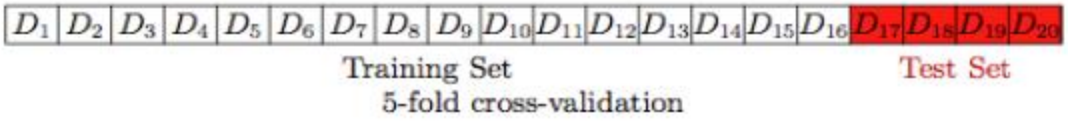
$E_g =$ 3.7



$E_g =$ 4.6



$E_g =$ 4.6



$E_g =$ 3.3

$$\langle E_g \rangle = \frac{5.1 + 3.7 + 4.6 + 4.6 + 3.3}{5} = 4.3$$

K Nearest Neighbor Classification

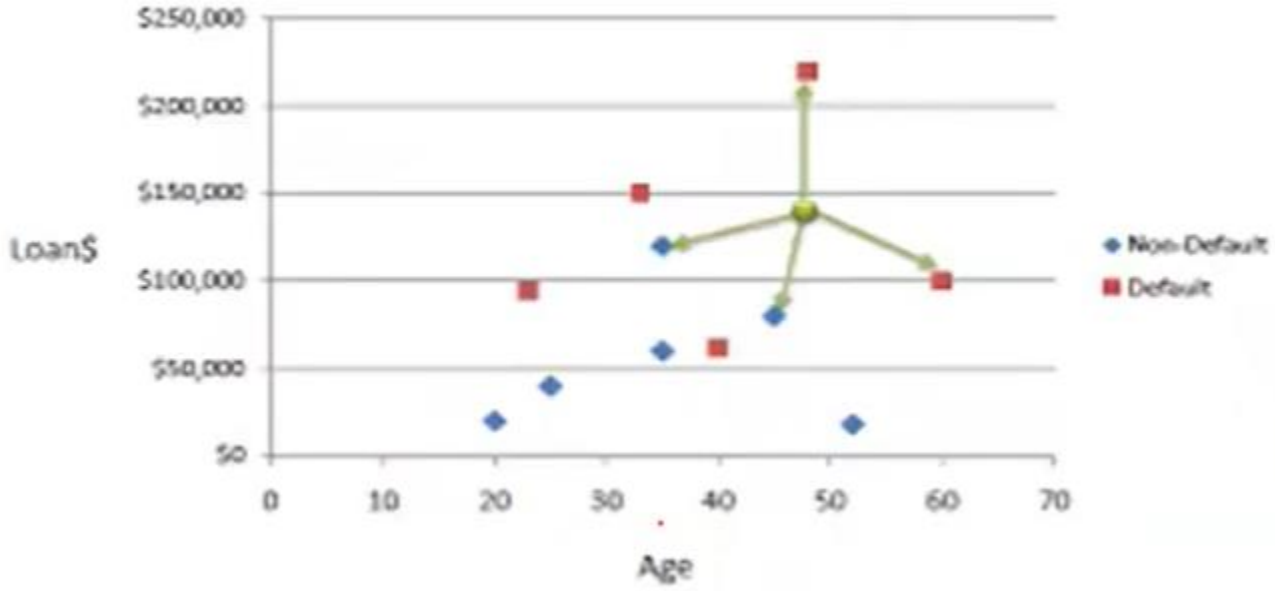
K Nearest Neighbor Classification (lazy Learner)

- A powerful classification algorithm used in pattern recognition.
- K nearest neighbors stores all available cases and classifies new cases based on a *similarity measure*(e.g **distance function**)
- One of the top data mining algorithms used today.
- A non-parametric lazy learning algorithm (An Instance-based Learning method).

KNN Classification Approach

- An object (a new instance) is classified by a majority votes for its neighbor classes.
- The object is assigned to the most common class amongst its K nearest neighbors.(*measured by a distant function*)

KNN Classification Approach



KNN Classification Algorithm

Step 1 – For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 – Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 – For each point in the test data do the following –

- **3.1** – Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- **3.2** – Now, based on the distance value, sort them in ascending order.
- **3.3** – Next, it will choose the top K rows from the sorted array.
- **3.4** – Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 – End

KNN Classification Algorithm – example

X1	X2	Actual Class	Manhattan Dist	K=1	K=3	K=5
1	6	0	|1-4|+||6-2| =7			
2	4	0	|2-4|+||4-2| =4			
3	7	0	|3-4|+||7-2| =6			
6	8	1	|6-4|+||8-2| =8			
7	1	1	|7-4|+||1-2| =4			
8	4	1	|8-4|+||4-2| =6			

Input: 4,2

KNN Regression- Example

X1	X2	Actual Values	Manhattan Dist	K=1	K=3
1	6	7	|1-4|+||6-2| =7	(8+50)/2=29	(8+16=50=68)/4=
2	4	8	|2-4|+||4-2| =4		
3	7	16	|3-4|+||7-2| =6		
6	8	44	|6-4|+||8-2| =8		
7	1	50	|7-4|+||1-2| =4		
8	4	68	|8-4|+||4-2| =6		

Input: 4,2

KNN - Example

Example : Classify whether a customer will respond to a survey question using a 3-Nearest Neighbor classifier

