Machine learning algorithms

Regression

SVM

Naïve bayes

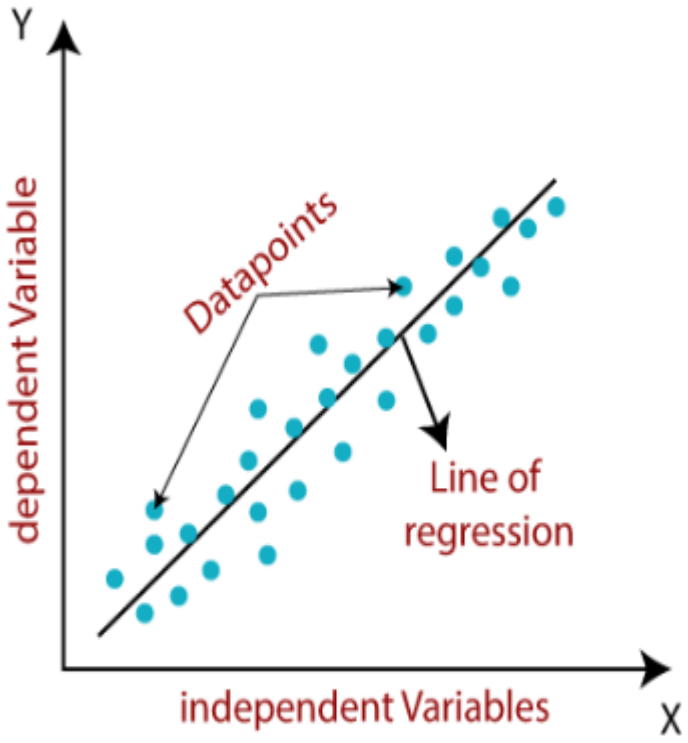
KNN

ANN

Decision trees

Linear regression

- Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression.



$$y= a_0+a_1x+ \varepsilon$$

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a0= intercept of the line (Gives an additional degree of freedom)

a1 = Linear regression coefficient (scale factor to each input value).

ε = random error

The values for x and y variables are training datasets for Linear Regression model representati

Mathematically, we can represent a linear regression as:

$$y= a_0+a_1x+ \varepsilon$$

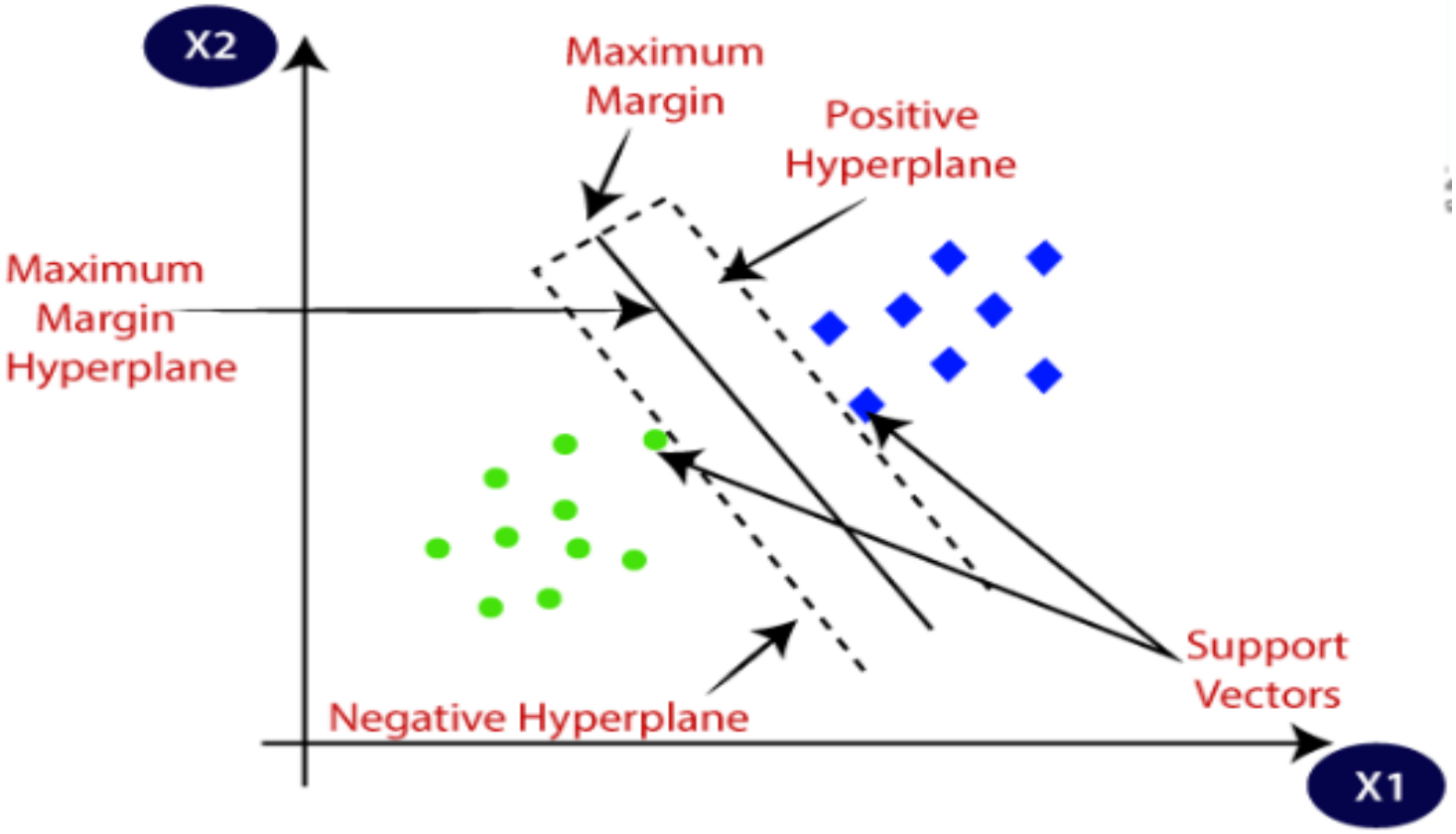
Types

- Simple linear regression (one predictor)
- Multiple linear regression (multiple predictor)

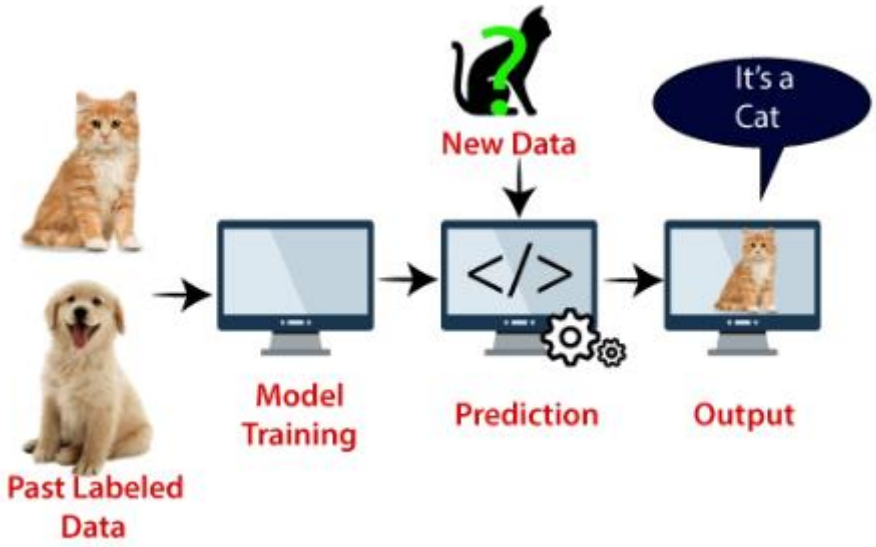
Support vector machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.


X2
Maximum
Margin
Positive
Hyperplane
Maximum
Margin
Hyperplane
Support
Vectors
Negative Hyperplane
X1

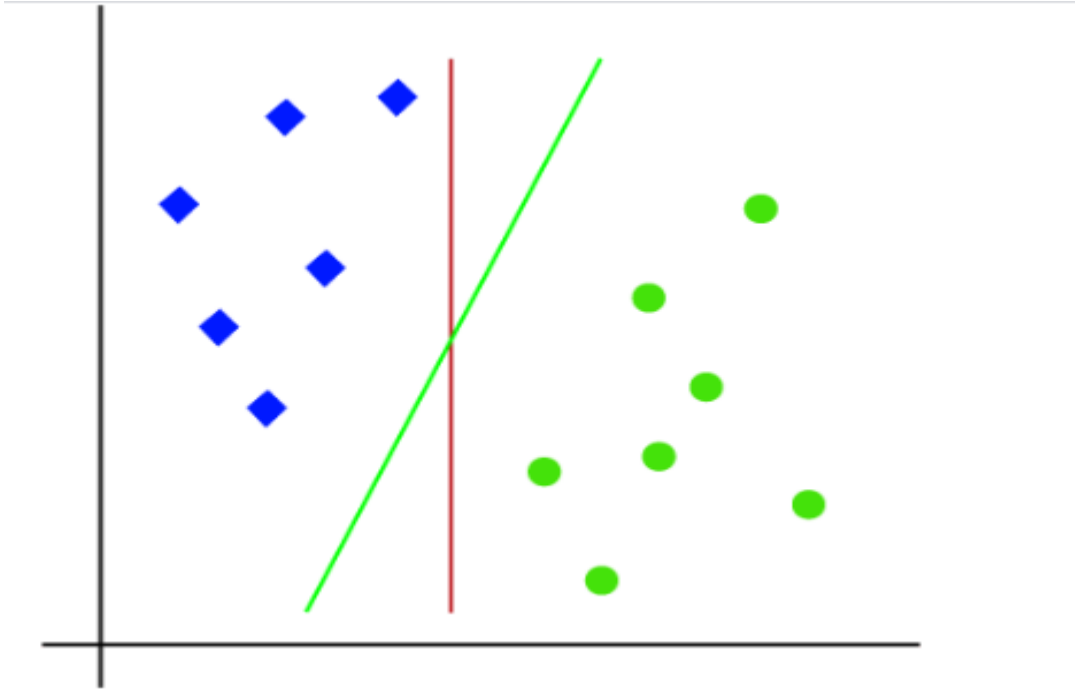
- SVM algorithm can be used for **Face detection, image classification, text categorization,**



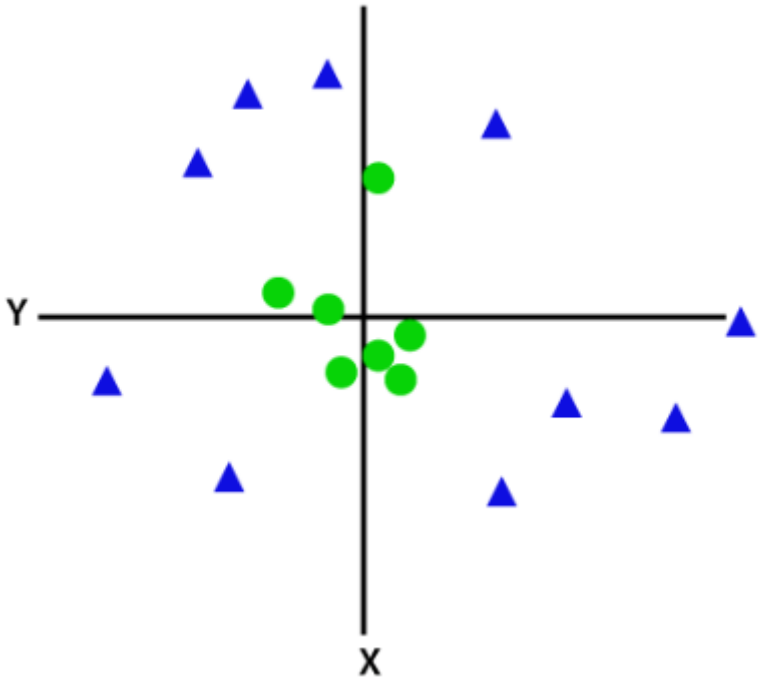
Types

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Linear SVM



Non linear SVM



Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

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

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

NB example

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

- Convert the given dataset into frequency tables.
- Generate Likelihood table by finding the probabilities of given features.
- Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying the theorem

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

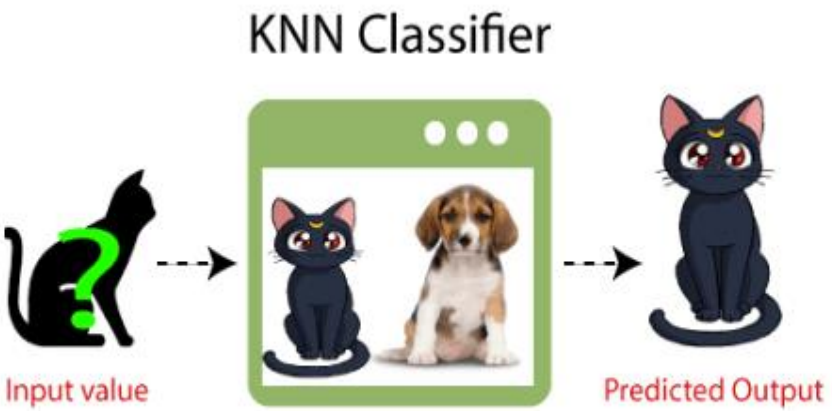
- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

