Nanak Dev Engineering College

Ludhiana, Punjab, India

Machine Learning Report:

Name: Arshpreet Singh

Class: I.T. A1

Urn: 2104478

Crn: 2121021

(Btech) Information Technology

Contents

1 Program no.1(Print) **1**

2 Program no.2(If Else) **1**

3 Program no.3(For loop) **2**

4 Program no.4(NumPy) **2**

5 Program no.5(Pandas) **3**

6 Program no.6(MatploLib) **4**

7 Program no.7(Scikit Learn) **4**

8 Report On Aritificial Intelligence(A.I): **5**

8.1 What is AI? 5

8.2 Types of AI: 5

9 Tools for Applications of AI: **6**

9.0.1 Automation. 6

9.0.2 Machine learning. 7

9.0.3 Machine vision. 7

9.0.4 Natural language processing (NLP). 7

9.0.5 Robotics. 8

10 What are the applications of AI? **8**

10.1 AI in healthcare. 8

10.2 AI in business. 9

10.3 AI in education. 10

10.4 AI in finance. 10

10.5 AI in manufacturing. 11

10.6 AI in banking. 12

11 Different Tools of A.I. **12**

11.1 Pytorch: 12

11.1.1 What Is PyTorch, and How Does It Work? 12

11.1.2 The two main features of PyTorch are: 13

11.1.3 Basics of PyTorch 13

11.1.4 Common PyTorch Modules: 14

12 TensorFlow: 14

12.1 Tensor processing unit (TPU) Tensor processing unit: 15

12.2 Features : 15

12.2.1 AutoDifferentiation: 15

12.2.2 Eager execution: 16

12.2.3 Losses: 16

12.2.4 Metrics 16

12.2.5 TF.nn 17

12.2.6 Optimizers 17

13 Scikit-learn: 17

13.1 Features: 18

13.1.1 Datasets 18

13.1.2 Data Splitting 18

13.1.3 Linear Regression 18

13.1.4 Decision Trees 18

14 Supervised Learning techniques: 19

14.1 KNN(K-Nearest Neighbours): 19

14.1.1 Points to Note: 19

14.1.2 Example: 19

14.1.3 Why do we need a K-NN Algorithm? 20

14.1.4 How does K-NN work? 20

14.1.5 KNN Code: 21

14.1.6 Output: 21

14.2 SVM (Support Vector Machine): 22

14.2.1 Points to Note: 22

14.2.2 Example: 23

14.2.3 Hyperplane and Support Vectors in the SVM algorithm: 24

14.2.4 Hyperplane: 24

14.2.5 Support Vectors: 24

14.2.6 SVM Code: 25

14.2.7 Output: 25

14.3 Confusion Matrix in Machine Learning : 26

14.3.1 Making the Confusion Matrix: 26

14.3.2 Output: 27

14.4 Decision Tree Classification: . . . 27
14.4.1 Decision Tree Terminologies . . . 28
14.4.2 Example: . . . 28
14.5 Decision Tree Regression Code: . . . 29
14.5.1 Output: . . . 30
14.6 Random Forest Regression: . . . 30
14.6.1 Example: . . . 31
14.6.2 Random Forest Regression Code: . . . 32
14.6.3 Output: . . . 32

15 Clustering in Machine Learning: 32
15.1 Use of clustering: . . . 33

16 Clustering Algorithms: 33
16.1 K-Means Clustering Algorithm: . . . 34
16.2 K-means++ algorithm: (K++) . . . 34

17 K-Means Clustering code: 34
17.1 Elbow Method(Graph): . . . 35
17.2 Visualising the clusters : . . . 35
17.3 Output: . . . 36

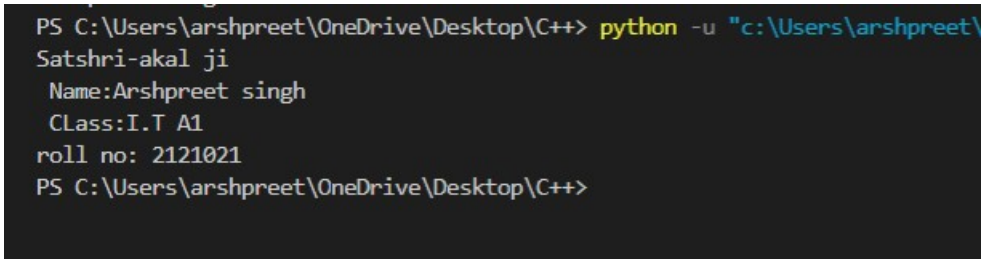
18 References: 36

1 Program no.1(Print)

To Print using Python:

```
print("Satshri-akal ji\n
Name:Arshpreet singh\n
CLass:I.T A1\n
roll no: 2121021")
```

Output:



2 Program no.2(If Else)

To Use If Else Conditions:

```
i = 10
if (i == 10):
          if (i < 15):
              print("i is smaller than 15")
                  if (i < 12):
                      print("i is smaller than 12 too")
          else:         print("i is greater than 15"):
```

Output:

```
PS C:\Users\arshpreet\OneDrive\Desktop\C++> python -u "c:\Users\arshpreet\C
i is smaller than 15
i is smaller than 12 too
PS C:\Users\arshpreet\OneDrive\Desktop\C++>
```

3 Program no.3(For loop)

To use for loop in Python:

```
marks=[98,99,100]
for i in range (len(marks)):
    marks[i]+=5
    print(marks[i])
```

