DEPARTMENT OF

ARTIFICIAL INTELLIGENCE

"Computers are able to see, hear and learn.. Wecome to the future"

Understanding Machine Learning

BY MACHINE LEARNING CLUB

I am your model, Train me with your love..
Our memories is my dataset, Test me with different case..
I will give accuracy of 100%,
You won't need any gradient descent..

About book

This is not written by any specialist authors. Content in this book is easy to understand because we use simple English language to understand the concepts clearly. Students have learned concepts by using various resources through the internet and some courses and they were sharing knowledge with all of us. You may find some copy content and images through the internet.

This is done by students based on their knowledge and they were representing their knowledge and work effectively with clear and basic level implementation of ml models.

I can't assure you that you will become an expert in ML after reading this book but at a basic level, you get an idea of how to implement ML models using the python sklearn package. we provide you code and dataset through the github link so that you can read the book and implement them parallelly. Even you can follow our contributors on GitHub and comment your doubts. I recommend highly reading this content for ML beginners.

Thank you, students, for your valuable contributions. This book could not have made without you.

Contents:

NumPy

- NumPy is a python library
- Used for working with arrays
- NumPy stands for Numerical Python
- Was created in 2005 by Travis Oliphant
- It is an open-source project and you can use it freely
- It also has functions for working in the domain of linear algebra, Fourier transform, and matrices

Lists VS NumPy

NumPy aims to provide an array object which is 50x faster than lists. Lists also used to work with arrays, but are slow to process.

- NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This is called the locality of reference in computer science.

- This is the main reason why NumPy is faster than lists. Also, it is optimized to work with the latest CPU architectures.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

- Import NumPy in your applications by adding the import keyword

```
import numpy
```

- NumPy is usually imported under the np alias.

```
import numpy as np
```

- 1-D Array using array() function

```
import numpy as np
arr=np.array([1,2,3,4,5])
print(arr)
```

```
[1 2 3 4 5]
```

- Checking version using `__version__` attribute

```
import numpy as np
print(np.__version__)
```

```
1.19.5
```

The array object in NumPy is called ndarray. We can create a NumPy ndarray object by using the array() function.

```
import numpy as np
arr=np.array([1,2,3,4,5])
print(arr)
type(arr)
```

```
[1 2 3 4 5]
numpy.ndarray
```

type(): This is a python built-in function that tells us the type of the object.

INDEXING:

Accessing 1-D Arrays

The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1, etc.

- Printing the first element

```
import numpy as np
arr=np.array([1,2,3,4,5])
print(arr[0])
```

```
1
```

- Printing the sum of given indices

```
import numpy as np
arr=np.array([1,2,3,4,5])
print(arr[2]+arr[4])
```

```
8
```

Accessing 2-D Arrays

To access elements from 2-D arrays we can use comma-separated integers representing the dimension and the index of the element.

- Printing 3rd element from 1st Dimension and 5th element from 2nd Dimension

```
import numpy as np
arr=np.array([[1,2,3,4,5],[6,7,8,9,10]])
print(arr[0,2])
print(arr[1,4])
```

```
3
10
```

Accessing 3-D Arrays

To access elements from 3-D arrays we can use comma-separated integers representing the dimensions and the index of the element.

- Printing the third element of the second array of the first array.

```
import numpy as np
arr=np.array([[[1,2,3,4],[5,6,7,8]],[[9,10,11,12],[13,14,15,16]]])
print(arr[0,1,2])
```

```
7
```

How indexing works in the above experiments

The first number represents the first dimension, which contains two arrays:
[[1, 2, 3, 4], [5, 6, 7, 8]]
and:
[[9, 10, 11, 12], [13, 14, 15, 16]]
Since we selected 0, we are left with the first array:
[[1, 2, 3, 4], [5, 6, 7, 8]]

The second number represents the second dimension, which also contains two arrays:
[1, 2, 3, 4]
and:

[5, 6, 7, 8]
Since we selected 1, we are left with the second array:
[5, 6, 7, 8]

The third number represents the third dimension, which contains three values:
5, 6, 7, 8
Since we selected 2, we end up with the third value:
7

Negative Indexing

- Printing the last element from 2nd Dimension

```
import numpy as np
arr=np.array([[1,2,3,4],[5,6,7,8]])
print(arr[1,-1])
```

```
8
```

SLICING:

Slicing in python means taking elements from one given index to another given index.

- We pass slice instead of an index like this: [*start*:*end*].
- We can also define the step, like this: [*start*:*end*:*step*].
- If we don't pass start it is considered 0
- If we don't pass the end it is considered the length of the array in that dimension
- If we don't pass step it is considered 1

- Slicing elements from index 1 to 5

```
import numpy as np
arr=np.array([1,2,3,4,5,6,7,8,9])
print(arr[1:5])
```

```
[2 3 4 5]
```

✔ The result includes the start index but excludes the end index

- Slicing elements from the 4th index to the end of the array

```
import numpy as np
arr=np.array([1,2,3,4,5,6,7,8,9])
print(arr[4:])
```

```
[5 6 7 8 9]
```

Slicing in 2-D Arrays

- From the second Array, slice elements from index 1 to 4

```
import numpy as np
arr=np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4])
```

```
[7 8 9]
```

NUMPY ARRAY SHAPE:

- The shape of an array is the number of elements in each dimension.
- NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

- Printing the shape of 2-D Array

```
import numpy as np
arr=np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.shape)
```

```
(2, 4)
```

Here (2,4) means that the array has 2 dimensions, and each dimension has 4 elements.

NUMPY ARRAY RESHAPE:

- Reshaping means changing the shape of an array.
- The shape of an array is the number of elements in each dimension.
- By reshaping we can add or remove dimensions or change the number of elements in each dimension.

- Reshaping 1-D array to 2-D array

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr = arr.reshape(4, 3)
print(newarr)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

The outermost dimension will have 4 arrays, each with 3 elements.

- Reshaping 1-D to 3-D Array

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr = arr.reshape(2,3,2)
print(newarr)
```

```
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]
```

The outermost dimension will have 2 arrays that contain 3 arrays, each with 2 elements.

Unknown Dimension:

- This means that we do not have to specify an exact number for one of the dimensions in the reshape method.
- Pass -1 as the value and NumPy will calculate this number.

- Reshaping 1-D Array with 8 elements to 3-D Array with 2x2 elements

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
newarr = arr.reshape(2, 2, -1)
print(newarr)
```

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

RANDOM DATA DISTRIBUTION:

- Data Distribution is a list of all possible values, and how often each value occurs.
- Such lists are important when working with statistics and data science.

- The random module offer methods that returns randomly generated data distributions.

- Random distribution is a set of random numbers that follow a certain *probability density function*.

```
from numpy import random
x = random.choice([3, 5, 7, 9], p=[0.1, 0.3, 0.6, 0.0], size=(100))
print(x)
```

```
[7 5 7 7 5 7 7 3 5 5 7 3 5 5 5 5 7 7 7 7 7 7 3 5 7 3 7 7 7 3 5 7 5 5 5 7 5
 3 7 7 7 7 7 5 7 5 7 3 5 5 7 3 5 7 7 5 7 7 7 7 7 3 7 7 7 5 5 7 7 7 5 5 7 7
 5 7 5 7 7 3 7 5 7 7 7 3 3 7 7 3 7 5 7 5 7 7 7 3 7 5]
```

This output represents a 1-D array containing 100 values, where each value has to be 3, 5, 7, or 9.

The probability for the value to be 3 is set to be 0.1

The probability for the value to be 5 is set to be 0.3

The probability for the value to be 7 is set to be 0.6

The probability for the value to be 9 is set to be 0

NOTE: The sum of all probability numbers should be 1.

- Returning 2-D Array with 3 rows, each containing 5 values using the size parameter

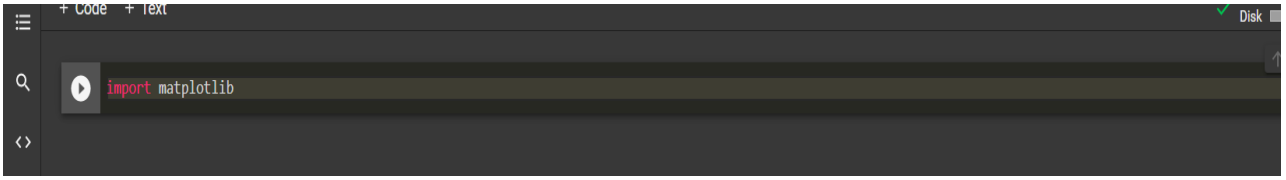
```
from numpy import random
x = random.choice([3, 5, 7, 9], p=[0.1, 0.3, 0.6, 0.0], size=(3, 5))
print(x)
```

```
[[7 7 7 7 5]
 [3 7 5 3 7]
 [3 7 7 5 7]]
```

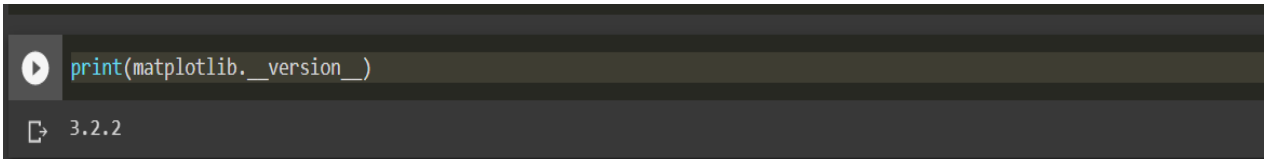
Matplotlib

- Matplotlib is a plotting library in python that serves as a visualization utility.
- It is mostly written in python, a few segments are written in C, Objective-C, and Javascript for Platform compatibility.
- Matplotlib was created by John D. Hunter.

⮑ Import it in your applications by adding the import keyword.

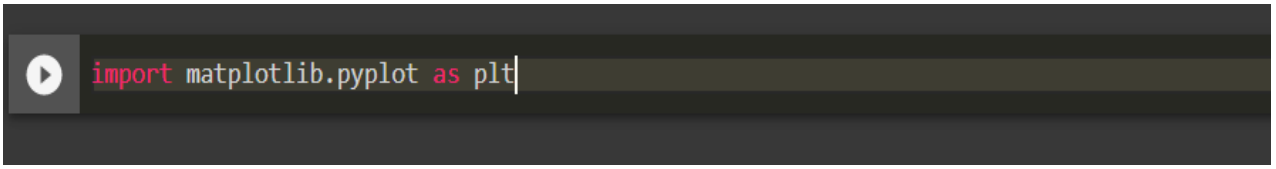


⮑ The version string is stored under the _version__ attribute.



Pyplot :

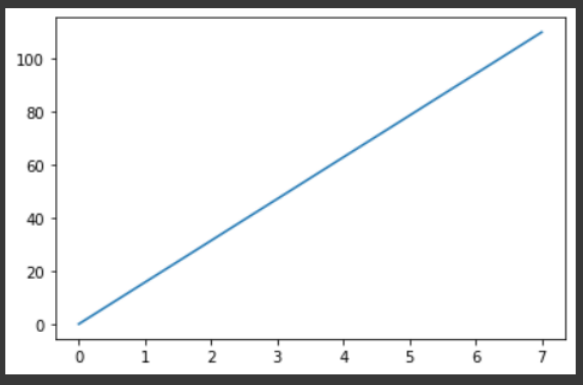
⮑ Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt



⟹Now Pyplot can be referred as plt

⇒Let us take an example to draw a line from position (0,0) to (7,110) :

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

xpoints = np.array([0, 7])
ypoints = np.array([0, 110])
plt.plot(xpoints, ypoints)
plt.show()
```



Matplotlib Plotting :

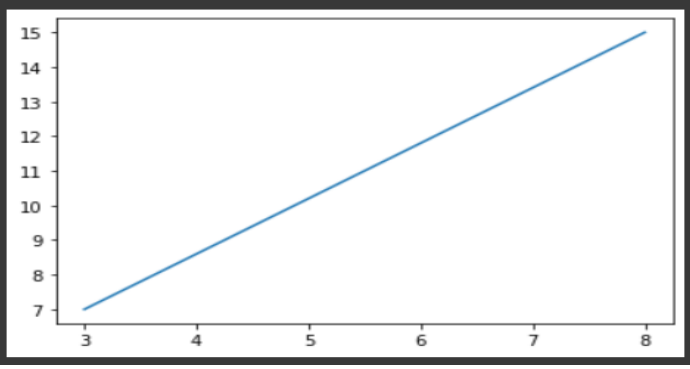
⊆ The plot() function is used to draw points (markers) in a diagram and it draws line from point to point.

⊆ Parameter 1 is an array containing the points on the x-axis.

⊆ Parameter 2 is an array containing the points on the y-axis.

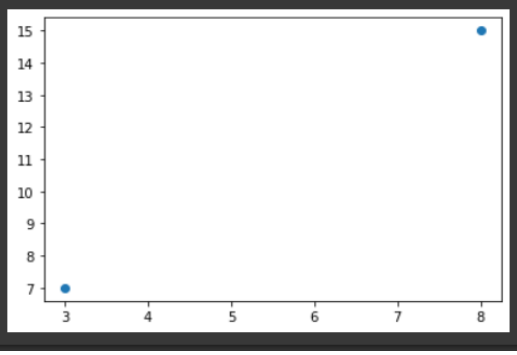
⇒Let us take an example by drawing line from position (3,7) to (8,15)

```
[11] import matplotlib.pyplot as plt
     import numpy as np

     xpoints = np.array([3, 8])
     ypoints = np.array([7, 15])
     plt.plot(xpoints, ypoints)
     plt.show()
```



⇒we can also plot without line :

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

xpoints = np.array([3, 8])
ypoints = np.array([7, 15])

plt.plot(xpoints, ypoints, 'o')
plt.show()
```

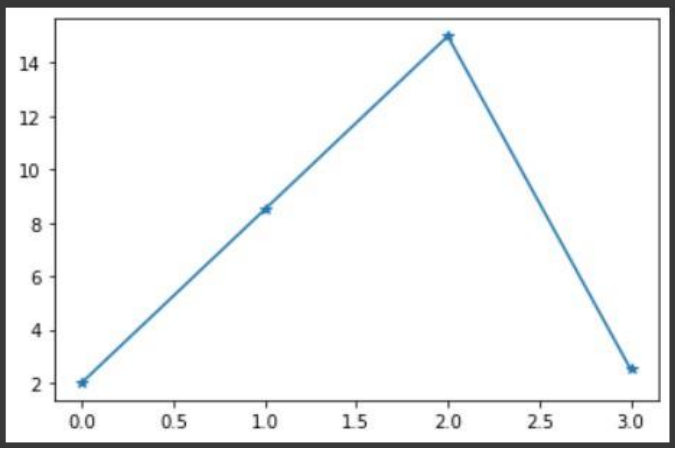


Matplotlib Markers :

☆ You can use the keyword argument marker to emphasize each point with a specified marker :

Marking each point with asterisk.

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

ypoints = np.array([2, 8.5, 15, 2.5])

plt.plot(ypoints, marker = '*')
plt.show()
```



We can also mark points with ‘o’, ‘.’, ‘,’, ‘x’, ‘+’ etc.

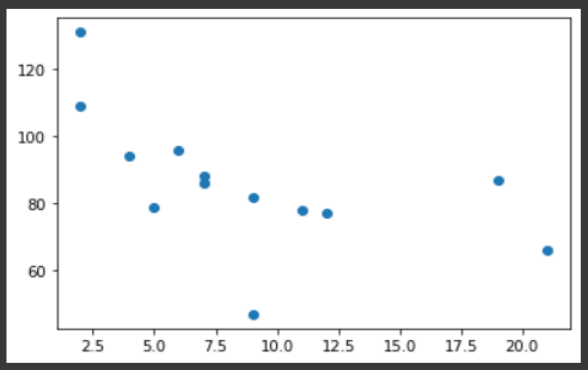
Matplotlib Scatter :

☆ scatter() function is used to draw a scatter plot, It plots one dot for each observation.

☆ It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis.

A simple scatter plot :

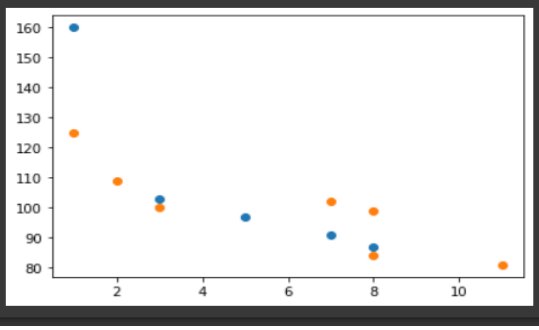
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,9,7,2,21,2,9,4,11,12,19,6])
y = np.array([79,86,47,88,131,66,109,82,94,78,77,87,96])

plt.scatter(x, y)
plt.show()
```



0s completed at 9:35 PM

We can also compare two plots by drawing them on same figure using scatter :

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

x = np.array([3,7,8,5,1])
y = np.array([103,91,87,97,160])
plt.scatter(x, y)

x = np.array([3,2,8,1,11,8,7])
y = np.array([100,109,84,125,81,99,102])
plt.scatter(x, y)

plt.show()
```



Pandas

- Pandas is used to analyse data
- Pandas is a Python library used for working with data sets. It has functions for analysing, cleaning, exploring, and manipulating data.
- Pandas is usually imported under the pd alias.

```
import pandas as pd
```

Now the Pandas package can be referred to as pd instead of pandas.

```
import pandas as pd

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

Checking Pandas Version

The version string is stored under __version__attribute.

```
import pandas as pd

print(pd.__version__)
```

```
1.1.5
```

Pandas generally provide two data structure for manipulating data, They are:

- **Series**
- **Data Frame**

Series:

A Pandas Series is like a column in a table. It is a one-dimensional array holding data of any type

Example

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

```
0    1
1    7
2    2
dtype: int64
```

:

label can be used to access a specified value.

```
print(myvar[0])
```

```
1
```

Create Labels:

You can create your own labels by using index argument

```python
import pandas as pd

a = [11, 17, 12]

myvar = pd.Series(a, index = ["xy", "yz", "zx"])

print(myvar)
#you can access an item by referring to the label.
print(myvar["yz"])
```

```
xy    11
yz    17
zx    12
dtype: int64
17
```

You can also use key value pairs like in dict.

```python
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
#To select only some of the items in the dictionary, use the index argument and
# specify only the items you want to include in the Series.
myvar1 = pd.Series(calories, index = ["day1", "day2"])

print(myvar1)
```

```
day1    420
day2    380
day3    390
dtype: int64
day1    420
day2    380
dtype: int64
```

Data Frames

- Data sets in Pandas are usually multi-dimensional tables, called DataFrames.
- Series is like a column, a DataFrame is the whole table.
- Pandas use the loc attribute to return one or more specified row(s)
- With the index argument, you can name your own indexes.
- Use the named index in the loc attribute to return the specified row(s)

```python
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

myvar = pd.DataFrame(data)

print(myvar)
#refer to the row index:
print(df.loc[0])
#use a list of indexes:
print(df.loc[[0, 1]])
#With the index argument, you can name your own indexes.
df = pd.DataFrame(data, index = ["day1", "day2", "day3"])
#refer to the named index:
print(df.loc["day2"])
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
Name                Jai
Age                  27
Address           Delhi
Qualification       Msc
Name: 0, dtype: object
     Name  Age Address Qualification
0     Jai   27   Delhi           Msc
1  Princi   24  Kanpur            MA
calories    380
duration     40
Name: day2, dtype: int64
```

Load Files Into a DataFrame

If your data sets are stored in a file, Pandas can load them into a DataFrame.

Read CSV Files

A simple way to store big data sets is to use CSV files (comma separated files).
CSV files contains plain text and is a well know format that can be read by everyone including Pandas

```python
import pandas as pd

df = pd.read_csv('data.csv')

print(df)
```

- Here we have used **print(df)**
- It will give you only up to 5 rows and the last 5 rows
- But we use **df.to_string()** to print entire Data Frame

Read JSON

- Big data sets are often stored, or extracted as JSON.
- JSON is plain text, but has the format of an object, and is well known in the world of programming, including Pandas.

```
import pandas as pd

df = pd.read_json('data.json')

print(df.to_string())
```

use **to_string()** to print the entire DataFrame.

Dictionary as JSON
JSON = Python Dictionary
JSON objects have the same format as Python dictionaries.
If your JSON code is not in a file, but in a Python Dictionary, you can load it into a DataFrame directly
Pandas - Analyzing DataFrames

Viewing the Data

- The head() method returns the headers and a specified number of rows, starting from the top.
- if the number of rows is not specified, the head() method will return the top 5 rows.
- The tail() method returns the headers and a specified number of rows, starting from the bottom.
- The DataFrames object has a method called info(), that gives you more information about the data set.(like how many rows and columns)
- The info() method also tells us how many Non-Null values there are present in each column

```
import pandas as pd

df = pd.read_csv('data.csv')
#gives 10 rows
print(df.head(10))
#gives top 5 rows
print(df.head())
#returns the headers and a specified number of rows, starting from the bottom.
print(df.tail())
#gives you info about data
print(df.info())
```

Data Cleaning

Data cleaning means fixing bad data in your data set.

Bad data could be:

- Empty cells
- Data in wrong format
- Wrong data
- Duplicates

Empty cells can potentially give you a wrong result when you analyse data.

Remove Rows

By default, the dropna() method returns a new DataFrame, and will not change the original.

If you want to change the original DataFrame, use the inplace = True argument

the dropna(inplace = True) will NOT return a new DataFrame, but it will remove all rows containing NULL values from the original DataFrame.

Replace Empty Values

Another way of dealing with empty cells is to insert a new value instead.