K-Nearest Neighbor(KNN) Algorithm

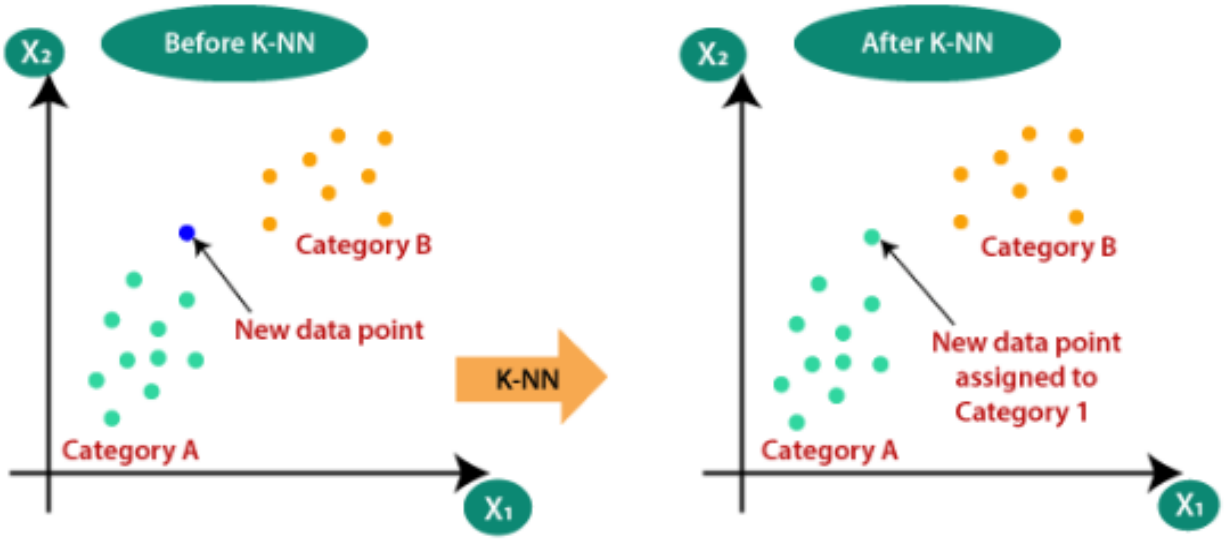
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. The KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1

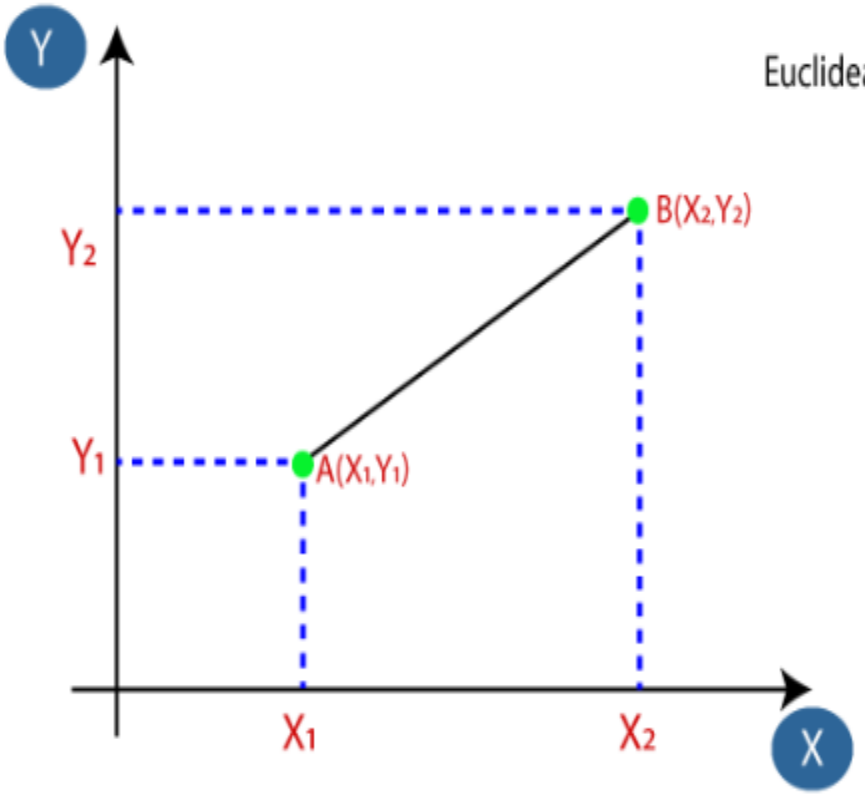


Working of KNN

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.

Firstly, we will choose the number of neighbors, so we will choose the k=5.

Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



$$\text{Euclidean Distance between } A_1 \text{ and } B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

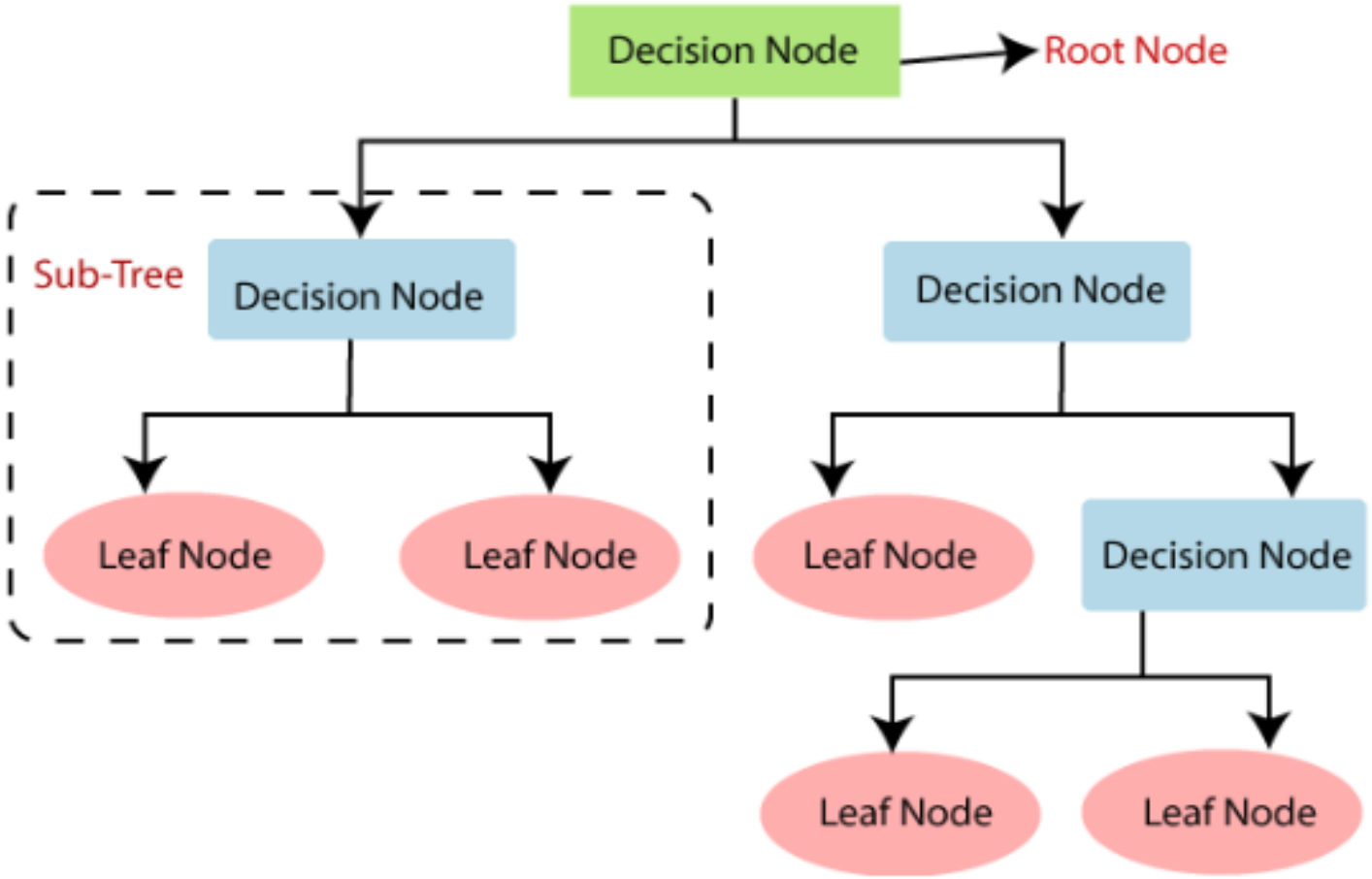
- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- Eg J48.

Tree building

- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.


Decision Node
Root Node
Sub-Tree
Decision Node
Decision Node
Leaf Node
Leaf Node
Leaf Node
Decision Node
Leaf Node
Leaf Node

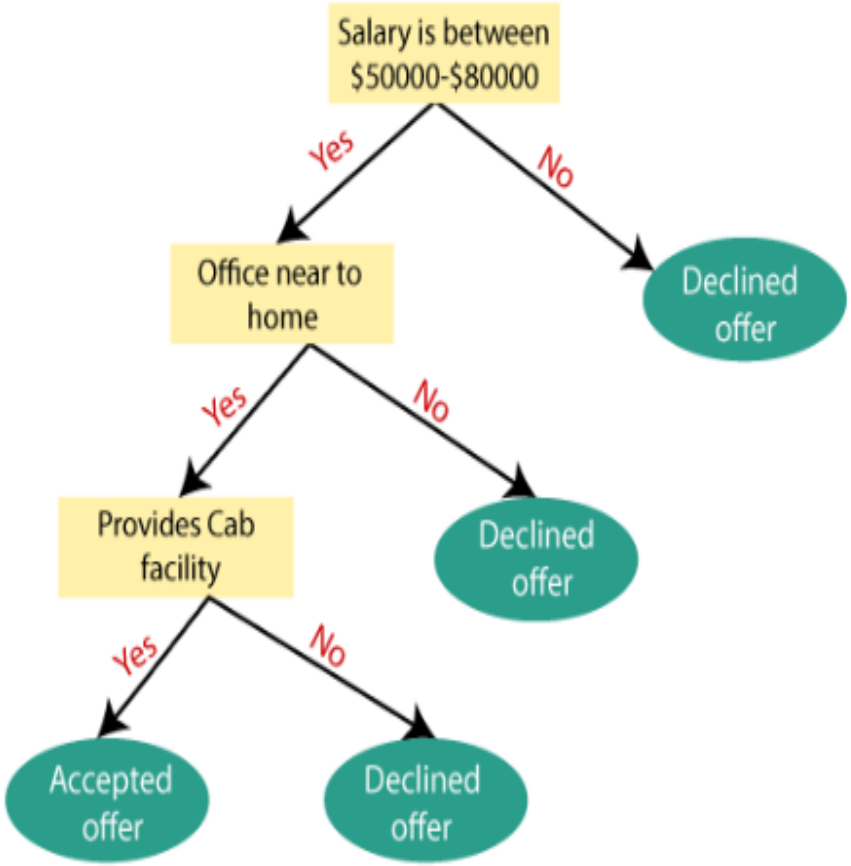
How it works

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree.

Steps

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures(ASM)

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**
- By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:
  - **Information Gain**
  - **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

> Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$Entropy(s)= -P(yes)log2\ P(yes)- P(no)\ log2\ P(no)$$

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

K means Clustering

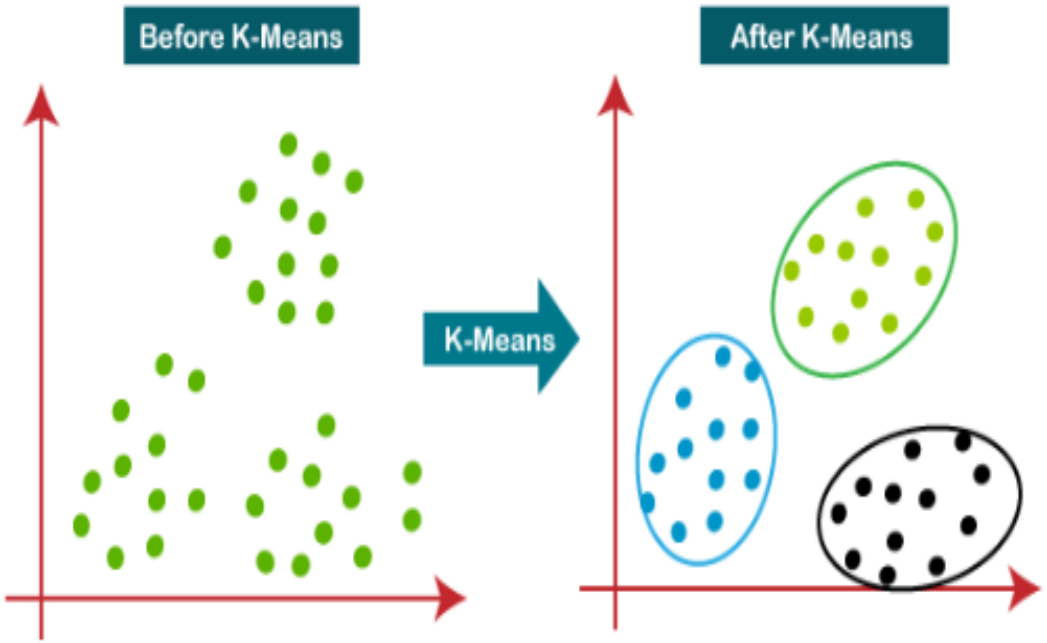
- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How it works

- **Step-1:** Select the number K to decide the number of clusters.
- **Step-2:** Select random K points or centroids. (It can be other from the input dataset).
- **Step-3:** Assign each data point to their closest centroid, which will form the predefined K clusters.
- **Step-4:** Calculate the variance and place a new centroid(average) of each cluster.
- **Step-5:** Repeat the third steps, which means reassign each data point to the new closest centroid of each cluster.
- **Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.
- **Step-7**: The model is ready.