Output:

```
PS C:\Users\arshpreet\OneDrive\Desktop\C++> python -u "c:\Users
103
104
105
PS C:\Users\arshpreet\OneDrive\Desktop\C++>
```

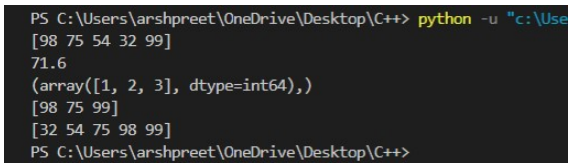
4 Program no.4(NumPy)

To implement NumPy Library

```
import numpy as np
marks=[98,75,54,32,99]
c=np.array((marks))
d=np.mean(c)
e=np.where(c<98)
```

```
filter_arr=c>74
new_arr=c[filter_arr]
sort_arr=np.sort(c)
print(c)
print(d)
print(e)
print(new_arr)
print(sort_arr)
```

Output:

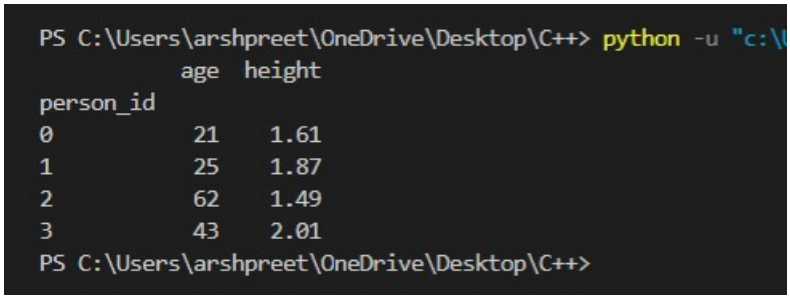


5 Program no.5(Pandas)

To implete Pandas Library:

```
import pandas as pd
df = pd.DataFrame({'person_id': [0, 1, 2, 3],
'age': [21, 25, 62, 43],
'height': [1.61, 1.87, 1.49, 2.01]} ).set_index('person_id')
print(df)
```

Output:

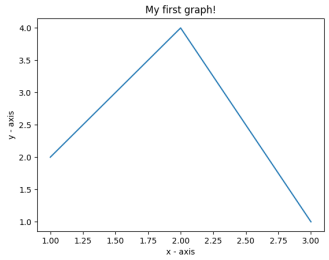


6 Program no.6(MatploLib)

To implement Matplolib Library:

```
import matplotlib.pyplot as plt
x = [1,2,3]
y = [2,4,1]
plt.plot(x, y)
plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.title('My first graph!')
plt.show()
```

Output:



7 Program no.7(Scikit Learn)

To implement Scikit Learn:

```
from sklearn.datasets import load_iris
iris = load_iris() \
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("\nFirst 10 rows of X:\n", X[:10])
```

Output:

```
PS C:\Users\arshpreet\OneDrive\Desktop\C++> python -u "c:\U
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'p
Target names: ['setosa' 'versicolor' 'virginica']

First 10 rows of X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
PS C:\Users\arshpreet\OneDrive\Desktop\C++>
```

8 Report On Aritificial Intelligence(A.I):

8.1 What is AI?

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems.

Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

8.2 Types of AI:

Strong AI vs. weak AI

Strong AI vs. weak AI AI can be categorized as either weak or strong.

Weak AI, also known as narrow AI, is an AI system that is designed and trained to complete

a specific task. Industrial robots and virtual personal assistants, such as Apple's Siri, use weak AI.

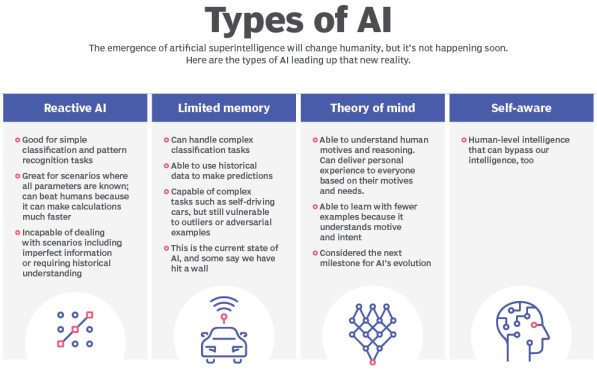
Strong AI, also known as artificial general intelligence (AGI), describes programming that can

replicate the cognitive abilities of the human brain. When presented with an unfamiliar task,

a strong AI system can use fuzzy logic to apply knowledge from one domain to another and

find a solution autonomously. In theory, a strong AI program should be able to pass both

a Turing Test and the Chinese room test.



9 Tools for Applications of AI:

AI is incorporated into a variety of different types of technology. Here are six examples:

9.0.1 Automation.

When paired with AI technologies, automation tools can expand the volume and types of tasks performed.

An example is robotic process automation (RPA), a type of software that automates repetitive,

rules-based data processing tasks traditionally done by humans.

When combined with machine learning and emerging AI tools, RPA can automate bigger portions of enterprise jobs,

enabling RPA's tactical bots to pass along intelligence from AI and respond to process changes.

9.0.2 Machine learning.

This is the science of getting a computer to act without programming.

Deep learning is a subset of machine learning that, in very simple terms,

can be thought of as the automation of predictive analytics.

