

# Machine Learning

Practical File

BACHELOR OF TECHNOLOGY

Information Technology

SUBMITTED BY

RAJBIR SINGH BEDI

University Roll Number-2104555

College Roll Number- 2121095



GURU NANAK DEV ENGINEERING COLLEGE

LUDHIANA-141006, INDIA

## Contents

<b>1</b>	<b>BASIC PYTHON PROGRAM</b>	<b>5</b>
1.1	Print HELLO WORLD .....	5
1.2	Print Biodata.....	5
1.3	Else If Condition .....	6
<b>2</b>	<b>NUMPY LIBRARY</b>	<b>7</b>
2.1	Numpy library to add two number. ....	7
2.2	Numpy library to find mean of an array.....	7
2.3	Numpy library to find shape of an array .....	8
2.4	Numpy library to find median of an array.....	8
<b>3</b>	<b>PANADA LIBRARY</b>	<b>9</b>
3.1	Using Panda for dataframing. ....	9
3.2	Series in Panda Library .....	9
3.3	Adding series in Panda Library .....	10
3.4	Data clean-up in Panda Library .....	11
<b>4</b>	<b>MATPLOTLIB</b>	<b>12</b>
4.1	Plot.....	12
4.2	Scatter Plot.....	13
4.3	Pie Chart. ....	14
4.4	Bar Graph.....	15
<b>5</b>	<b>SCIKIT-LEARN</b>	<b>17</b>
5.1	What is Scikit-Learn (Sklearn)? .....	17
<b>6</b>	<b>PYTORCH</b>	<b>20</b>
6.1	What is pytorch? .....	20
<b>7</b>	<b>TENSORFLOW</b>	<b>24</b>
7.1	What is TensorFlow?.....	24
<b>8</b>	<b>AI IN BANKING HOW ARTIFICIAL INTELLIGENCE IS USED IN BANKS</b>	<b>29</b>
8.1	Cybersecurity and fraud detection.....	30
8.2	Chatbots.....	31
8.3	Loan and credit decisions.....	32
8.4	Tracking market trends .....	32
8.5	Data collection and analysis .....	33
8.6	Customer experience .....	33
8.7	Risk management .....	34
8.8	Regulatory compliance .....	34
8.9	Predictive analytics .....	35
8.10	Process automation .....	35

<b>I</b>	<b>SUPERVISED LEARNING.</b>	<b>37</b>
<b>9</b>	<b>CLASSIFICATION.</b>	<b>38</b>
9.1	Decision Tree.....	38
9.2	K-Nearest Neighbor .....	39
9.3	Kernel Support Vector Machine .....	40
9.3.1	Hyperplane.....	42
9.3.2	Support Vectors.....	42
9.4	Confusion Matrix in Machine Learning .....	42
<b>10</b>	<b>REGRESSION</b>	<b>45</b>
10.1	Simple linear regression.....	45
10.2	Polynomial regression.....	45
10.3	Support vector regression .....	45
10.4	Decision trees.....	46
10.5	Random forest .....	46
<b>II</b>	<b>UNSUPERVISED LEARNING.</b>	<b>47</b>
<b>11</b>	<b>CLUSTERING</b>	<b>48</b>
11.1	K-mean Clustering .....	49
11.2	K-mode Clustering.....	50
<b>III</b>	<b>PROGRAMS</b>	<b>50</b>
<b>12</b>	<b>SIMPLE LINEAR REGRESSION</b>	<b>51</b>
12.1	Importing the library: .....	51
12.2	Importing the dataset.....	51
12.3	Splitting the dataset into the Training set and Test set .....	51
12.4	Training the Simple Linear Regression model on the Training set	51
12.5	Predicting the Test set result.....	51
12.6	Visualising the Training set result .....	51
12.7	Visualising the Testing set result. ....	52
<b>13</b>	<b>K- MEANS CLUSTERING.</b>	<b>53</b>
13.1	Importing the libraries .....	53
13.2	Importing the dataset .....	53
13.3	Using the elbow method to find the optimal number of clusters .	53
13.4	Training the K-Means model on the dataset.....	54
13.5	Visualising the clusters .....	54

14 Polynomial Regression	54
14.1 Importing the library: .....	54
14.2 Importing the dataset .....	55
14.3 Training the Linear Regression model on the whole dataset.....	55
14.4 Training the Polynomial Regression model on the whole dataset .	55
14.5 Visualising the Linear Regression results.....	55
14.6 Visualising the Polynomial Regression results.....	56
14.7 Visualising the Polynomial Regression results (for higher resolution and smoother curve).....	57
15 Support Vector Regression (SVR)	58
15.1 Importing the library: .....	58
15.2 Importing the dataset .....	58
15.3 Feature Scaling .....	58
15.4 Training the SVR model on the whole dataset .....	58
15.5 Predicting a new result .....	59
15.6 Visualising the SVR results .....	59
15.7 Visualising the SVR results (for higher resolution and smoother curve) .....	59
16 Hierarchical Clustering.	61
16.1 Importing the libraries .....	61
16.2 Importing the dataset .....	61
16.3 Using the dendrogram to find the optimal number of clusters.....	61
16.4 Training the Hierarchical Clustering model on the dataset.....	61
16.5 Visualising the clusters .....	62

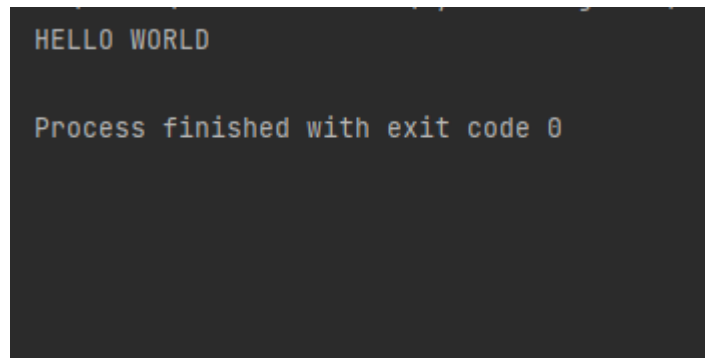
# 1 BASIC PYTHON PROGRAM

## 1.1 Print HELLO WORLD .

CODE :

```
print("HELLO WORLD")
```

OUTPUT :

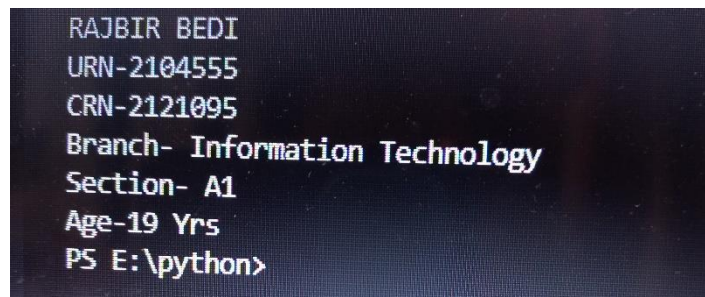


## 1.2 Print Biodata.

CODE:

```
print("RAJBIR BEDI")  
print("URN-2104555")  
print("CRN-2121095")  
print("Branch- Information Technology")  
print("Section- A1")  
print("Age-19 Yrs")
```

OUTPUT :

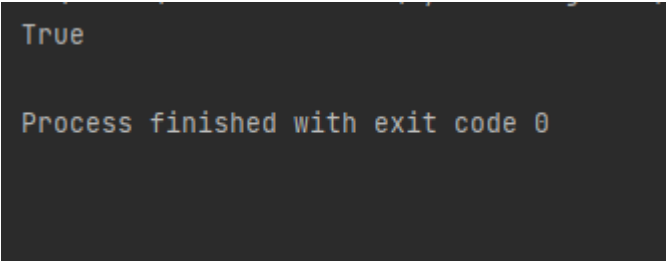


### 1.3 Else If Condition

CODE :

```
a=10  
if(1):print("True")  
else:print("False")
```

OUTPUT :



```
True  
  
Process finished with exit code 0
```

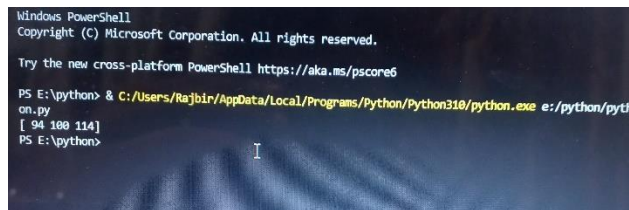
## 2 NUMPY LIBRARY

### 2.1 Numpy library to add two number.

CODE :

```
import numpy as np
a=np.array([90,96,110])
b=np.array([4])
c=np.array(a+b)
print(c)
```

OUTPUT :



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

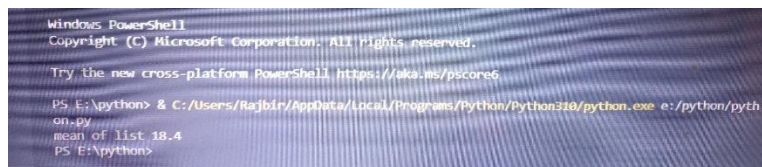
PS E:\python> & C:/Users/Rajbir/AppData/Local/Programs/Python/Python310/python.exe e:/python/pyth
on.py
[ 94 100 114]
PS E:\python>
```

### 2.2 Numpy library to find mean of an array.

CODE :

```
import numpy as np
marks=[12,19,15,18,28]
marks_arr=np.array(marks)
m=np.mean(marks_arr)
print("mean of list",m)
```

OUTPUT :



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

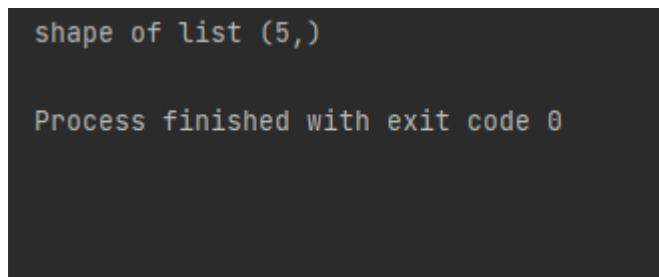
PS E:\python> & C:/Users/Rajbir/AppData/Local/Programs/Python/Python310/python.exe e:/python/pyth
on.py
mean of list 18.4
PS E:\python>
```