Hands on in python

IRIS dataset.

Problem: *predicting the species of a flower based on the petal/sepal* width and size

Iris Versicolor

Iris Setosa

Iris Virginica

Iris data set

The numeric parameters which the dataset contains are **Sepal width**, **Sepal length**, **Petal width and Petal length**. In this data we will be predicting the species of the flowers based on these parameters. We will be building a machine learning project to determine the species of the flower.

NB. **This is not a image-classifier algorithm**

Data set extract

Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)	Class
7	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor
4.9	2.4	3.3	1	Iris-versicolor
6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor
5	2	3.5	1	Iris-versicolor
5.9	3	4.2	1.5	Iris-versicolor
6	2.2	4	1	Iris-versicolor
6.1	2.9	4.7	1.4	Iris-versicolor
5.6	2.9	3.6	1.3	Iris-versicolor
6.7	3.1	4.4	1.4	Iris-versicolor
5.6	3	4.5	1.5	Iris-versicolor
5.8	2.7	4.1	1	Iris-versicolor
6.2	2.2	4.5	1.5	Iris-versicolor
5.6	2.5	3.9	1.1	Iris-versicolor
5.9	3.2	4.8	1.8	Iris-versicolor
6.1	2.8	4	1.3	Iris-versicolor

load ML libraries and algo

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

Loading the data using pandas

```
url = "/iris.data.txt"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = pd.read_csv(url, names=names)
```

Summarize the data and perform analysis

- Dimensions of data set: Find out how many rows and columns our dataset has using the shape property

  ```
  # shape
  print(dataset.shape)
  ```

Result: (150,5), Which means our dataset has 150 rows and 5 columns

To check the first 20 rows of our dataset

print(dataset.head(20))

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa

statistical summary

```
print(dataset.describe())
```

Result:

```
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

Class distribution

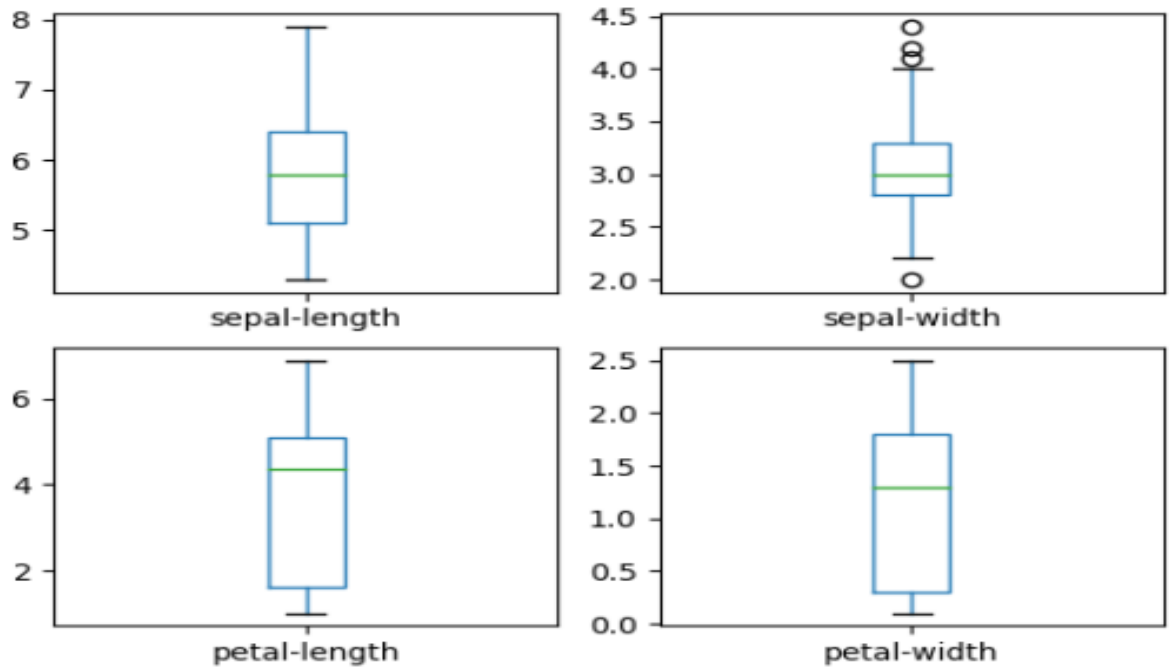
```
#class distribution
print(dataset.groupby('class').size())
```

```
Result:
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Which implies our dataset has an even distribution of 50 records for each species type.

Visual data analysis

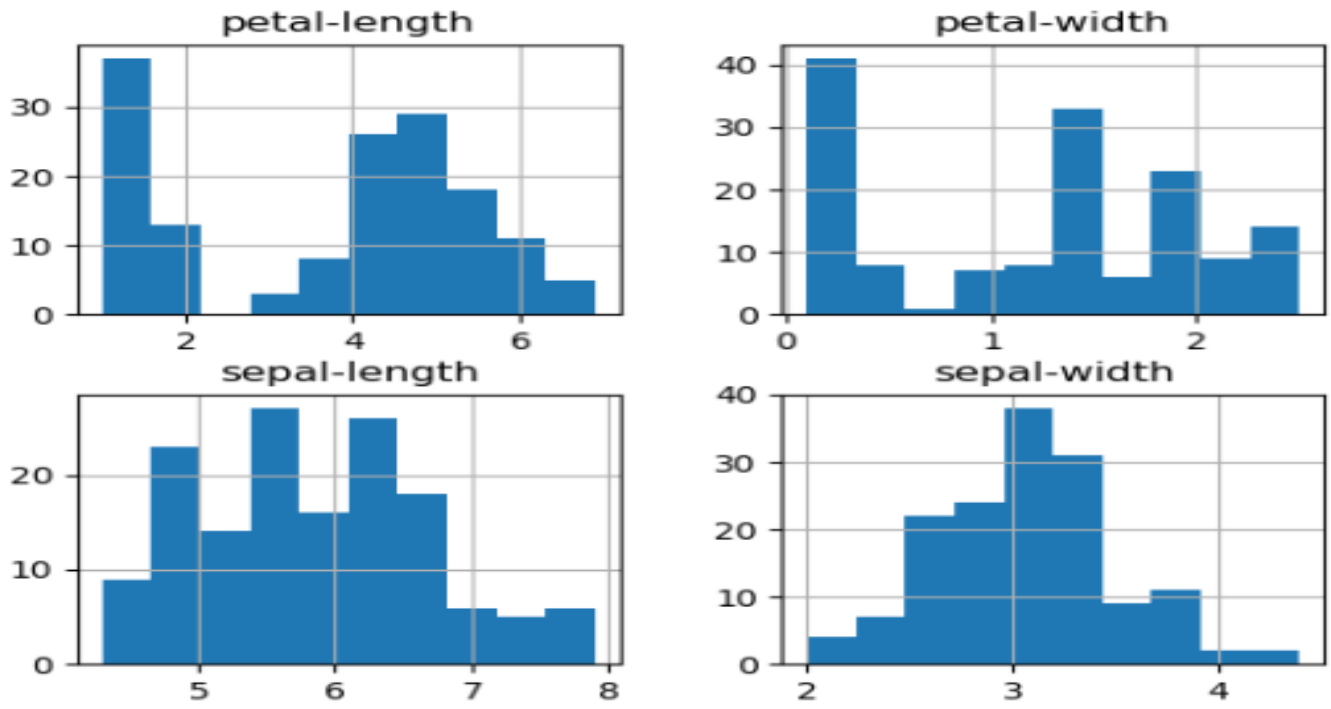
```
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```



Box plot is a percentile-based graph, which divides the data into four quartiles of 25% each

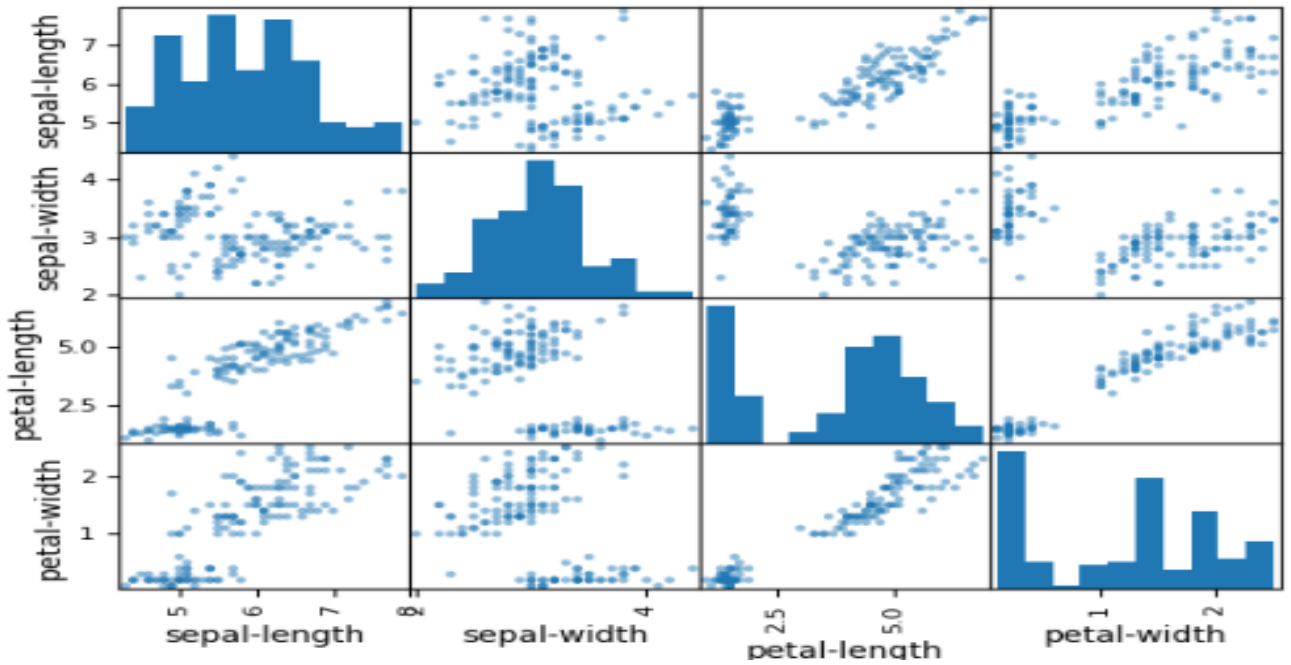
Histogram

```
# histograms
dataset.hist()
plt.show()
```



Scatter plot

```
# scatter plot matrix
from pandas.plotting import scatter_matrix
scatter_matrix(dataset)
plt.show()
```



shows correlation with respect to other features

Splitting the Data

- Two data set

  -Training dataset

  -**Testing dataset**

  Since the the dataset is small (150 records) , we will use 120 (80%) records for training the model and 30 (20%) records to evaluate the model.

```
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
x_train, x_test, y_train, y_test = model_selection.train_test_split(X, Y, test_size=0.2,
random_state=7)
```

Evaluating the model and training the Model

- K – Nearest Neighbour (KNN)
- Support Vector Machine (SVM)
- Random forest
- Logistic Regression

KNN Model

```
model = KNeighborsClassifier()
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(accuracy_score(y_test, predictions))
```

SVM

```
model = SVC()
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(accuracy_score(y_test, predictions))
```

Random forest

```
model = RandomForestClassifier(n_estimators=5)
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(accuracy_score(y_test, predictions))
```

Logistic regression