There are three types of machine learning algorithms:

Supervised learning. Data sets are labeled so that patterns can be detected and used to label new data sets.

Unsupervised learning. Data sets aren't labeled and are sorted according to similarities or differences.

Reinforcement learning. Data sets aren't labeled but, after performing an action or several actions,

the AI system is given feedback.

9.0.3 Machine vision.

This technology gives a machine the ability to see. Machine vision captures and analyzes visual

information using a camera, analog-to-digital conversion and digital signal processing

. It is often compared to human eyesight, but machine vision isn't bound by biology and can

be programmed to see through walls, for example. It is used in a range of applications from

signature identification to medical image analysis.

Computer vision, which is focused on machine-based image processing, is often conflated with machine vision.

9.0.4 Natural language processing (NLP).

This is the processing of human language by a computer program. One of the older and best-known examples

of NLP is spam detection, which looks at the subject line and text of an email and decides if it's junk.

Current approaches to NLP are based on machine learning. NLP tasks include text translation, sentiment analysis and speech recognition.

9.0.5 Robotics.

This field of engineering focuses on the design and manufacturing of robots.

Robots are often used to perform tasks that are difficult for humans to perform or

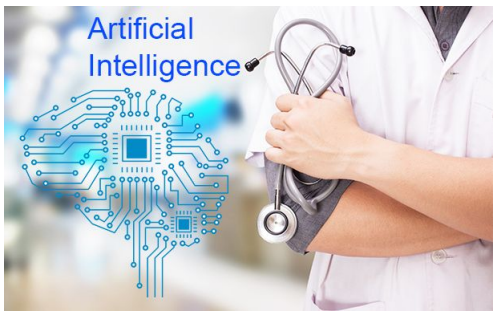
perform consistently. For example, robots are used in assembly lines for car production or by

NASA to move large objects in space. Researchers are also using machine learning to build robots that can interact in social settings.

10 What are the applications of AI?

Artificial intelligence has made its way into a wide variety of markets. Here are nine examples.

10.1 AI in healthcare.



The biggest bets are on improving patient outcomes and reducing costs.

Companies are applying machine learning to make better and faster diagnoses than humans.

One of the best-known healthcare technologies is IBM Watson. It understands natural language

and can respond to questions asked of it. The system mines patient data and other available data sources

to form a hypothesis, which it then presents with a confidence scoring schema. Other AI applications

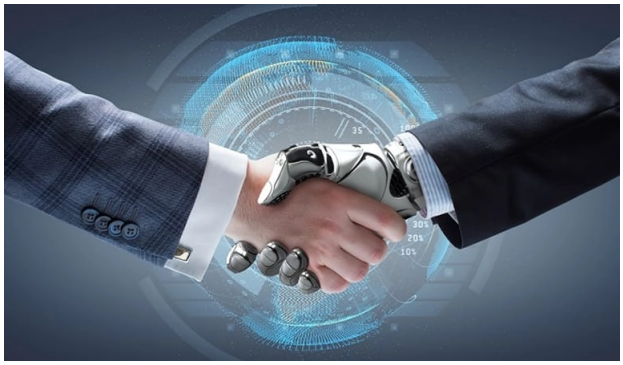
include using online virtual health assistants and chatbots to help patients and healthcare customers

find medical information, schedule appointments, understand the billing process and complete other

administrative processes. An array of AI technologies is also being used to predict, fight and

understand pandemics such as COVID-19.

10.2 AI in business.



Machine learning algorithms are being integrated into analytics and customer relationship

management (CRM) platforms to uncover information on how to better serve customers.

Chatbots have been incorporated into websites to provide immediate service to customers.

Automation of job positions has also become a talking point among academics and IT analysts.

10.3 AI in education.

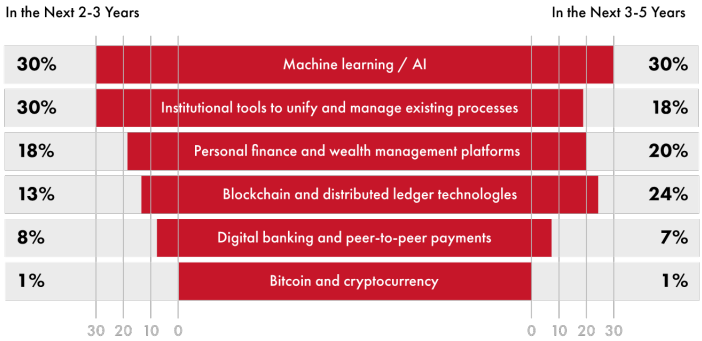
AI can automate grading, giving educators more time. It can assess students and adapt to

their needs, helping them work at their own pace. AI tutors can provide additional support to

students, ensuring they stay on track. And it could change where and how students learn,

perhaps even replacing some teachers.

10.4 AI in finance.



AI in personal finance applications, such as Intuit Mint or TurboTax, is disrupting financial institutions.