### 2.3 Numpy library to nd shape of an array.

CODE :

```
import numpy as np
s=[12,23,15,10,20]
arr=np.array(s)
shp=arr.shape
print("shape of list",shp)
```

OUTPUT :



```
shape of list (5,)

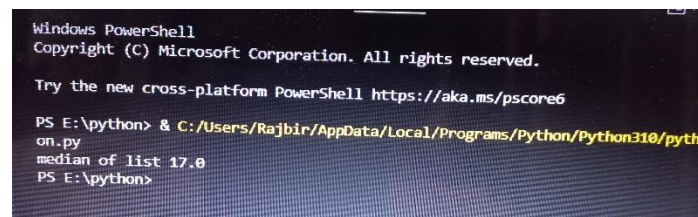
Process finished with exit code 0
```

### 2.4 Numpy library to nd median of an array.

CODE :

```
import numpy as np
marks=[17,13,17,19,25]
marks_arr=np.array(marks)
m=np.median(marks_arr)
print("median of list",m)
```

OUTPUT :



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\python> & C:/Users/Rajbir/AppData/Local/Programs/Python/Python310/python
on.py
median of list 17.0
PS E:\python>
```



## 3 PANADA LIBRARY

### 3.1 Using Panda for dataframing.

CODE :

```
import pandas as pd
s=pd.array([23,41,77,98,16,48],[55,86,93,98,7,1])
print(pd.DataFrame(s))
```

OUTPUT :



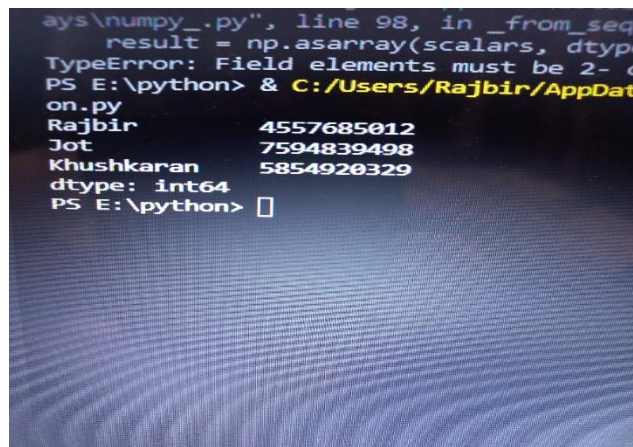
	0
0	23
1	43
2	76
3	98
4	12
5	45

### 3.2 Series in Panda Library.

CODE :

```
import pandas as pd
phonebook={
'Rajbir':4557685012,
'Jot':7594839498,
'Khushkaran':5854920329
}
print(pd.Series(phonebook))
```

OUTPUT :



```
ays\numpy_.py", line 98, in _from_sequ
result = np.asarray(scalars, dtype=
TypeError: Field elements must be 2- d
PS E:\python> & C:/Users/Rajbir/AppDat
on.py
Rajbir      4557685012
Jot         7594839498
Khushkaran  5854920329
dtype: int64
PS E:\python> □
```



### 3.3 Adding series in Panda Library.

CODE :

```
import pandas as pd
first=pd.series([1,2,3],['a','b','c'])
second=pd.series([4,5,6],['d','e','f'])
thied=first+second
print(third)
```

OUTPUT :

```
a    5.0
b    NaN
c    9.0
e    NaN
dtype: float64

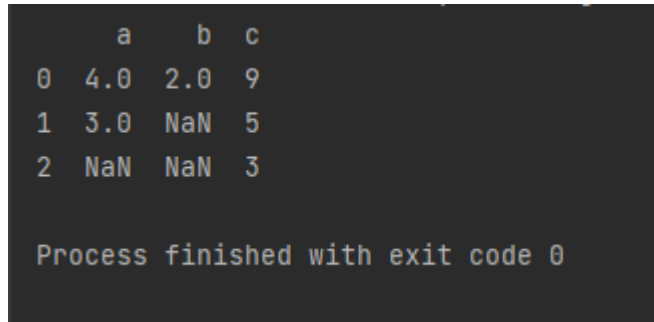
Process finished with exit code 0
```

### 3.4 Data clean-up in Panda Library.

CODE :

```
import panda as pd
import numpy as np
data={ 'a':[4,3,np.NaN],
       'b':[2,np.NaN,np.NaN]
       'c':[9,5,3]
}
dataframe=pd.DataFrame(data) print(dataframe)
```

OUTPUT :



```
      a      b      c
0  4.0  2.0  9
1  3.0  NaN  5
2  NaN  NaN  3

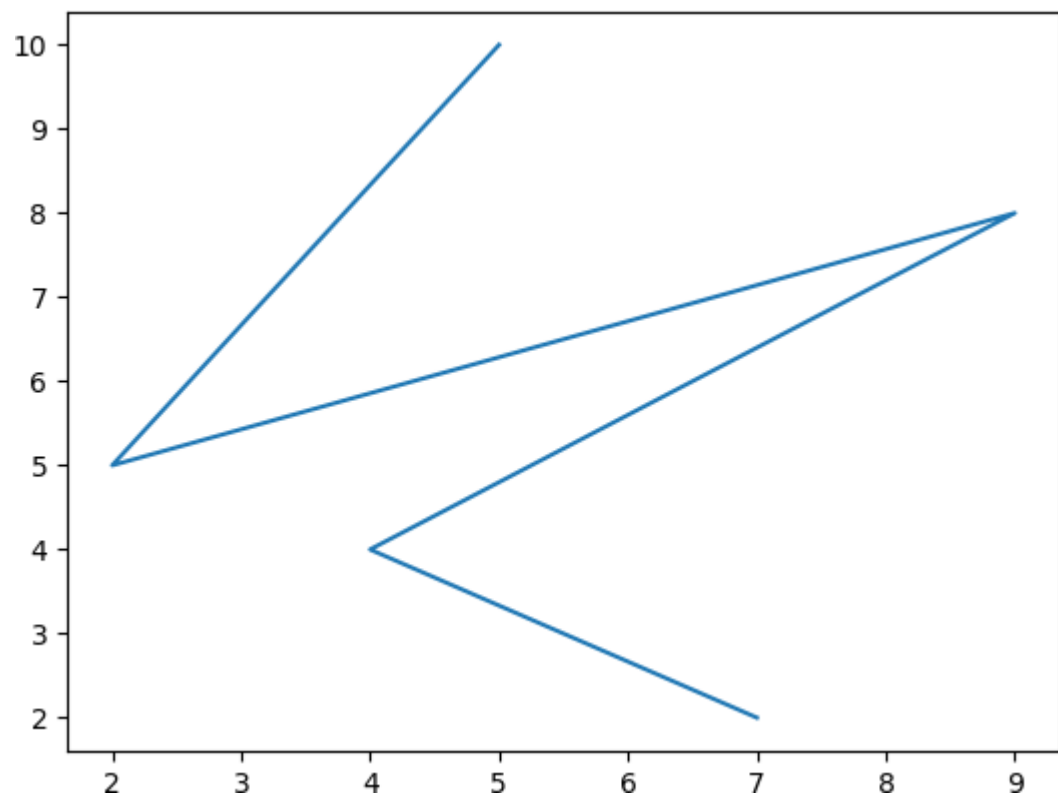
Process finished with exit code 0
```

## 4 MATPLOTLIB

### 4.1 Plot.

```
from matplotlib import pyplot as plt  
x=[5,2,9,4,7]  
y=[10,5,8,4,2]  
plt.plot(x,y)  
plt.show()
```

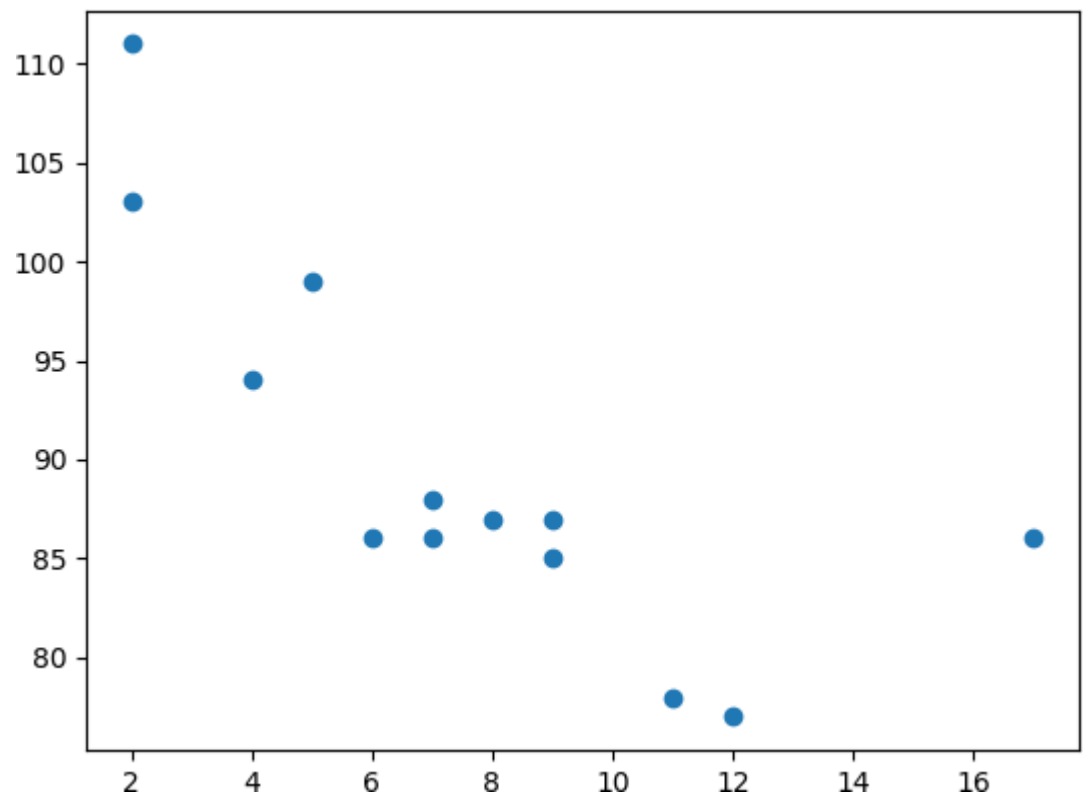
OUTPUT :



## 4.2 Scatter Plot.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)
plt.show()
```