```
model = LogisticRegression()
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(accuracy_score(y_test, predictions))
```

Model accuracy

K – Nearest Neighbour (KNN) = 0.9

Support Vector Machine (SVM)= 0.93333333

Randomforest=0.8666666666666667

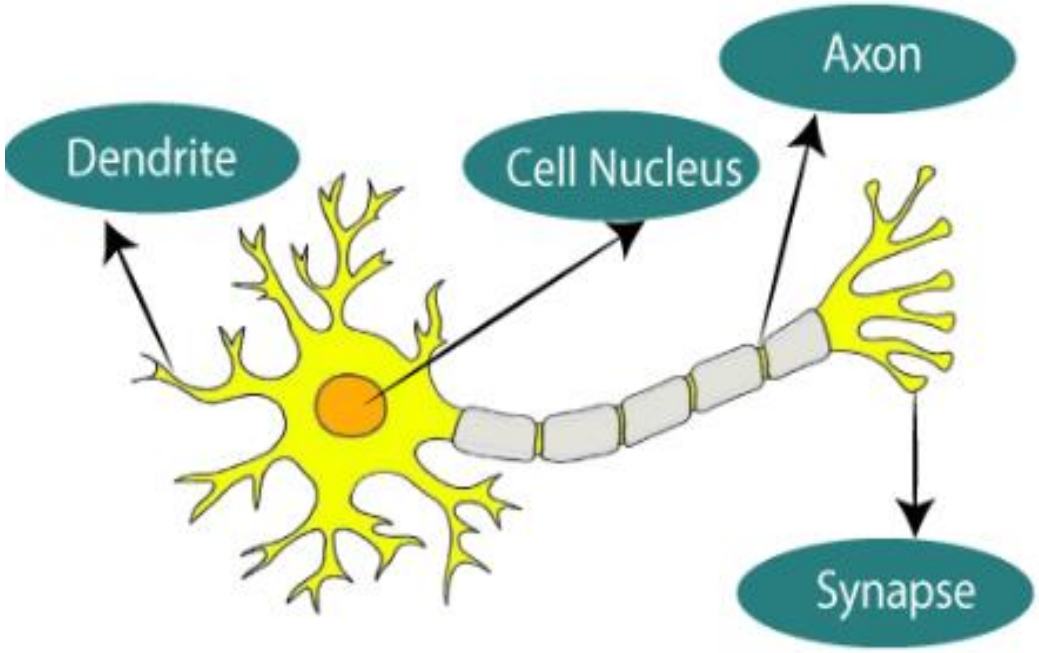
Logistic Regression= 0.8

Add a confusion matrix to the models

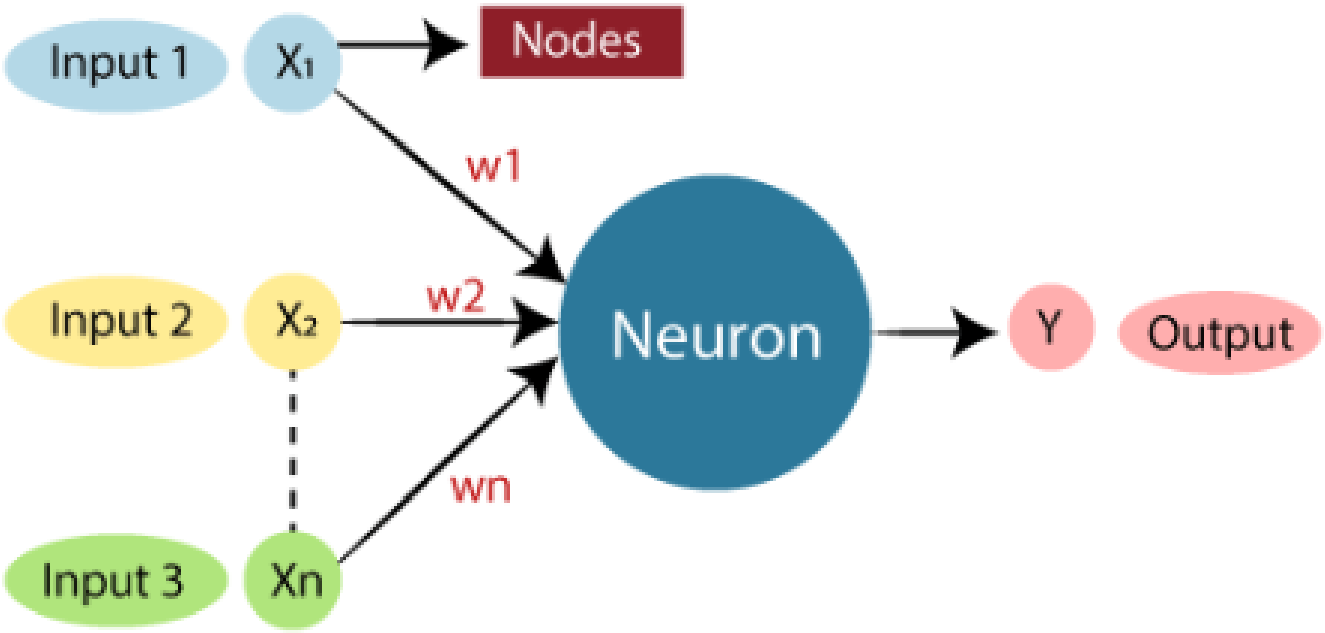
Artificial neuron network(ANN)

- The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

Biological Neural Network.



Artificial neuron networks (ANN)

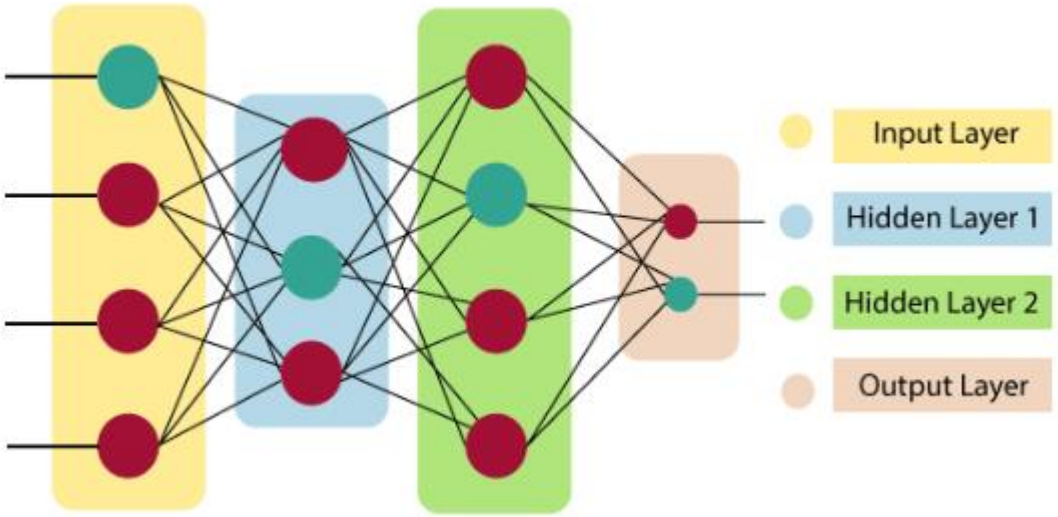


BNN vs ANN

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

Artificial neural network architecture

- Artificial Neural Network primarily consists of three layers:



- **Input Layer:**
- As the name suggests, it accepts inputs in several different formats provided by the programmer.
- **Hidden Layer:**
- The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.
- **Output Layer:**
- The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

Advantages of Artificial Neural Network (ANN)

- **Parallel processing capability:**

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

- **Storing data on the entire network:**

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

- **Capability to work with incomplete knowledge:**

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

- **Having a memory distribution:**

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

Having fault tolerance:

- Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages of ANN

- **Assurance of proper network structure:**

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

- **Unrecognized behavior of the network:**

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

- **Hardware dependence:**

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

- **Difficulty of showing the issue to the network:**

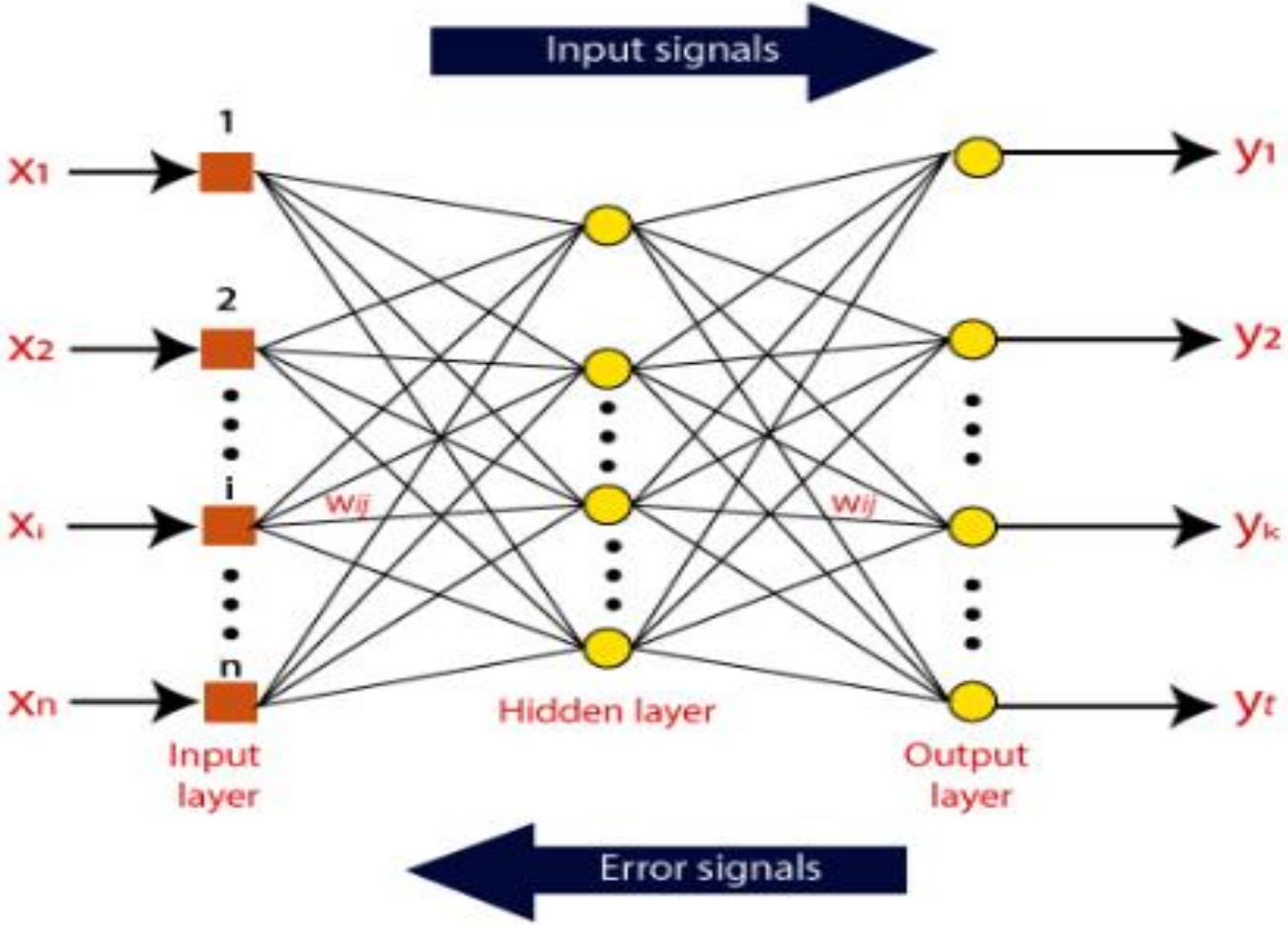
ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

- **The duration of the network is unknown:**

The network is reduced to a specific value of the error, and this value does not give us optimum results.

How they work?

- Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights.
- The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.


Input signals
1
2
i
n
x1
x2
xi
xn
wij
wij
y1
y2
yk
yt
Input layer
Hidden layer
Output layer
Error signals

Activation function

- The activation function(mathematical equations) refers to the set of transfer functions used to determine the desired output.
- Sigmoid function (0-1)
- Hyperbolic tangent(**tanh**) (-1 , 1)
- ReLU (**Rectified Linear Unit)** (0- infinity)

Sigmoid

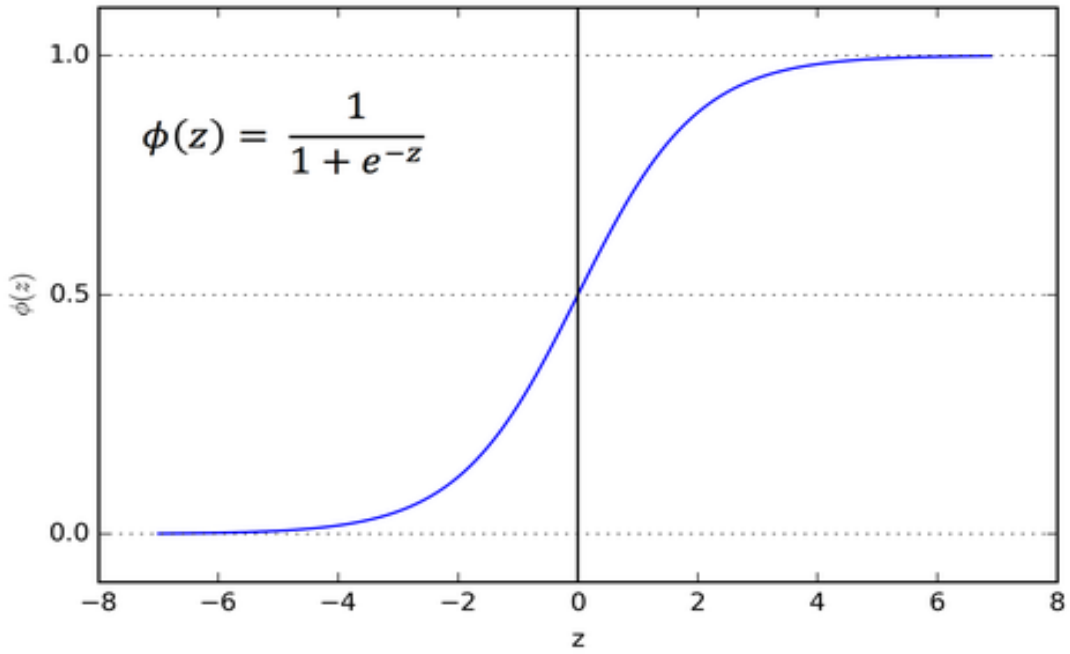


Fig: Sigmoid Function

The main reason why we use sigmoid function is because it exists between **(0 to 1).** Therefore, it is especially used for models where we have to **predict the probability** as an output. Since probability of anything exists only between the range of **0 and 1,** sigmoid is the right choice.

Hyperbolic tangent

The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped)

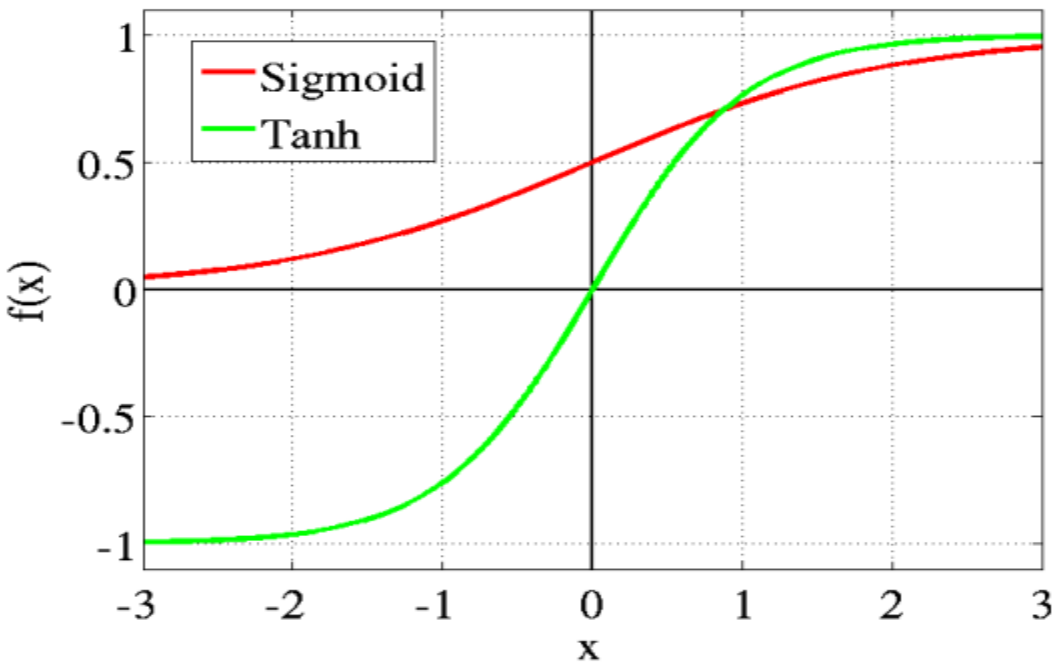


Fig: tanh v/s Logistic Sigmoid

The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

ReLU

- The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning. the ReLU is half rectified (from bottom). F(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

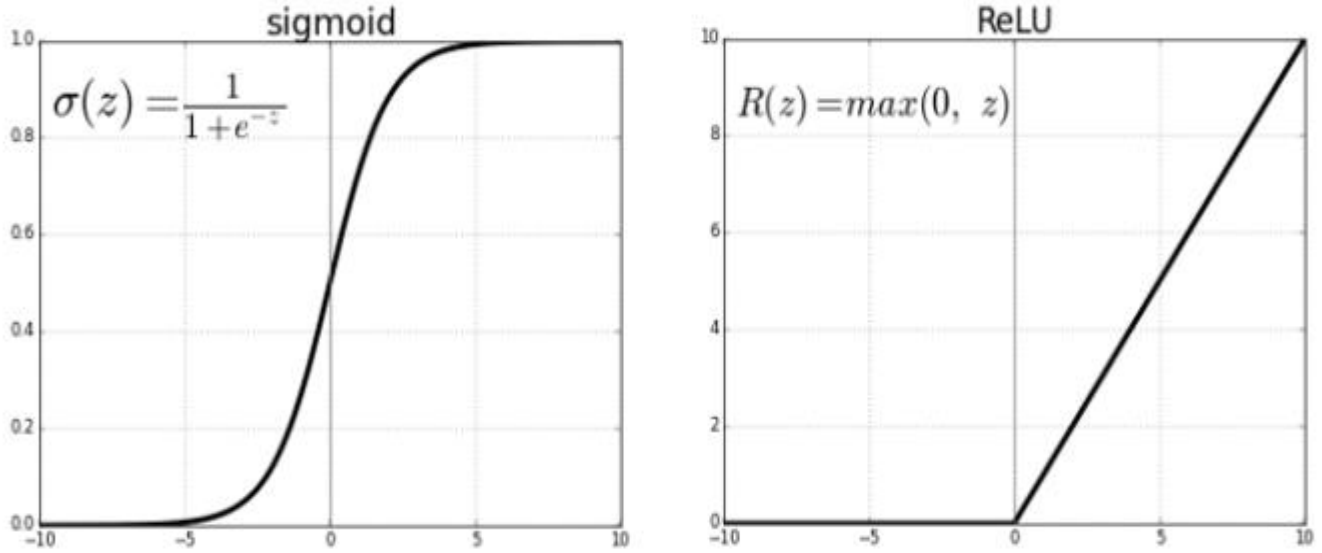


Fig: ReLU v/s Logistic Sigmoid

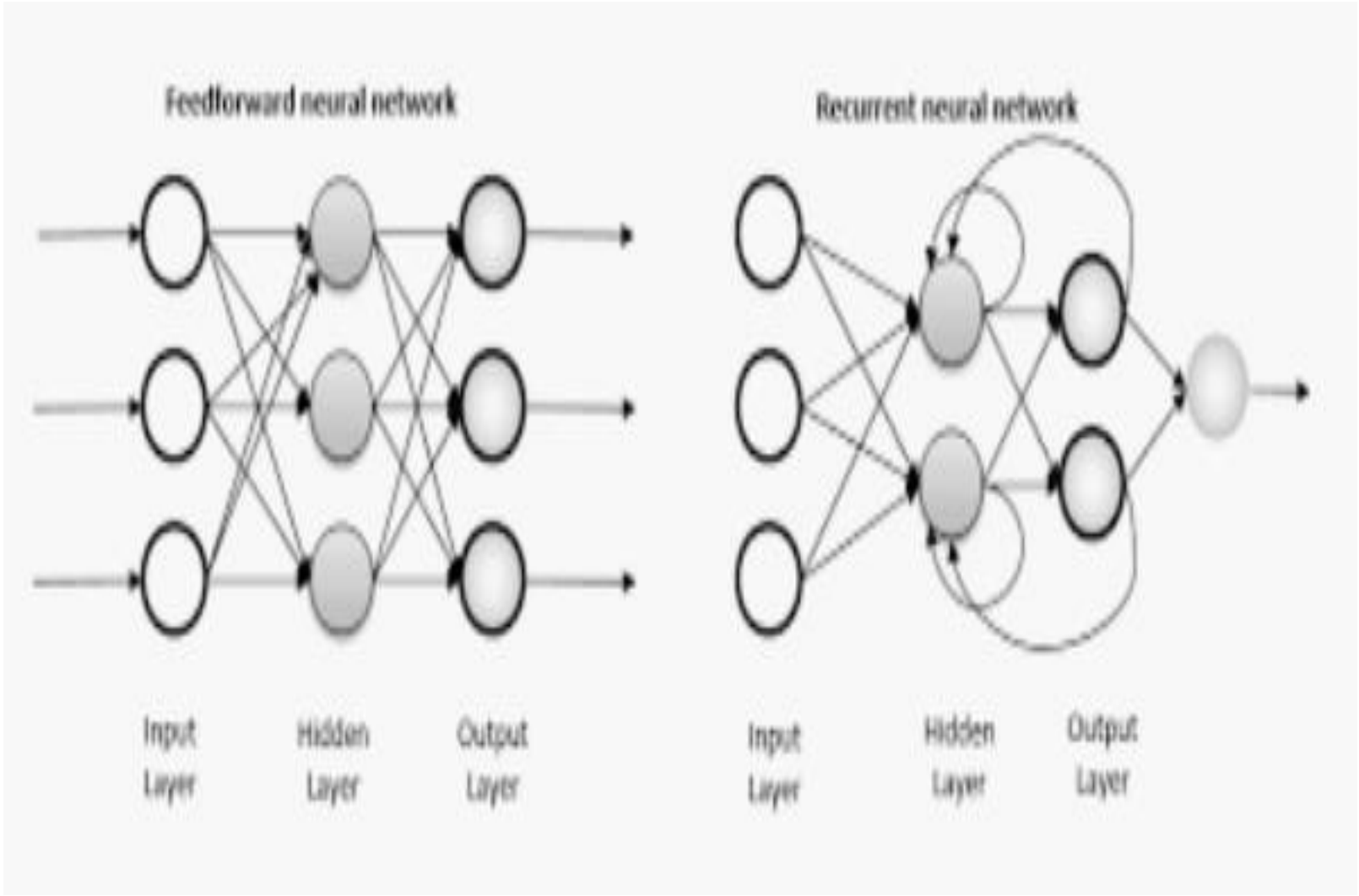