Applications such as these collect personal data and provide financial advice. Other programs, such as IBM Watson,

have been applied to the process of buying a home. Today, artificial intelligence software performs much

of the trading on Wall Street.AI in law. The discovery process -- sifting through documents -- in law is often

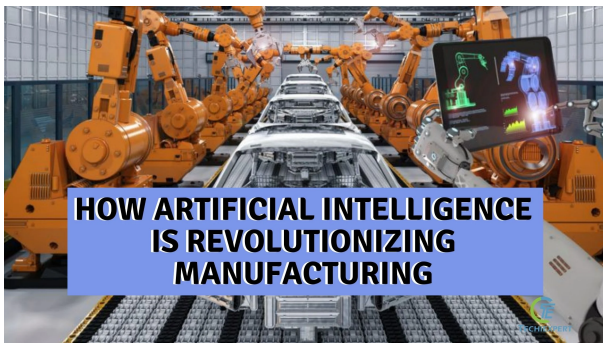
overwhelming for humans. Using AI to help automate the legal industry's labor-intensive processes is saving

time and improving client service. Law firms are using machine learning to describe data and predict outcomes,

computer vision to classify and extract information from documents and natural language

processing to interpret requests for information.

10.5 AI in manufacturing.



Manufacturing has been at the forefront of incorporating robots into the workflow.

For example, the industrial robots that were at one time programmed to perform single tasks and

separated from human workers, increasingly function as cobots: Smaller, multitasking robots that

collaborate with humans and take on responsibility for more parts of the job in warehouses,

factory floors and other workspaces.

10.6 AI in banking.

Banks are successfully employing chatbots to make their customers aware of services

and offerings and to handle transactions that don't require human intervention.

AI virtual assistants are being used to improve and cut the costs of compliance with banking regulations.

Banking organizations are also using AI to improve their decision-making for loans,

and to set credit limits and identify investment opportunities.

11 Different Tools of A.I.

11.1 Pytorch:

11.1.1 What Is PyTorch, and How Does It Work?

PyTorch is an optimized Deep Learning tensor library based on Python and Torch and is mainly used for applications using

GPUs and CPUs. PyTorch is favored over other Deep Learning frameworks like TensorFlow and Keras since it uses dynamic computation graphs and is completely Pythonic. It allows scientists, developers, and neural network debuggers to run and test portions of the code in real-time. Thus, users don't have to wait for the entire code to be implemented to check if a part of the code works or not.

11.1.2 The two main features of PyTorch are:

- Tensor Computation (similar to NumPy) with strong GPU (Graphical Processing Unit) acceleration support Automatic
- Differentiation for creating and training deep neural networks.

11.1.3 Basics of PyTorch

The basic PyTorch operations are pretty similar to Numpy. Let's understand the basics first.

- Introduction to Tensors

  In machine learning, when we represent data, we need to do that numerically. A tensor is simply a container that can hold data in multiple dimensions. In mathematical terms, however, a tensor is a fundamental unit of data that can be used as the foundation for advanced mathematical operations. It can be a number, vector, matrix, or multi-dimensional array like Numpy arrays. Tensors can also be handled by the CPU or GPU to make operations faster. There are various types of tensors like Float Tensor, Double Tensor, Half Tensor, Int Tensor, and Long Tensor, but PyTorch uses the 32-bit Float Tensor as the default type.

- Mathematical Operations

  The codes to perform mathematical operations are the same in PyTorch as in Numpy. Users need to initialize two tensors and then perform operations like addition, subtraction, multiplication, and division on them.

- Matrix Initialization and Matrix Operations

To initialize a matrix with random numbers in PyTorch, use the function randn() that gives a tensor filled with random numbers from a standard normal distribution. Setting the random seed at the beginning will generate the same numbers every time you run this code. Basic matrix operations and transpose operation in PyTorch are also similar to NumPy.

11.1.4 Common PyTorch Modules:

In PyTorch, modules are used to represent neural networks.

- Autograd

  The autograd module is PyTorch's automatic differentiation engine that helps to compute the gradients in the forward pass in quick time. Autograd generates a directed acyclic graph where the leaves are the input tensors while the roots are the output tensors.

- Optim

  The Optim module is a package with pre-written algorithms for optimizers that can be used to build neural networks.

- nn

  The nn module includes various classes that help to build neural network models. All modules in PyTorch subclass the nn module

12 TensorFlow:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence.

It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production.

The initial version was released under the Apache License 2.0 in 2015.[1][9] Google released the updated version of TensorFlow,

named TensorFlow 2.0, in September 2019.TensorFlow can be used in a wide variety of programming languages,

most notably Python, as well as Javascript, C++, and Java.

This flexibility lends itself to a range of applications in many different sectors.

12.1 Tensor processing unit (TPU) Tensor processing unit:

In May 2016, Google announced its Tensor processing unit (TPU), an application-specific integrated circuit (ASIC, a hardware chip) built specifically for machine learning and tailored for TensorFlow. A TPU is a programmable AI accelerator designed to provide high throughput of low-precision arithmetic (e.g., 8-bit), and oriented toward using or running models rather than training them. Google announced they had been running TPUs inside their data centers for more than a year, and had found them to deliver an order of magnitude better-optimized performance per watt for machine learning.[22]

12.2 Features :

12.2.1 AutoDifferentiation:

AutoDifferentiation is the process of automatically calculating the gradient vector of a model with respect to each of its parameters. With this feature, TensorFlow can automatically compute the gradients for the parameters in a model, which is useful to algorithms such as backpropagation which require gradients to optimize performance.[32] To do so, the framework must keep

track of the order of operations done to the input Tensors in a model, and then compute the gradients with respect to the appropriate parameters.[32]

12.2.2 Eager execution:

TensorFlow includes an "eager execution" mode, which means that operations are evaluated immediately as opposed to being added to a computational graph which is executed later.[33] Code executed eagerly can be examined step-by step-through a debugger, since data is augmented at each line of code rather than later in a computational graph.[33] This execution paradigm is considered to be easier to debug because of its step by step transparency.[33] Distribute

In both eager and graph executions, TensorFlow provides an API for distributing computation across multiple devices with various distribution strategies.[34] This distributed computing can often speed up the execution of training and evaluating of TensorFlow models and is a common practice in the field of AI.[34][35]

12.2.3 Losses:

To train and assess models, TensorFlow provides a set of loss functions (also known as cost functions).[36] Some popular examples include mean squared error (MSE) and binary cross entropy (BCE).[36] These loss functions compute the "error" or "difference" between a model's output and the expected output (more broadly, the difference between two tensors). For different datasets and models, different losses are used to prioritize certain aspects of performance.