The fillna() method allows us to replace empty cells with a value

This will replaces all empty cells in the whole Data Frame

To only replace empty values for one column, specify the *column name* for the DataFrame

```
import pandas as pd

df = pd.read_csv('data.csv')
#the dropna() method returns a new DataFrame, and will not change the original
new_df = df.dropna()

print(new_df.to_string())
#want to change the original DataFrame, use the inplace = True argument
df.dropna(inplace = True)

print(df.to_string())
#The fillna() method allows us to replace empty cells with a value
df.fillna(130, inplace = True)
#To only replace empty values for one column, specify the column name for the DataFrame
df["Calories"].fillna(130, inplace = True)
```

Replace Using Mean, Median, or Mode

Pandas uses the mean() median() and mode() methods to calculate the respective values for a specified column

Mean = the average value (the sum of all values divided by number of values).

Median = the value in the middle, after you have sorted all values ascending.

Mode = the value that appears most frequently.

```python
import pandas as pd

df = pd.read_csv('data.csv')
#mean
x = df["Calories"].mean()

df["Calories"].fillna(x, inplace = True)
#median
x = df["Calories"].median()

df["Calories"].fillna(x, inplace = True)
#mode
x = df["Calories"].mode()[0]

df["Calories"].fillna(x, inplace = True)
```

Data of Wrong Format

Cells with data of wrong format can make it difficult, or even impossible, to analyze data.

To fix it, you have two options: remove the rows, or convert all cells in the columns into the same format.

- Convert Into a Correct Format
- Removing Rows

The result from the converting in the example above gave us a NaT value, which can be handled as a NULL value, and we can remove the row by using the `dropna()` method.

Wrong Data

"Wrong data" does not have to be "empty cells" or "wrong format", it can just be wrong, like if someone registered "199" instead of "1.99".

Replacing Values

One way to fix wrong values is to replace them with something else.

In our example, it is most likely a typo, and the value should be "45" instead of "450", and we could just insert "45" in row 7

To replace wrong data for larger data sets you can create some rules, e.g. set some boundaries for legal values, and replace any values that are outside of the boundaries.

Removing Rows

Another way of handling wrong data is to remove the rows that contains wrong data.

This way you do not have to find out what to replace them with, and there is a good chance you do not need them to do your analyses.

```
#replacing values
df.loc[7, 'Duration'] = 45
#replacing values
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.loc[x, "Duration"] = 120
#removing rows
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.drop(x, inplace = True)
```

Pandas - Removing Duplicates

Discovering Duplicates:

To discover duplicates, we can use the duplicated() method.

The duplicated() method returns a Boolean values for each row

Removing Duplicates

To remove duplicates, use the `drop_duplicates()` method

The (inplace = True) will make sure that the method does NOT return a new DataFrame, but it will remove all duplicates from the original DataFrame.

```
#discovering duplicates
print(df.duplicated())
#removing duplicates
df.drop_duplicates(inplace = True)
```

Pandas - Data Correlations

- The corr() method calculates the relationship between each column in your data set.
- The **corr()** method ignores "not numeric" columns.

df.corr()
for this input we get output like:

```
          Duration     Pulse  Maxpulse  Calories
Duration  1.000000 -0.155408  0.009403  0.922721
Pulse    -0.155408  1.000000  0.786535  0.025120
Maxpulse  0.009403  0.786535  1.000000  0.203814
Calories  0.922721  0.025120  0.203814  1.000000
```

Perfect Correlation:

We can see that "Duration" and "Duration" got the number `1.000000`, which makes sense, each column always has a perfect relationship with itself.

Good Correlation:

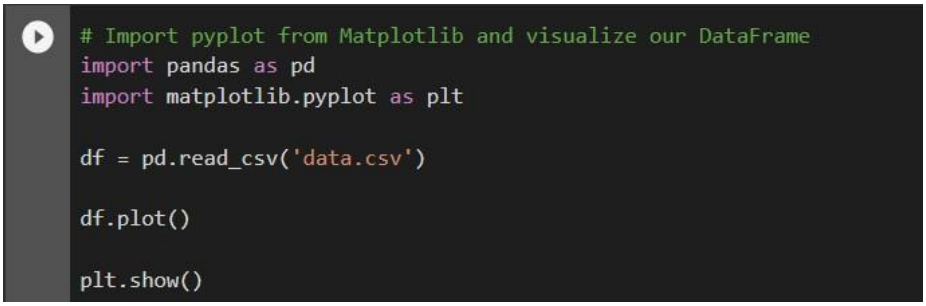
"Duration" and "Calories" got a `0.922721` correlation, which is a very good correlation, and we can predict that the longer you work out, the more calories you burn, and the other way around: if you burned a lot of calories, you probably had a long workout.

Bad Correlation:

"Duration" and "Maxpulse" got a `0.009403` correlation, which is a very bad correlation, meaning that we can not predict the max pulse by just looking at the duration of the workout, and vice versa.

Pandas – Plotting

Pandas uses the `plot()` method to create diagrams.



```
# Import pyplot from Matplotlib and visualize our DataFrame
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot()

plt.show()
```

Scatter Plot

Specify that you want a scatter plot with the `kind` argument:

```
kind = 'scatter'
```

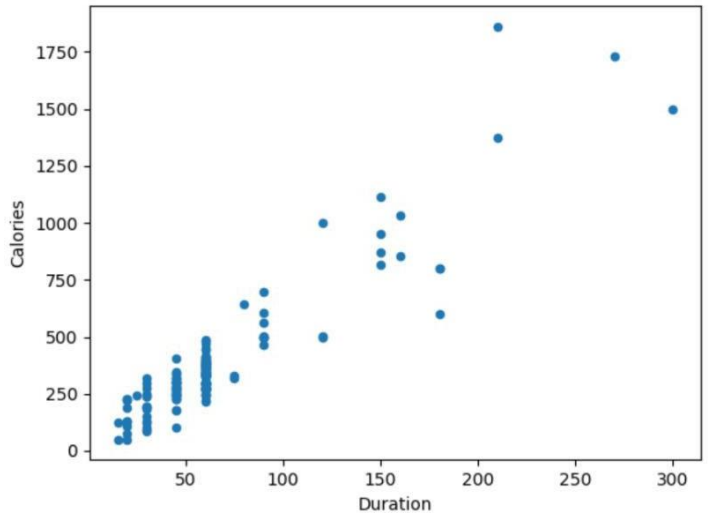
A scatter plot needs an x- and a y-axis.

In the example below we will use "Duration" for the x-axis and "Calories" for the y-axis.

Include the x and y arguments like this:

```
x = 'Duration', y = 'Calories'
```

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

df = pd.read_csv('data.csv')

df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

plt.show()
```



Histogram

Use the `kind` argument to specify that you want a histogram:

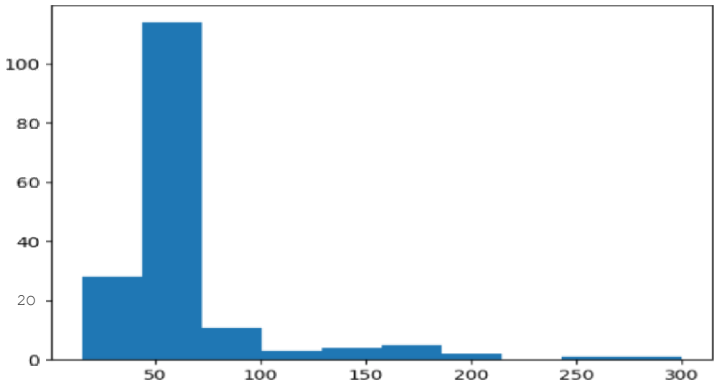
```
kind = 'hist'
```

A histogram needs only one column.

A histogram shows us the frequency of each interval, e.g. how many workouts lasted between 50 and 60 minutes?

In the example below we will use the "Duration" column to create the histogram:

```
df["Duration"].plot(kind = 'hist')
```


100
80
60
40
20
0
50
100
150
200
250
300

Vidya Jyothi Institute of Technology (Autonomous)

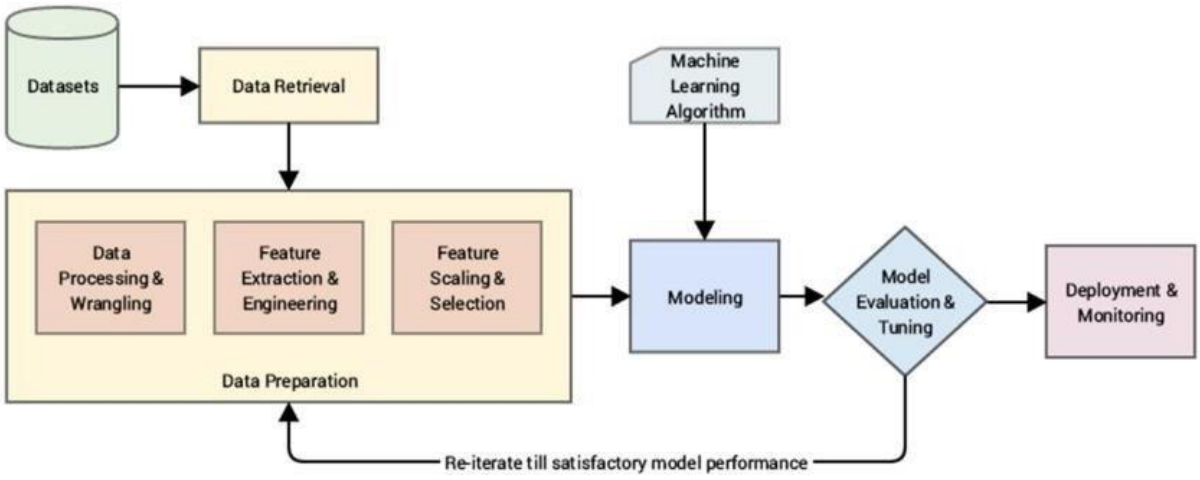
(Accredited by NBA, Approved by AICTE New Delhi)
Aziz nagar Gate, C.B. Post, Hyderabad-500 075.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE.

Introduction to ML:

Machine Learning is the science (and art) of programming computers so they can learn from data. Here is a slightly more general definition:[Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed]. Arthur Samuel, 1959And a more engineering-oriented one: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.Tom Mitchell, 1997

Stages of ML



We can observe complete ML lifecycle to put our model in production

The first stage of ML is to train an ML model with examples. we start from examples. An example consists of a label and an input. For example, suppose we want to train a machine learning model to look at images an identify

what's in those images. The true answer is called the label. So cat for the first image, and dog for the second image, those are the labels. The image itself, the pixels of the image are the input to the model. The model itself is a mathematical function of a form that can be applied to a wide variety of problems. There are many such mathematical functions. The models used in machine learning have a bunch of adjustable parameters though, all of them do. Then when we train a model, what we're doing is that we're making tiny adjustments to the model. So that the output of the model, the output of the mathematical function, is as close as possible to the true answer for any given input. Of course, we don't do this on one image at a time. The idea is to adjust the mathematical function so that overall, the outputs of the model for the set of training inputs is as close as possible to the training labels. And by labeled examples,

we mean the input and true answer. And after the model is trained, we can use it to predict the label of images that it has never seen before. Here, we are inputting to the trained model this image. And because the network has been trained, it is correctly able to output cat. Notice that the cat image on this slide is different from the one before it. It still works because the machine learning model has generalized from the specific examples of cat images that we showed it to a more general idea of what a cat is and what it looks like. The key to making a machine learning model generalized is data, and lots and lots of it. Having labeled the data is a precondition for successful machine learning. It is important to realize that machine learning has two stages, training and inference. Sometimes people refer to prediction as inference, because prediction seems to imply a future state. In the case of images like this, we're not really predicting that it's a cat, just inferring that it's a cat based on the pixel data. It can be tempting as a data scientist to focus all your energy on the first stage, on training. But this is not enough, you need to be able to operationalize the model, put the model into production so that you can run inferences.

.

Type of ML:

1. SUPERVISED LEARNING

In supervised learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data by associating patterns to the unlabeled new data.
Supervised learning can be divided into two categories: classification and regression.

CLASSIFICATION PREDICTS THE CATEGORY THE DATA BELONGS TO.

Some examples of classification include spam detection, churn prediction, sentiment analysis, dog breed detection and so on.

REGRESSION PREDICTS A NUMERICAL VALUE BASED ON PREVIOUSLY OBSERVED DATA.

Some examples of regression include house price prediction, stock price prediction, height-weight prediction and so on.

Linear regression for regression problems.
Random forest for classification and regression problems.
Support vector machines for classification problems.

Suppose consider above data is feed into ML model and as discussed above it learns from labeled data and when new coin comes it is going predict based on features that ML model trained**.**

2. Unsupervised Machine Learning

Unsupervised learning is where you only have input data (X) and no corresponding output variables.The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data. Unsupervised learning problems can be further grouped into clustering and association problems.

- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Some popular examples of unsupervised learning algorithms are:

- k-means for clustering problems.
- Apriori algorithm for association rule learning problems.

Now if we feed names and scores and wickets of respective players there is no labeled data as we seen before now model itself need to predict it in such as way low score and high wickets as bowlers and high score and low wickets as batsman as per our example.

2. Reinforcement Learning

Reinforcement Learning is defined as a Machine Learning method that is concerned with how software agents should take actions in an environment. Reinforcement Learning is a part of the deep learning method that helps you to maximize some portion of the cumulative reward

Here are some important terms used in Reinforcement AI:

- **Agent:** It is an assumed entity which performs actions in an environment to gain some reward.
- **Environment (e):** A scenario that an agent has to face.
- **Reward (R):** An immediate return given to an agent when he or she performs specific action or task.
- **State (s):** State refers to the current situation returned by the environment.
- **Policy (π):** It is a strategy which applies by the agent to decide the next action based on the current state.
- **Value (V):** It is expected long-term return with discount, as compared to the short-term reward.
- **Value Function:** It specifies the value of a state that is the total amount of reward. It is an agent which should be expected beginning from that state.
- **Model of the environment:** This mimics the behavior of the environment. It helps you to make inferences to be made and also determine how the environment will behave.
- **Model based methods:** It is a method for solving reinforcement learning problems which use model-based methods.
- **Q value or action value (Q):** Q value is quite similar to value. The only difference between the two is that it takes an additional parameter as a current action.

Let understand with one example

If we train our model based on dog image and if model predicts it as cat then our model gets learned from feedback and experience i.e collects reward and maximize reward may be positive reward or negative reward

Now let see how to implement this kind models

Vidya Jyothi Institute of Technology (Autonomous)
(Accredited by NBA, Approved by AICTE New Delhi)
Aziz nagar Gate, C.B. Post, Hyderabad-500 075.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE.

Algorithm name:

Simple Linear Regression

Dataset Download link:

https://www.kaggle.com/rsadiq/salary

Algorithm explanation:

Regression:

Regression is a type of supervised machine learning problem, under which the relationship between features in the data-space and the outcome is established, to predict the values for the outcome, which are continuous by nature.

Linear Regression:

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.

Types of Linear Regression

Simple Linear Regression:

Simple linear regression establishes a linear relationship between one input and one output variable, along with a constant co-efficient.

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression:

When multiple input variables are linearly combined to get the value of outcome, it is called multiple linear regression.

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

Mathematical part :

In the above equations,

y = the target variable.

x = the input variable.(x1,x2,...xn in case of multiple regression)

b0 = the intercept value.

b1, b2,.. bn = Coefficients describing the linear relationship between a combination of the input variables and target variable.

Minimizing the Error

The error is defined as the difference of values between actual points and the points on the straight line). Ideally., we'd like to have a straight line where the error is minimized across all points.

Least Square Regression

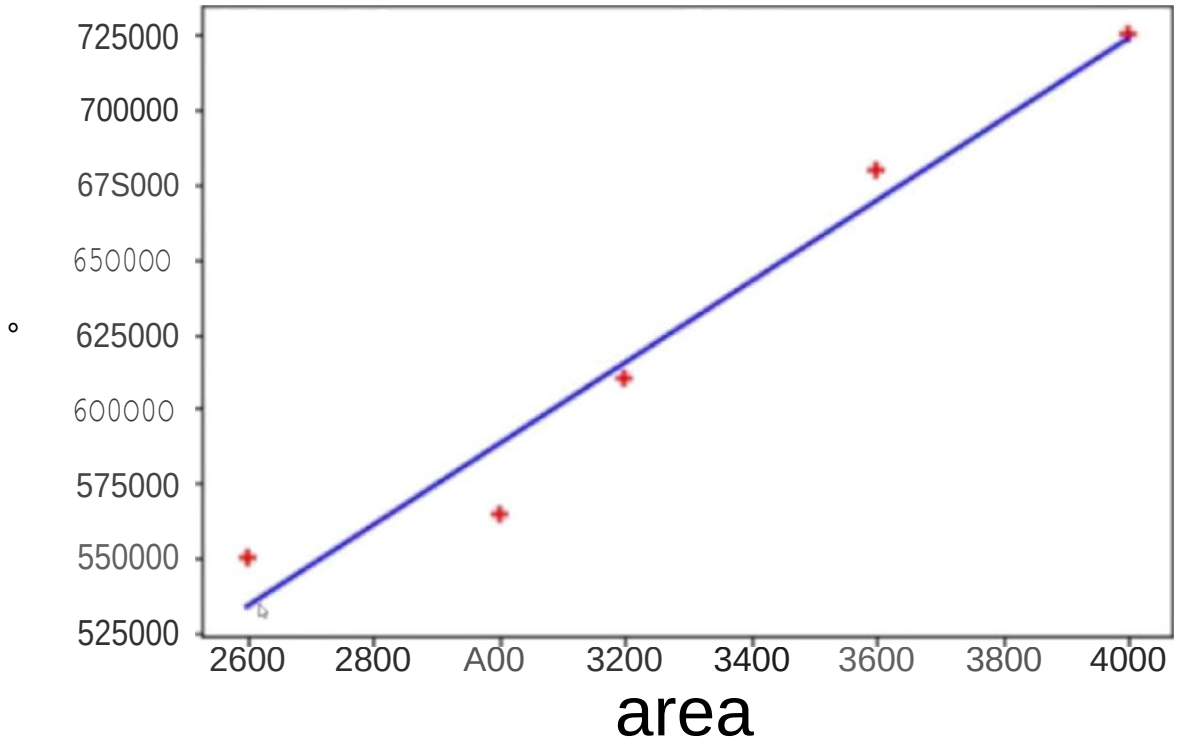
Least Square Regression is a method which minimizes the error in such a way that the sum of all square error is minimized. Here are the steps you use to calculate the Least square regression

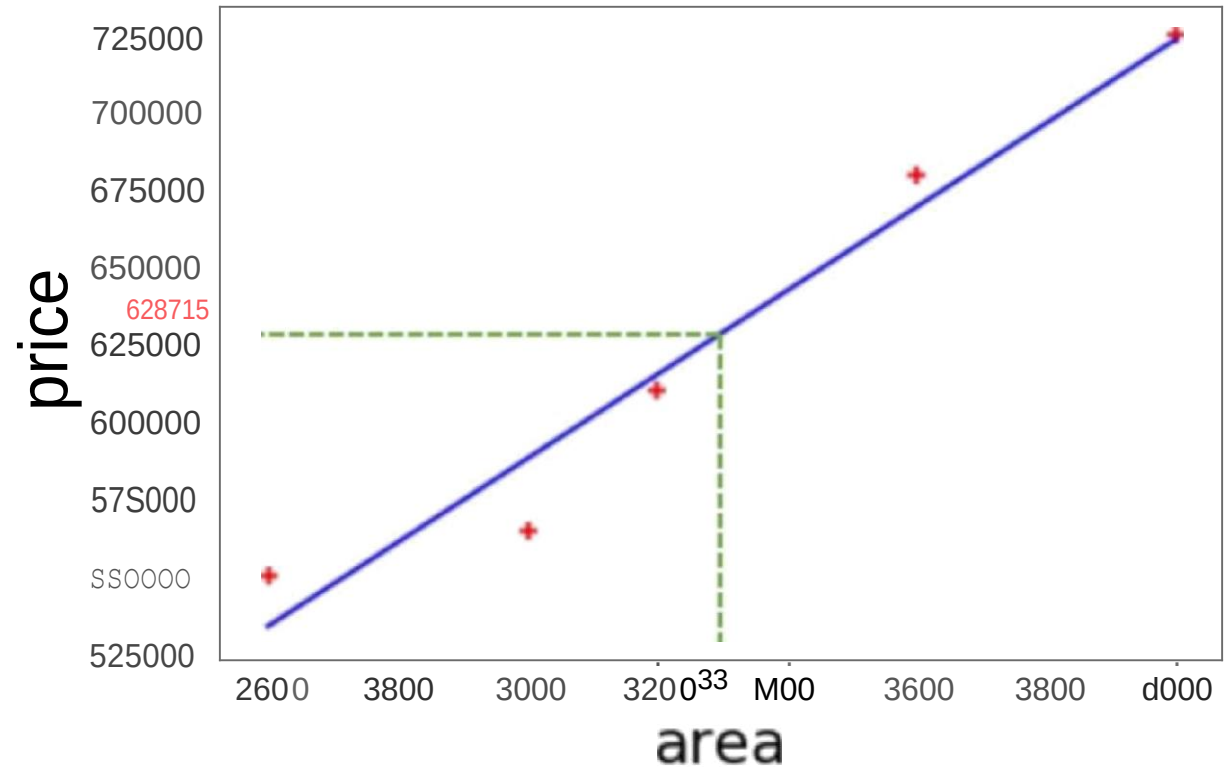
Home prices in Monroe Twp, NJ (USA)

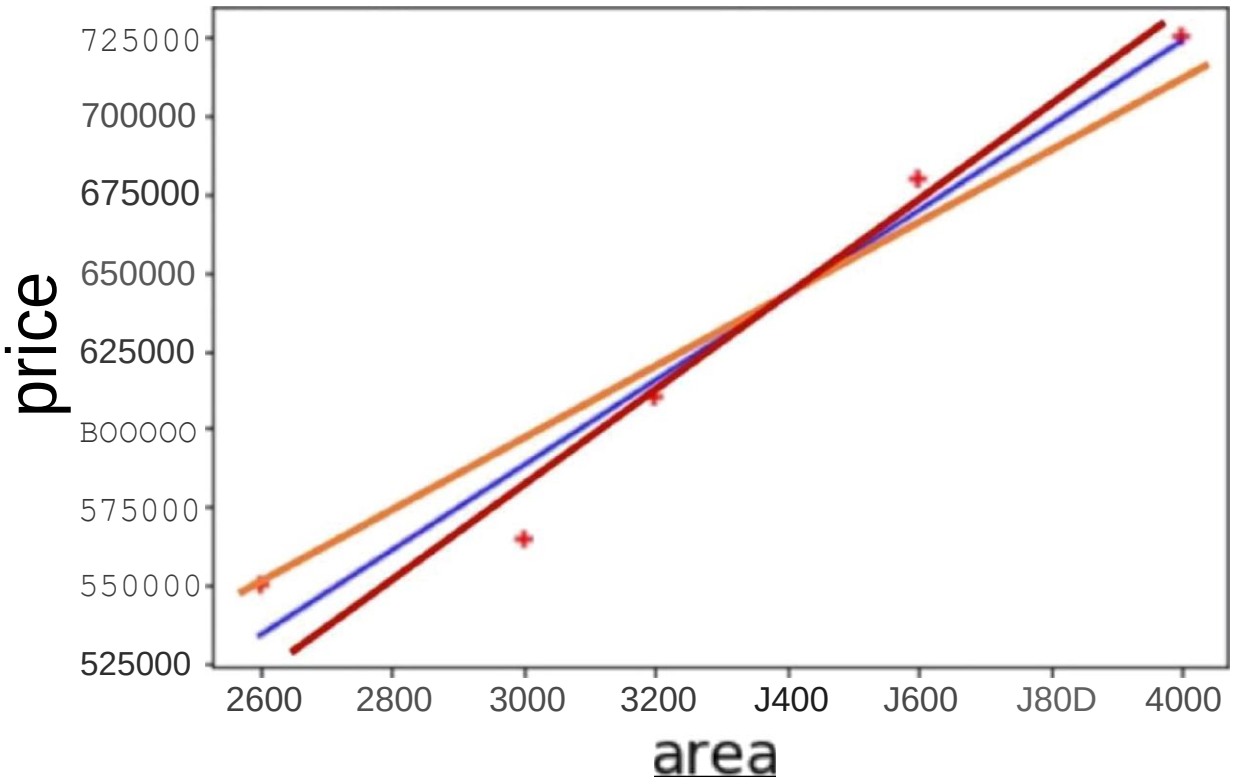
area	price
2600	SS0000
3000	565000
3200	610000
3600	680000
4000	725000

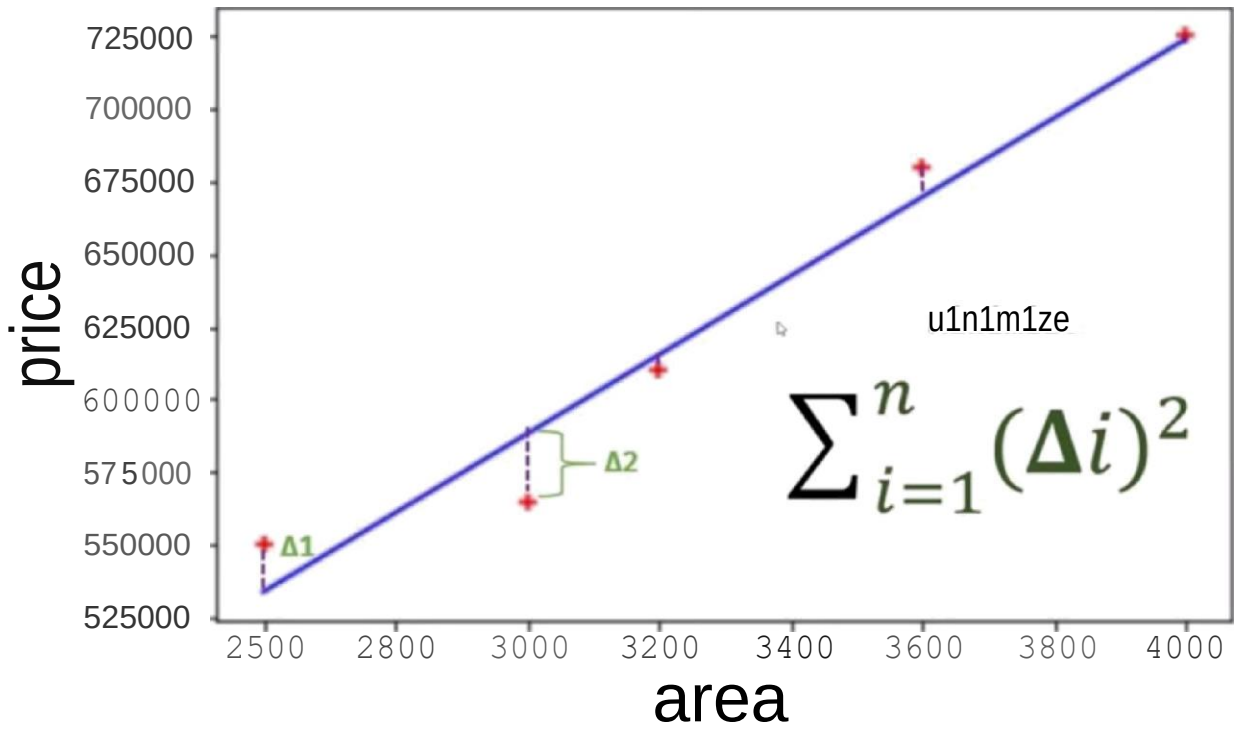
Given these home prices find out prices of homes whose area is,

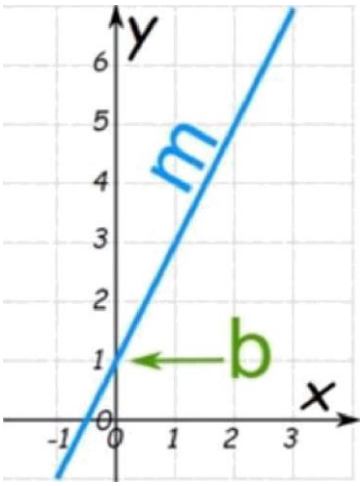
3300 square feet
5000 square feet




725000
700000
675000
650000
628715
625000
600000
57S000
SS0000
525000
price
2600
3800
3000
3200
33
M00
3600
3800
d000
area


725000
700000
675000
650000
625000
BOOOOO
575000
550000
525000
price
2600
2800
3000
3200
J400
J600
J80D
4000
area

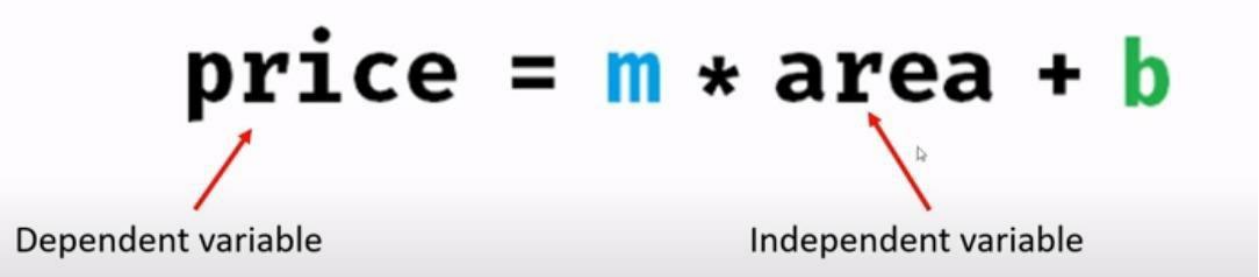




pr1ce = m + area + b

y = m x + b

Slope (or Gradient) Y Internet



Code explanation with screen shots:(also try to keep your code in GitHub repository and give link)