Types of Artificial Neural Network

Feedback ANN (recurrent)

- In this type of ANN, the output returns into the network to accomplish the best-evolved results internally. The feedback networks feed information back into itself and are well suited to solve optimization issues.

Feed-Forward ANN:

- A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.


Feedforward neural network
Input Layer
Hidden Layer
Output Layer
Recurrent neural network
Input Layer
Hidden Layer
Output Layer

Uses of Neural Networks

- Optical character recognition
- Image recognition
- Fingerprint recognition
- Face recognition
- Stock prediction
- Sport bets prediction
- Computer controlled game characters
- Statistical modelling
- Data mining

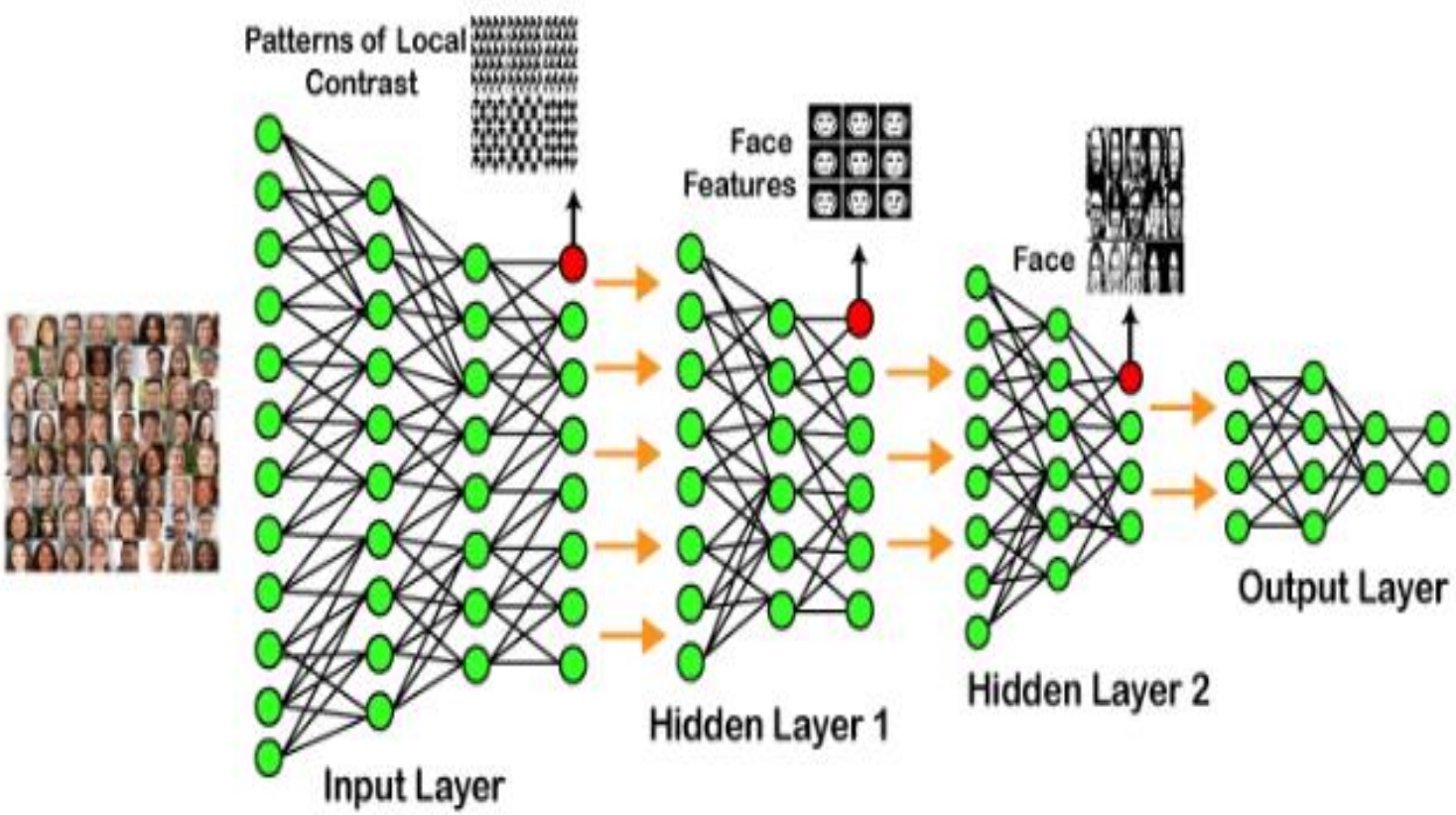
Deep learning

Deep learning is based on the branch of machine learning, which is a subset of artificial intelligence.

Deep learning *is a collection of statistical techniques of machine learning for learning feature hierarchies that are actually based on artificial neural networks.*

It basically neural networks with **multiple hidden layers.**

Example of Deep Learning



1st hidden layer will determine the face feature, i.e., it will fixate on eyes, nose, and lips, etc.

2nd hidden layer, will actually determine the correct face

.

. More hidden layers and so on.

as More hidden layers increase, we are able to solve complex problems.

Deep learning applications

- ***Self-Driving Cars***
  In self-driven cars, it is able to capture the images around it by processing a huge amount of data, and then it will decide which actions should be incorporated to take a left or right or should it stop. So, accordingly, it will decide what actions it should take, which will further reduce the accidents that happen every year.
- ***Voice Controlled Assistance***
  When we talk about voice control assistance, then **Siri** is the one thing that comes into our mind. So, you can tell Siri whatever you want it to do it for you, and it will search it for you and display it for you.
- ***Automatic Image Caption Generation***
  Whatever image that you upload, the algorithm will work in such a way that it will generate caption accordingly. If you say blue colored eye, it will display a blue-colored eye with a caption at the bottom of the image.
- ***Automatic Machine Translation***
  With the help of automatic machine translation, we are able to convert one language into another with the help of deep learning.

Advantages

- It lessens the need for feature engineering.
- It eradicates all those costs that are needless.
- It easily identifies difficult defects.
- It results in the best-in-class performance on problems.

Disadvantages

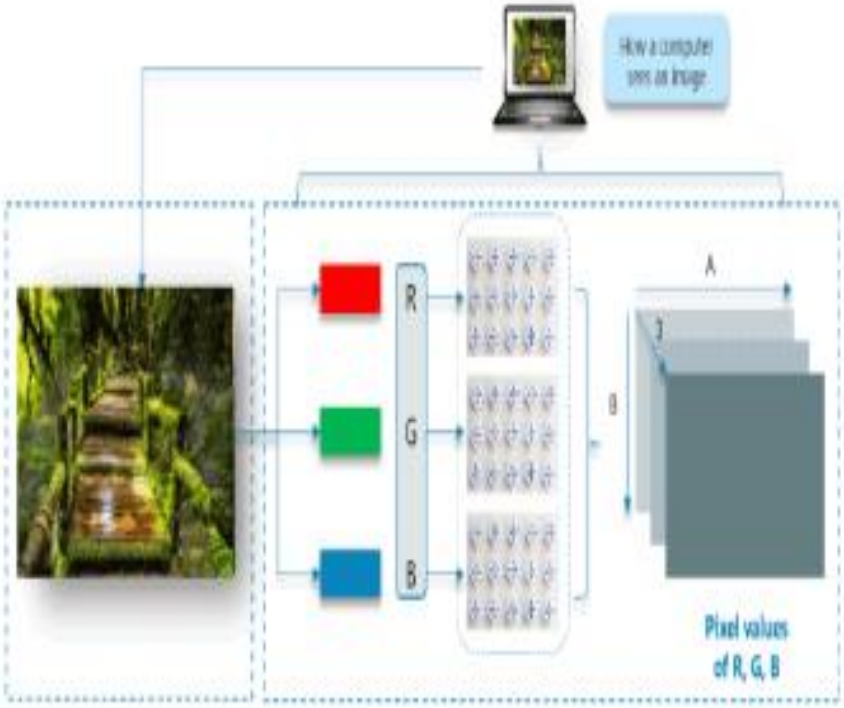
- It requires an ample amount of data.
- It is quite expensive to train.
- It does not have strong theoretical groundwork.

Convolutional Neural Network

- **Convolutional Neural Network** is one of the main categories to do image classification and image recognition in neural networks. Scene labeling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.

How Does a Computer read an image?

The image is broken into 3 color-channels which is **Red**, **Green**, and **Blue**. Each of these color channels is mapped to the image's pixel.



- Some neurons fires when exposed to vertices edges and some when shown horizontal or diagonal edges. CNN utilizes spatial correlations which exist with the input data. Each concurrent layer of the neural network connects some input neurons. This region is called a local receptive field. The local receptive field focuses on hidden neurons.