12.2.4 Metrics

In order to assess the performance of machine learning models, TensorFlow gives API access to commonly used metrics. Examples include various accuracy metrics (binary, categorical, sparse categorical) along with other metrics such as Precision, Recall, and Intersection-over-Union (IoU).[37]

12.2.5 TF.nn

TensorFlow.nn is a module for executing primitive neural network operations on models.[38] Some of these operations include variations of convolutions (1/2/3D, Atrous, depthwise), activation functions (Softmax, RELU, GELU, Sigmoid, etc.) and their variations, and other Tensor operations (max-pooling, bias-add, etc.).[38]

12.2.6 Optimizers

TensorFlow offers a set of optimizers for training neural networks, including ADAM, ADAGRAD, and Stochastic Gradient Descent (SGD).[39] When training a model, different optimizers offer different modes of parameter tuning, often affecting a model's convergence and performance.[40]

13 Scikit-learn:

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.

13.1 Features:

13.1.1 Datasets

Scikit-learn comes with several inbuilt datasets such as the iris dataset, house prices dataset, diabetes dataset, etc.

The main functions of these datasets are that they are easy to understand and you can directly implement ML models on them.

These datasets are good for beginners.

13.1.2 Data Splitting

Sklearn provided the functionality to split the dataset for training and testing.

Splitting the dataset is essential for an unbiased evaluation of prediction performance.

We can define what proportion of our data to be included in train and test datasets.

13.1.3 Linear Regression

This supervised ML model is used when the output variable

is continuous and it follows linear relation with dependent variables.

It can be used to forecast sales in the coming months by analyzing the sales data for previous months.

13.1.4 Decision Trees

A Decision Tree is a powerful tool that can be used for both classification and regression problems.

It uses a tree-like model to make decisions and predict the output. It consists of roots and nodes.

Roots represent the decision to split and nodes represent an output variable value. A decision tree is an important concept.

14 Supervised Learning techniques:

14.1 KNN(K-Nearest Neighbours):

14.1.1 Points to Note:

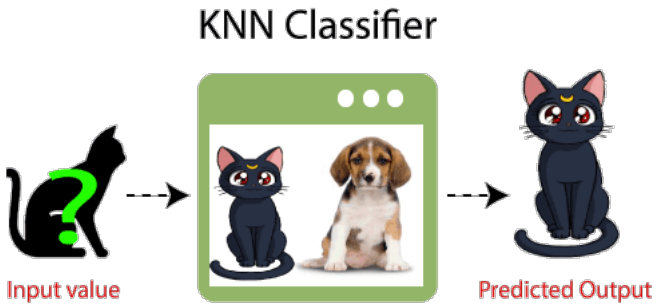
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on **Supervised Learning technique**.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the **Classification problems**.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a lazy **learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

14.1.2 Example:

Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog.

So for this identification, we can use the KNN algorithm, as it works on a similarity measure.

Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

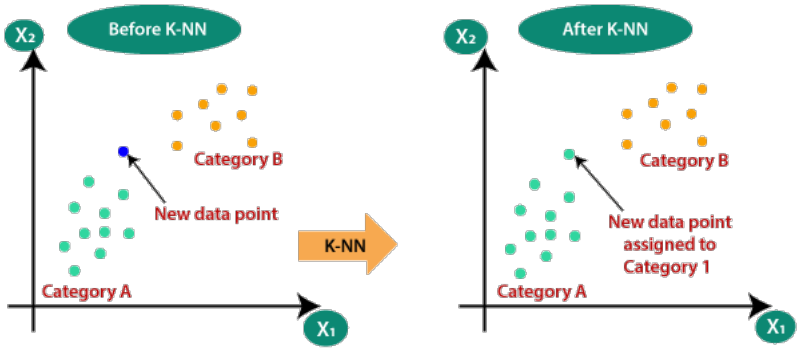


14.1.3 Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories.

To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.

Consider the below diagram:



14.1.4 How does K-NN work?

The K-NN working can be explained on the basis of the below 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.