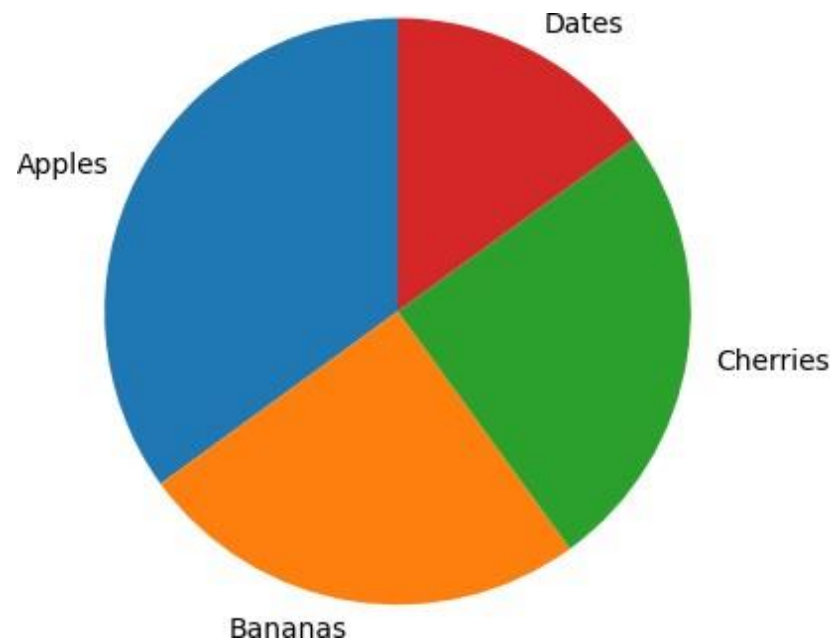
OUTPUT:



### 4.3 Pie Chart.

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
plt.pie(y, labels = mylabels, startangle = 90)
plt.show()
```

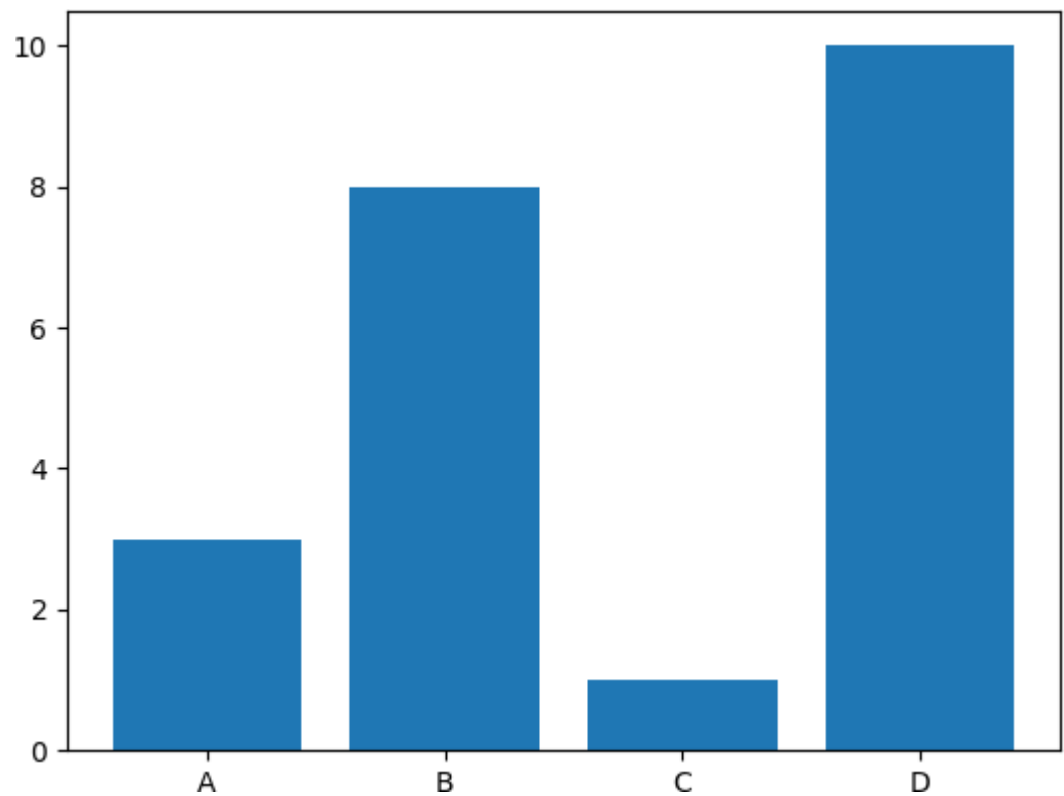
OUTPUT:



#### 4.4 Bar Graph.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x,y)
plt.show()
```

OUTPUT:





# Arti cial Intelligence

July 20, 2022



Arti cial Intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks as, for example, discovering proofs for mathematical theorems or playing chess with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human exibility over wider domains or in tasks requiring much everyday knowledge.

## 5 SCIKIT-LEARN



### 5.1 What is Scikit-Learn (Sklearn)?

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

## Features

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows -

1. **Supervised Learning algorithms** - Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.
2. **Unsupervised Learning algorithms** On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
3. **Clustering** - This model is used for grouping unlabeled data.
4. **Cross Validation** - It is used to check the accuracy of supervised models on unseen data.
5. **Dimensionality Reduction** - It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.
6. **Ensemble methods** As name suggest, it is used for combining the predictions of multiple supervised models.
7. **Feature extraction** - It is used to extract the features from data to define the attributes in image and text data.
8. **Feature selection** - It is used to identify useful attributes to create supervised models.
9. **Open Source** - It is open source library and also commercially usable under BSD license.

## Pros and cons of scikit-learn

### Pros:

- The library is distributed under the BSD license, making it free with minimum legal and licensing restrictions.
- It is easy to use.
- The scikit-learn library is very versatile and handy and serves real-world purposes like the prediction of consumer behavior, the creation of neuroimages, etc.
- Scikit-learn is backed and updated by numerous authors, contributors, and a vast international online community.
- The scikit-learn website provides elaborate API documentation for users who want to integrate the algorithms with their platforms.

### Con:

- It is not the best choice for in-depth learning.
- It is not optimized for graph algorithms, and it is not very good at string processing. For example, scikit-learn does not provide a built-in way to produce a simple word cloud. Scikit-learn doesn't have a strong linear algebra library, hence scipy and numpy are used.

## 6 PYTORCH

### 6.1 What is pytorch?



PyTorch is an open source machine learning library used for developing and training neural network based deep learning models. It is primarily developed by Facebook's AI research group. PyTorch can be used with Python as well as a C++. Naturally, the Python interface is more polished. Pytorch (backed by biggies like Facebook, Microsoft, SalesForce, Uber) is immensely popular in research labs. Not yet on many production servers that are ruled by frameworks like TensorFlow (Backed by Google) Pytorch is picking up fast. Unlike most other popular deep learning frameworks like TensorFlow, which use static computation graphs, PyTorch uses dynamic computation, which allows greater exibility in building complex architectures. Pytorch uses core Python concepts like classes, structures and conditional loops that are a lot familiar to our eyes, hence a lot more intuitive to understand. This makes it a lot simpler than other frameworks like TensorFlow that bring in their own programming style.

## Why do we need PyTorch?

The pyTorch framework can be seen as the future of the deep learning framework. There are many deep learning frameworks getting introduced, and the most preferred frameworks are Tensor ow and PyTorch, but among all, PyTorch is emerging as a winner due to its exibility and computation power. For machine learning and Artificial Intelligence enthusiast, PyTorch is easy to learn and will be very useful to build models.

Here are some of the reasons why developers and researchers learn PyTorch:

1. **Easy to Learn** PyTorch has the same structure as traditional programming, and it has been brilliantly documented with the developer community continuously working to improve. Due to this, it is easy to learn for the programmer and non-programmer.
2. **Developers Productivity** It has an interface with python and with different powerful APIs and can be implemented in Windows or Linux OS. With some programming, knowledge developer can improve their productivity as most of the tasks from PyTorch can be Automated.
3. **Easy to Debug** It can use debugging tools like pdb and ipdb tools of python. As PyTorch develops a computational graph at runtime programmer can use Pythons IDE PyCharm for debugging.
4. **Data Parallelism** It can distribute the computational tasks among multiple CPUs or GPU. This is possible using the data parallelism(`torch.nn.DataParallel`) feature, which wraps any module and helps us do parallel processing.
5. **Useful Libraries** It has a large community of developers and researchers who built tools and libraries to extend PyTorch. This community helps in developing computer vision, reinforcement learning, NLP for research and production purposes. some of the popular libraries are GPyTorch, BoTorch, and Allen NLP. The rich set of powerful APIs helps to extend the PyTorch framework.

## Features

The major features of PyTorch are mentioned below .

1. **Easy Interface** . PyTorch offers easy to use API; hence it is considered to be very simple to operate and runs on Python. The code execution in this framework is quite easy.
2. **Python usage** . This library is considered to be Pythonic which smoothly integrates with the Python data science stack. Thus, it can leverage all the services and functionalities offered by the Python environment.
3. **Computational graphs** . PyTorch provides an excellent platform which offers dynamic computational graphs. Thus a user can change them during runtime. This is highly useful when a developer has no idea of how much memory is required for creating a neural network model.

PyTorch is known for having three levels of abstraction as given below .

- **Tensor** . Imperative n-dimensional array which runs on GPU.
- **Variable** . Node in computational graph. This stores data and gradient.
- **Module** . Neural network layer which will store state or learnable weights.

## Advantages

1. It is easy to debug and understand the code.
2. It includes many layers as Torch.
3. It includes lot of loss functions.
4. It can be considered as NumPy extension to GPUs.
5. It allows building networks whose structure is dependent on computation itself.

## Disadvantages

1. It has been released in 2016, so it's new compared to others and has fewer users, and is not widely known.
2. Absence of monitoring and visualization tools like a tensor board.
3. The developer community is small compared to other frameworks.