```
[1]  import pandas as pd

[2]  df=pd.read_csv("Salary_Data.csv")

[3]  df.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
[4]  X=df.iloc[:,:-1]
     y=df.iloc[:,-1]

[5]  X.head()
```

	YearsExperience
0	1.1
1	1.3
2	1.5
3	2.0
4	2.2

```
[6]
```

[1]
```
import pandas as pd
```

[2]
```
df=pd.read_csv("Salary_Data.csv")
```

[3]
```
df.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

[4]
```
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

[5]
```
X.head()
```

	YearsExperience
0	1.1
1	1.3
2	1.5
3	2.0
4	2.2

[6]
```
y.tail()
```

```
25    105582.0
26    116969.0
27    112635.0
28    122391.0
29    121872.0
Name: Salary, dtype: float64
```

[25]
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y)
```

[29]
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

[30]
```
import numpy as np
```

[32]
```
model.fit(X_train,y_train)
```

```
LinearRegression()
```

[34]
```
model.fit_intercept
```

```
True
```

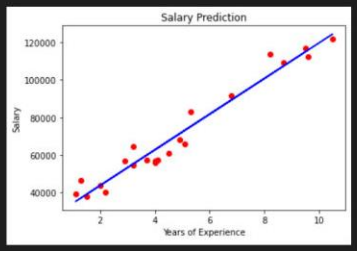
[35]
```
model.intercept_  #b
```

```
25151.09200962919
```

[23]
```
import matplotlib.pyplot as plt
```

[26]
```
plt.scatter(X_train,y_train,color='red')
plt.plot(X_train,model.predict(X_train),color='blue')
plt.title('Salary Prediction')
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
```

```
Text(0, 0.5, 'Salary')
```



```
[18]
model.intercept_  #b

24703.94447572597

[19]
model.coef_        #slope(m)

array([9505.02560239])

[20]
model.predict([[10]])        #y=mx+c

array([119754.20049962])

[21]
import joblib

[22]
joblib.dump(model,"model.sav")

['model.sav']
```

Github Link repository link:
https://github.com/Dhana10/SalaryPrediction

Application or use cases:

1. **House Price Prediction**
2. **Crop yield prediction**
3. **Predicting salary on years of experience**
4. **Sales Forcasting**
5. **Predicting the marks scored by students on basis of no of hours studied.**

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NBA, Approved by AICTE New Delhi)
Aziz nagar Gate, C.B. Post, Hyderabad-500 075.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE.

Algorithm name:

K NEAREST NEIGHBOURS CLASSIFICATION

Dataset Download link:

https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/ML0101ENv3/labs/teleCust1000t.csv

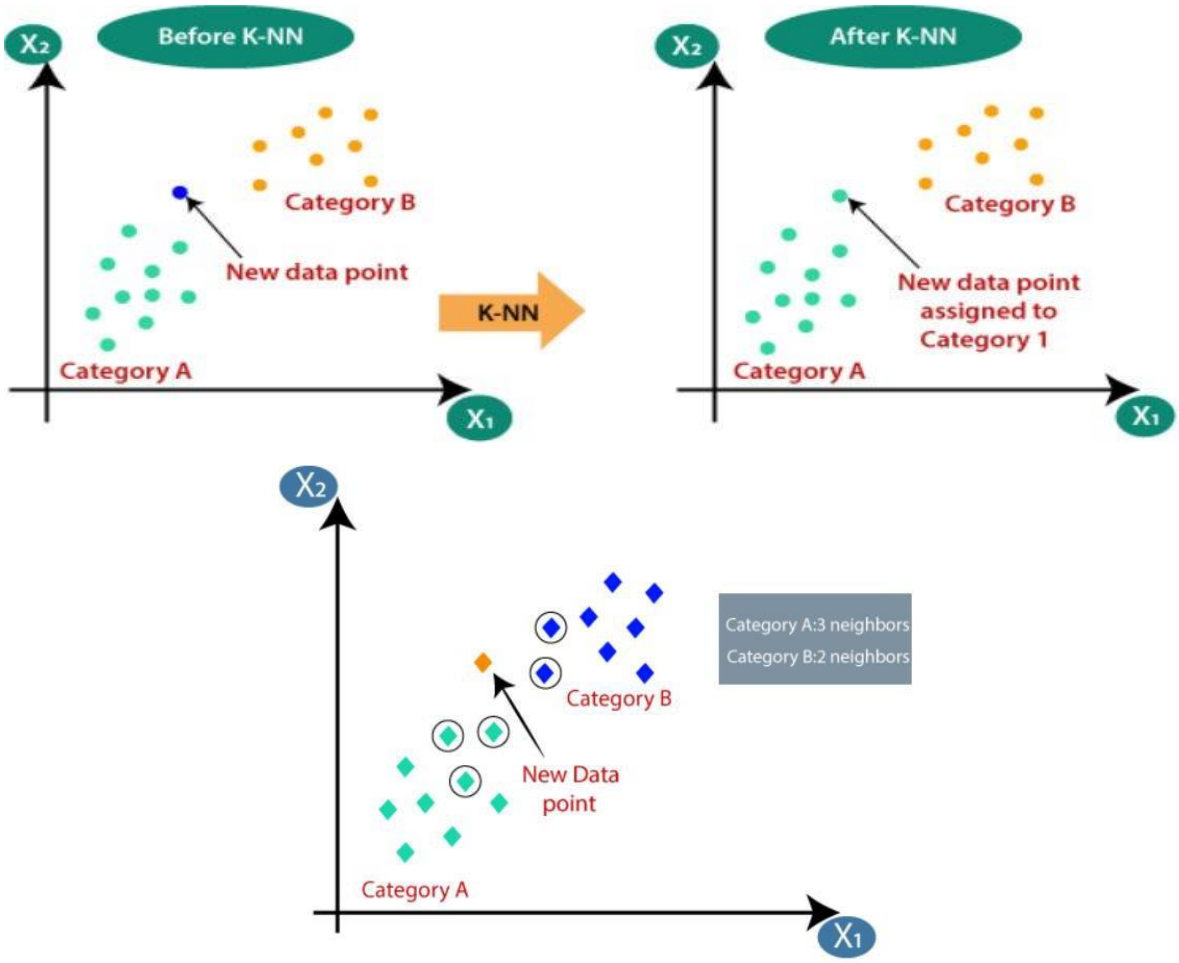
Algorithm explanation:

K-Nearest Neighbors is an algorithm for supervised learning. Where the data is 'trained' with data points corresponding to their classification. Once a point is to be predicted, it takes into account the 'K' nearest points to it to determine it's classification.

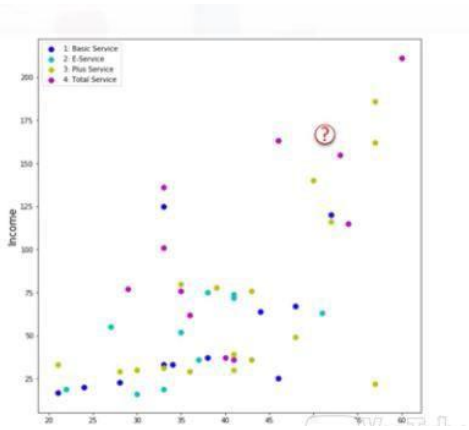
Algorithm :

- **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.

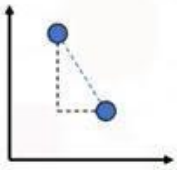
- **Step-6:** Our model is ready.



Mathematical part (Euclidean distance):



Customer 1	
Age	Income
54	190

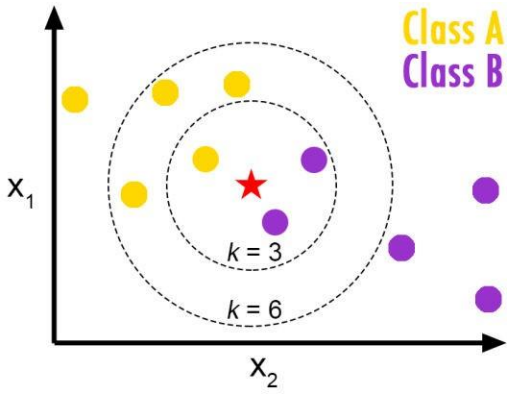


Customer 2	
Age	Income
50	200

$$\text{Dis}\,(x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

$$= \sqrt{(54-50)^2+(190-200)^2} = 10.77$$

Here's an visualization of the K-Nearest Neighbors algorithm.



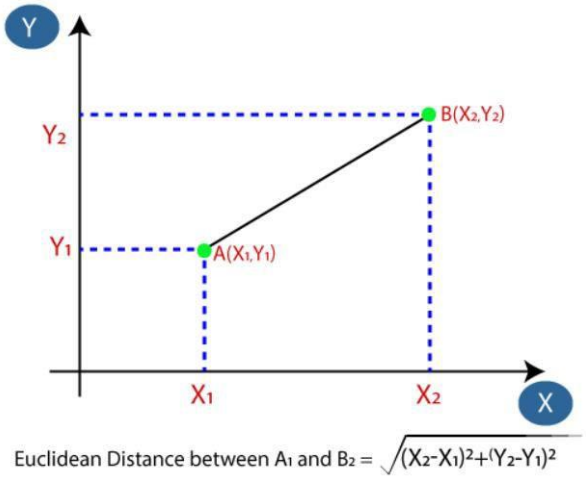
1. In this case, we have data points of Class A and B.

2. We want to predict what the star (test data point) is.

3. If we consider a k value of 3 (3 nearest data points) we will obtain a prediction of Class B. Yet if we consider a k value of 6, we will obtain a prediction of Class A.

In this sense, it is important to consider the value of k. But hopefully from this diagram, you should get a sense of what the K-Nearest Neighbors algorithm is. It considers the 'K' Nearest Neighbors (points) when it predicts the classification of the test point.

Distance is calculated as follows:



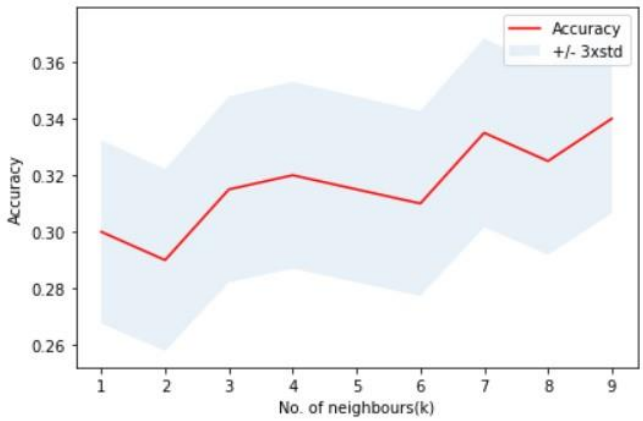
Code explanation with screen shots:

```
df:pd.read_csv(r"C:*,Users\komsu\Downloads\teleCust1OOOt.csv")
df.head()
```

	region	tenure	age	marital	address	income	ed		retire	gender	reside	custcat
D	2	13	44	1	9		4	5	0.D	D	2	1
1	S	11	33	1	7	136.D	5	5	0.D	D	6	4
2	3	68	52	1	24		1	29	0.D	1	2	3
3	2	33	33	D	12	33.D	2	D	0.D	1	1	1
4	2	23	3D	1	9	30.D	1	2	0.D	D	4	3

```
from sklearn.neighbors import KNeighborsClassifier

k=4
neigh:KNeighborsClassifier(n neighbors: k).fit(X train,ytrain)
neigh

KNeighborsClassifier(n neighbors=4)

yhat=neigh . pred1ct (X test)
yhat [0 : 5 ]

array([1, 1, 3, Z, 4], dtype=int64)

from sklearn import metrics
print("Train set accuracy: ",metrics.accuracy score(y train,neigh.predict(X train)))
point ( "Test set ar c uracy : ",met r1c s . accuracy scone(y test, yhat) )

Train setaccuracy: 0.5475
Test setaccuracy: 0.32
```

```
plt.plot(range(1,ks),mean_acc,'r')
plt.fill_between(range(1,ks),mean_acc-1* std_acc,mean_acc+1 * std_acc, alpha=0.10)
plt.legend(("Accuracy","+/- 3xstd"))
plt.ylabel("Accuracy")
plt.xlabel("No. of neighbours(k)")
plt.tight_layout()
plt.show()
```




```
print("The best accuracy is with k=",mean_acc.argmax()+1, " which is ",mean_acc.max())
```

```
The best accuracy is with k= 9  which is  0.34
```

https://github.com/Suchetha21/KNN-classifier

Application or use cases:

- Forecasting stock market: Predict the price of a stock, on the basis of company performance measures and economic data.
- Currency exchange rate
- Bank bankruptcies
- Understanding and managing financial risk
- Trading futures
- Credit rating
- Loan management
- Bank customer profiling
- Money laundering analyses

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.

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NBA, Approved by AICTE New Delhi)
Aziz nagar Gate, C.B. Post, Hyderabad-500 075.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE.

Algorithm name:

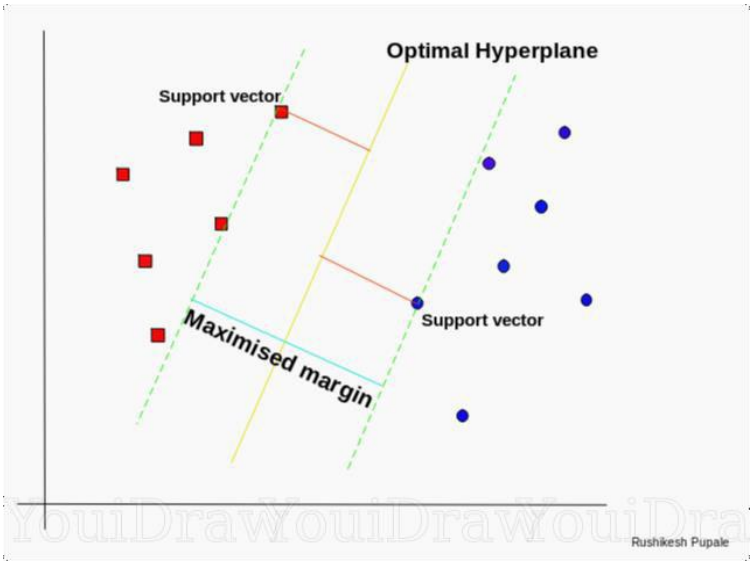
SUPPORT VECTOR MACHINE(SVM)

Dataset Download link:

https://github.com/Sindhuja-AI/SVM/blob/main/spam%20(1).csv

Algorithm explanation:

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each featurebeingthevalueofaparticularcoordinate.