- Convolutional Neural Networks has the following 4 layers:
    - **Convolutional**
    - **ReLU Layer**
    - **Pooling**
    - **Fully Connected**

- CNN takes an image as input, which is classified and process under a certain category such as dog, cat, lion, tiger, etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution, it will see as **h * w * d**, where h= height w= width and d= dimension. For example, An RGB image is **6 * 6 * 3** array of the matrix, and the grayscale image is **4 * 4 * 1** array of the matrix.
- In CNN, each input image will pass through a sequence of convolution layers along with pooling, fully connected layers, filters (Also known as kernels). After that, we will apply the Soft-max function to classify an object with probabilistic values 0 and 1.

Convolution Layer

Convolution layer is the first layer to extract features from an input image. By learning image features using a small square of input data, the convolutional layer preserves the relationship between pixels. It is a mathematical operation which takes two inputs such as image matrix and a kernel or filter.

- The dimension of the image matrix is $h \times w \times d$.
- The dimension of the filter is $f_h \times f_w \times d$.
- The dimension of the output is $(h-f_h+1) \times (w-f_w+1) \times 1$.

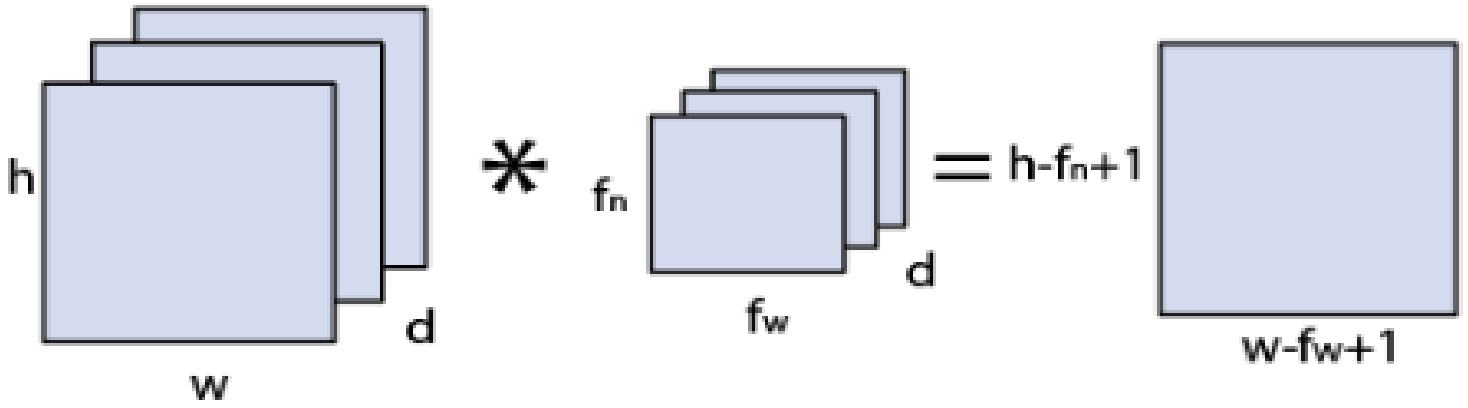


Image matrix multiplies kernl or filter matrix

Let's start with consideration a 5*5 image whose pixel values are 0, 1, and filter matrix 3*3 as:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

5 X 5 – Image Matrix 3 X 3 – Filter Matrix

The convolution of 5*5 image matrix multiplies with 3*3 filter matrix is called "**Features Map**" and show as an output.

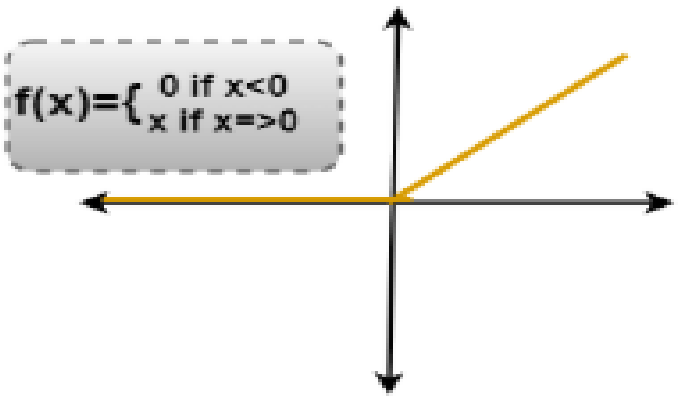
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{bmatrix}$$

Convolved Feature

Convolution of an image with different filters can perform an operation such as blur, sharpen, and edge detection by applying filters.

ReLU

Rectified Linear unit(ReLU) transform functions only activates a node if the input is above a certain quantity. While the data is below zero, the output is zero, but when the input rises above a certain threshold. It has a linear relationship with the dependent variable.
In this layer, we remove every negative value from the filtered images and replaces them with zeros.
It is happening to avoid the values from adding up to zero.

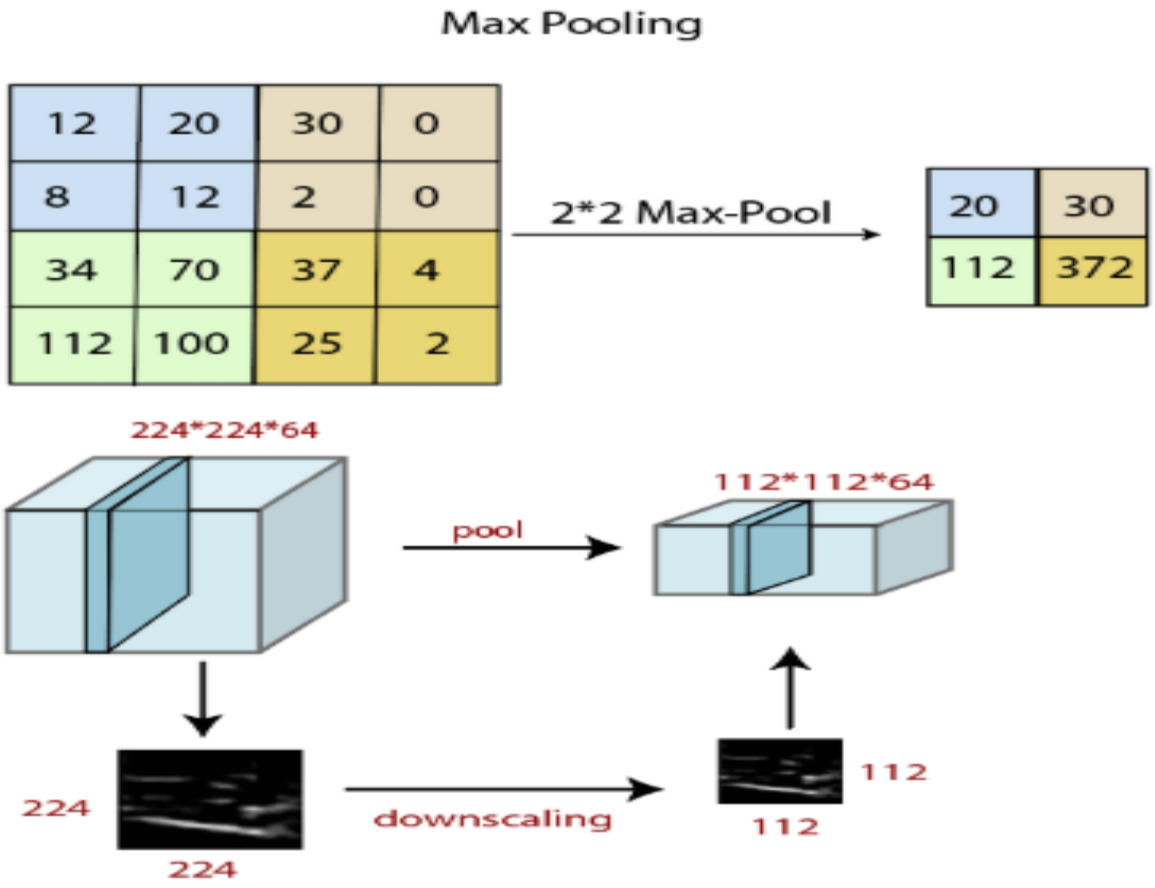


Pooling layer

- Pooling layer plays a vital role in pre-processing of any image. Pooling layer reduces the number of the parameter when the image is too large. Pooling is **"downscaling"** of the image achieved from previous layers. It can be compared to shrink an image to reduce the image's density. Spatial pooling is also called downsampling and subsampling, which reduce the dimensionality of each map but remains essential information. These are the following types of spatial pooling.
- We do this by implementing the following 4 steps:
  - Pick a **window size** (usually 2 or 3)
  - Pick a **stride** (usually 2)
  - **Walk** your window **across** your **filtered** images
  - From each **window**, take the **maximum** value

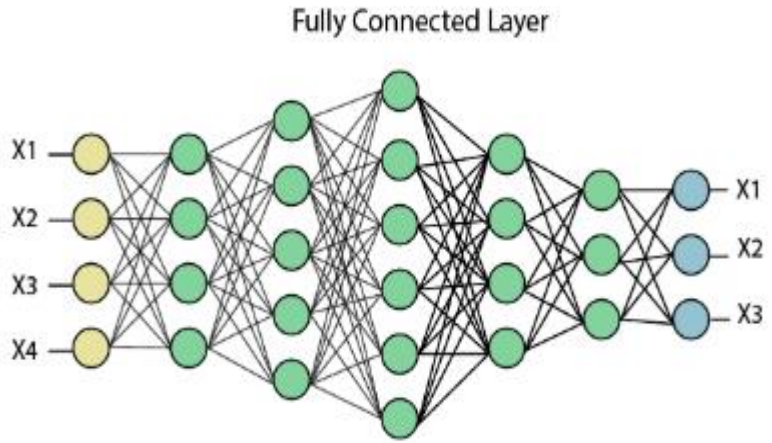
Max pooling

- Max pooling is a **sample-based discretization process**. The main objective of max-pooling is to downscale an input representation, reducing its dimension and allowing for the assumption to be made about feature contained in the sub-region binned.