## Application of PyTorch

1. **Computer Vision** It is used a convolution neural network to develop image classification, object detection, and generative application. Using PyTorch, a programmer can process images and videos to develop a highly accurate and precise computer vision model.
2. **Natural Language Processing** It can be used to develop the language translator, language modeling, and to develop a chatbot. It uses RNN, LSTM, etc. Architecture to develop natural language, processing models.
3. **Reinforcement Learning** It is used to develop Robotics for automation, Business strategy planning or robot motion control, etc. It uses Deep Q learning architecture to build a model.



## 7 TENSORFLOW



### 7.1 What is TensorFlow?

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses data flow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources.

Currently, the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

**Features of TensorFlow** Let us learn some exciting TensorFlow features:

1. **Open-source Library** It is an open-source library that allows rapid and easier calculations in machine learning. It eases the switching of algorithms from one tool to another TensorFlow tool.

With the help of python, it provides the front-end API for the development of various machines and deep learning algorithms.

2. **Easy to run** We can execute TensorFlow applications on various platforms such as Android, Cloud, IOS and various architectures such as CPUs and GPUs. This allows it to be executed on various embedded platforms.

TensorFlow has its own designed hardware to train the neural models known as Cloud TPUs (TensorFlow Processing unit).

3. **Fast Debugging** It allows you to re-ect each node, i.e., operation individually concerning its evaluation. Tensor Board works with the graph to visualize its working using its dashboard. It provides computational graphing methods that support an easy to execute paradigm.

4. **Efctive** It works with multi-dimensional arrays with the help of data structure tensor which represents the edges in the ow graph. Tensor identi es each structure using three criteria: rank, type, shape.

5. **Scalable** It provides room for prediction of stocks, products, etc with the help of training using the same models and di erent data sets. It also allows for synchronous and asynchronous learning techniques and data ingestion. The graphical approach secures the distributed execution parallelism.

6. **Easy Experimentation** TensorFlow transforms the raw data to the estimators-a form of data neural networks understand. TensorFlow feature columns allow the bridge between raw data and estimators to train the model. This adds the agility to the model for fast developmental insights.

7. **Abstraction** TensorFlow provides a de ned level of abstraction by reducing the code length and cutting the development time. The user needs to focus on logic disregarding the proper way of providing input to functions. A user can choose the model apt according to the system's requirement.

8. **Flexibility** TensorFlow provides the process of resolving complex topologies with the support of Keras API and data input pipelines. Keras provides easy prototyping and suits best for object-oriented neural networks.

TensorFlow eases the mechanism of machine learning with the assistance of such characteristics. It allows the user to create and manipulate the system to create di erent types of real-time models.

# Advantages of TensorFlow

- **Open-source platform** It is an open-source platform that makes it available to all the users around and ready for the development of any system on it.
- **Data visualization** TensorFlow provides a better way of visualizing data with its graphical approach. It also allows easy debugging of nodes with the help of TensorBoard. This reduces the effort of visiting the whole code and effectively resolves the neural network.
- **Keras friendly** TensorFlow has compatibility with Keras, which allows its users to code some high-level functionality sections in it. Keras provides system-specific functionality to TensorFlow, such as pipelining, estimators, and eager execution. The Keras functional API supports a variety of topologies with different combinations of inputs, output, and layers. Scalable Almost every operation can be performed using this platform. With its characteristic of being deployed on every machine and graphical representation of a model allows its users to develop any kind of system using TensorFlow. Hence TensorFlow has been able to develop systems like Airbnb, Dropbox, Intel, Snapchat, etc.
- **Compatible** It is compatible with many languages such as C++, JavaScript, Python, C#, Ruby, and Swift. This allows a user to work in an environment they are comfortable in.
- **Parallelism** TensorFlow finds its use as a hardware acceleration library due to the parallelism of work models. It uses different distribution strategies in GPU and CPU systems.
- **A user can choose to run its code on either of the architecture based on the modeling rule.** A system chooses a GPU if not specified. This process reduces the memory allocation to an extent.
- **Architectural support** TensorFlow also has its architecture TPU, which performs computations faster than GPU and CPU. Models built using TPU can be easily deployed on a cloud at a cheaper rate and executed at a faster rate.

- Graphical support Deep learning uses TensorFlow for its development as it allows building neural networks with the help of graphs that represent operations as nodes.

## Disadvantages of TensorFlow

- Frequent updates TensorFlow releases different updates every 2-3 month, increasing the overhead for a user to install it and bind it with the existing system.
- Inconsistent TensorFlow provides homonyms that share similar names but different implementations, which makes it confusing to remember and use. For eg: `tf.nn.conv2d`, `tf.nn.convolution`, `tf.layers.conv2d`, `tf.layers.Conv2d` has varying meanings and often makes it inconsistent with its usability.
- Architectural limitation TensorFlow's architecture TPU only allows the execution of a model not to train it.
- Dependency Although TensorFlow reduces the length of code and makes it easier for a user to access it, it adds a level of complexity to its use. Every code needs to be executed using any platform for its support which increases the dependency for the execution.
- Symbolic loops TensorFlow lags at providing the symbolic loops for indefinite sequences. It has its usage for definite sequences, which makes it a usable system. Hence it is referred to as a low-level API.
- GPU Support TensorFlow has only NVIDIA support for GPU and python support for GPU programming. It does not have any other support.
- Slow speed TensorFlow has low speed with respect to its competitors. It has less usability in comparison to other frameworks.
- Support for Windows TensorFlow does not provide much features for the Windows Operating System users. It opens a wide range of features for the Linux users. But still, Windows users can download TensorFlow using the anaconda prompt or using the pip package.

## Scikit-Learn vs. TensorFlow

Scikit-learn is a widely used open source machine learning library for Python. It's built on top of and integrates with commonly used libraries such as NumPy, SciPy, Matplotlib and pandas, making it accessible and versatile. TensorFlow, also an open-source machine learning library, specializes in deep learning and neural networks. TensorFlow has support for several programming languages, such as Python, C/C++, Java and Javascript, and others. Consider scikit-learn if you're new to machine learning or developing something using non-neural network algorithms. Consider TensorFlow if you want to use a deep learning approach in conjunction with hardware acceleration through GPUs and TPUs, or on a cluster of computers (which scikit-learn doesn't natively support).

## PyTorch vs. Scikit-Learn

PyTorch is a deep learning software library for Python, C++ and Julia. PyTorch is primarily used for end-to-end building and training of deep neural networks with the ability to create custom models and learning algorithms. Scikit-learn is a library for traditional machine learning algorithms used for clustering, classification, regression, etc. Scikit-learn's black-box nature makes it more accessible to those who are relatively new to machine learning. Consider PyTorch if you're developing applications that have computationally expensive tasks. With PyTorch, you're also able to use GPU acceleration to your advantage. Consider scikit-learn if you're developing a small, exploratory project that doesn't require a substantial amount of data. With scikit-learn your focus would not be on customization, but on the speed and user-friendliness of the machine learning algorithms.

## PyTorch vs. TensorFlow

PyTorch is a deep learning framework with a pythonic and object oriented approach. PyTorch has more debugging and testing options than TensorFlow. TensorFlow is a low-level deep learning library that provides workarounds to high-level APIs such as Keras - albeit with less computational power. TensorFlow is currently more widely used than PyTorch. Consider PyTorch for its many debugging capabilities, if Python is central to your development. Consider TensorFlow if you want a library compatible with various coding languages. TensorFlow also has extensive multi-platform support.

## 8 AI IN BANKING HOW ARTIFICIAL INTELLIGENCE IS USED IN BANKS

Artificial intelligence (AI) technology has become a critical disruptor in almost every industry and banking is no exception. The introduction of AI in banking apps and services has made the sector more customer-centric and technologically relevant.

AI-based systems can help banks reduce costs by increasing productivity and making decisions based on information unfathomable to a human agent. Also, intelligent algorithms are able to spot anomalies and fraudulent information in a matter of seconds.

A report by Business Insider suggests that nearly 80% of banks are aware of the potential benefits that AI presents to their sector. Another report suggests that by 2023, banks are projected to save \$447 billion by using AI apps. These numbers indicate that the banking and finance sector is swiftly moving towards AI to improve efficiency, service, productivity, and ROI and reduce costs.

In this article, we will find out the key applications of AI in the finance and banking sector and how this technology is redefining customer experience with its exceptional benefits.

Applications of AI in banking and finance Artificial intelligence technologies have become an integral part of the world we live in, and banks have started integrating these technologies into their products and services at scale to remain relevant.

Here are some major AI applications in the banking industry through which you can reap the numerous benefits



of the technology. So, let's dive in!

### 8.1 Cybersecurity and fraud detection

Every day, huge quantities of digital transactions take place as users pay bills, withdraw money, deposit checks, and do a lot more via apps or online accounts. Thus, there is an increasing need for the banking sector to ramp up its cybersecurity and fraud detection efforts.

This is when artificial intelligence in banking comes to play. AI can help banks improve the security of online finance, track the loopholes in their systems, and minimize risks. AI along with machine learning can easily identify fraudulent activities and alert customers as well as banks.

For instance, Danske Bank, Denmark's largest bank, implemented a fraud detection algorithm to replace its old rules-based fraud detection system. This deep learning tool increased the bank's fraud detection capability by 50%

and reduced false positives by 60%. The system also automated a lot of crucial decisions while routing some cases to human analysts for further inspection.

AI can also help banks to manage cyber threats. In 2019, the financial sector accounted for 29% of all cyber attacks, making it the most-targeted industry. With the continuous monitoring capabilities of artificial intelligence in financial services, the banks can respond to potential cyberattacks before they affect employees, customers, or internal systems.

## 8.2 Chatbots

Undoubtedly, chatbots are one of the best examples of practical applications of artificial intelligence in banking. Once deployed, they can work 24\*7, unlike humans who have fixed working hours.

Additionally, they keep on learning about the usage pattern of a particular customer. It helps them understand the requirements of a user in an efficient manner.

By integrating chatbots into banking apps, the banks can ensure that they are available for their customers round the clock. Moreover, by understanding customer behavior, chatbots are able to offer personalized customer support and recommend suitable financial services and products accordingly.