SVM Hyperplane

There may be multiple lines/decision boundaries to segregate the classes in n-dimensional space. Still, we want to search out the simplest decision boundary that helps to classify the information points.

This best boundary is considered to be the hyperplane of SVM. The dimensions of the hyperplane rely on the features present within the dataset. These features suggest if there are 2 target labels in the dataset.

Then the hyperplane is going to be a line. And if there are 3 target labels, then the hyperplane is going to be a 2-dimension plane. We always create a hyperplane that provides maximum margin. This margin simply means there should be a maximum distance between the data points.

SVM Support Vectors

Support vectors are defined as the data points, which are closest to the hyperplane and have some effect on its position. As these vectors are supporting the hyperplane, therefore named as Support vectors.

Kernel: Kernel is used due to the set of mathematical functions used in the Support Vector Machine that provides the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces.

Types of SVMs:

There are two different types of SVMs, each used for different things:

- Simple SVM: Typically used for linear regression and classification problems.
- Kernel SVM: Has more flexibility for non-linear data because you can add more features to fit a hyperplane instead of a two-dimensional space

Popular SVM Kernel Functions:

Linear Kernel

It is the most basic type of kernel, usually one dimensional nature. It proves to be the best function when there are lots of features. The linear kernel is mostly preferred for text-classification problems as most of these kinds of classification problems can be linearly separated.

Linear kernel functions are faster than other functions.

Linear Kernel Formula

F(x, xj) = sum( )Here, x, xj

represents the data you're trying to classify. **Polynomial Kernel**

It is a more generalized representation of the linear kernel and is not as preferred as other kernel functions as it is less efficient and accurate.

Polynomial Kernel Formula

$$F(x, x_j) = (x.x_j+1)^d$$

Here '.' shows the dot product of both the values, and d denotes the degree. F(x, xj) representing the decision boundary to separate the given classes.

Gaussian Radial Basis Function (RBF)

It is one of the most preferred and used kernel function in svm. It is usually chosen for non-linear data. It helps to make proper separation when there is no prior knowledge of data.

Gaussian Radial Basis Formula

$$F(x, x_j) = \exp(-gamma * ||x - x_j||^2)$$

The value of gamma varies from 0 to 1. You have to manually specify the value of gamma in the code. The most preferred value for gamma is 0.1.

Sigmoid Kernel

It is mostly preferred for neural networks. This kernel function is similar to a two-layer perceptron model of the neural network, which works as an activation function for neurons. It can be shown as,

Sigmoid Kernel F(x, xj) =

tanh(αxay + c) Gaussian

Kernel

It is a commonly used kernel. It is used when there is no prior knowledge of a given dataset.

Gaussian Kernel Formula

$$k(x,y) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right)$$

Bessel function kernel

It is mainly used for removing the cross term in mathematical functions.

Bassel Kernel Formula

$$k(x,y) = \frac{J_{v+1}(\sigma||x-y||)}{||x-y||^{-n(v+1)}}$$

Here J is the Bessel function.

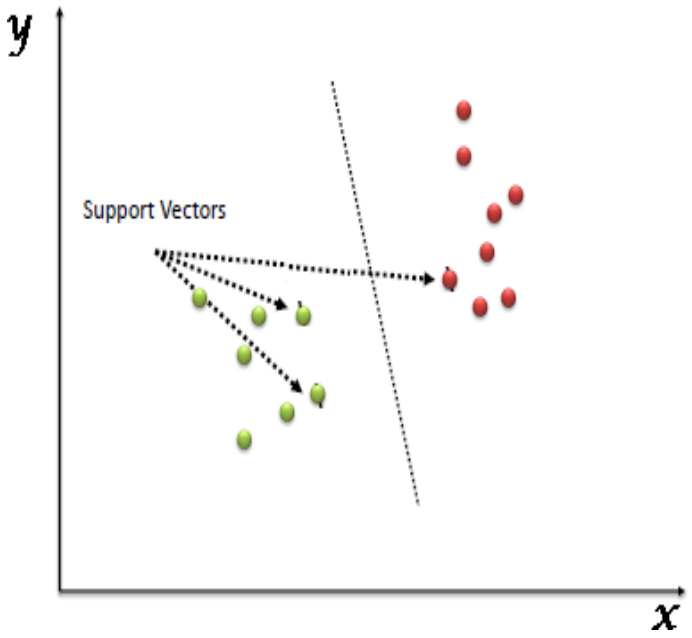
ANOVA kernel

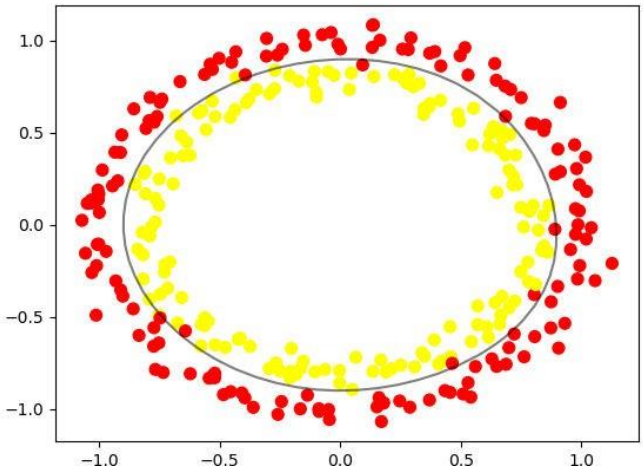
It is also known as a radial basis function kernel. It performs well in multidimensional regression problems.

Anova Kernel Formula

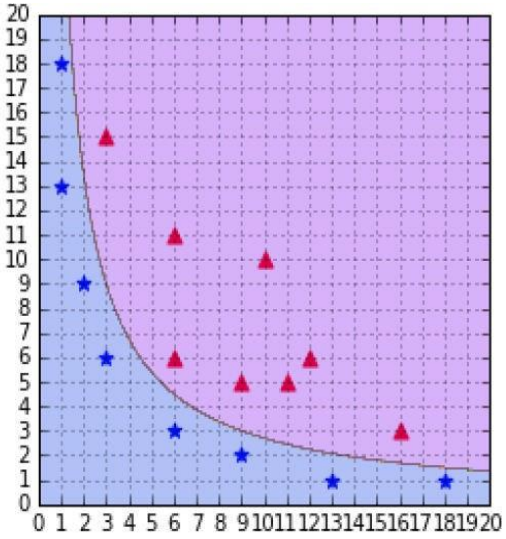
$$k(x,y) = \sum_{k=1}^{n} \exp(-\sigma(x^k - y^k)^2)^d$$

Here's an visualization of the Support Vector Machine:



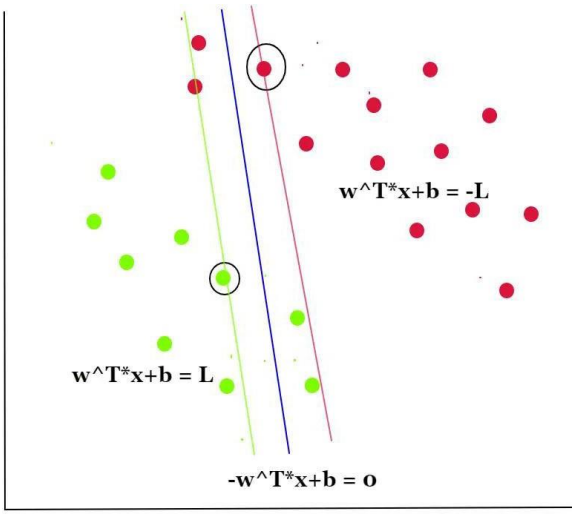


non-linear SVM using RBF kernel



A SVM using a polynomial kernel is able to separate the data (degree=2)

Mathematical part :



Now let's take two other pieces of information we know, namely that $w^T x_r + b = -L$ and $w^T x_g + b = L$ as both the red and green support vectors lies on their respective hyperplanes and thus satisfy their equations. Subtracting both equations we get

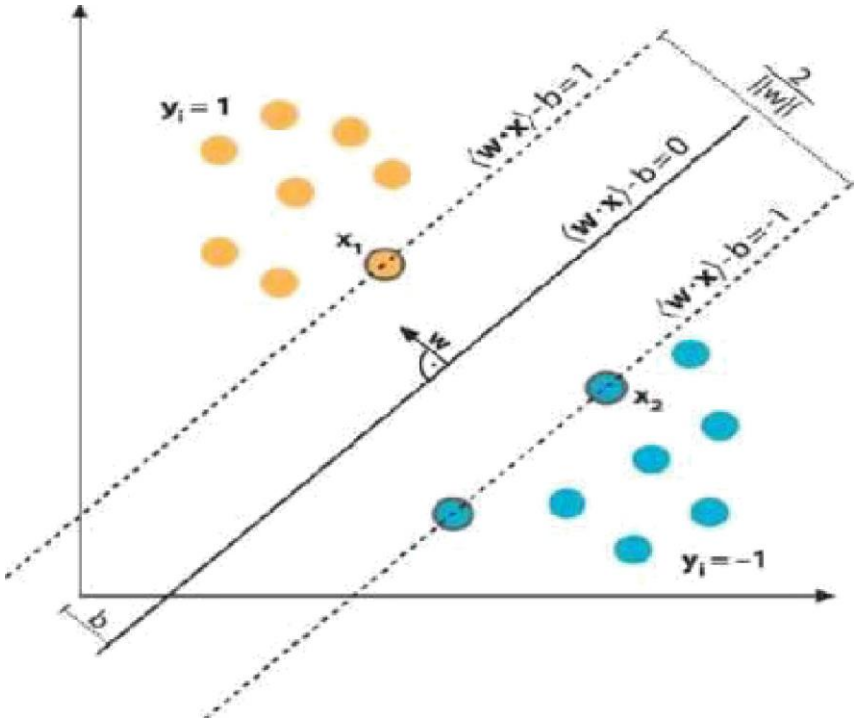
$$(w^T x_r + b = -L) - (w^T x_g + b = L) = w^T(x_g - x_r) = 2L$$

Dividing by $||w^T||$, we get

$$\frac{w^T}{||w^T||}(x_g - x_r) = \frac{2L}{||w^T||}$$

Amazing!!! We already proved the left side of the above equation was the magnitude of the purple vector, meaning the distance between both hyperplanes is $\frac{2L}{||w^T||}$ and that's what we have to maximize!

Consider here the value of L as 1



Jnymnn *term SVM optimization can be written as,*

i[y ; w. x;+b) —1=o:

then (• , y,)issupport vectors

then save the parameters w, b

else i[y ! w . x +b) —i>0:

then save the parameters w ,b

eJsei/ $y_i(w. x_f+b) —1<0:$

then update the parameters w ,b

Code explanation with screen shots:(also try to keep your code in GitHub repository and give link)

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

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import GridSearchCV
from sklearn import svm
```

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

```
data=pd.read_csv('spam (1).csv')
```

```
data.head()
```

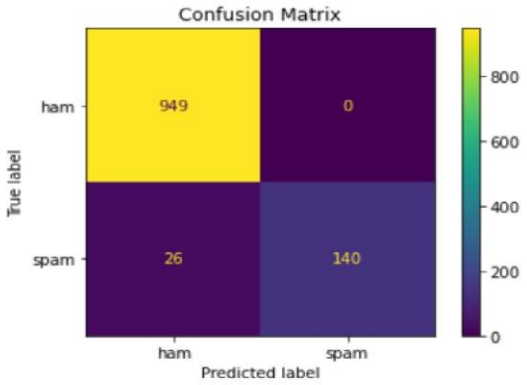
	Label	EmailText
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 10)
classifier.fit(X_train, y_train)
print(classifier.score(X_test,y_test))
```

```
0.9766816143497757
```

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier,X_test,y_test)
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```




```
X= oata['EzailText'].values
y= oath[ ' Label' .va1ues
```

```
array(['Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...',
       'ok lar. . . Joking off u oni. . . ' ,
       "Free entry In 2 a xkTy conp to min *A Cup final tkts 21st May 2005. Text FA tD 87121 to receive entry question(std txt
rate)T&C' s apply 68432810675over18 's",
       .   ,  'Pity,     was in nood for that. So. . . any other suggestions ? ' ,
       "The guy did soae bitching but I acted like i' d be interested in buying soneth1ng e1se next week ana he gave it  to us to

       'Roll. Its true to its name ' ], dtype=object)
```

```
y
```

```
array([ ' han', ' han' , 'span' , . . . , ' han' , ' han', ' han' ], otype=object)
```

```
frm skbarn.mOJel seection import tran test spit
X train, X test, y train, y test = train test split( x, y, test size:0.2, random state=B)
```

```
cv = CountVectorizer ()
X train = cv. f1t transform(X train)
X test - cv. transform(X test)
```

```
print(x train)
```

```
  (0, 4711)     Z
  (0, 6732)     1
  (0, 36g9)     2
  (0, 3827)     1

  (0, 3203)     1
```

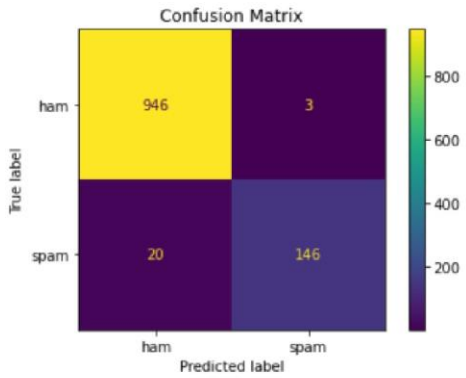
```
tuned parameters= {'kernel': ['rbf','linear'], 'gamma': |1e-3, 1e-4],
                   'C': [1, 18, 100, 1008]}

nodel= GridSearchCV(svn.SvC(), tuned parameters)
nodel.fit(X train,y train)

GridSearchCV(est1mator=SVC(),
             param grid=( ' C' : [1, 16, 100, 1000], ' gama' : [0.061, 0.6001],
                          'kernel' : [ 'rbf', ' linear' ]))

 print(model. score(y test,y test))

0.979372197309417

y pred=0ode1. pred1ct(y test)

array(['ham', ' ham' , ' han', . .., 'ham', ' ham', 'han'], dtypembject)

accuracy=accuracy score(y test,y pred)*160
accuracy

97.9372197309417
```

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

```
array([  946,    3],
       [ 20, 146]], dtype=int64)
```

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(model,X_test,y_test)
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```



https://github.com/Sindhuja-AI/SVM/blob/main/SVM.ipynb

Application or use cases:

- Face detection – SVM classify parts of the image as a face and non-face and create a square boundary around the face.
- Classification of images :

  Use of SVMs provides better search accuracy for image classification. It provides better accuracy in comparison to the traditional query-based searching techniques.

- Protein fold and remote homology detection
- Handwriting recognition
- Generalized predictive control(GPC)
- Text and hypertext categorization
- Cancer Diagnosis and Prognosis

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NBA, Approved by AICTE New Delhi)
Aziz nagar Gate, C.B. Post, Hyderabad-500 075.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE.

Algorithm name:

K fold cross validation

Dataset Download link:

https://github.com/shriakshita/K-Fold-cross-validation.git

Algorithm explanation:

Cross-validation is a statistical method used to estimate the skill of machine learning models.

It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

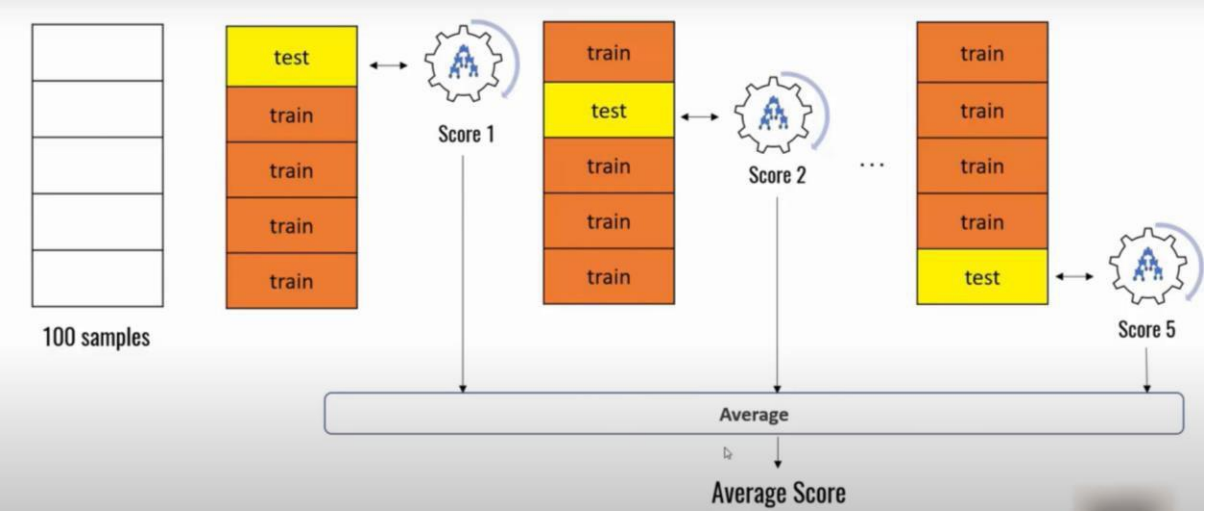
Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.



Configuration of k

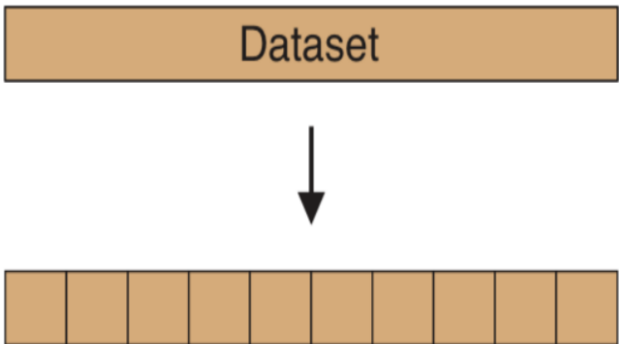
The k value must be chosen carefully for your data sample.

A poorly chosen value for k may result in a mis-representative idea of the skill of the model, such as a score with a high variance (that may change a lot based on the data used to fit the model), or a high bias, (such as an overestimate of the skill of the model).

Three common tactics for choosing a value for k are as follows:

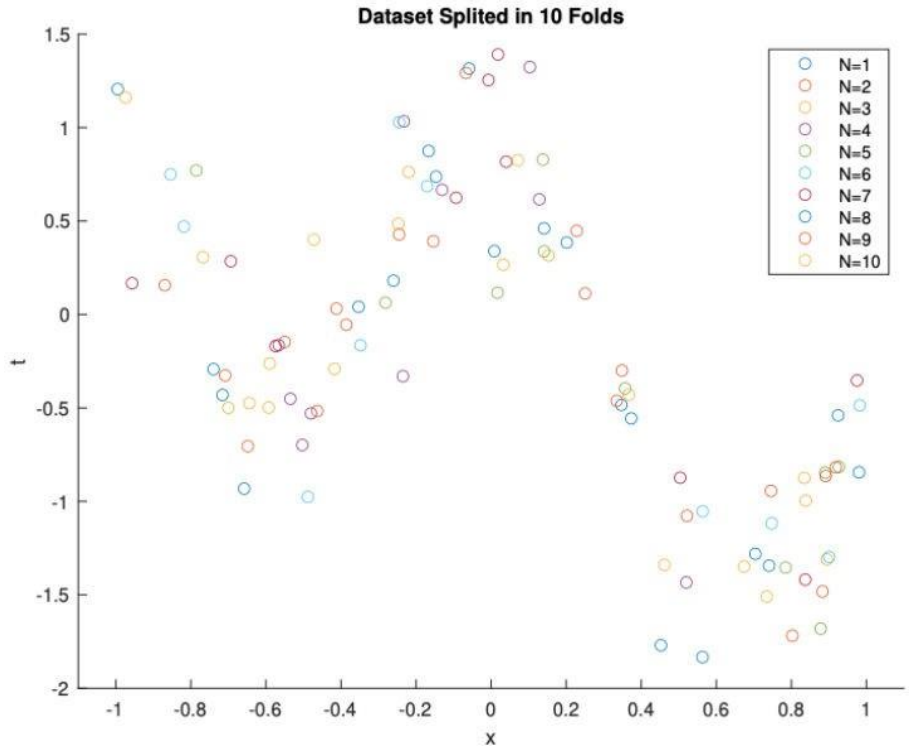
- **Representative**: The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.
- **k=10**: The value for k is fixed to 10, a value that has been found through experimentation to generally result in a model skill estimate with low bias a modest variance.
- **k=n**: The value for k is fixed to n, where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called leave-one-out cross-validation.

Mathematical part:



Splitting of data into 10 folds

Algorithm for 10-fold cross validation



The solution of the ERM is defined in the equation below.

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$$

The vector **w** is our polynomial coefficients, **X** is the design matrix and **t** is the vector of outputs.

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \dots & x_1^W \\ 1 & x_2^1 & x_2^2 & \dots & x_2^W \\ \dots & & & & \\ \dots & & & & \\ 1 & x_{100}^1 & x_{100}^2 & \dots & x_{100}^W \end{bmatrix}$$

For the sake of simplifying our calculations, we'll solve the linear equation for **w** manually in MATLAB **(**by multiplying and inverting the matrices**).** Be aware this is not the most efficient way of solving linear equations. Also, we'll analyse the polynomials of order W = 1, 2, …, 30.

So, for 10-fold cross-validation, we execute the process of ERM 10 times in a loop and storage all test scores of each execution in the cell array **E**. The empirical square loss is calculated by the equation below.

$$\mathcal{L}_S(\mathbf{y_w}) = \frac{1}{N}\frac{1}{2}\|\mathbf{Xw} - \mathbf{t}\|^2$$

In this process, we are using one-fold for testing and the remaining 9 folds for training. Inside this loop, I also compute both the training and the testing design matrices.