14.1.5 KNN Code:

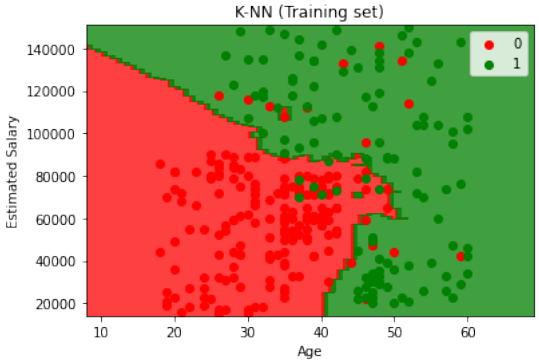
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split
(X, y, test_size = 0.25, random_state = 0)
print(y_train)
print(X_train)
print(y_test)
print(y_test)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the K-NN model on the Training set

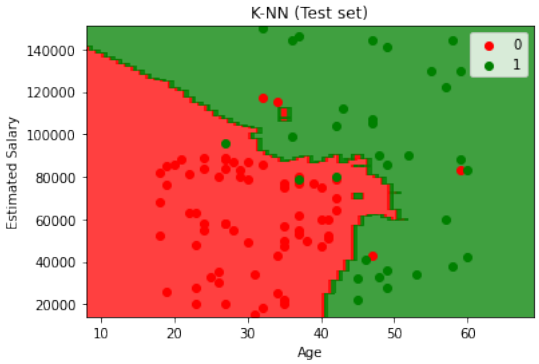
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5,
metric = 'minkowski', p = 2) classifier.fit(X_train, y_train)
print(classifier.predict(sc.transform([[30,87000]])))
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

14.1.6 Output:

KNN Tranning set results:



KNN Test set results:

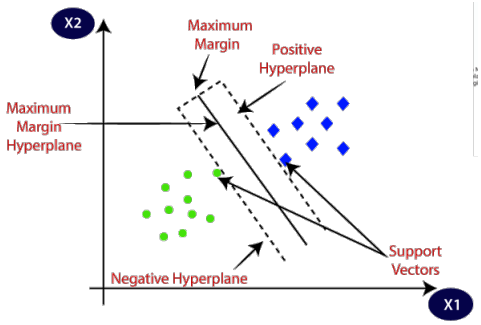


14.2 SVM (Support Vector Machine):

14.2.1 Points to Note:

- Support Vector Machine or SVM is one of the most **popular Supervised Learning algorithms**, which is used for Classification as well as Regression problems.
- However, primarily, it is used for **Classification** problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or **decision boundary** that can segregate n-dimensional space into classes
- so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a **HYPERPLANE.**.
- 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:



14.2.2 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,

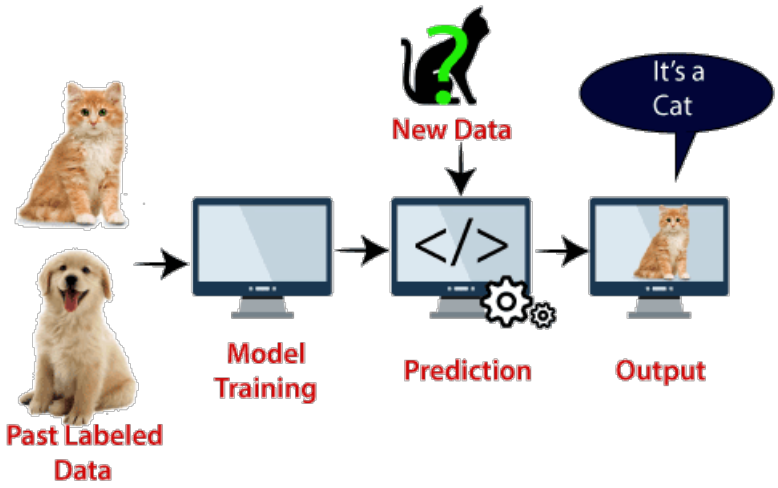
so such a model can be created by using the SVM algorithm.

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:



14.2.3 Hyperplane and Support Vectors in the SVM algorithm:

14.2.4 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.

14.2.5 Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the

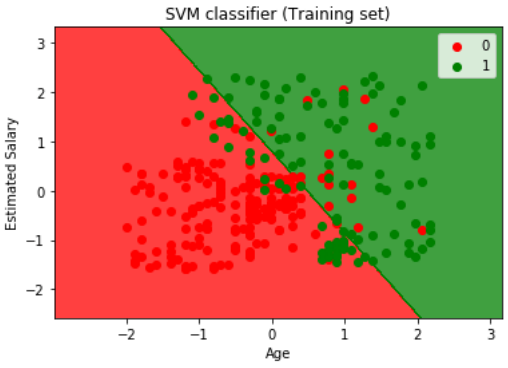
position of the hyperplane are termed as Support Vector.

Since these vectors support the hyperplane, hence called a Support vector.

14.2.6 SVM Code:

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train x1,
x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min()-1,
stop = x_set[:, 0].max() + 1, step =0.01),
nm.arange(start = x_set[:, 1].min() - 1,
stop = x_set[:, 1].max() + 1,step0.01))mtp.contourf(x1, x2,
classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape), alpha = 0.75,
cmap = ListedColormap(('red', 'green'))) mtp.xlim(x1.min(),
x1.max()) mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)): mtp.scatter(x_set[y_set == j, 0],
x_set[y_set == j, 1],
c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('SVM classifier (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

14.2.7 Output:



14.3 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: Confusion Matrix in Machine Learning