One of the best examples of AI chatbot in banking apps is Erica, a virtual assistant from the Bank of America. This AI chatbot can handle tasks like credit card debt reduction and card security updates. Erica managed over 50 million client requests in 2019.



### 8.3 Loan and credit decisions

Banks have started incorporating AI-based systems to make more informed, safer, and profitable loan and credit decisions. Currently, many banks are still too connected to the use of credit history, credit scores, and customer references to determine the creditworthiness of an individual or company.

However, one cannot deny that these credit reporting systems are often riddled with errors, missing real-world transaction history, and misclassifying creditors.

An AI-based loan and credit system can look into the behavior and patterns of customers with limited credit history to determine their creditworthiness. Also, the system sends warnings to banks about specific behaviors that may increase the chances of default. In short, such technologies are playing a key role in changing the future of consumer lending.

### 8.4 Tracking market trends

Artificial intelligence in financial services helps banks to process large volumes of data and predict the latest market trends, currencies, and stocks. Advanced machine learning techniques help evaluate market sentiments and suggest investment options.

AI for banking also suggests the best time to invest in stocks and warns when there is a potential risk. Due to its high data processing capacity, this emerging technology also helps speed up decision-making and makes trading convenient for both banks and their clients.

## 8.5 Data collection and analysis

Banking and finance institutions record millions of transactions every single day. Since the volume of information generated is enormous, its collection and registration turn into an overwhelming task for employees. Structuring and recording such a huge amount of data without any error becomes impossible.

In such scenarios, AI-based innovative solutions can help in efficient data collection and analysis. This, in turn, improves the overall user experience. The information can also be used for detecting fraud or making credit decisions.

## 8.6 Customer experience

Customers are constantly looking for a better experience and convenience. For example, ATMs were a success because customers could avail essential services of depositing and withdrawing money even when banks were closed.

This level of convenience has only inspired more innovation. Customers can now open bank accounts from the comfort of their homes using their smartphones.

Integrating artificial intelligence in banking and finance services will further enhance consumer experience and increase the level of convenience for users. AI technology reduces the time taken to record Know Your Customer (KYC) information and eliminate errors. Additionally, new products and financial offers can be released on time.

Eligibility for cases such as applying for a personal loan or credit gets automated using AI, which means clients can eliminate the hassle of going through the entire process manually. In addition, AI-based software can reduce

approval times for facilities such as loan disbursement.

AI banking also helps to accurately capture client information to set up accounts without any error, ensuring a smooth experience for the customers.

#### 8.7 Risk management

External global factors such as currency fluctuations, natural disasters, or political unrest have serious impacts on banking and financial industries. During such volatile times, it's crucial to take business decisions extra cautiously. AI-driven analytics can give a reasonably clear picture of what is to come and help you stay prepared and make timely decisions.

AI also helps in risky applications by evaluating the probability of a client failing to pay back a loan. It predicts this future behavior by analyzing past behavioral patterns and smartphone data.

#### 8.8 Regulatory compliance

Banking is one of the highly regulated sectors of the economy worldwide. Governments use their regulatory authority to ensure that banking customers are not using banks to perpetrate financial crimes and that banks have acceptable risk profiles to avoid large-scale defaults.

In most cases, banks maintain an internal compliance team to deal with these problems, but these processes take a lot more time and require huge investment when done manually. The compliance regulations are also subject to frequent change, and banks need to update their processes and workflows following these regulations constantly.

AI uses deep learning and NLP to read new compliance

requirements for financial institutions and improve their decision-making process. Even though AI banking can't replace a compliance analyst, it can make their operations faster and efficient.

#### 8.9 Predictive analytics

One of AI's most common use cases includes general-purpose semantic and natural language applications and broadly applied predictive analytics. AI can detect specific patterns and correlations in the data, which traditional technology could not previously detect.

These patterns could indicate untapped sales opportunities, cross-sell opportunities, or even metrics around operational data, leading to a direct revenue impact.

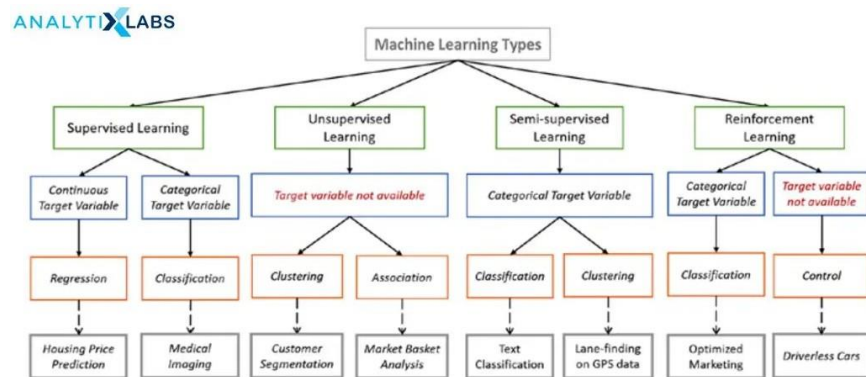
#### 8.10 Process automation

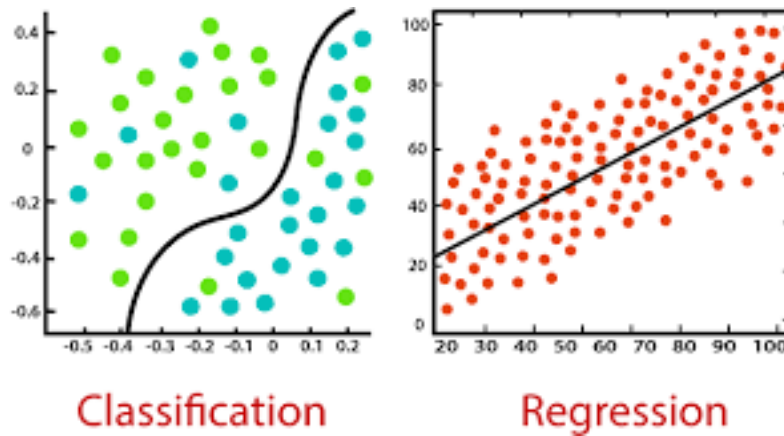
Robotic process automation (RPA) algorithms increase operational efficiency and accuracy and reduce costs by automating time-consuming repetitive tasks. This also allows users to focus on more complex processes requiring human involvement.

As of today, banking institutions successfully leverage RPA to boost transaction speed and increase efficiency. For example, JPMorgan Chase's CoiN technology reviews documents and derives data from them much faster than humans can.

# TYPES OF MACHINE LEARNING

1. SUPERVISED LEARNING.
2. UNSUPERVISED LEARNING.



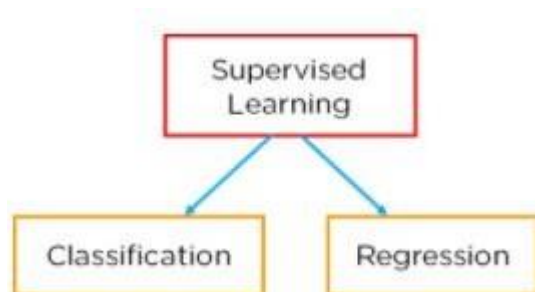


## Part I

# **SUPERVISED LEARNING.**

Suppose you are trying to learn a new concept in maths and after solving a problem, you may refer to the solutions to see if you were right or not. Once you are confident in your ability to solve a particular type of problem, you will stop referring to the answers and solve the questions put before you by yourself.

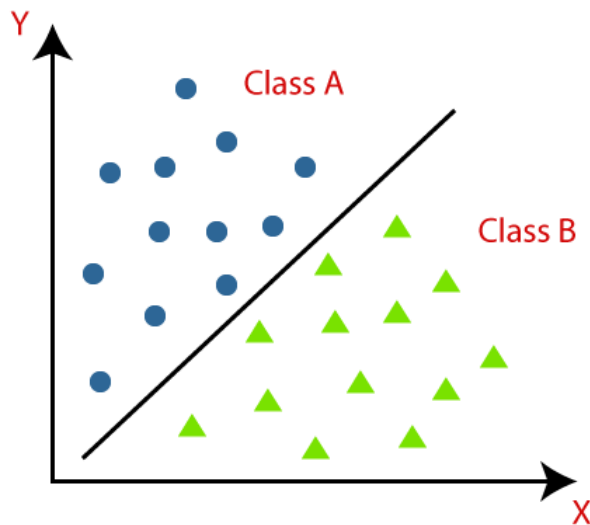
This is also how Supervised Learning works with machine learning models. In Supervised Learning, the model learns by example. Along with our input variable, we also give our model the corresponding correct labels. While training, the model gets to look at which label corresponds to our data and hence can find patterns between our data and those labels.



## 9 CLASSIFICATION.

Classification is a process of categorizing a given set of data into classes. It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

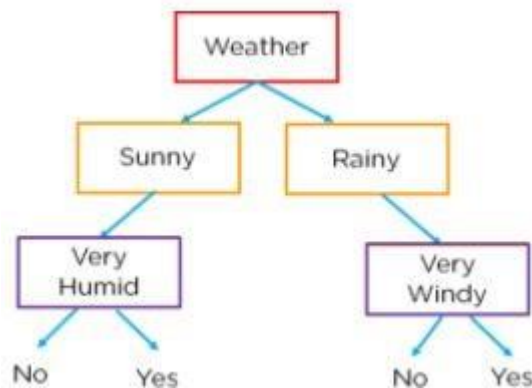
Classification is defined as the process of recognition, understanding, and grouping of objects and ideas into preset categories a.k.a sub-populations. With the help of these pre-categorized training datasets, classification in machine learning programs leverage a wide range of algorithms to classify future datasets into respective and relevant categories.



### Classification Models

#### 9.1 Decision Tree

A Decision Tree is an algorithm that is used to visually represent decision-making. A Decision Tree can be made by asking a yes/no question and splitting the answer to lead to another decision. The question is at the node and it places the resulting decisions below at the leaves. The tree depicted below is used to decide if we can play tennis.