- Max pooling is complete by applying a max filter in non-overlapping sub-regions of initial representation.

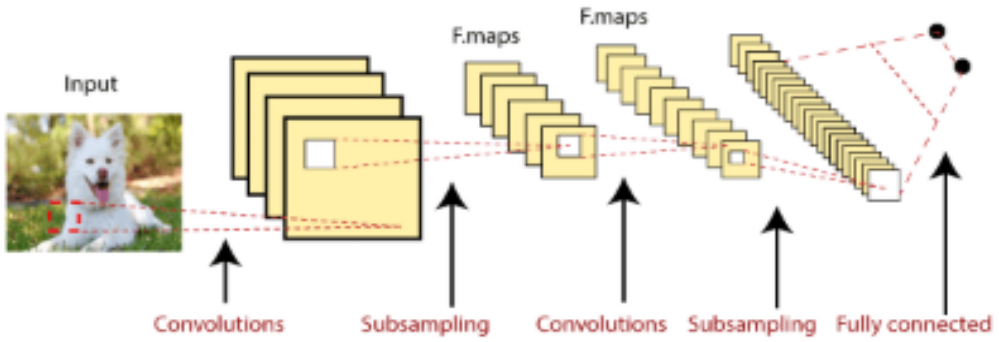

Max Pooling
12
20
30
0
8
12
2
0
34
70
37
4
112
100
25
2
2*2 Max-Pool
20
30
112
372
224*224*64
pool
112*112*64
224
224
downscaling
112
112

Fully Connected (Dense) Layer

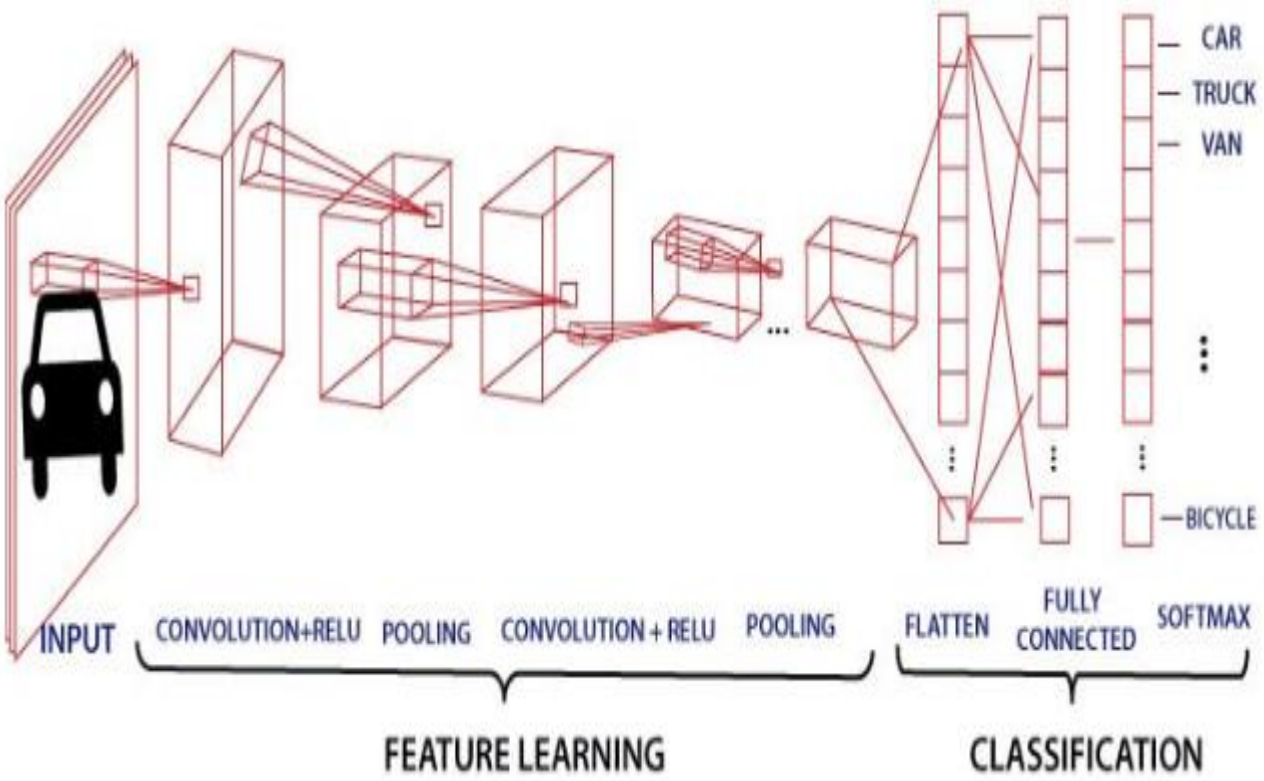
The fully connected layer (dense layer) is a layer where the input from other layers will be depressed into the vector. It will transform the output into any desired number of classes into the network.



- In the above diagram, the map matrix is converted into the vector such as **x1, x2, x3... xn** with the help of a fully connected layer. We will combine features to create any model and apply activation function like as **softmax** or **sigmoid** to classify the outputs as a car, dog, truck, etc.

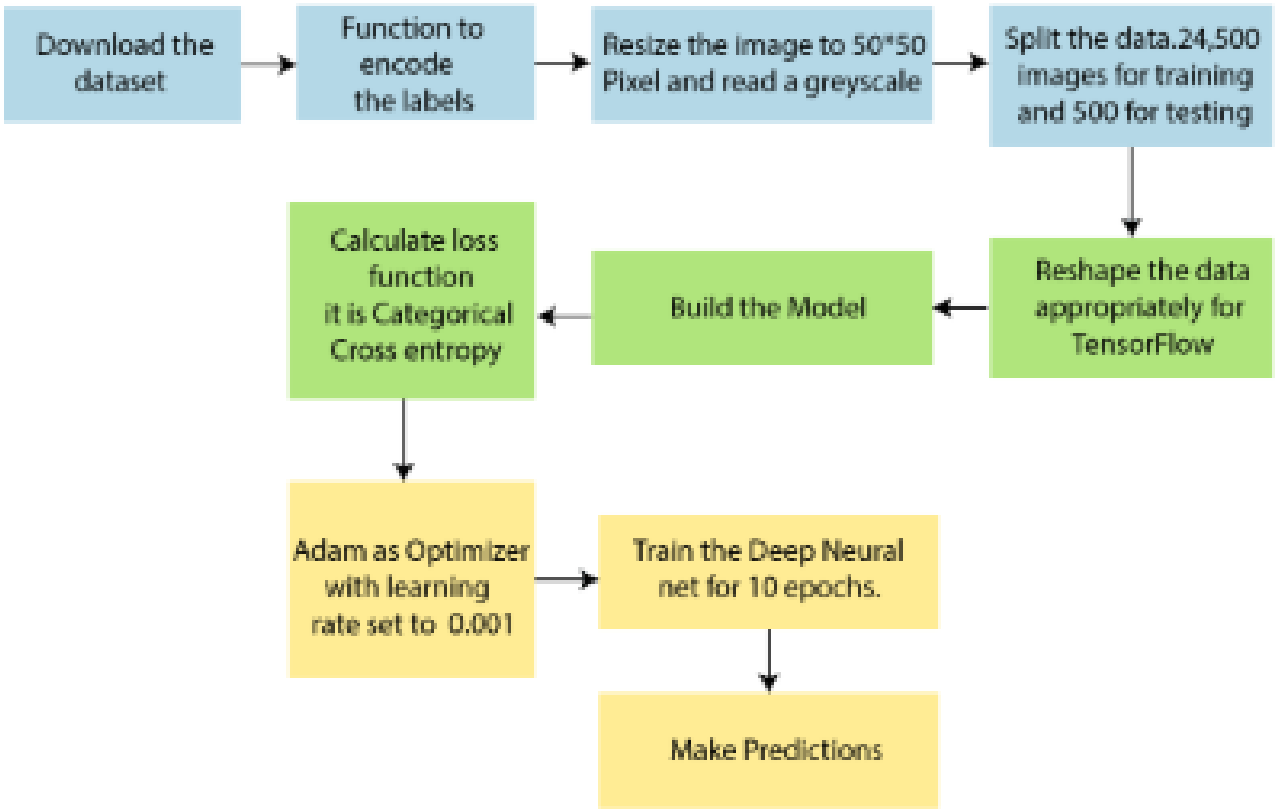


This is the final where the actual classification happens.

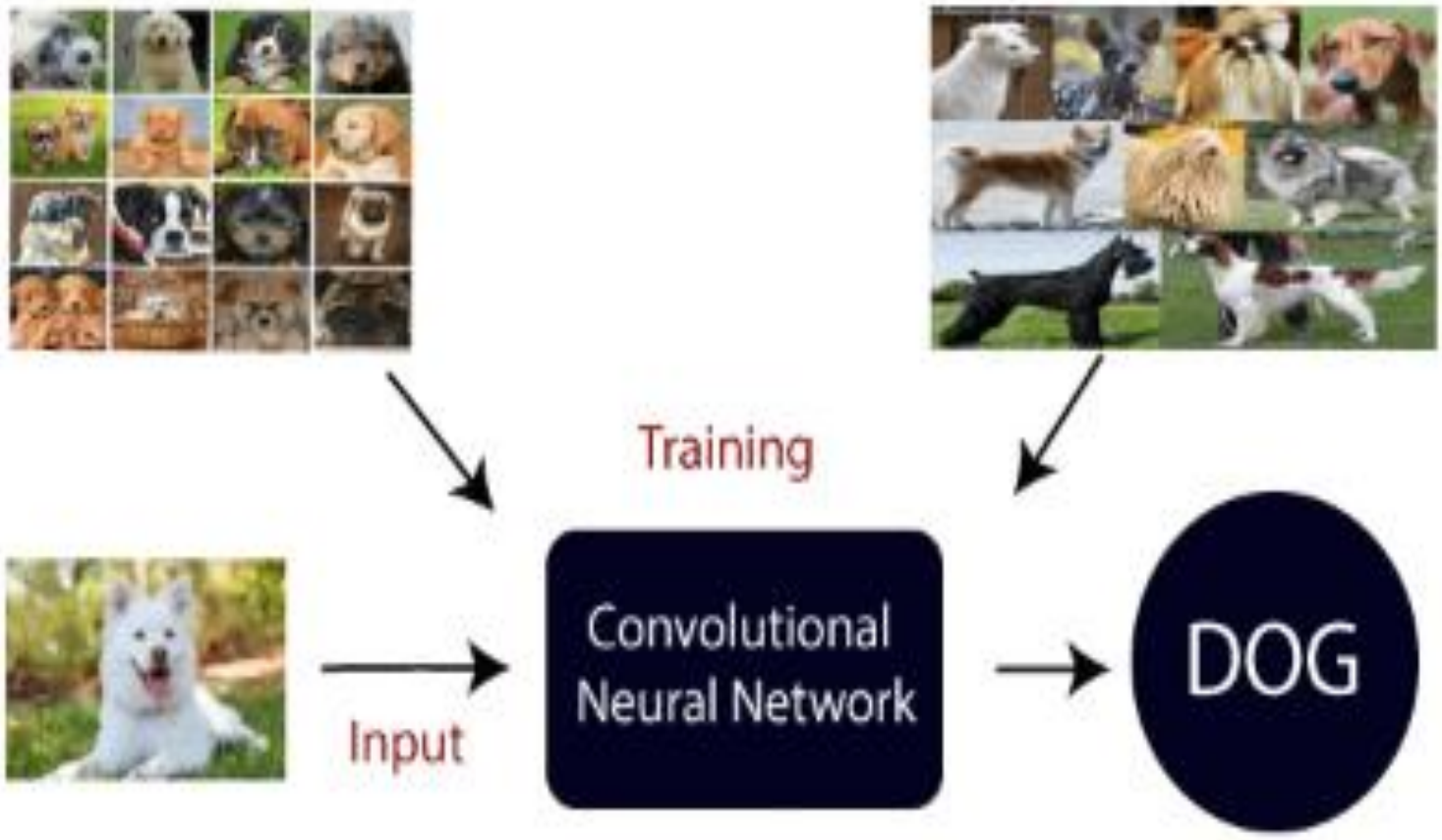

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION+RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

CNN use case

Steps:

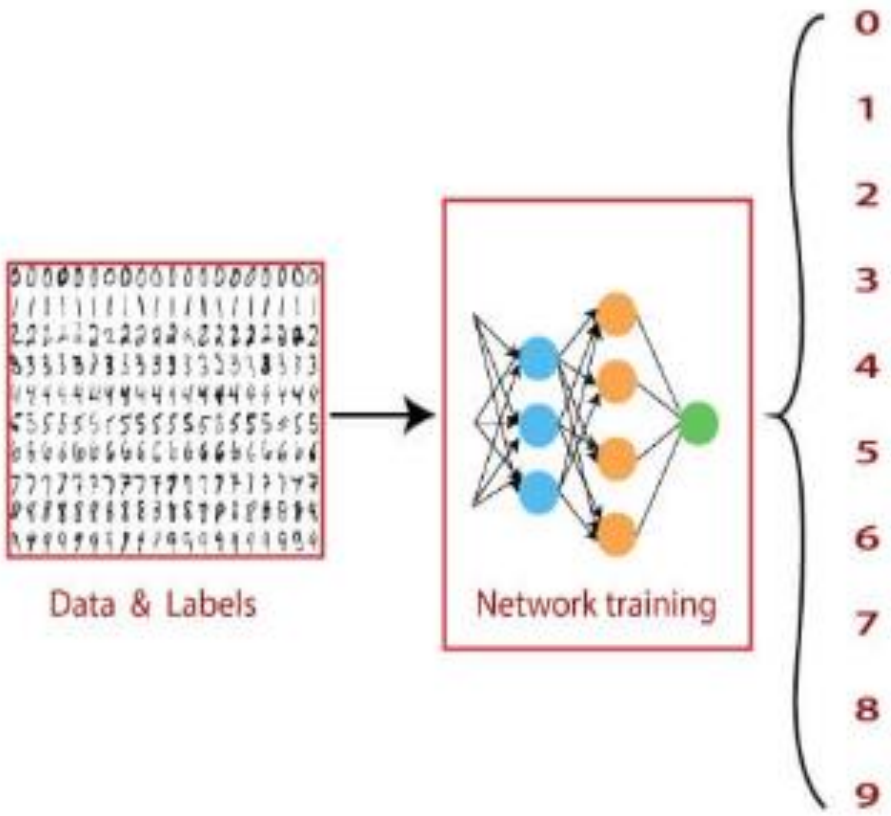


Model training



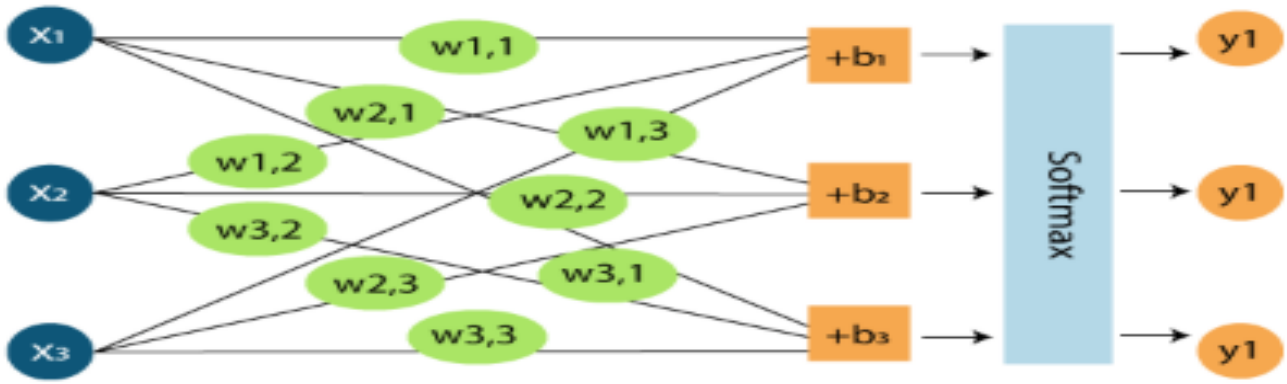
Training of CNN in TensorFlow

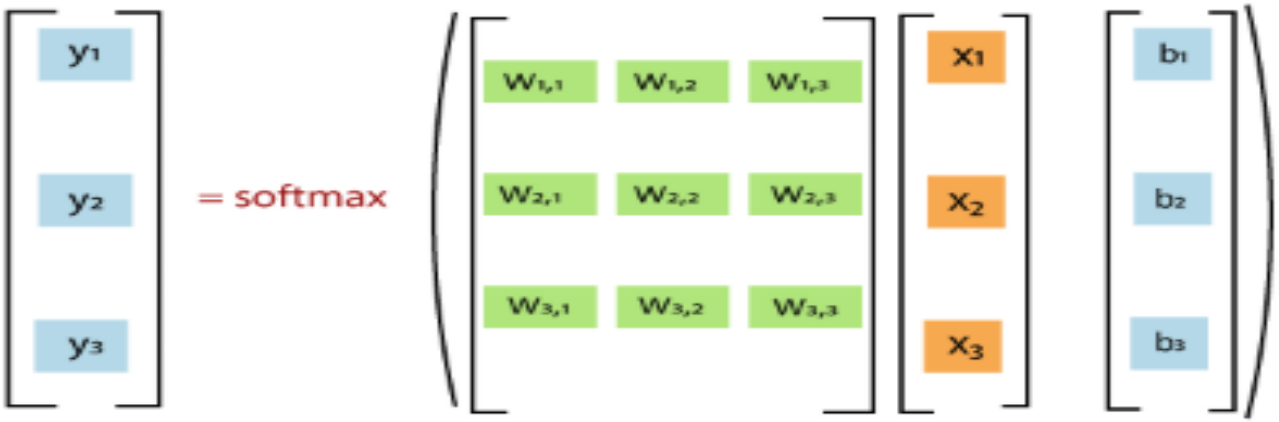
- The MNIST database (**Modified National Institute of Standard Technology database**) is an extensive database of handwritten digits, which is used for training various image processing systems. It was created by "**reintegrating**" samples from the original dataset of the **MNIST**.
- MNIST is the equivalent *Hello World* of image analysis. It consists of hand written numbers, 0-9, in 28x28 pixel squares.
  Each gray-scale pixel contains an integer 0-255 to indicate darkness, with 0 white and 255 black.
  There are about 60,000 training records, and about 10,000 test records.


Data & Labels
Network training
0
1
2
3
4
5
6
7
8
9

Softmax regression in tensorflow

- There are only ten possibilities of a TensorFlow MNIST to be from 0 to 9. Our aim is to look at an image and say with the particular probability that a given image is a particular digit. Softmax is used when there is a possibility as the regression gives us values between 0 and 1 that sum up to 1. Therefore, our approach should be simple


x1
x2
x3
w1,1
w2,1
w1,2
w3,2
w1,3
w2,2
w2,3
w3,1
w3,3
+b1
+b2
+b3
Softmax
y1
y1
y1


y1
y2
y3
= softmax
w1,1
w1,2
w1,3
w2,1
w2,2
w2,3
w3,1
w3,2
w3,3
x1
x2
x3
b1
b2
b3

Code execution

- executing our code in **Google Colab** (an online editor of machine learning).
- Link

: https://colab.research.google.com

Or jupyter notebook

Steps in training CNN

- **Step 1:** Upload Dataset
- **Step 2:** The Input layer
- **Step 3:** Convolutional layer
- **Step 4:** Pooling layer
- **Step 5:** Convolutional layer and Pooling Layer
- **Step 6:** Dense layer
- **Step 7:** Logit Layer

CNN & TF


Model Architecture
Evaluation
Prediction
Training a Model Using Multiple GPU Cards
CNN Tensorflow
Training
Launching and Training the Model on Multiple GPU cards
Launching the Model
Code Example

Step 1: Upload Dataset

- The MNIST dataset is available with scikit for learning in this URL (Unified Resource Locator). We can download it and store it in our downloads. We can upload it with fetch_mldata ('MNIST Original').
- Create a test/train set
- We need to split the dataset into **train_test_split**.
- **Scale the features**
- Finally, we scale the function with the help of **MinMax Scaler**.

Load code(jupyter)