Code explanation with screen shots:

https://github.com/shriakshita/K-Fold-cross-validation.git

```
In [1]: frs st1earn.linear nxiel irport Logisticiegression
        frs stlearn.sn report svt
        frs stlearn.ensec£le inpirt 8aréonForest£lassifier
        inport nocpy as np
        frs  st1earn.datasets in|iort load ti¿iti
        import matplotlib.pyplot as plt
        digits =1oaé digits()
```

```
I  [ ] : frs stlearn.odel selection inport train test split
         I trair,I test,y train,y test=train test split(digits.data,digits.target,test size=8.S)
```

logistic regression

```
In [3]: lr = t0gisticsegressior(S0lVer='liblinear',crlti class='ovr')
        1r.fit(I train, y train)
        1r.score(I test, y test)
```

```
Out[3]: 8.9 216987675194fi6
```

svm (suppoR Vectormachine)

```
In [4]: svn - SvC(ganIila-'auto')
        svn.fit(X train, y train)
        svn.score(X test, y test)
```

```
08t[4]: 8. I72SI58I78886763
```

r4ndomforestcl4ssifier

```
In [S]: rd = 8aodonForesttlassifier(r estinators=48)
        rLfit(I train, y train)
        rf.score(I test, y test)
```

```
Out[5]: 8.9699666195884316
```

K **FOLD VALIDATION**

```
In [6): Iron sk1earn.itode1_seJect1on inport KFold
        kf = KFOld(u splits=10)
        kf
```

Out[6] : KFold(n_splits-10, *randon_state-IJone,* shuffle-False)

```
In [7] : for train_Tndex, test_index :in kf. split( [1,2,3,4,5,6,7,8,9,16]) :
             print(train index, test index)
```

```
[1 z 1 4 s 6 7 g sj [e]
[B 2 3 4 5  6 7 8 9j  [l]
[B 1 3 4 5  6 7 8 9]  [z]
[B 1  2 4 5  6 7 8  9j [s]
[B 1 2 3  5 6 7 8  9j  [4]
[0 1 2  3 4 6 7  g  sj  [sj
[B 1  2 3 4 5 7 8 9j  [6]
[B 1  2 3 4 5    8  9] [7]
[B 1  2 3 4 5 6 7 9j  [8]
[0 1 2 3 4 5 6 7 8] [9]
```

```
In [8) : def get score(node1, X train, X test, y          y test) :
             model.fit(x train, y train)
           return rode1. score(X test, y test)
```

```
In [9] : Iron sk1earn.liode1_seJect1on inpol-t StratifiedKFold
         folds 5tratifiedKFotd(n splits=18)

         scores_logTstic- []
         scores svn = [j
         scores rd = []

         for train index, test index in folds. split(dig1ts.data, digits.target):
               train, X test, y train, y test digits.data[train Index], dIg1ts.data[test Index],dIgits.target[train Index], digits. target[test 1
         ıdex
             scores_logistic.appeud(get_score(LogisticRegressiOn(sOlver='liblinear',multiclass='ovr'), X_train, X_test, y_train, y test))
             scores svn. append(get score(SVc(ganna-'auto'), X train, X test, y train, y test))
             scores rf.append(get score(RandonForestc1assifier(n estinators=40), X train, X test, y train, y test))
```

```
In      : SCO°2S_lOgiStiC

Out[10]: [0.9111111111111111,
          0.9388888888888889,
          0.89444444444*4145,
          0.8666666666666667,

          0.9722222222222222,
          0.9777777777777777,
          0.9A97206703910615,
          0.8603351955307262,
          0.9AA1340782122905)

In [11]: scores_svm

         [0.4388888888888889,
          0.5777777777777777,
          0.46B6666b66666667,
          0.3888888888888889,
          0.4722222222222222,
          0.4,
          0.5055555555553555,
          0.5754189944134078,
          0.5586592178770949,
          0.43575418994413406]



Out[12]: [0.9055555555555556,
          0.9611T1111111T111,
          0.9333333333333333,
          0.9166666666666666,

          0.9833333333333333,
          0.9777777777777777,
          0.972067039106T452,
          0.9385474860335196,
          0.9273743016750777)
```

cross_val_score function

```
In [13]: from sklearn.model_selection import cross_val_score
```

for logistic regression

```
In [14]: cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'), digits.data, digits.target,cv=10)
Out[14]: array([0.91111111, 0.93888889, 0.89444444, 0.86666667, 0.94444444,
                0.97222222, 0.97777778, 0.94972067, 0.8603352 , 0.94413408])
```

for svm

```
In [15]: cross_val_score(SVC(gamma='auto'), digits.data, digits.target,cv=10)
Out[15]: array([0.43888889, 0.57777778, 0.46666667, 0.38888889, 0.47222222,
                0.4       , 0.50555556, 0.57541899, 0.55865922, 0.43575419])
```

randomforestclassifier

```
In [16]: cross_val_score(RandomForestClassifier(n_estimators=40),digits.data, digits.target,cv=9)
Out[16]: array([0.895     , 0.96      , 0.945     , 0.95      , 0.98      ,
                0.965     , 0.96984925, 0.94974874, 0.93467337])
```

parameter turning

```
In [115]: scores1 = cross_val_score(RandomForestClassifier(n_estimators=5),digits.data, digits.target, cv=10)
          np.average(scores1)
Out[115]: 0.8692209807572937

In [80]: scores2 = cross_val_score(RandomForestClassifier(n_estimators=20),digits.data, digits.target, cv=10)
         np.average(scores2)
Out[80]: 0.9304345127250155

In [81]: scores3 = cross_val_score(RandomForestClassifier(n_estimators=30),digits.data, digits.target, cv=10)
         np.average(scores3)
Out[81]: 0.9410148975791433

In [82]: scores4 = cross_val_score(RandomForestClassifier(n_estimators=40),digits.data, digits.target, cv=10)
         np.average(scores4)
```

Application or use cases:

Applications of Cross-Validation

- This technique can be used to compare the performance of different predictive modeling methods.
- It has great scope in the medical research field.
- It can also be used for the meta-analysis, as it is already being used by the data scientists in the field of medical statistics.

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NBA, Approved by AICTE New Delhi)
Aziz nagar Gate, C.B. Post, Hyderabad-500 075.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE.

Algorithm Name: Decision Tree

We are well known with trees in data structures just a quick recap.

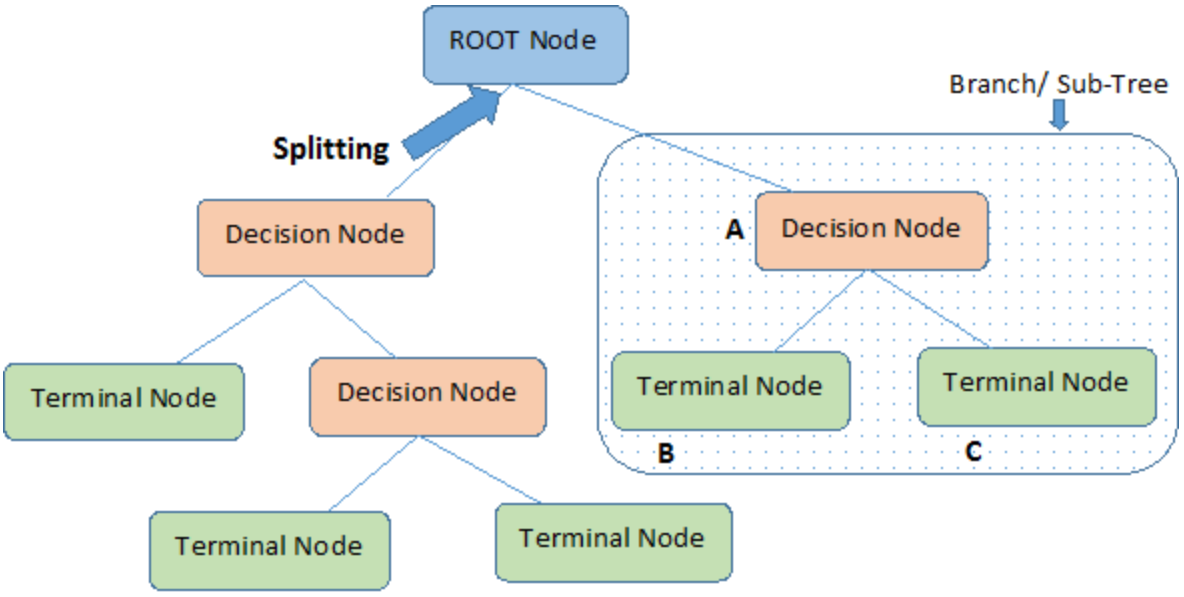
Tree Definition: A tree is a non-linear data structure in which items are arranged in a hierarchical manner.it is a very flexible and powerful data structure that can be used for a variety of applications.

This tree concept in data structures helps to develop ML algorithm Decision Tree.

Important Terminology related to Tree based Algorithms

Let's look at the basic terminology used with Decision trees

1. **Root Node:** It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
4. **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.



Note:- A is parent node of B and C.

Reason to use decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Explanation: The decision tree Algorithm belongs to the family of supervised machine learning algorithms. It can be used for both a classification problem as well as for regression problem.
Decision tree is a type of supervised learning algorithm (having a predefined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on the most significant splitter / differentiator in input variables.

Algorithm:

- **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 call the final node as a leaf node.

Types of Decision Trees:

Types of decision trees are based on the type of target variable we have. It can be of two types:

1. **Categorical Variable Decision Tree:** Decision Tree which has categorical target variable then it called as categorical variable decision tree. Example:- In above scenario of student problem, where the target variable was "Student will play cricket or not" i.e. YES or NO.
2. **Continuous Variable Decision Tree:** Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.

Example:- Let's say we have a problem to predict whether a customer will pay his renewal premium with an insurance company (yes/ no). Here we know that income of customer is a significant variable but insurance company does not have income details for all customers. Now, as we know this is an important variable, then we can build a decision tree to predict customer income based on occupation, product and various other variables. In this case, we are predicting values for continuous variables.

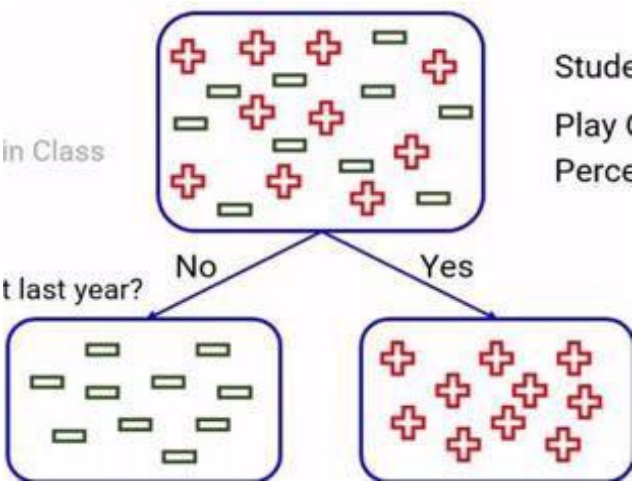
Are tree based algorithms better than linear models?

"If I can use logistic regression for classification problems and linear regression for regression problems, why is there a need to use trees"? Many of us have this question. And, this is a valid one too.

Actually, you can use any algorithm. It is dependent on the type of problem you are solving. Let's look at some key factors which will help you to decide which algorithm to use:

1. If the relationship between dependent & independent variables is well approximated by a linear model, linear regression will outperform tree based models.
2. If there is a high non-linearity & complex relationship between dependent & independent variables, a tree model will outperform a classical regression method.
3. If you need to build a model which is easy to explain to people, a decision tree model will always do better than a linear model. Decision tree models are even simpler to interpret than linear regression!

   Ideal Split: split which exactly separates data points as shown in below figure.

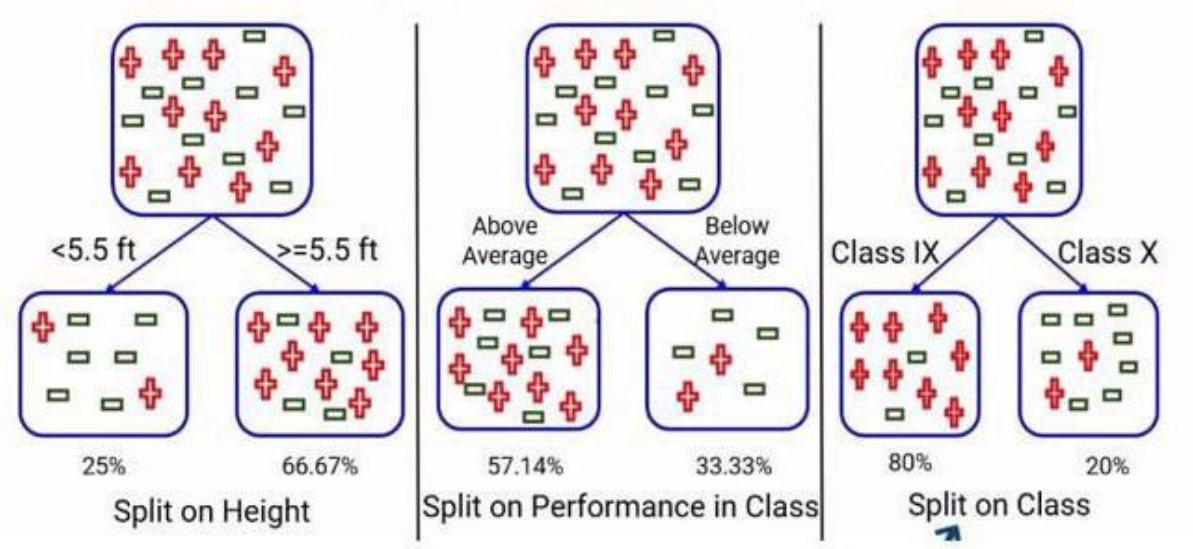


How does a tree based algorithm decide where to split?

Decision trees use multiple algorithms to decide to split a node in two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that purity of the node increases with respect to the target variable. Decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

Mathematical formulas or techniques to decide the best split .Let us consider one example and perform formulas on it.

Example: There are 20 students. Students who play cricket are denoted as red and do not play cricket are denoted with green.



From the figure we can understand, for example student will play cricket or not .We have a root node with 20 students, 10 red plays cricket and 10 do not play cricket. We split based on different features like height of students, performance in class, and also based on the class students belong to.

By just observing we can directly say split based on class is best because split more accurately compared to the other two i.e:80 and 20 percent. Then how can we achieve the same thing with our model here comes the role of techniques to split.

There are 3 techniques we can use any of these

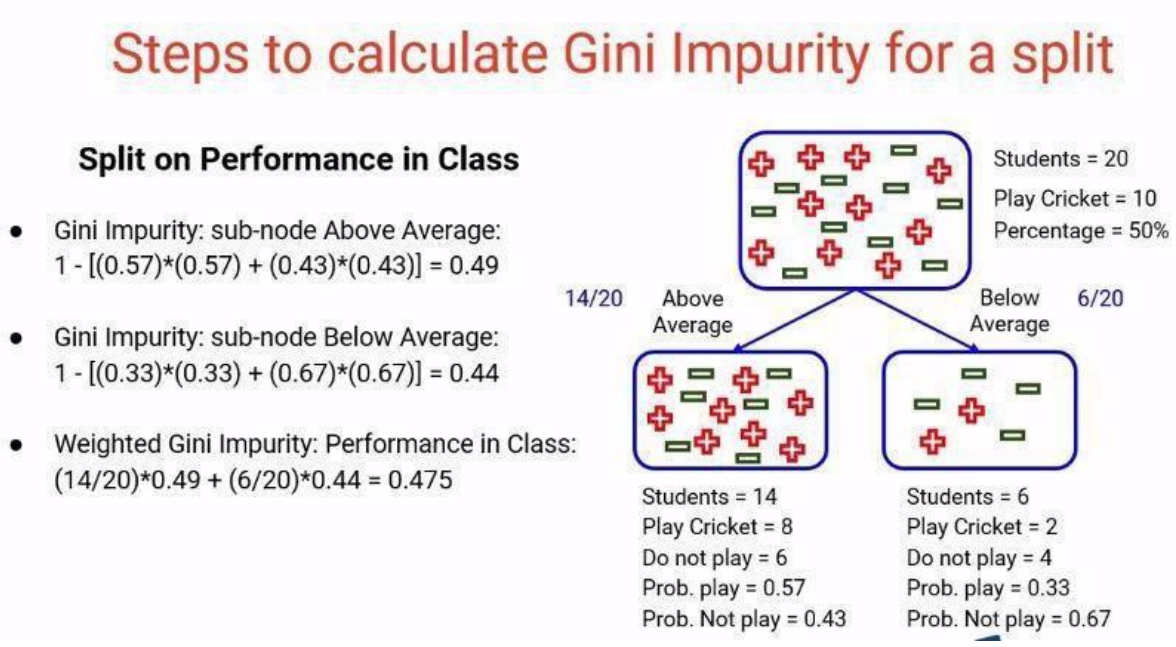
1. Gini Impurity

Gini Impurity=1-Gini

Calculate Gini for sub-nodes, using the formula sum of square of probability for success and failure (p^2+q^2).

Gini says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

1. It works with categorical target variables “Success” or “Failure”.
2. It performs only Binary splits
3. Higher the value of Gini higher the homogeneity.
4. CART (Classification and Regression Tree) uses the Gini method to create binary splits.



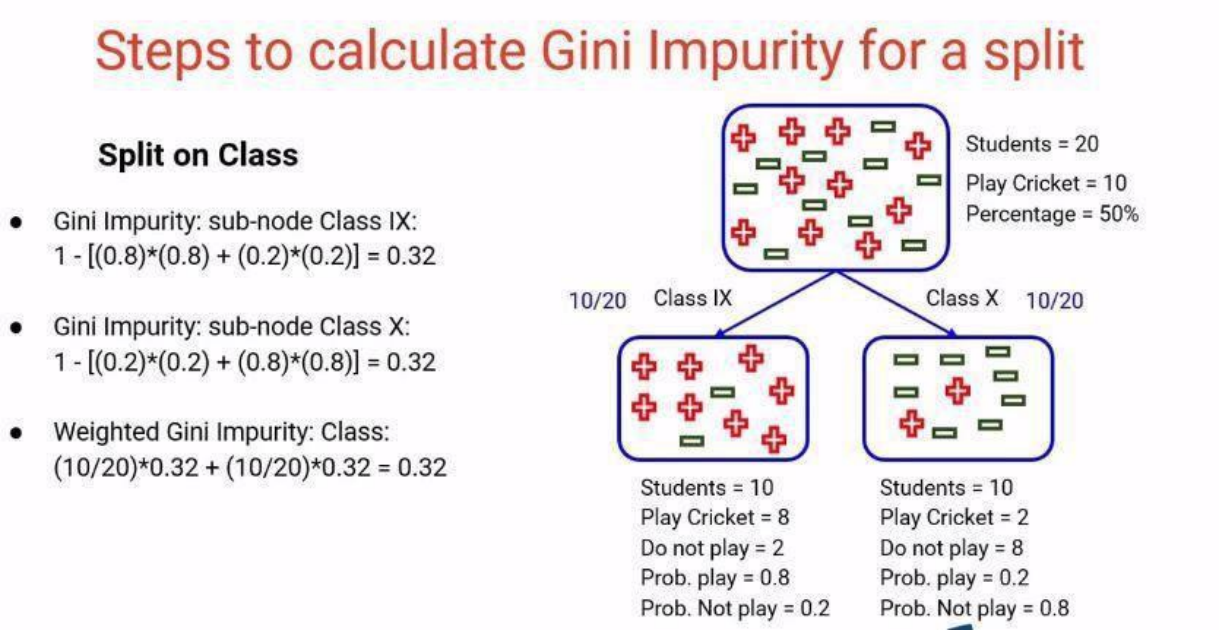
The above picture shows how we calculate Gini Impurity. First we calculate both left and right sub nodes individually and add both with their weights.

Weights are taken as no of data points on that node to total no of data points.

For both sub nodes we need to calculate probability for playing and not playing and find Gini= (p^2+q^2). And subtract from 1 to find Gini impurity.

As we are done for only one split based on performance of students.

In the similar manner we can find weighted Gini impurity value for split on class the student belongs to the below figure gives that



By just observing Gini impurity values we can say lower the Gini impurity higher the homogeneity and purity.so as we compare Gini impurity 0.32 is lesser compare to 0.475 then we can conclude split based on class is best split the same thing we achieved by observing 3 splits on 1 figure class splits 80-20 percent.
Gini is used for only binary split so we have chi-square

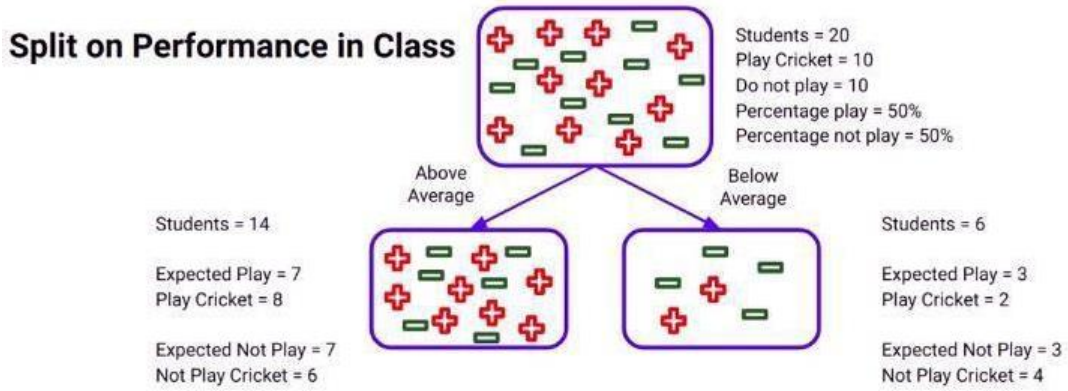
2. Chi-Square

It is an algorithm to find out the statistical significance between the differences between sub-nodes and parent nodes. We measure it by sum of squares of standardized differences between observed and expected frequencies of target variable.