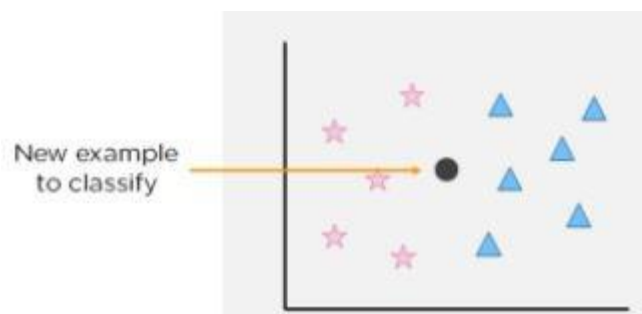


### Decision Tree

In the above figure, depending on the weather conditions and the humidity and wind, we can systematically decide if we should play tennis or not. In decision trees, all the False statements lie on the left of the tree and the True statements branch off to the right. Knowing this, we can make a tree which has the features at the nodes and the resulting classes at the leaves.

## 9.2 K-Nearest Neighbor

- K-Nearest Neighbor is a classification and prediction algorithm that is used to divide data into classes based on the distance between the data points. K-Nearest Neighbor assumes that data points which are close to one another must be similar and hence, the data point to be classified will be grouped with the closest cluster. data-classified



Data to be classified





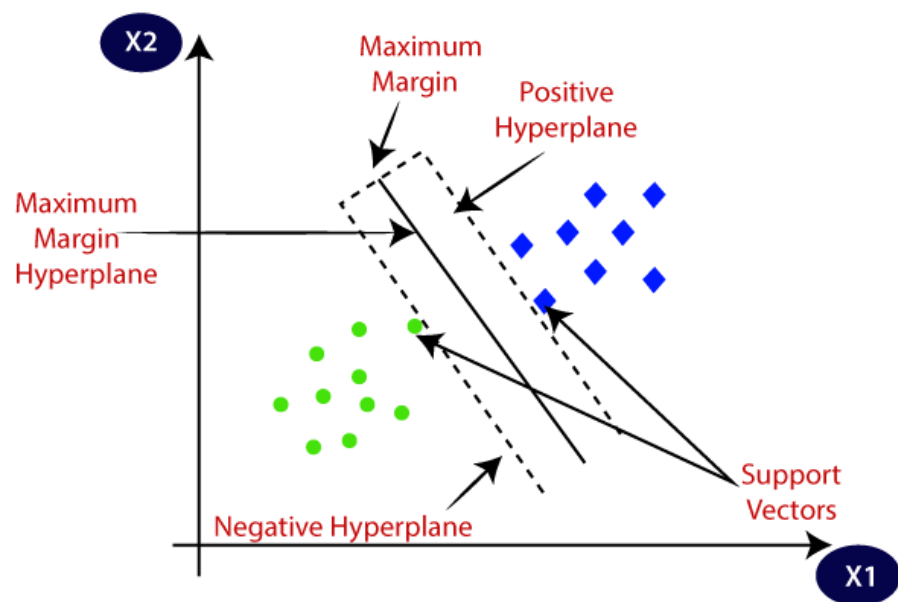
Classification using K-Nearest Neighbours

### 9.3 Kernel Support Vector Machine

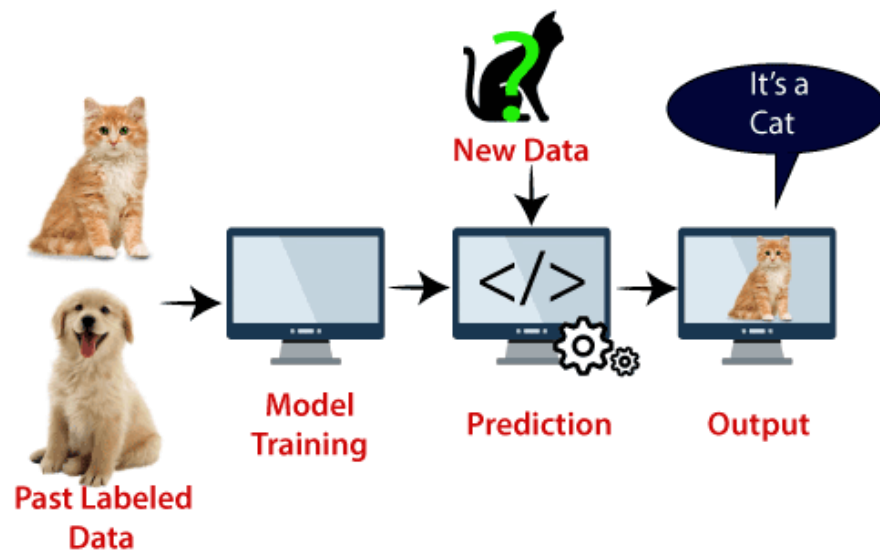
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

Support Vector Machine Algorithm



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



Hyperplane and Support Vectors in the SVM algorithm:

### 9.3.1 Hyperplane

There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

### 9.3.2 Support Vectors

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

## 9.4 Confusion Matrix in Machine Learning

Confusion Matrix in Machine Learning The confusion matrix is a

matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix. Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifiers, the matrix is of 2\*2 table, for 3 classes, it is 3\*3 table, and so on.
- The matrix is divided into two dimensions, that are predicted values and actual values along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.
- It looks like the below table:

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

#### Confusion Matrix in Machine Learning

The above table has the following cases:

- True Negative: Model has given prediction No, and the real or actual value was also No.
- True Positive: The model has predicted yes, and the actual value was also true.
- False Negative: The model has predicted no, but the actual value was Yes, it is also called as Type-II error.
- False Positive: The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

n = 100	Actual: No	Actual: Yes	
Predicted: No	TN: 65	FP: 3	68
Predicted: Yes	FN: 8	TP: 24	32
	73	27	

### Need for Confusion Matrix in Machine learning?

- It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is.
- It not only tells the error made by the classifiers but also the type of errors such as it is either type-I or type-II error.
- With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

Example: We can understand the confusion matrix using an example.

Suppose we are trying to create a model that can predict the result for the disease that is either a person has that disease or not. So, the confusion matrix for this is given as:

#### Confusion Matrix in Machine Learning

- From the above example, we can conclude that:
- The table is given for the two-class classifier, which has two predictions "Yes" and "NO." Here, Yes denotes that patient has the disease, and No denotes that patient does not have that disease.
- The classifier has made a total of 100 predictions. Out of 100 predictions, 89 are true predictions, and 11 are incorrect predictions.
- The model has given prediction "yes" for 32 times, and "No" for 68 times. Whereas the actual "Yes" was 27, and actual "No" was 73 times.

## 10 REGRESSION

Regression models are used to predict a continuous value. Predicting prices of a house given the features of house like size, price etc is one of the common examples of Regression. It is a supervised technique. A detailed explanation on types of Machine Learning and some important concepts is given in my previous article.

Types of Regression

1. Simple Linear Regression
2. Polynomial Regression
3. Support Vector Regression
4. Decision Tree Regression
5. Random Forest Regression

### 10.1 Simple linear regression

This is one of the most common and interesting type of Regression technique. Here we predict a target variable  $Y$  based on the input variable  $X$ . A linear relationship should exist between target variable and predictor and so comes the name Linear Regression.

### 10.2 Polynomial regression

In polynomial regression, we transform the original features into polynomial features of a given degree and then apply Linear Regression on it

### 10.3 Support vector regression

In SVR, we identify a hyperplane with maximum margin such that the maximum number of data points are within that margin. SVRs are almost similar to the SVM classification algorithm. We will discuss the SVM algorithm in detail in my next article.

Instead of minimizing the error rate as in simple linear regression, we try to find the error within a certain threshold. Our objective in SVR is to basically consider the points that are within the margin. Our best line is the hyperplane that has the maximum number of points.

## 10.4 Decision trees

Decision trees can be used for classification as well as regression. In decision trees, at each level, we need to identify the splitting attribute. In the case of regression, the ID3 algorithm can be used to identify the splitting node by reducing the standard deviation (in classification information gain is used).

A decision tree is built by partitioning the data into subsets containing instances with similar values (homogenous). Standard deviation is used to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous, its standard deviation is zero.

## 10.5 Random forest

Random forest is an ensemble approach where we take into account the predictions of several decision regression trees.

- Select K random points

- Identify n where n is the number of decision tree regressors to be created.

- Repeat steps 1 and 2 to create several regression trees.

The average of each branch is assigned to the leaf node in each decision tree.

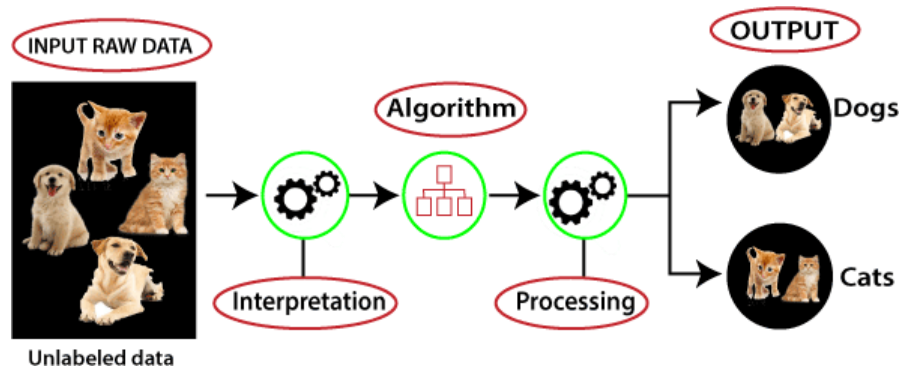
To predict output for a variable, the average of all the predictions of all decision trees are taken into consideration.

Random Forest prevents over fitting (which is common in decision trees) by creating random subsets of the features and building smaller trees using these subsets.

## Part II

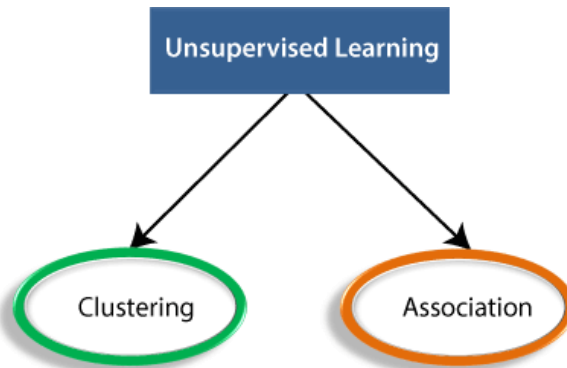
### UNSUPERVISED LEARNING.