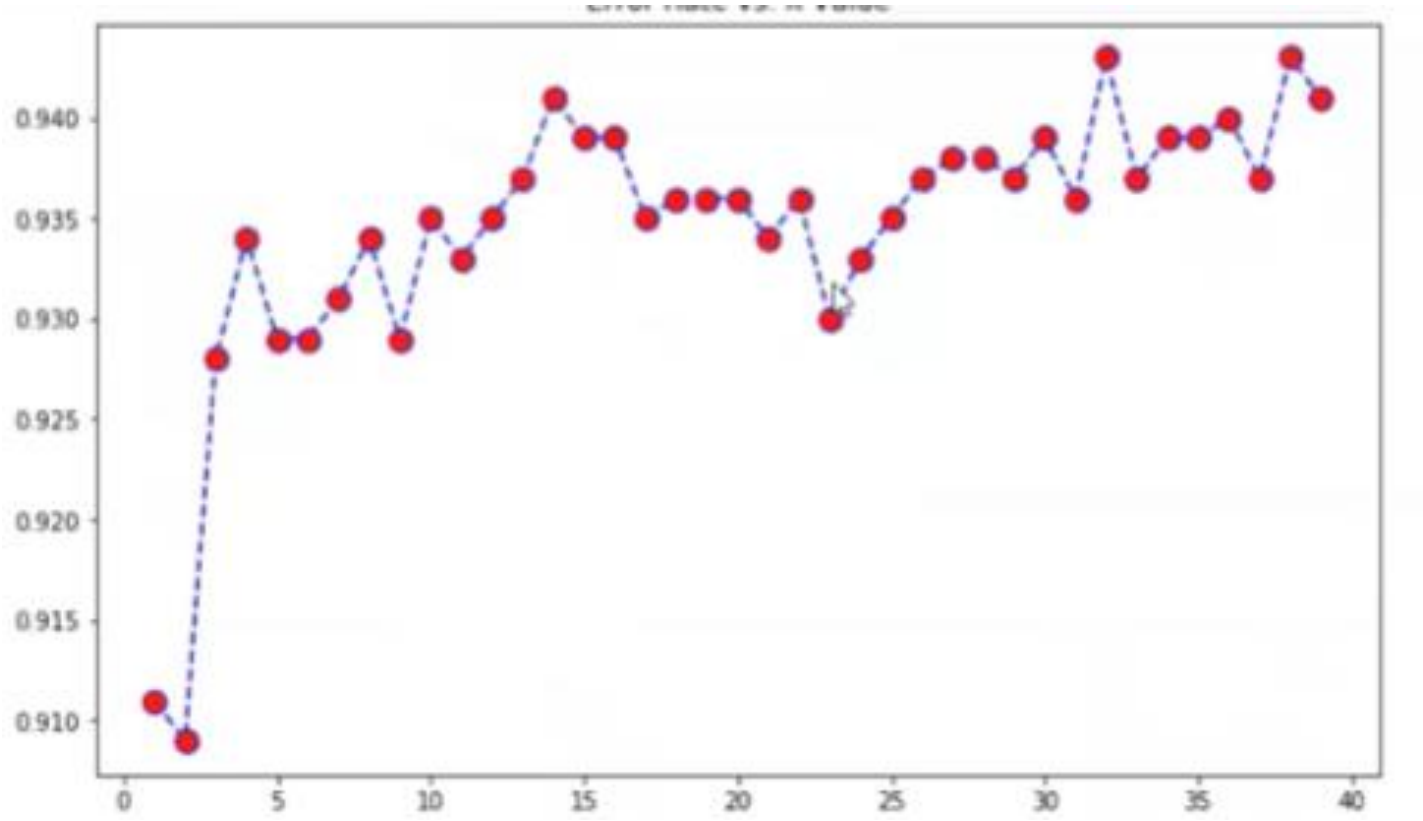
Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

Picking K

- Use N fold cross validation – Pick K to minimize the cross validation error

- For each of N training example
  - Find its K nearest neighbours
  - Make a classification based on these K neighbours
  - Calculate classification error
  - Output average error over all examples
  - Use the K that gives lowest average error over the N training examples

Picking K



Pros and Cons of KNN

Pros

- It is very simple algorithm to understand and interpret.
- It is very useful for nonlinear data because there is no assumption about data in this algorithm.
- It is a versatile algorithm as we can use it for classification as well as regression.
- It has relatively high accuracy but there are much better supervised learning models than KNN.

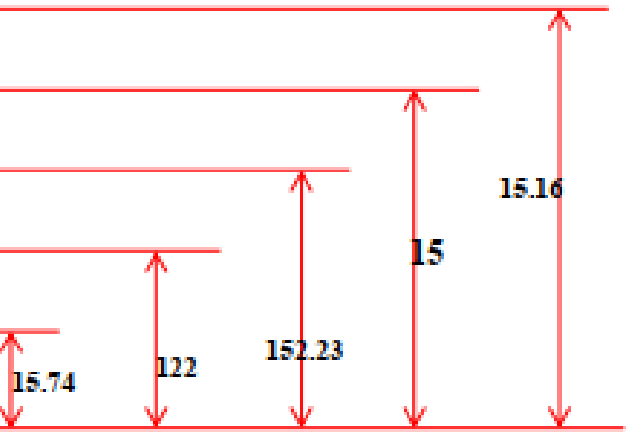
Cons

- It is computationally a bit expensive algorithm because it stores all the training data.
- High memory storage required as compared to other supervised learning algorithms.
- Prediction is slow in case of big N.
- It is very sensitive to the scale of data as well as irrelevant features.

KNN - EXAMPLE

- Example : 3-Nearest Neighbors

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?



Some of the above calculated values are wrong.