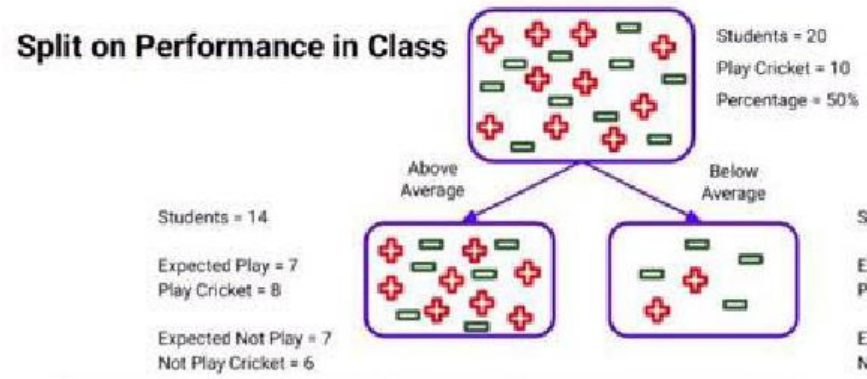
1. It works with categorical target variables “Success” or “Failure”.
2. It can perform two or more splits.
3. Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.

We need to calculate draw tables for each split table is chi table and higher chi value is considered as best split.

Steps to calculate Chi-Square for a split



Chi-Square = √[(Actual - Expected)² / Expected]



Node	Actual Play	Actual Not Play	Expected Play	Expected Not Play	Deviation Play	Deviation Not Play	Chi-Square (Play)	Chi-Square (Not Play)
Above Average	8	6	7	7	1	-1	0.38	0.38
Below Average	2	4	3	3	-1	1	0.58	0.58

Tota1=0.38+0.38+0.58+0.58=1.92

Chi-table for split on Class:

Node	Actual Play	Actual Not Play	Expected Play	Expected Not Play	Deviation Play	Deviation Not Play	Chi-Square (Play)	Chi-Square (Not Play)
IX	8	2	5	5	3	-3	1.34	1.34
X	2	8	5	5	-3	3	1.34	1.34

Total=1.34+1.34+1.34+1.34=5.36

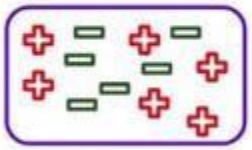
5.36 is higher compare to 1.92 so we can conclude split on class is best split again so we have achieved same with Gini and chi-square and also by observation we got same result

3. Information Gain:

Information Gain=1- Entropy

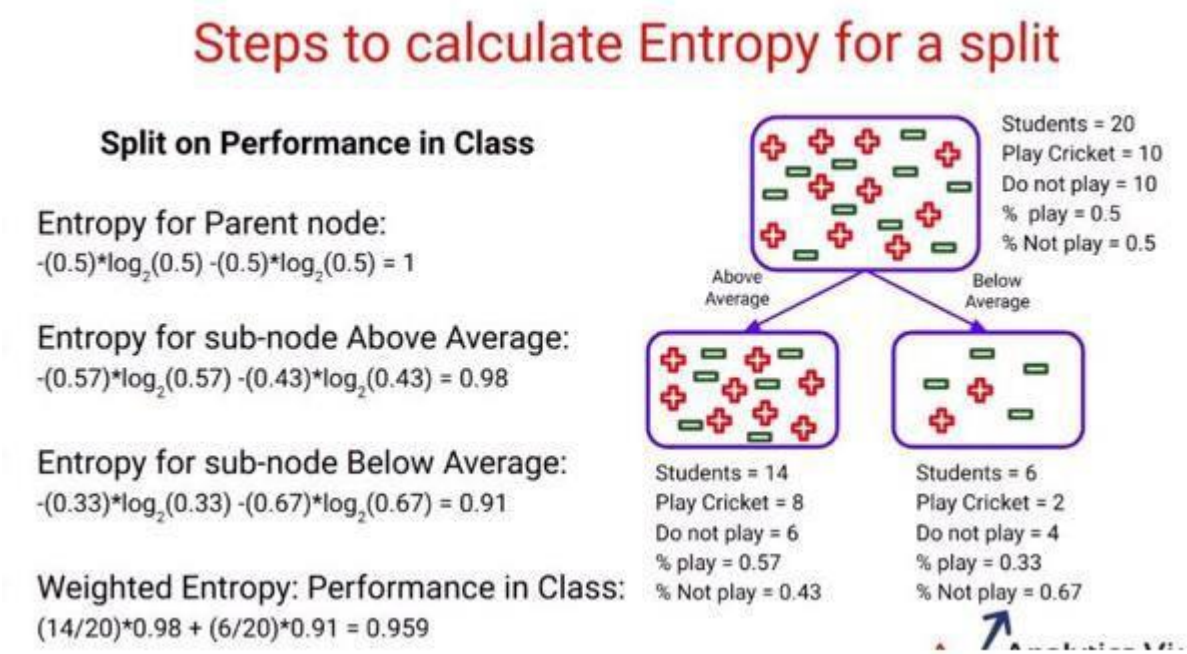
Entropy

$$- p_1 * \log_2 p_1 - p_2 * \log_2 p_2 - p_3 * \log_2 p_3 - ..... - p_n * \log_2 p_n$$



% Play = 0.50
% Not play = 0.50

Entropy = - (0.5) * log2(0.5) - (0.5) * log2(0.5)



We need to perform same calculation for each split then lower entropy is taken as best split.

So far this 3 techniques are for classification (target variable is categorical and discrete) now we will see for regression problems (target value is continuous and numerical).

Reduction in Variance

Till now, we have discussed the algorithms for categorical target variable. Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

Above X-bar is the mean of the values, X is actual and n is the number of values.

Steps to calculate Variance:

1. Calculate variance for each node.

2. Calculate variance for each split as a weighted average of each node variance.

Note: in our case replace playing cricket yes as 1 and no as 0

So finally after finding variance for each split lower variance is selected as best split

Disadvantages:

1. **Over fitting:** Overfitting is one of the most practical difficulty for decision tree models. This problem gets solved by setting constraints on model parameters and pruning (discussed in detail below).
2. **Not fit for continuous variables**: While working with continuous numerical variables, decision tree loses information when it categorizes variables in different categories.

How can we avoid over-fitting in decision trees?

Overfitting is one of the key challenges faced while using tree based algorithms. If there is no limit set of a decision tree, it will give you 100% accuracy on the training set because in the worst case it will end up making 1 leaf for each observation. Thus, preventing overfitting is pivotal while modeling a decision tree and it can be done in 2 ways:

1. Setting constraints on tree size
2. Tree pruning

Problem Statement: Predicting whether a person purchase SUV car or not based on the provided features

Dataset Download Link:

https://github.com/Sindhuja-AI/Decision-Tree/blob/main/Purchased_Dataset.csv

Github link for code:

https://github.com/Sindhuja-AI/Decision-Tree/blob/main/Decision%20Tree.ipynb

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NBA, Approved by AICTE New Delhi)
Aziz nagar Gate, C.B. Post, Hyderabad-500 075.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE.

Algorithm name:

Random Forest

Dataset Download link:

https://www.kaggle.com/lodetomasi1995/income-classification

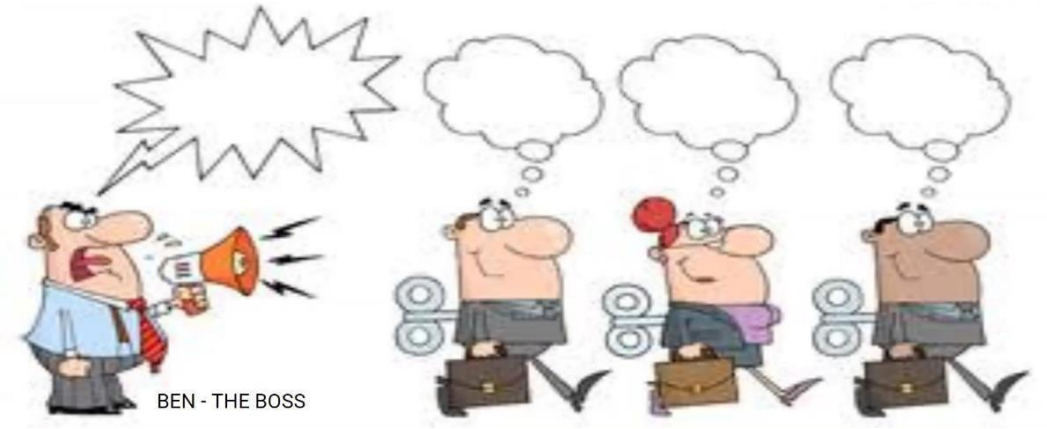
Algorithm explanation:

Random Forest is an ensemble tree based algorithm. Random Forest use set of decision tree from randomly selected set of training set in order to decide the final outcome.

What is Ensemble modeling?
A mixture of some algorithms with their own answers and performing majority voting. The most likely answer is choosen.

Example:(a made-up story)

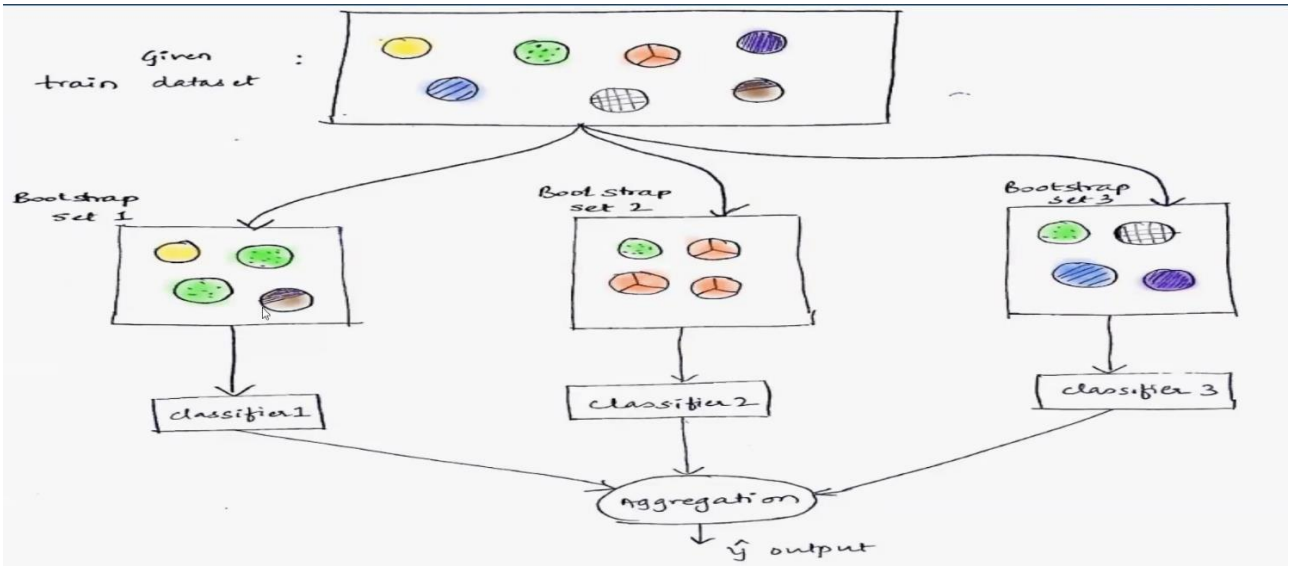
There is a boss called ‘Ben’. So he is interested in investing money in a company ‘Amazon’. He sends his 3 employees for finding whether investing in that company is worthy or not. Check the given below figure:



Each of the three employees has different ways to find the solution for the given problem. Out of them 2 employees say ‘Yes’ for investing in that company one employee say ‘No’. According to the model most likely answer would be yes. The boss invests the that company.

‘While coming to random forest every classifier must be in the form of tree’.

This distribution of data and performing the majority voting is called as **‘Bagging and Bootstrap Aggregation’**.



Mathematical part behind Random Forest:

For each decision tree, Scikit-learn calculates a nodes importance using Gini Importance:

What is Gini?

Gini is a cost function used to split in the data set.

Gini(t) = 1- summation(p(i | t)^2)

Impurity	Task	Formula	Description
Gini impurity	Classification	$\sum_{i=1}^{C} -f_i(1-f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Entropy	Classification	$\sum_{i=1}^{C} -f_i \log(f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Variance / Mean Square Error (MSE)	Regression	$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$	y_i is label for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$
Variance / Mean Absolute Error (MAE) (Scikit-learn only)	Regression	$\frac{1}{N}\sum_{i=1}^{N}\lvert y_i-\mu\rvert$	y_i is label for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$

Code explanation with screen shots:

https://github.com/bharadwajvsd/Randomforest/blob/main/Random%20Forest%20Classifier%20Algo%20ML%20Club-Copy1.ipynb

Application or use cases:

Applications of Random Forest

It can be used for both regression and classification tasks, and it's also easy to view the relative importance it assigns to the input features.

Random forest is also a very handy algorithm because the default hyperparameters it uses often produce a good prediction result. Understanding the hyperparameters is pretty straightforward, and there's also not that many of them.

One of the biggest problems in machine learning is overfitting, but most of the time this won't happen thanks to the random forest classifier. If there are enough trees in the forest, the classifier won't overfit the model.

The main limitation of random forest is that a large number of trees can make the algorithm too slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. In most real-world applications, the random forest algorithm is fast enough but there can certainly be situations where run-time performance is important and other approaches would be preferred.

Use cases:

The random forest algorithm is used in a lot of different fields, like banking, the stock market, medicine and e-commerce.

In finance, for example, it is used to detect customers more likely to repay their debt on time, or use a bank's services more frequently. In this domain it is also used to detect fraudsters out to scam the bank. In trading, the algorithm can be used to determine a stock's future behavior.

In the healthcare domain it is used to identify the correct combination of components in medicine and to analyze a patient’s medical history to identify diseases.

Random forest is used in e-commerce to determine whether a customer will actually like the product or not.

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NBA, Approved by AICTE New Delhi)
Aziz nagar Gate, C.B. Post, Hyderabad-500 075.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE.

Algorithm name:
Association Rule Mining via Apriori Algorithm in Python

Dataset Download link:
https://github.com/rakeshrohan-123/Apriori/blob/main/store_data.csv

Algorithm explanation:

Association rule mining is a technique to identify underlying relations between different items. Take an example of a Super Market where customers can buy variety of items. Usually, there is a pattern in what the customers buy. For instance, mothers with babies buy baby products such as milk and diapers. Damsels may buy makeup items whereas bachelors may buy beers and chips etc. In short, transactions involve a pattern. More profit can be generated if the relationship between the items purchased in different transactions can be identified.

For instance, if item A and B are bought together more frequently then several steps can be taken to increase the profit. For example:

1. A and B can be placed together so that when a customer buys one of the product he doesn't have to go far away to buy the other product.
2. People who buy one of the products can be targeted through an advertisement campaign to buy the other.

3. Collective discounts can be offered on these products if the customer buys both of them.
4. Both A and B can be packaged together.

The process of identifying an associations between products is called association rule mining.

Theory of Apriori Algorithm

There are three major components of Apriori algorithm:

- Support
- Confidence
- Lift

We will explain these three concepts with the help of an example.

Suppose we have a record of 1 thousand customer transactions, and we want to find the Support, Confidence, and Lift for two items e.g. burgers and ketchup. Out of one thousand transactions, 100 contain ketchup while 150 contain a burger. Out of 150 transactions where a burger is purchased, 50 transactions contain ketchup as well. Using this data, we want to find the support, confidence, and lift using mathematical formulas .this formulas are in mathematical part.

Steps involved are:

1. Set a minimum value for support and confidence. This means that we are only interested in finding rules for the items that have certain default existence (e.g. support) and have a minimum value for co-occurrence with other items (e.g. confidence).
2. Extract all the subsets having higher value of support than minimum threshold.
3. Select all the rules from the subsets with confidence value higher than minimum threshold.
4. Order the rules by descending order of Lift.

Mathematical part

Support

Support refers to the default popularity of an item and can be calculated by finding number of transactions containing a particular item divided by total number of transactions. Suppose we want to find support for item B. This can be calculated as:

```
Support(B) = (Transactions containing (B))/(Total Transactions)
```

For instance if out of 1000 transactions, 100 transactions contain Ketchup then the support for item Ketchup can be calculated as:

```
Support(Ketchup) = (Transactions containingKetchup)/(Total Transactions)

Support(Ketchup) = 100/1000
                 = 10%
```

Confidence

Confidence refers to the likelihood that an item B is also bought if item A is bought. It can be calculated by finding the number of transactions where A and B are bought together, divided by total number of transactions where A is bought. Mathematically, it can be represented as:

```
Confidence(A→B) = (Transactions containing both (A and B))/(Transactions containing A)
```

Lift

`Lift(A -> B)` refers to the increase in the ratio of sale of B when A is sold. Lift(A –> B) can be calculated by dividing `Confidence(A -> B)` divided by `Support(B)`. Mathematically it can be represented as:

```
Lift(A→B) = (Confidence (A→B))/(Support (B))
```

Coming back to our Burger and Ketchup problem, the `Lift(Burger -> Ketchup)` can be calculated as:

```
Lift(Burger→Ketchup) = (Confidence (Burger→Ketchup))/(Support (Ketchup))

Lift(Burger→Ketchup) = 33.3/10
                     = 3.33
```

Lift basically tells us that the likelihood of buying a Burger and Ketchup together is 3.33 times more than the likelihood of just buying the ketchup. A Lift of 1 means there is no association between products A and B. Lift of greater than 1 means products A and B are more likely to be bought together. Finally, Lift of less than 1 refers to the case where two products are unlikely to be bought together.

Code explanation with screen shots:(also try to keep your code in GitHub repository and give link)

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
```

```
store_data = pd.read_csv('store_data.csv')
```

```
store_data.head()
```

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	fro smoo
0	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
3	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

Just imported required packages and loaded dataset in store_data variable and used head() method to see first five from dataset

```
In [27]: store_data = pd.read_csv('store_data.csv', header=None)
```

```
In [28]: store_data.head()
```

Out[28]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	frozen smoothie	spin
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

As we can see above pic we took header is equal to none because they are not column names

```
In [43]: records = []
         for i in range(0, 7501):
             records.append([str(store_data.values[i,j]) for j in range(0, 20)])
```

```
In [44]: records
```

```
Out[44]: [['shrimp',
           'almonds',
           'avocado',
           'vegetables mix',
           'green grapes',
           'whole weat flour',
           'yams',
           'cottage cheese',
           'energy drink',
           'tomato juice',
           'low fat yogurt',
           'green tea',
           'honey',
           'salad',
           'mineral water',
           'salmon',
           'antioxydant juice',
           'frozen smoothie',
           'spinach',
           'olive oil'],
```

We took empty list as record variable and using for loop we appended all items in to list

```
In [32]: association_rules = apriori(records, min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2)
         association_results = list(association_rules)
```

```
In [20]: association_results
```

```
Out[20]: [RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.004532728969470737, ordered_statistics=[OrderedStatist
         ic(items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=4.8439506172
         8395)]),
          RelationRecord(items=frozenset({'escalope', 'mushroom cream sauce'}), support=0.005732568990801226, ordered_statistics=[Orde
         redStatistic(items_base=frozenset({'mushroom cream sauce'}), items_add=frozenset({'escalope'}), confidence=0.300699300699300
         7, lift=3.790832696715049)]),
          RelationRecord(items=frozenset({'escalope', 'pasta'}), support=0.005865884548726837, ordered_statistics=[OrderedStatistic(it
         ems_base=frozenset({'pasta'}), items_add=frozenset({'escalope'}), confidence=0.3728813559322034, lift=4.700811850163794)]),
          RelationRecord(items=frozenset({'herb & pepper', 'ground beef'}), support=0.015997866951073192, ordered_statistics=[OrderedS
         tatistic(items_base=frozenset({'herb & pepper'}), items_add=frozenset({'ground beef'}), confidence=0.3234501347708895, lift=
         3.2919938411349285)]),
          RelationRecord(items=frozenset({'ground beef', 'tomato sauce'}), support=0.005332622317024397, ordered_statistics=[OrderedSt
         atistic(items_base=frozenset({'tomato sauce'}), items_add=frozenset({'ground beef'}), confidence=0.3773584905660377, lift=3.8
         40659481324083)]),
          RelationRecord(items=frozenset({'olive oil', 'whole wheat pasta'}), support=0.007998933475536596, ordered_statistics=[Ordere
         dStatistic(items_base=frozenset({'whole wheat pasta'}), items_add=frozenset({'olive oil'}), confidence=0.2714932126696833, li
         ft=4.122410097642296)]),
          RelationRecord(items=frozenset({'pasta', 'shrimp'}), support=0.005065991201173177, ordered_statistics=[OrderedStatistic(item
         s_base=frozenset({'pasta'}), items_add=frozenset({'shrimp'}), confidence=0.3220338983050847, lift=4.506672147735896)]),
          RelationRecord(items=frozenset({'nan', 'light cream', 'chicken'}), support=0.004532728969470737, ordered_statistics=[Ordered
```

The `apriori` class requires some parameter values to work. The first parameter is the list of list that you want to extract rules from. The second parameter is the `min_support` parameter. This parameter is used to select the items with support values greater than the value specified by the parameter. Next, the `min_confidence` parameter filters those rules that have confidence greater than the confidence threshold specified by the parameter. Similarly, the `min_lift` parameter specifies the minimum lift value for the short listed rules. Finally, the `min_length` parameter specifies the minimum number of items that you want in your rules.

Let's suppose that we want rules for only those items that are purchased at least 5 times a day, or 7 x 5 = 35 times in one week, since our dataset is for a one-week time period. The support for those items can be calculated as 35/7500 = 0.0045. The minimum confidence for the rules is 20% or 0.2. Similarly, we specify the value for lift as 3 and finally `min_length` is 2 since we want at least two products in our rules. These values are mostly just arbitrarily chosen, so you can play with these values and see what difference it makes in the rules you get back out.