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models find the hidden patterns and insights from the given data. Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision. Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.





## Types of Unsupervised Learning.



- Clustering: Clustering is a method of grouping the objects into clusters such that objects with most similarities remain in a group and have less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- Association: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

### 11 CLUSTERING

Clustering is the process of grouping similar entities together. The goal of this unsupervised machine learning technique is to find similarities in the data point and group similar data points together.

Grouping similar entities together helps profile the attributes of different groups. In other words, this will give us insight into underlying patterns of different groups. There are many applications

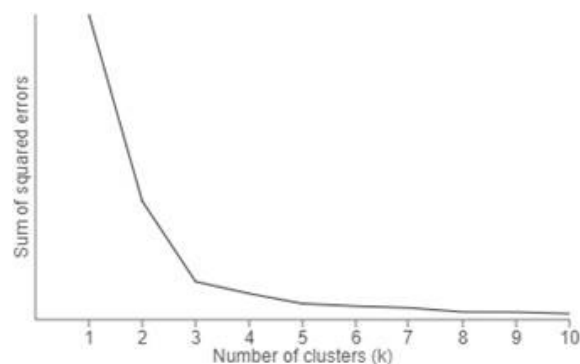
of grouping unlabeled data, for example, you can identify different groups/segments of customers and market each group in a different way to maximize the revenue. Another example is grouping documents together which belong to the similar topics etc. Clustering is also used to reduce the dimensionality of the data when you are dealing with a copious number of variables.

There are many algorithms developed to implement this technique but for this post, let's stick to the most popular and widely used algorithms in machine learning.

### 11.1 K-mean Clustering

It starts with  $K$  as the input which is how many clusters you want to find.

1. Place  $K$  centroids in random locations in your space.
2. Now, using the euclidean distance between data points and centroids, assign each data point to the cluster which is close to it.
3. Recalculate the cluster centers as a mean of data points assigned to it.
4. Repeat 2 and 3 until no further changes occur.
5. Now, you might be thinking that how do I decide the value of  $K$  in the first step.



One of the methods is called Elbow method can be used to decide an optimal number of clusters. Here you would run K-mean clustering on a range of K values and plot the percentage of variance explained on the Y-axis and K on X-axis.

In the picture below you would notice that as we add more clusters after 3 it doesn't give much better modeling on the data. The first cluster adds much information, but at some point, the marginal gain will start dropping.

## 11.2 K-mode Clustering

Most of the real world datasets are in categorical form. Let's say, if we are working on analysing the social media, we have categorical data like gender (male or female), profession and so on. So deal with all this categorical data or cluster the categorical variables we use K Modes Clustering. It is widely used algorithm for grouping the categorical data because it is easy to implement and efficiently handles large amount of data. . It defines clusters based on the number of matching categories between data points. (This is in contrast to the more well-known k-means algorithm, which clusters numerical data based on Euclidean distance.)

How is it used???

The k-modes clustering algorithm is an extension of k-means clustering algorithm. The k-means algorithm is the most widely used centre based partitioning clustering algorithm. Huang extends the k-means clustering algorithm to k-modes clustering algorithm to group the categorical data.

The modifications done in the k-means are -

- (i) using a simple matching dissimilarity measure for categorical objects,
- (ii) replacing means of clusters by modes, and
- (iii) using a frequency-based method to update the modes.

## Part III

# PROGRAMS

## 12 SIMPLE LINEAR REGRESSION

### 12.1 Importing the library:

```
[1] 1. import numpy as np
    2. import matplotlib.pyplot as plt
    3. import pandas as pd
```

### 12.2 Importing the dataset

```
[2] 1. dataset = pd.read_csv('Salary_Data.csv')
    2. X = dataset.iloc[:, :-1].values
    3. Y = dataset.iloc[:, :-1].values
```

### 12.3 Splitting the dataset into the Training set and Test set

```
[3] 1. from sklearn.model_selection import train_test_split
    2. X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=1/3,random_state= 0)
```

### 12.4 Training the Simple Linear Regression model on the Training set

```
[4] 1. from sklearn.linear_model import LinearRegression
    2. regressor = LinearRegression()
    3. regressor.fit(X_train, y_train)
    LinearRegression()
```

### 12.5 Predicting the Test set result.

```
[5] 1. y_pred = regressor.predict(X_test)
```

### 12.6 Visualising the Training set result

```
[6] 1. plt.scatter(X_train, y_train,color = 'red' )
    2. plt.plot(X_train,regressor.predict(X_train) , color = 'blue')
    3. plt.title('Salary vs Experience (Training Set)' )
    4. plt.xlabel('Years of Experience' )
```

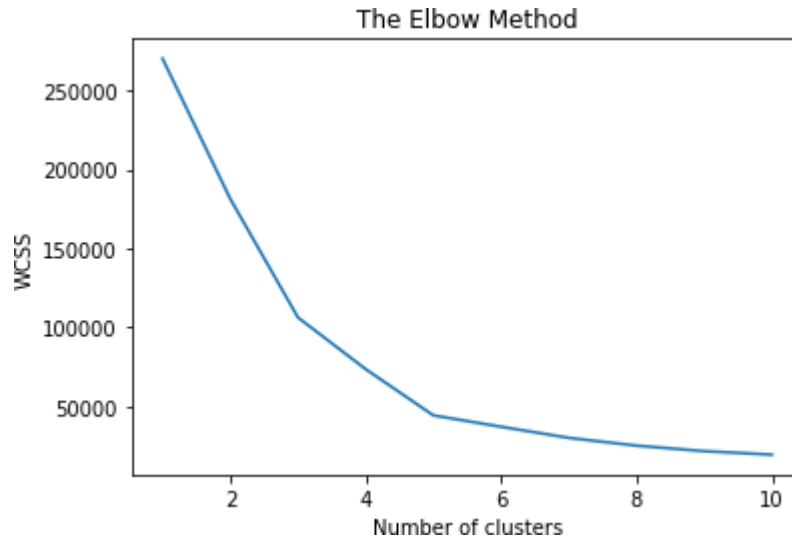
5. `plt.ylabel('Salary')`
6. `plt.show()`



## 12.7 Visualising the Testing set result.

- [7]
1. `plt.scatter(X_test, y_test, color = 'red')`
  2. `plt.plot(X_train, regressor.predict(X_train), color = 'blue')`
  3. `plt.title('Salary vs Experience (Test set)')`
  4. `plt.xlabel('Years of Experience')`
  5. `plt.ylabel('Salary')`
  6. `plt.show()`





## 13 K- MEANS CLUSTERING.

### 13.1 Importing the libraries

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

### 13.2 Importing the dataset

```
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
```

### 13.3 Using the elbow method to find the optimal number of clusters

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
```



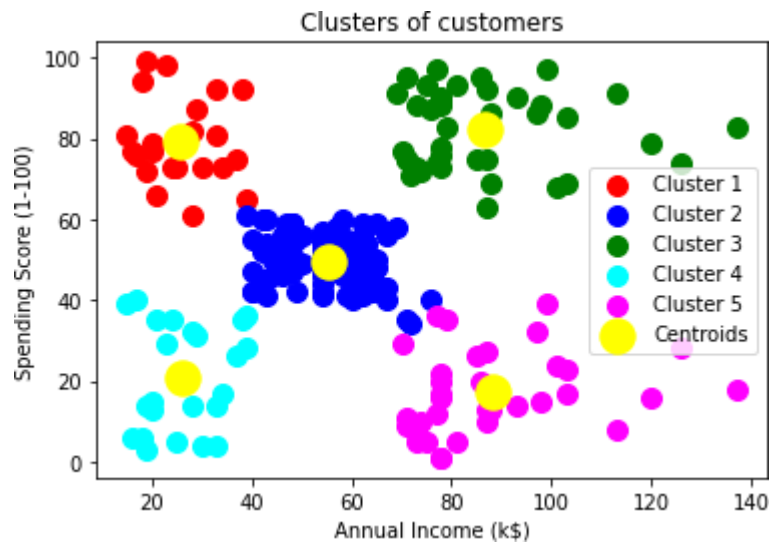
```
plt.ylabel('WCSS')  
plt.show()
```

## 13.4 Training the K-Means model on the dataset

```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42) y_kmeans =  
kmeans.fit_predict(X)
```

## 13.5 Visualising the clusters

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster  
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster  
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluste  
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster  
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Clus  
plt.scatter(kmeans.cluster_centers_[0], kmeans.cluster_centers_[1], s = 300, c = 'yel  
plt.title('Clusters of customers')  
plt.xlabel('Annual Income (k$)')  
plt.ylabel('Spending Score (1-100)') plt.legend()  
plt.show()
```



## 14 Polynomial Regression

### 14.1 Importing the library:

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

## 14.2 Importing the dataset

```
dataset = pd.read_csv('position_Salaries.csv') X =  
dataset.iloc[:, 1:-1].values  
Y = dataset.iloc[:, -1].values
```

## 14.3 Training the Linear Regression model on the whole dataset

```
from sklearn.model_selection import LinearRegression lin_reg =  
LinearRegression()  
lin_reg.fit(X, y)
```

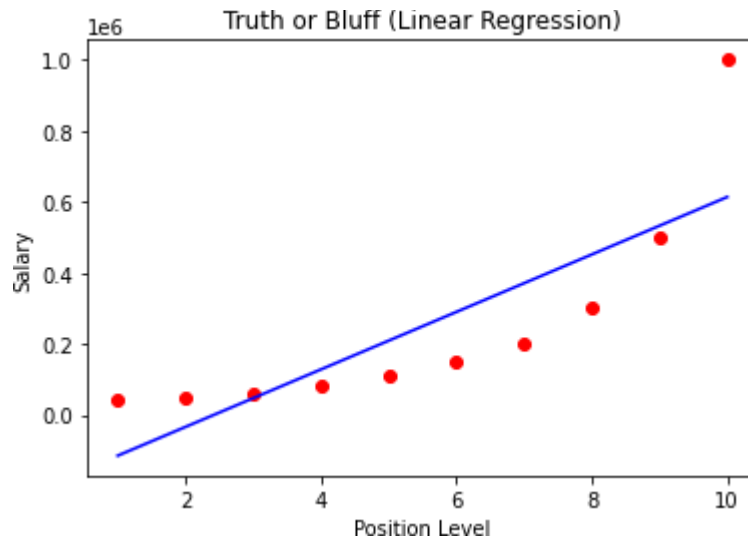
```
LinearRegression(copy_X=True, t_intercept=True, n_jobs=None,  
normalize=False)
```