```
import pickle
import numpy as np

with open('/Users/bob/Desktop/classes/kabarak/MAY AUG ONLINE NOTES/comp 413/mnist.pkl', 'rb') as f:
    (x_train, y_train), (x_test, y_test) = pickle.load(f, encoding='latin1')

print(type(x_train), x_train.size, x_train.shape)
print(type(y_train), y_train.size, y_train.shape)
print(type(x_test), x_test.size, x_test.shape)
print(type(y_test), y_test.size, y_test.shape)
print(y_train[55], y_test[583])
```

```
<class 'numpy.ndarray'> 47040000 (60000, 28, 28)
<class 'numpy.ndarray'> 60000 (60000,)
<class 'numpy.ndarray'> 7840000 (10000, 28, 28)
<class 'numpy.ndarray'> 10000 (10000,)
8 2
```

After load

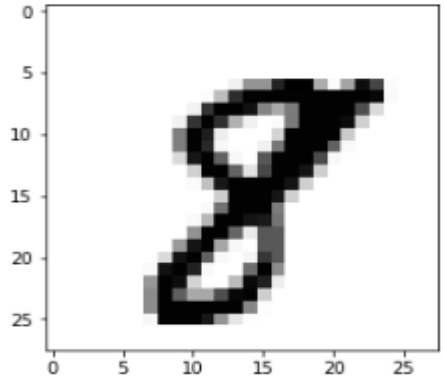
x_train contains 60k arrays of 28x28.
The y_train vector contains the corresponding labels for these.
x_test contains 10k arrays of 28x28.
The y_test vector contains the corresponding labels for these.

Matplotlib visualization

import matplotlib.cm as cm import matplotlib.pyplot as plt

plt.imshow(x_train[55].reshape(28, 28), cmap=cm.Greys)

NB there are many colormaps e.g Purples, Blues etc

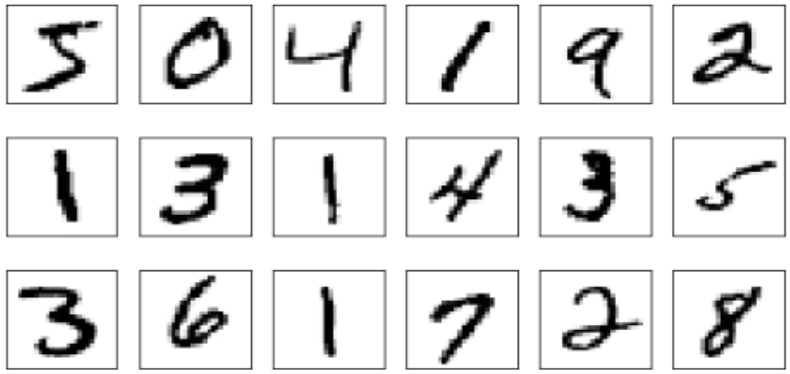


Plotting a bunch of records

Use Matplotlib for this task (18 records)

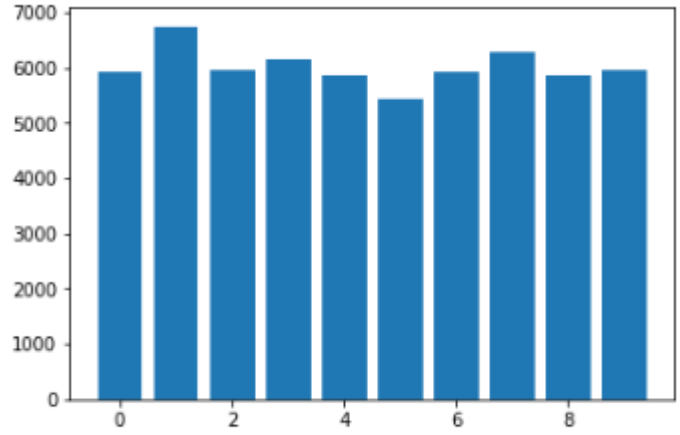
images = x_train[0:18] fig, axes = plt.subplots(3, 6, figsize=[9,5]) **for** i, ax **in** enumerate(axes.flat): ax.imshow(x_train[i].reshape(28, 28), cmap=cm.Greys) ax.set_xticks([]) ax.set_yticks([]) plt.show

```
<function matplotlib.pyplot.show(*args, **kw)>
```



Distribution of training data labels

counts = np.bincount(y_train) nums = np.arange(len(counts)) plt.bar(nums, counts) plt.show() print(counts)



[5923 6742 5958 6131 5842 5421 5918 6265 5851 5949]

Applying Keras/TensorFlow neural network

Use tensorflow to train the model with 60k training records, compile the model, and classify 10k test records with 98% accuracy.

Create the model

Build the keras model by stacking layers into the network. Our model here has four layers:

- Flatten reshapes the data into a 1-dimensional array
- Dense tells the model to use output arrays of shape (*, 512) and sets rectified linear activation function.
- Dropout applies dropout to the input to help avoid overfitting.
- The next Dense line condenses the output into probabilities for each of the 10 digits.

Compile the model

- Adam is an optimization algorithm that uses stochastic gradient descent to update network weights.
- Sparse categorical cross entropy is a loss function that is required to compile the model. The loss function measures how accurate the model is during training. We want to minimize this function to steer the model in the right direction.
- A metric is a function that is used to judge the performance of your model. We're using accuracy of our predictions as compared to y_test as our metric.
  Lastly, we fit our training data into the model, with several training repetitions (epochs), then evaluate our test data.

```
import tensorflow as tf
# Disable some deprecated error messages
tf.logging.set_verbosity(tf.logging.ERROR)

# Normalize the data to a 0.0 to 1.0 scale for faster processing
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(256, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.25),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=4)
model.evaluate(x_test, y_test)
```

output

```
Epoch 1/4
60000/60000 [==============================] - 5s 89us/sample - loss: 0.2581 - acc: 0.9244
Epoch 2/4
60000/60000 [==============================] - 5s 83us/sample - loss: 0.1180 - acc: 0.9644
Epoch 3/4
60000/60000 [==============================] - 5s 81us/sample - loss: 0.0867 - acc: 0.9736
Epoch 4/4
60000/60000 [==============================] - 5s 82us/sample - loss: 0.0697 - acc: 0.9785
10000/10000 [==============================] - 1s 59us/sample - loss: 0.0662 - acc: 0.9791
[0.06623600697717763, 0.9791]
```

CIFAR-10 and CIFAR-100 Dataset in TensorFlow

- The **CIFAR-10 (Canadian Institute for Advanced Research)** and CIFAR-100 are labeled subsets of the **80 million** tiny images dataset. They were collected by **Alex Krizhevsky, Geoffrey Hinton and Vinod Nair**. The dataset is divided into five training batches and only one test batch, each with **10000** images.
- The test batch contains **1000** randomly-selected images from each class. The training batches contain the remaining images in a random order, but some training batches contain the remaining images in a random order, but some training batches contain more images from one class to another. Between them, the training batches contain exactly 5000 images for each class.


airplane
automobile
bird
cat
deer
dog
frog
house
ship
truck

Generate predictions for test set

The predictions are in the form of a list of 10 floats, with probabilities for each value. We can get the prediction by picking the index of the list item with the highest probability. And we can visualize that item to verify our prediction.

```
predictions = model.predict(x_test)
print(predictions[88])
print(np.argmax(predictions[88]))
plt.imshow(x_test[88].reshape(28, 28),
cmap=cm.Greys)
```

```
[5.5930052e-09 1.6970777e-15 2.4897268e-10 2.3935108e-14 7.2053798e-09
 4.4642620e-10 1.0000000e+00 8.1785776e-12 2.4993282e-10 2.6947859e-13]
6
```

```
<matplotlib.image.AxesImage at 0x13fdb2668>
```