```
In [45]: len(association_results)
Out[45]: 48

In [46]: association_results[0]
Out[46]: RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.004532728969470737, ordered_statistics=[OrderedStatistic
         (items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=4.8439506172839
         5)])
```

We just seen length and sawn first element of our list

And then finally to see all relationship between items

The following script displays the rule, the support, the confidence, and lift for each rule in a more clear way:

```
In [48]: for item in association_results:

             # first index of the inner list
             # Contains base item and add item
             pair = item[0]
             items = [x for x in pair]
             print("Rule: " + items[0] + " -> " + items[1])

             #second index of the inner list
             print("Support: " + str(item[1]))

             #third index of the list located at 0th
             #of the third index of the inner list

             print("Confidence: " + str(item[2][0][2]))
             print("Lift: " + str(item[2][0][3]))
             print("====================================")
```

```
Rule: light cream -> chicken
Support: 0.004532728969470737
Confidence: 0.29059829059829057
Lift: 4.84395061728395
====================================
Rule: escalope -> mushroom cream sauce
Support: 0.005732568990801226
Confidence: 0.3006993006993007
```

You can also find full code and dataset in below github url.

https://github.com/rakeshrohan-123/Apriori/blob/main/Apriori.ipynb

https://github.com/rakeshrohan-123/Apriori/blob/main/store_data.csv

Application or use cases:

- ❑ Association rule mining algorithms such as Apriori are very useful for finding simple associations between our data items. They are easy to implement and have high explain-ability. However for more advanced

insights, such those used by Google or Amazon etc., more complex algorithms, such as recommender systems, are used. However, you can probably see that this method is a very simple way to get basic associations if that's all your use-case needs.

Machine Learning

Artificial Neural Networks

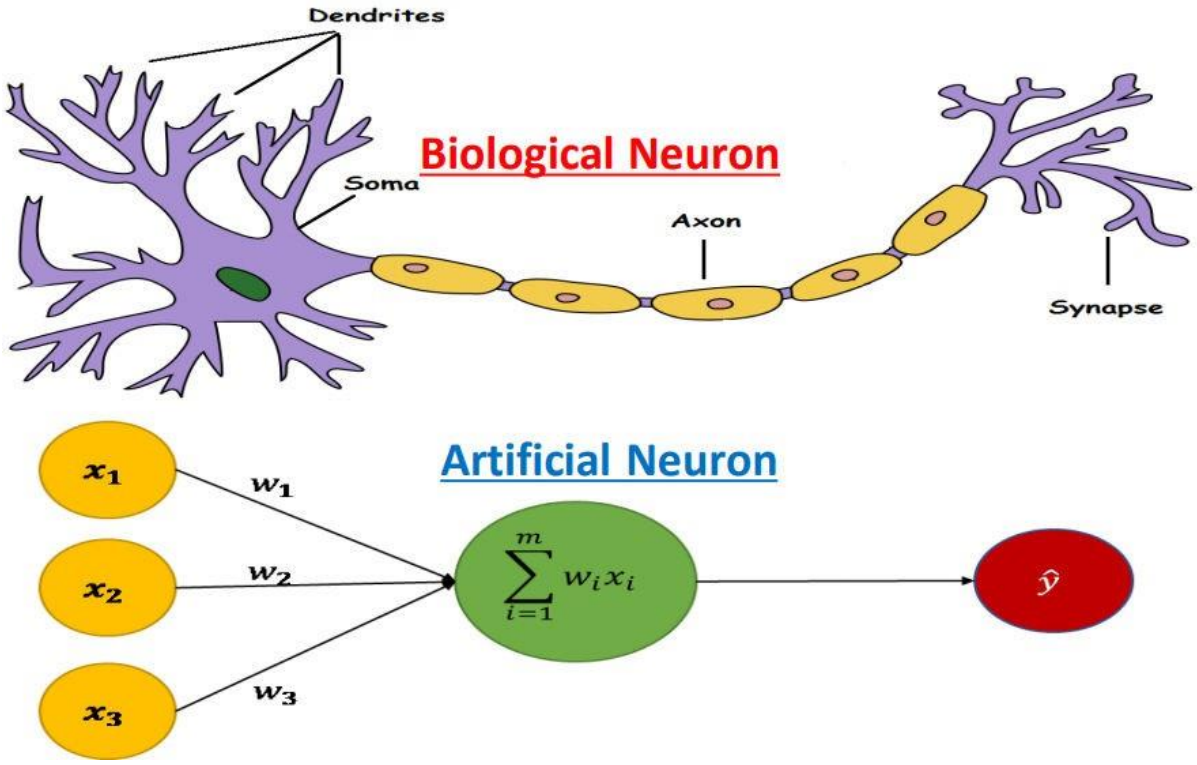
Department of Artificial Intelligence

Vidya Jyothi Institute of Technology

Index:

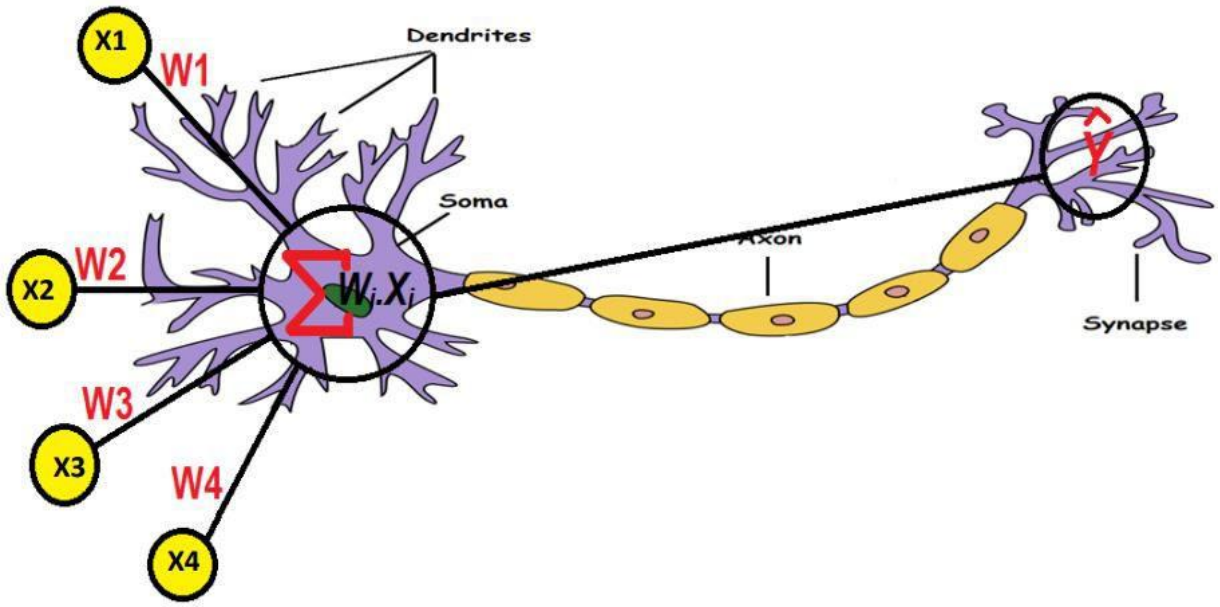
What is an Artificial Neuron?

An artificial neuron is a mathematical function conceived as a model of biological neurons, a neural network. Artificial neurons are elementary units in an artificial neural network. Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs and sums them to produce an output.



Comparison between Artificial Neuron and Biological Neuron

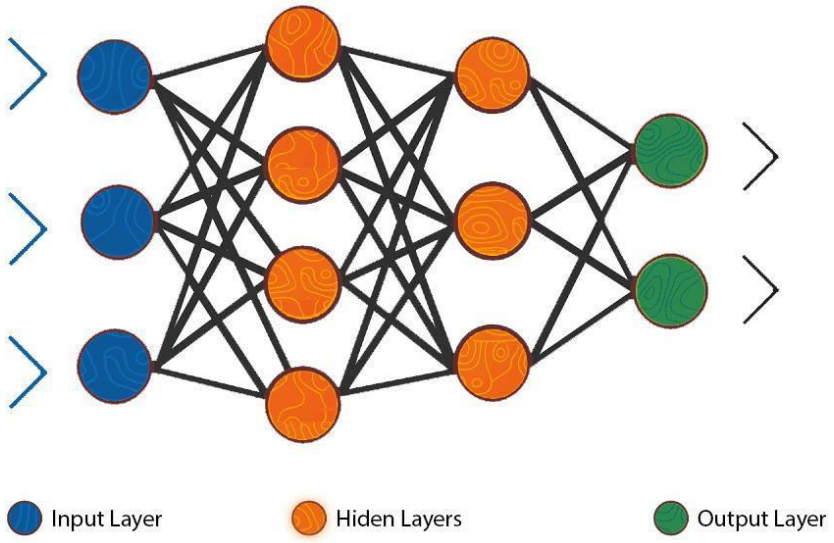
Artificial Neuron	Biological Neuron
Dendrites	Input
Synapse	Interconnections
Soma	Node
Axon	Output



What is an Artificial Neural Network?

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 also called nodes.

The architecture of an Artificial Neural Network



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.

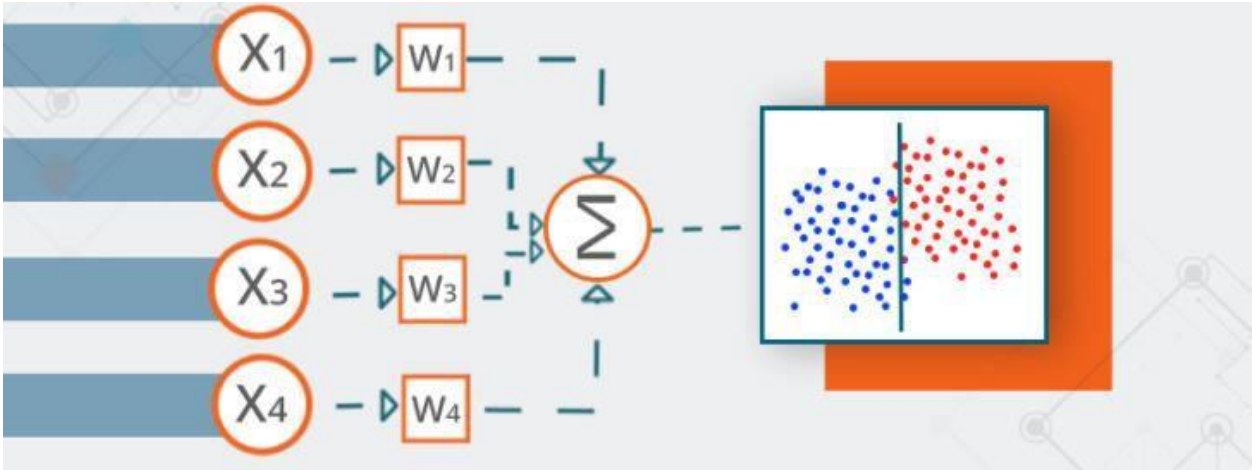
Perceptron

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.

A Perceptron is an algorithm for the supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.

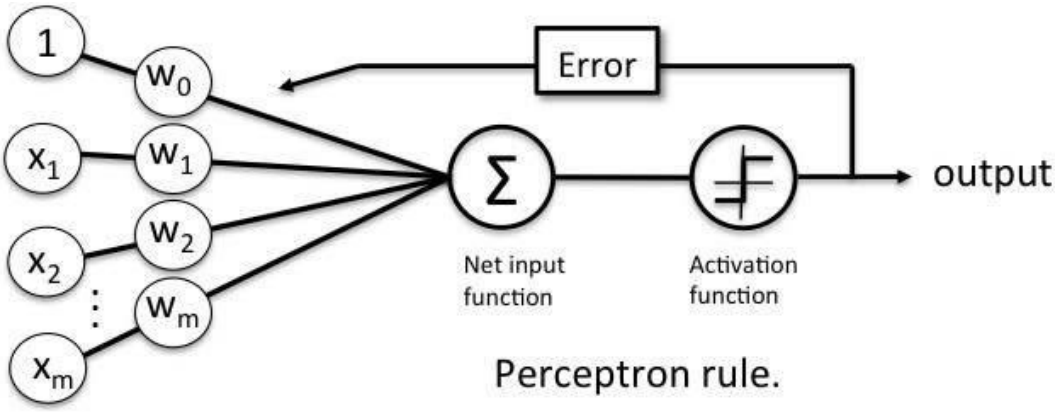
The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.

This enables you to distinguish between the two linearly separable classes +1 and -1.



Perceptron Learning Rule

Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients. The input features are then multiplied with these weights to determine if a neuron fires or not.



The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or

does not return an output. In the context of supervised learning and classification, this can then be used to predict the class of a sample.

Perceptron Function

Perceptron is a function that maps its input “x,” which is multiplied with the learned weight coefficient; an output value ”f(x)” is generated.
Perceptron is a function that maps its input “x,” which is multiplied with the learned weight coefficient; an output value ”f(x)” is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

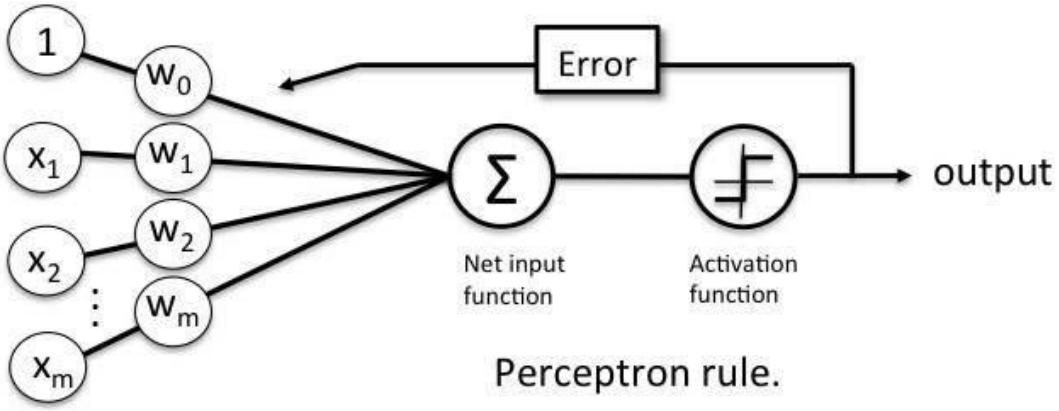
In the equation given above:
“w” = vector of real-valued weights
“b” = bias (an element that adjusts the boundary away from the origin without any dependence on the input value)
“x” = vector of input x values

$$\sum_{i=1}^{m} w_i x_i$$

“m” = number of inputs to the Perceptron
The output can be represented as “1” or “0.” It can also be represented as “1” or “-1” depending on which activation function is used.
Let us learn the inputs of a perceptron in the next section.

Inputs of a Perceptron

A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The above below shows a Perceptron with a Boolean output.



A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False. The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

Output of Perceptron

Perceptron with a Boolean output:

Inputs: x1...xn

Output: o(x1....xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

Weights: wi=> contribution of input xi to the Perceptron output;

w0=> bias or threshold

If∑w.x>0,outputis+1,else-1. The neuron gets triggered only when weighted input reaches a certain threshold value. An output of +1 specifies that the neuron is triggered. An output of -1

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

specifies that the neuron did not get triggered.
"sgn" stands for sign function with output +1 or -1.

Error in Perceptron

In the Perceptron Learning Rule, the predicted output is compared with the known output. If it does not match, the error is propagated backward to allow weight adjustment to happen.

Perceptron: Decision Function

A decision function φ(z) of Perceptron is defined to take a linear combination of x and w vectors.

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

The value z in the decision function is given by:

$$z = w_1 x_1 + \ldots + w_m x_m$$

The decision function is +1 if z is greater than a threshold θ, and it is -1 otherwise.

$$\phi(z) = \begin{cases} 1 \ \ if \ z \geq \theta \\ -1 \ \ otherwise \end{cases}$$

This is the Perceptron algorithm.

Bias Unit

For simplicity, the threshold θ can be brought to the left and represented as wOxO, where wO= -θ and x0= 1.

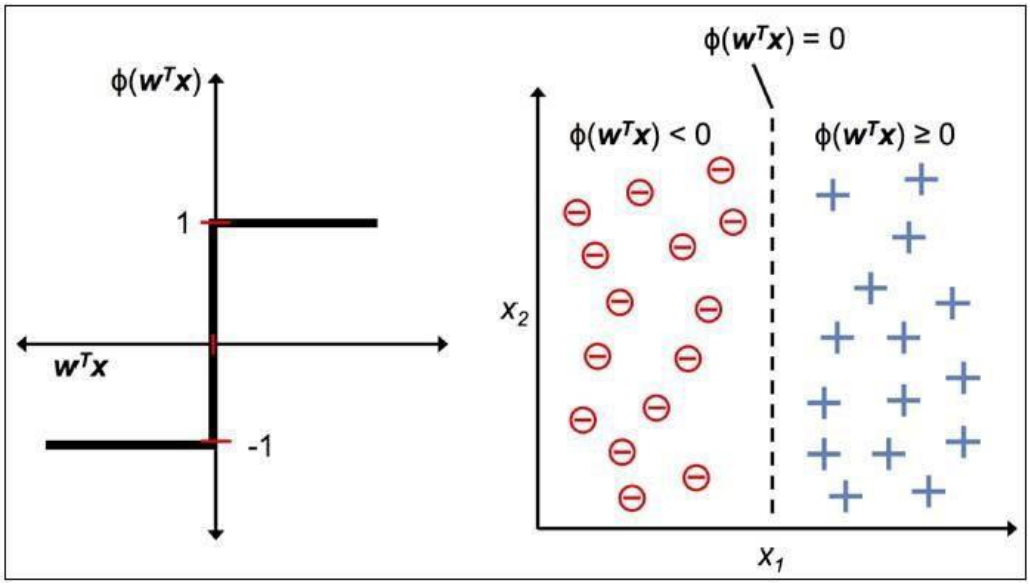
$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

The value wO is called the bias unit.

The decision function then becomes:

$$\phi(z) = \begin{cases} 1 \ if \ z \geq 0 \\ -1 \ otherwise \end{cases}$$

Perceptron linearly classifies two classes.

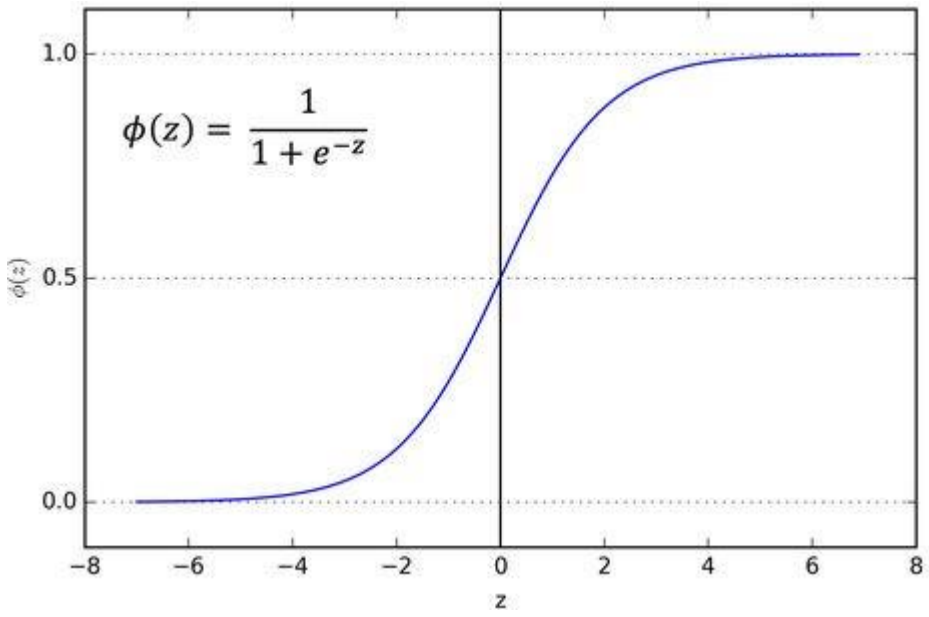


What is an Activation function?

The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

Most commonly used Activation function:

1.Sigmoid Function:



It is a function that is plotted as ‘S’ shaped graph.

Equation :

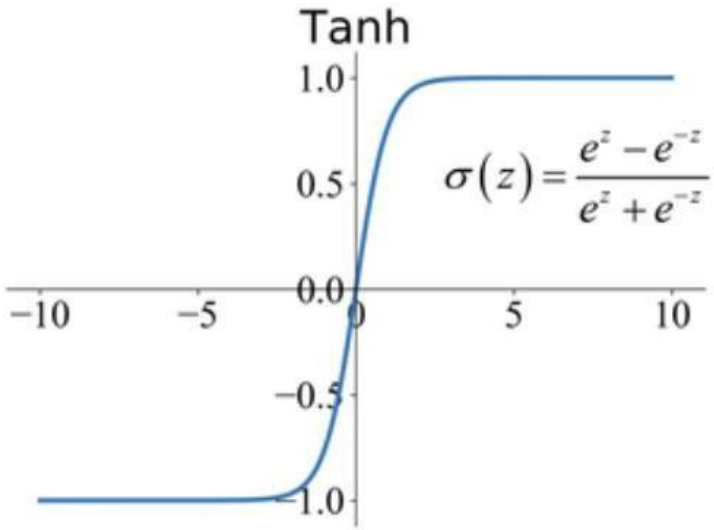
$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

Nature: Non-linear. Notice that X values lie between -2 to 2, Y values are very steep. This means small changes in x would also bring about large changes in the value of Y.

Value Range : 0 to 1

Uses: Usually used in the output layer of binary classification, where the result is either 0 or 1, as the value for the sigmoid function lies between 0 and 1 only so, the result can be predicted easily to be 1 if the value is greater than 0.5 and 0 otherwise.

2. Tanh Function:



The activation that works almost always better than sigmoid function is Tanh function also known as Tangent Hyperbolic function. It's actually a mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.

Equation :-

$$g(x) = \frac{e^x}{1 + e^x}$$

$$tanh(x) = 2g(2x) - 1$$

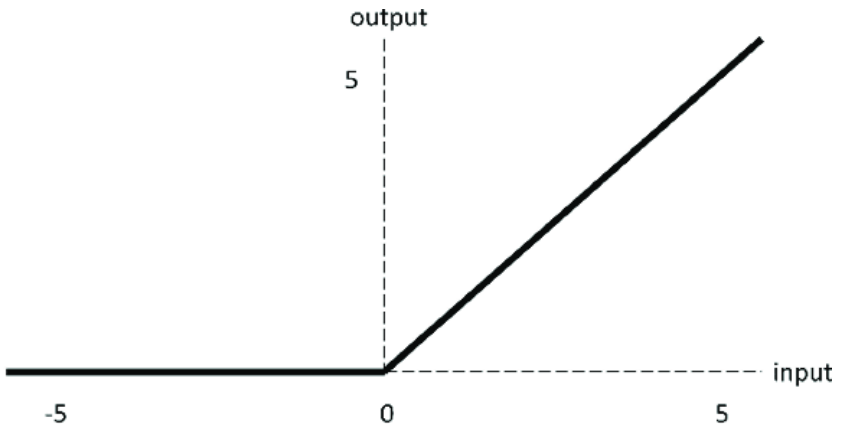
$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Value Range :--1 to +1

Nature :- non-linear

Uses :- Usually used in hidden layers of a neural network as it's values lies between -1 to 1 hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in centering the data by bringing mean close to 0. This makes learning for the next layer much easier.