## 14.4 Training the Polynomial Regression model on the whole dataset

```
from sklearn.preprocessing import PolynomialFeatures poly_reg =  
PolynomialFeatures(degree = 4)  
lin_reg_2 = LinearRegression()  
lin_reg_2.fit(X_poly, y)
```

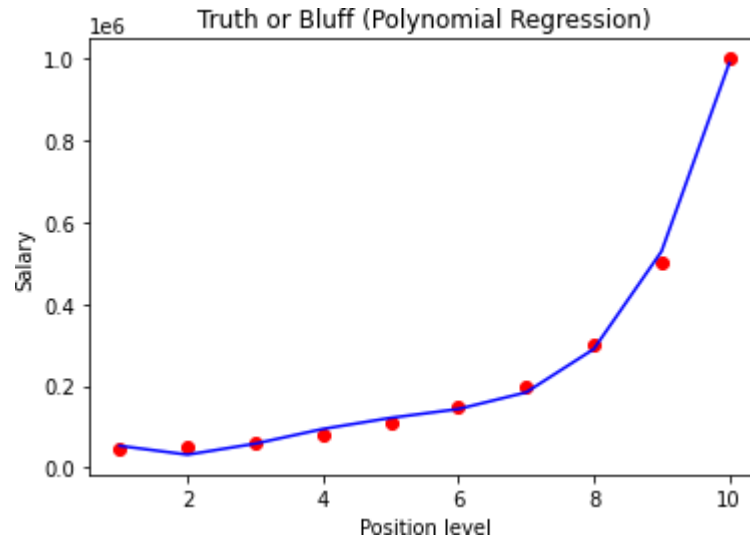
## 14.5 Visualising the Linear Regression results

```
plt.scatter(X, y, color = 'red')  
plt.plot(X, lin_reg.predict(X), color = 'blue') plt.title('Truth  
or Bluff (Linear Regression)') plt.xlabel('Position Level')  
plt.ylabel('Salary')  
plt.show()
```



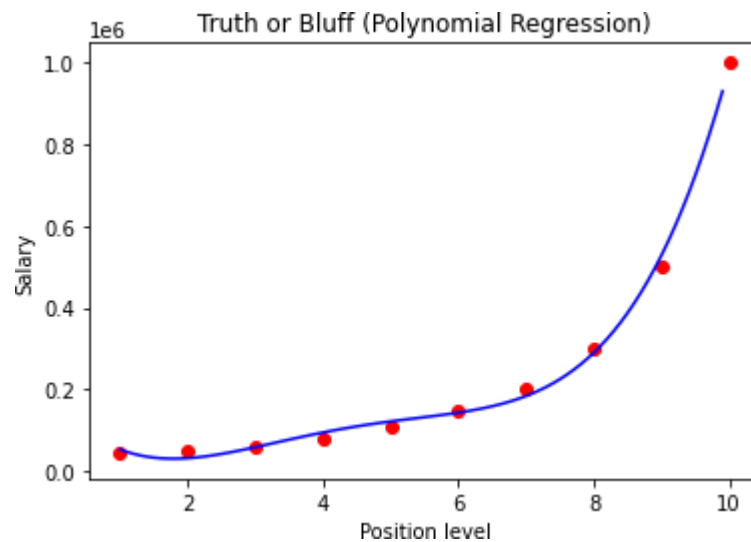
## 14.6 Visualising the Polynomial Regression results

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue') plt.title('Truth or Bluff
(Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary') plt.show()
```



#### 14.7 Visualising the Polynomial Regression results (for higher resolution and smoother curve)

```
X_grid = np.arange(min(X), max(X), 0.1) X_grid =
X_grid.reshape((len(X_grid), 1)) plt.scatter(X, y, color =
'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue') plt.title('Truth or Bluff (Polynomial
Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary') plt.show()
```



## 15 Support Vector Regression (SVR)

### 15.1 Importing the library:

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

### 15.2 Importing the dataset

```
dataset = pd.read_csv('position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
Y = dataset.iloc[:, -1].values
```

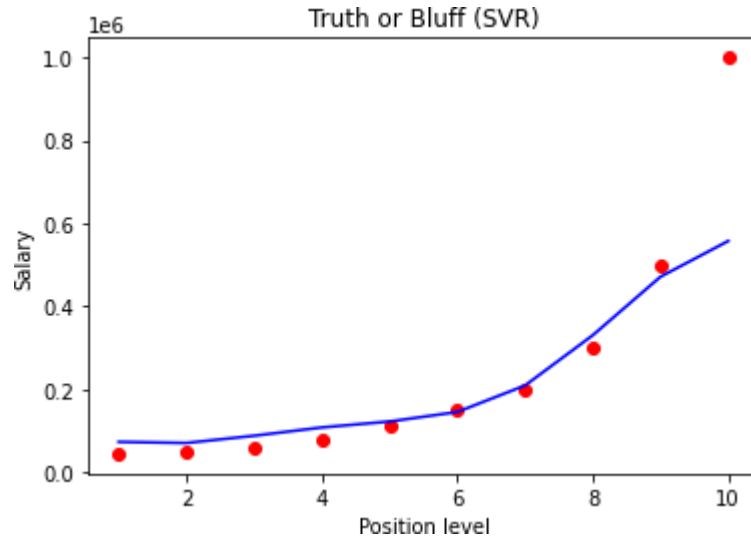
### 15.3 Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

### 15.4 Training the SVR model on the whole dataset

```
from sklearn.svm import SVR regressor =  
SVR(kernel = 'rbf') regressor.fit(X, y)
```





## 15.5 Predicting a new result

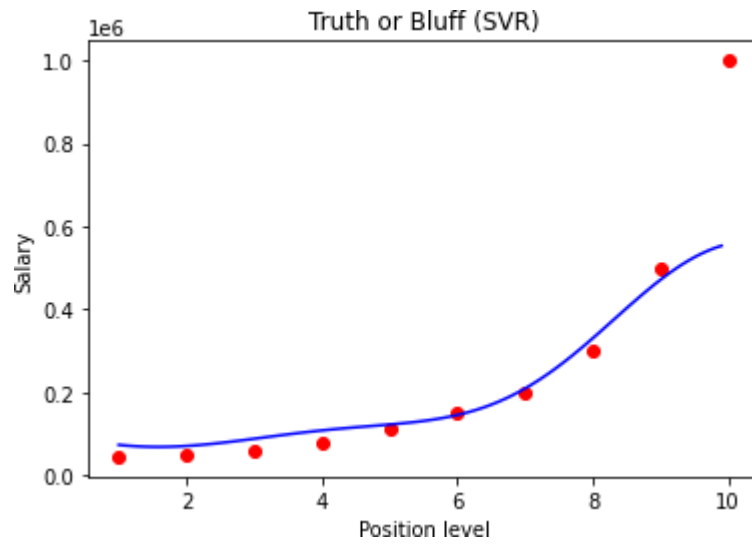
```
sc_y.inverse_transform(regressor.predict(sc_X.transform([[6.5]])))
```

## 15.6 Visualising the SVR results

```
plt.scatter(sc_X.inverse_transform(X), sc_y.inverse_transform(y), color = 'red') plt.plot(sc_X.inverse_transform(X),
sc_y.inverse_transform(regressor.predict(X)), color = 'blue') plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary') plt.show()
```

## 15.7 Visualising the SVR results (for higher resolution and smoother curve)

```
X_grid = np.arange(min(sc_X.inverse_transform(X)), max(sc_X.inverse_transform(X)), 0.1) X_g = X_grid
plt.plot(X_grid, sc_y.inverse_transform(regressor.predict(sc_X.transform(X_grid))), color = 'blue') plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary') plt.show()
```



## 16 Hierarchical Clustering.

### 16.1 Importing the libraries

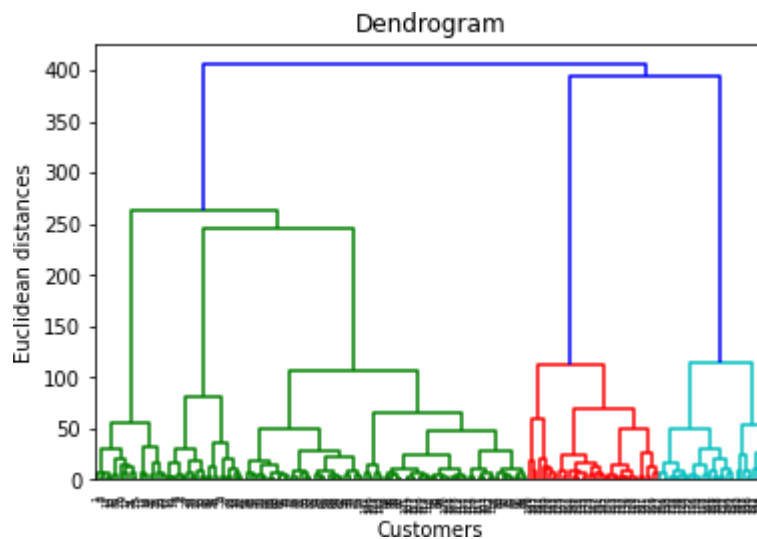
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

### 16.2 Importing the dataset

```
dataset = pd.read_csv('Mall_Customers.csv') X =
dataset.iloc[:, [3, 4]].values
```

### 16.3 Using the dendrogram to find the optimal number of clusters

```
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward')) plt.title('Dendrogram') plt.xlabel('Customers')
plt.ylabel('Euclidean distances') plt.show()
```



### 16.4 Training the Hierarchical Clustering model on the dataset

```
from sklearn.cluster import AgglomerativeClustering
```

```
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward') y_hc = hc.fit_predict(X)
```

## 16.5 Visualising the clusters

```
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Clusters of customers') plt.xlabel('Annual
Income (k$)') plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