3. RELU Function:



Stands for Rectified linear unit. It is the most widely used activation function. Chiefly implemented in hidden layers of Neural networks.

Equation :-

A(x) = max(0,x). It gives an output x if x is positive and 0 otherwise.

Value Range :- [0, inf)

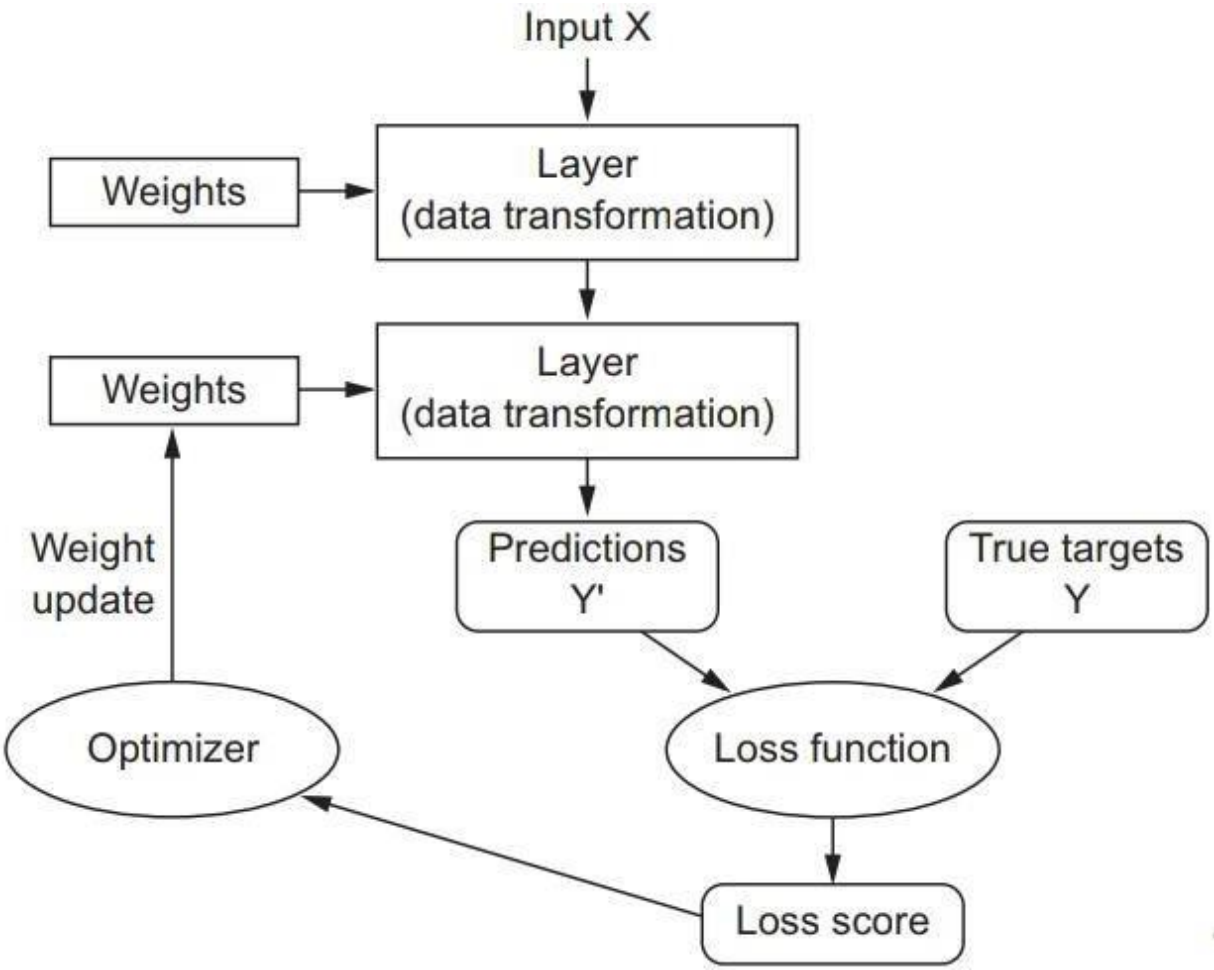
Nature :- non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.

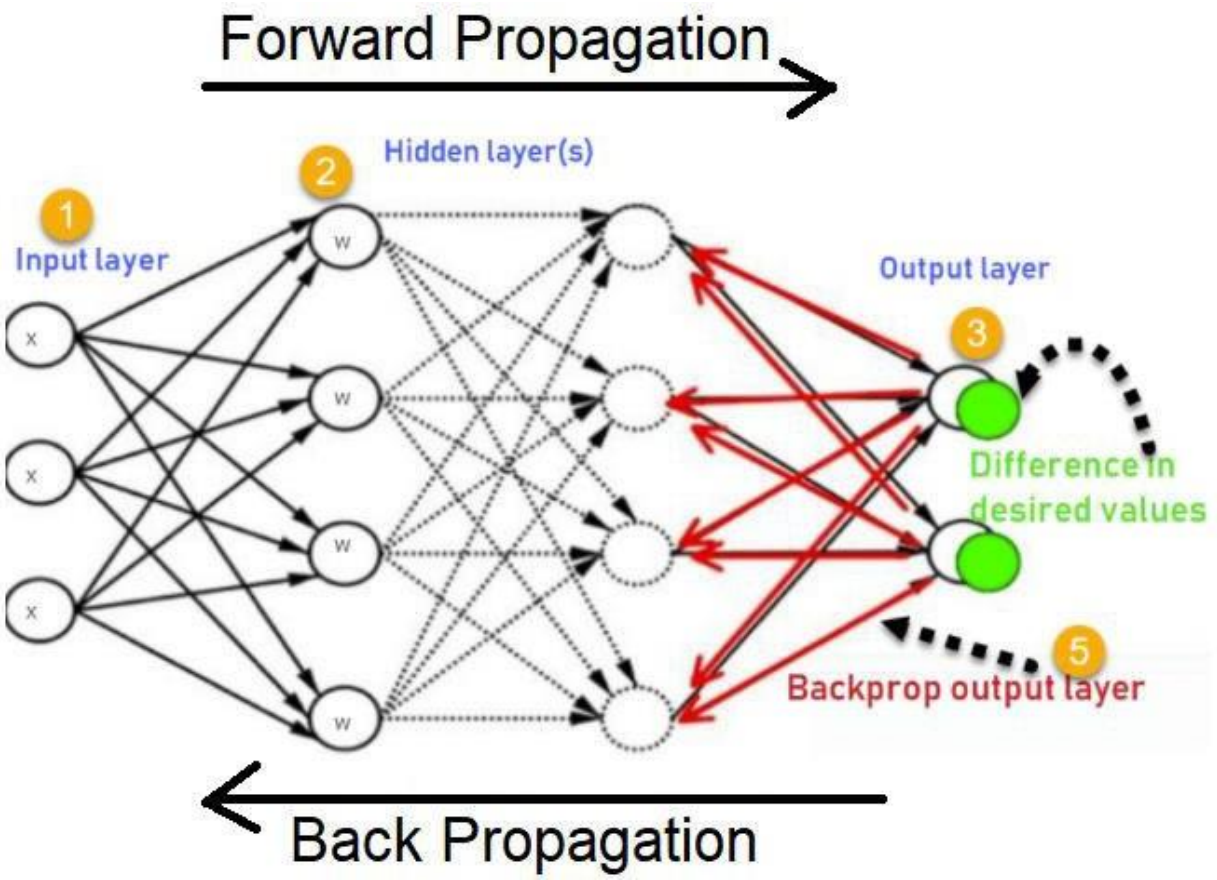
Uses :- ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation

Multilayer Neural Network

Multilayer 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.

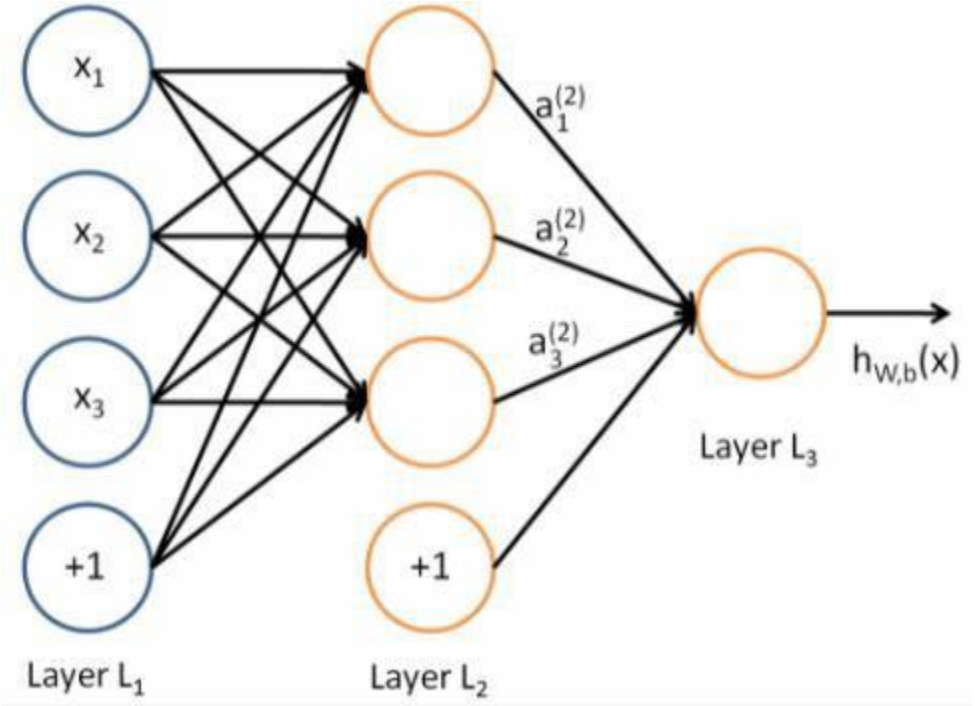
Working of Multilayer Neural Network


Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Weight
update
Predictions
Y'
True targets
Y
Optimizer
Loss function
Loss score



Afterward, each of the inputs is multiplied by its corresponding weights. In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.

Forward Propagation



The circles labeled "+1" are called bias units, and correspond to the intercept term. The leftmost layer of the network is called the input layer, and the rightmost layer the output layer (which, in this example, has only one node). The middle layer of nodes is called the hidden layer, because its values are not observed in the training set. We also say that our example neural network has 3 input units (not counting the bias unit), 3 hidden units, and 1 output unit.

Let nl denote the number of layers in our network; thus nl=3 in our example. We label layer l as Ll, so layer L1 is the input layer, and layer Lnl the output layer. Our neural network has parameters (W,b)=(W(1),b(1),W(2),b(2)), where we write W(l)ij to

denote the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer l+1. (Note the order of the indices.) Also, b(l)i is the bias associated with unit i in layer l+1. Thus,

inourexample,wehaveW(1)∈R3×3,andW(2)∈R1×3.Note that bias units don't have inputs or connections going into them, since they always output the value +1. We also let sl denote the number of nodes in layer l (not counting the bias unit).

We will write a(l)i to denote the activation (meaning output value) of unit i in layer l. For l=1, we also use a(1)i=xi to denote the i-th input. Given a fixed setting of the parameters W,b, our neural network defines a hypothesis hW,b(x) that outputs a real number. Specifically, the computation that this neural network represents is given by:

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

Let $z^{(l)}i$ denote the total weighted sum of inputs to unit i in layer l, including the bias term.

$$z_i^{(2)} = \sum_{j=1}^{n} W_{ij}^{(1)} x_j + b_i^{(1)}$$

$$
\begin{aligned}
z^{(2)} &= W^{(1)} x + b^{(1)} \\
a^{(2)} &= f(z^{(2)}) \\
z^{(3)} &= W^{(2)} a^{(2)} + b^{(2)} \\
h_{W,b}(x) &= a^{(3)} = f(z^{(3)})
\end{aligned}
$$

Equation for computing l + 1's activations:

$$
\begin{aligned}
z^{(l+1)} &= W^{(l)} a^{(l)} + b^{(l)} \\
a^{(l+1)} &= f(z^{(l+1)})
\end{aligned}
$$

Back Propagation Algorithm

Suppose we have a fixed training set {(x(1),y(1)),…,(x(m),y(m))} of m training examples. We can train our neural network using batch gradient descent. In detail, for a single training example (x,y), we define the cost function with respect to that single example to be:

$$
J(W,b;x,y) = \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2.
$$

Overall cost function:

$$
J(W,b) = \left[\frac{1}{m} \sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)}) \right]
$$

Derivative of the overall cost function J(W,b):

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \left[\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(i)},y^{(i)}) \right]$$

$$\frac{\partial}{\partial b_{i}^{(l)}} J(W,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x^{(i)},y^{(i)})$$

One iteration of gradient descent updates the parameters W,b as follows:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)$$

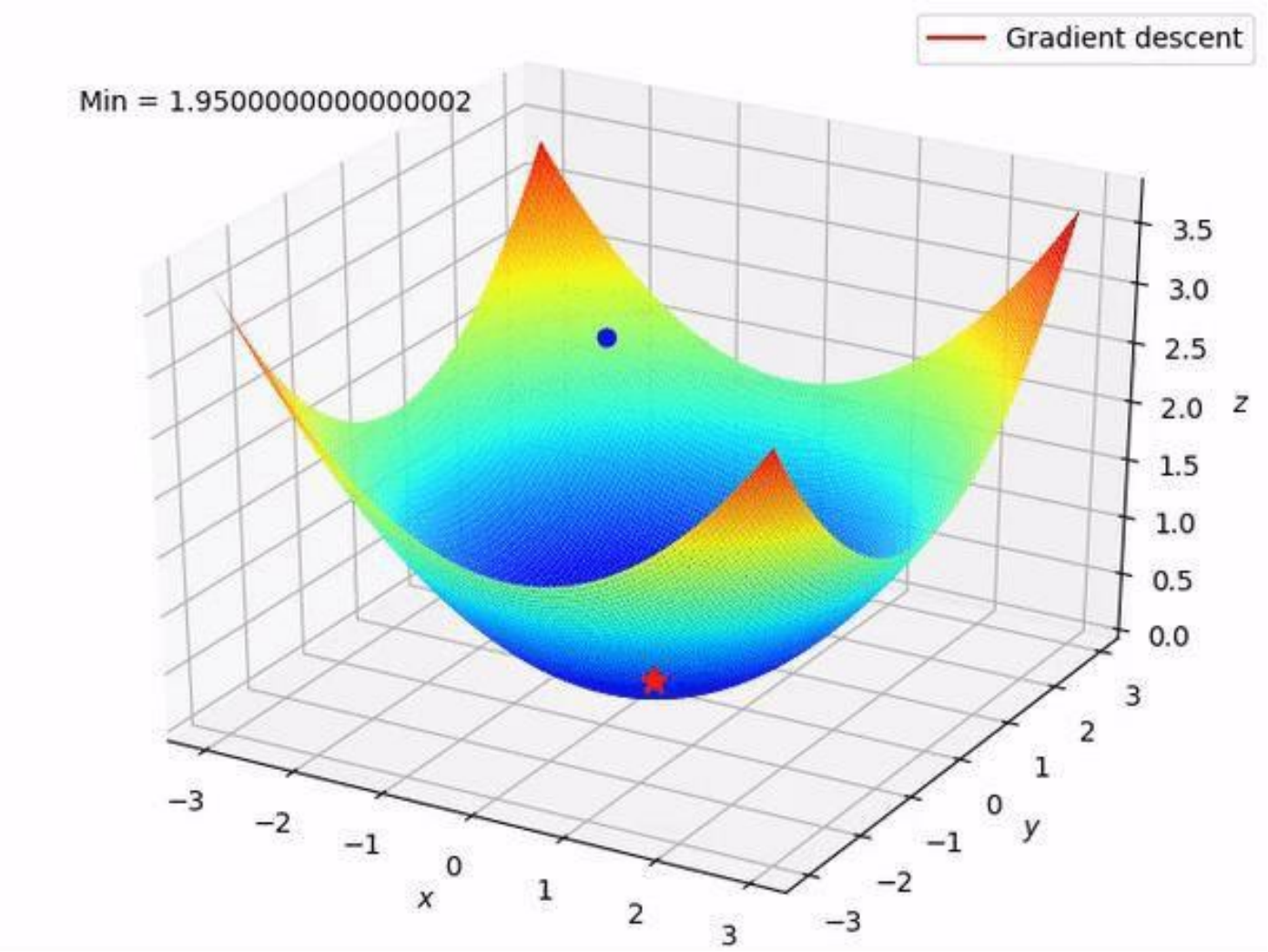
$$b_{i}^{(l)} = b_{i}^{(l)} - \alpha \frac{\partial}{\partial b_{i}^{(l)}} J(W,b)$$

Where α (alpha) is the learning rate.

Gradient Descent

Gradient Descent is an optimization algorithm which helps in finding the optimum values of weights (global minima) such that the value of cost function in minimum.

Graphical Intuition of Gradient Descent



Applications of Artificial Neural Networks:

1. Defense: Counterterrorism, facial recognition, feature extraction, noise suppression, object discrimination, sensors, sonar, radar and image signal processing, signal/image identification, target tracking, and weapon steering
2. Education: Adaptive learning software, dynamic forecasting, education system analysis and forecasting, student performance modeling, and personality profiling
3. Banking: Credit card attrition, credit and loan application evaluation, fraud and risk evaluation, and loan delinquencies
4. Medical: Cancer cell analysis, ECG and EEG analysis, emergency room test advisement, expense reduction and quality improvement for hospital systems, transplant process optimization, and prosthesis design
5. Character Recognition: The idea of character recognition has become very important as handheld devices like the Palm Pilot are becoming increasingly popular. Neural networks can be used to recognize handwritten characters.
6. Autonomous Vehicles
7. Speech Recognition

8. Stock Market Prediction : The day-to-day business of the stock market is extremely complicated. Many factors weigh in whether

a given stock will go up or down on any given day. Since neural networks can examine a lot of information quickly and sort it all out, they can be used to predict stock prices.

9. Travelling Salesman Problem: Interestingly enough, neural networks can solve the traveling salesman problem, but only to a certain degree of approximation.
10. Image Compression- Neural networks can receive and process vast amounts of information at once, making them useful in image compression. With the Internet explosion and more sites using more images on their sites, using neural networks for image compression is worth a look.

References:

1. Artificial Neural Network - Basic Concepts - Tutorialspoint
2. Artificial Neural Network Tutorial - Javatpoint
3. Artificial neuron - Wikipedia
4. Machine Learning (mcgill.ca)
5. What is Perceptron | Simplilearn
6. Artificial Neural Network Tutorial - Javatpoint
7. Activation functions in Neural Networks - GeeksforGeeks
8. Unsupervised Feature Learning and Deep Learning Tutorial (stanford.edu)

Required Softwares:

1. Python3
2. Git
3. Heroku CLI

Link to download Git for Windows:
Git - Downloading Package (git-scm.com)

Link to download Heroku CLI for windows:
The Heroku CLI | Heroku Dev Center

Required Python Packages:

1. **Flask**
   Installation Command: pip install flask

2. **Virtualenv**
   Installation Command: pip install virtualenv

   Command to create virtualenv:
   virtualenv environment_name

   Command to activate virtualenv:
   environment_name/Scripts/activate

   Command to deactivate virtualenv:
   deactivate

3. **Scikit-Learn**
   Installation Command: pip install -U scikit-learn

4. **Gunicorn**
   Installation Command: pip install gunicorn

Command for creating requirements.txt file:
pip freeze > requirements.txt

Steps to push the flask app to Heroku Cloud:

1. **Login to Heroku account in the terminal**
   Command: heroku login

2. **Initialize a git repository**
   Command: git init

3. **Connect to the created Heroku App**
   Command: heroku git:remote -a *app_name*

4. **Add all project files and folder to stage**
   Command: git add .

5. **Commit the project files and folders to the repository**
   Command: git commit -am “first commit”

6. **Deploy the app (final step)**
   Command: git push heroku master

Note:
If you are using virtualenv for the first time and your os is windows 10 there is a chance of getting an error in the terminal like **“permission denied”**
Solution:
Open Windows Powershell as Admin and run the following command:
Set-ExecutionPolicy Unrestricted