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

14.3.1 Making the Confusion Matrix:

```
from sklearn.metrics import confusion_matrix,
accuracy_score cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

14.3.2 Output:

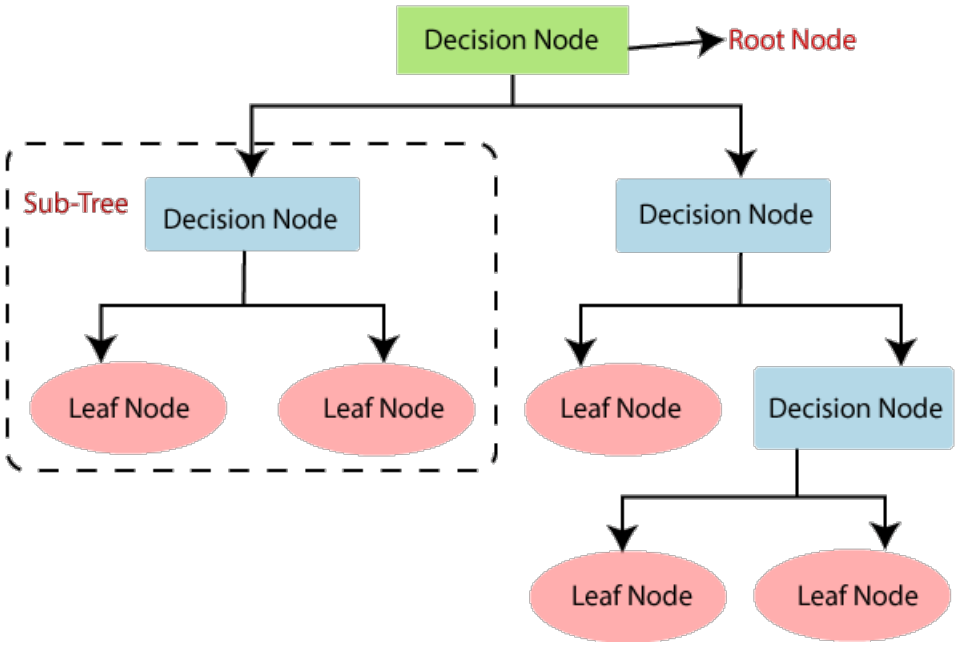
[[64 4]

[3 29]]

0.93

14.4 Decision Tree Classification:

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems,but mostly it is preferred for solving Classification problems.
- It is a **tree-structured classifier**, where internal nodes represent the features of a dataset, **branches** represent the decision rules and each **leaf node** represents the outcome. In a **Decision tree**, there are two nodes, which are the **Decision Node** and **Leaf Node**.
- Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed **on the basis of features** of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.



14.4.1 Decision Tree Terminologies

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

Branch/Sub Tree: A tree formed by splitting the tree. Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

14.4.2 Example:

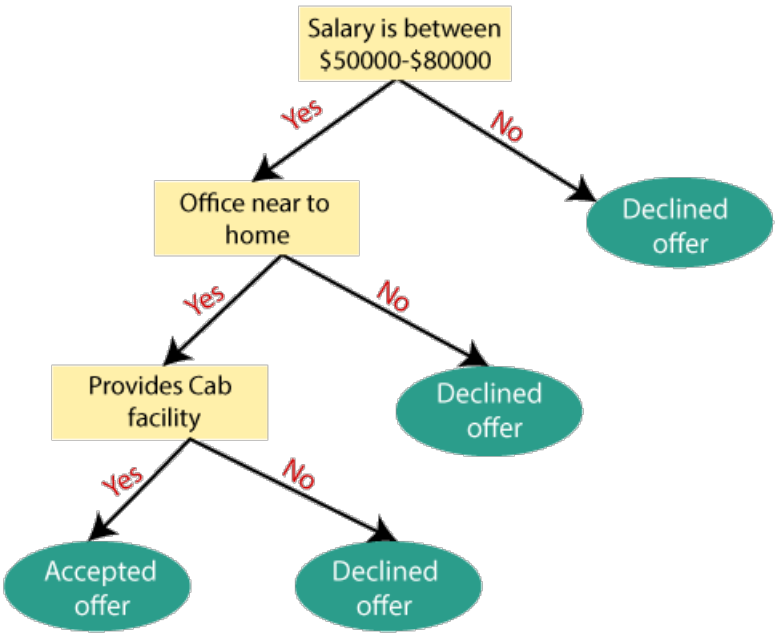
Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not.

So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM).

The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels.

The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer).

Consider the below diagram:

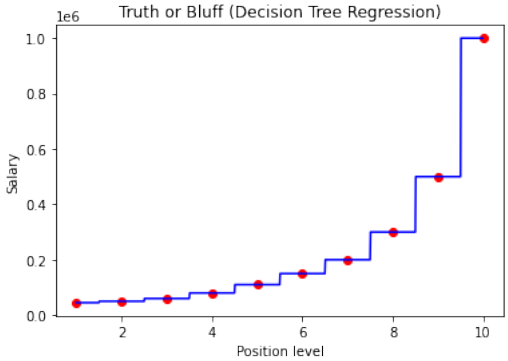


14.5 Decision Tree Regression Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)
X_grid = np.arange(min(X), max(X), 0.01)
```

```
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid),color'blue')
plt.title('Truth or Bluff (Decision TreeRegression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

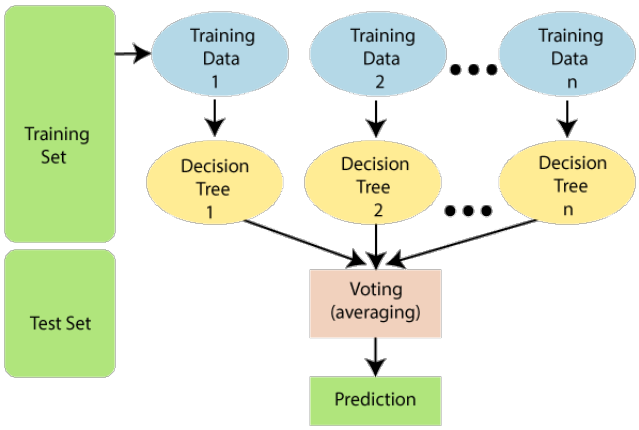
14.5.1 Output:



14.6 Random Forest Regression:

- **Random Forest** is a popular machine learning algorithm that belongs to the **supervised learning** technique.
- It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning,
- which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
- As the name suggests, "**Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.**" Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting

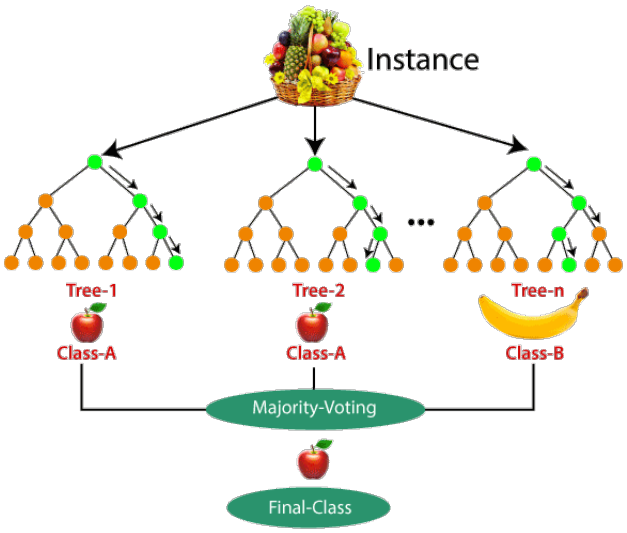


14.6.1 Example:

Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier.

The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs,

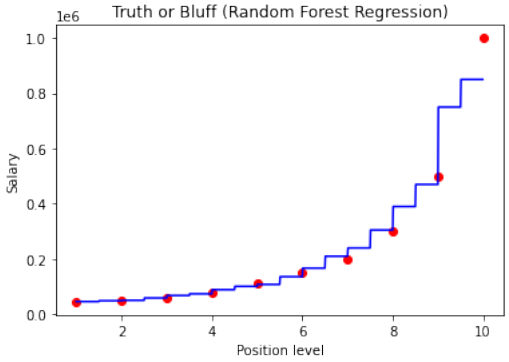
then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



14.6.2 Random Forest Regression Code:

```
import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    dataset = pd.read_csv('Position_Salaries.csv')
    X = dataset.iloc[:, 1:-1].values
    y = dataset.iloc[:, -1].values
    from sklearn.ensemble import RandomForestRegressor
    regressor = RandomForestRegressor(n_estimators = 10, random_state =
0) regressor.fit(X, y)
    X_grid = np.arange(min(X), max(X), 0.01)
    X_grid = X_grid.reshape((len(X_grid), 1))
    plt.scatter(X, y, color = 'red')
    plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
    plt.title('Truth or Bluff (Random Forest Regression)')
    plt.xlabel('Position level') plt.ylabel('Salary')
    plt.show()
```

14.6.3 Output:



15 Clustering in Machine Learning:

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset.

It can be defined as "**A way of grouping the data points into different clusters, consisting of similar data points.**

The objects with the possible similarities remain in a group that has less or no similarities with another group."

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an **unsupervised learning method,** hence no supervision is provided to the algorithm, and it deals with the **unlabeled dataset**.

15.1 Use of clustering:

The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

- Market Segmentation
- Statistical data analysis
- Social network analysis
- Image segmentation
- Anomaly detection, etc.

Apart from these general usages, it is used by the **Amazon** in its recommendation system to provide the recommendations as per the past search of products.

Netflix also uses this technique to recommend the movies and web-series to its users as per the watch history.

The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.

16 Clustering Algorithms:

The Clustering algorithms can be divided based on their models that are explained above.

There are different types of clustering algorithms published, but only a few are commonly used.

The clustering algorithm is based on the kind of data that we are using. Such as, some algorithms need to guess the

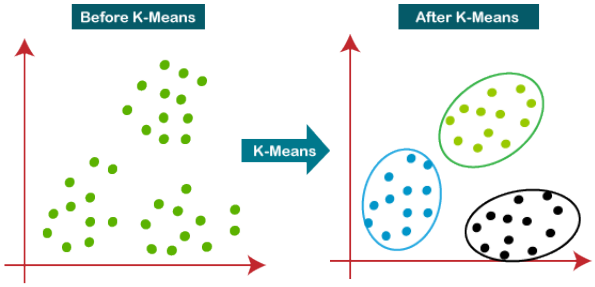
number of clusters in the given dataset, whereas some are required to find the minimum distance between the observation of the dataset.

16.1 K-Means Clustering Algorithm:

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.

K-Means Clustering is an Unsupervised Learning algorithm , which groups the unlabeled dataset into different clusters.

Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.



16.2 K-means++ algorithm: (K++)

In data mining, k-means++ is an algorithm for choosing the initial values (or "seeds") for the k-means clustering algorithm.

It was proposed in 2007 by David Arthur and Sergei Vassilvitskii, as an approximation algorithm for the NP-hard k-means problem—

a way of avoiding the sometimes poor clusterings found by the standard k-means algorithm. It is similar to the first of three seeding methods proposed,

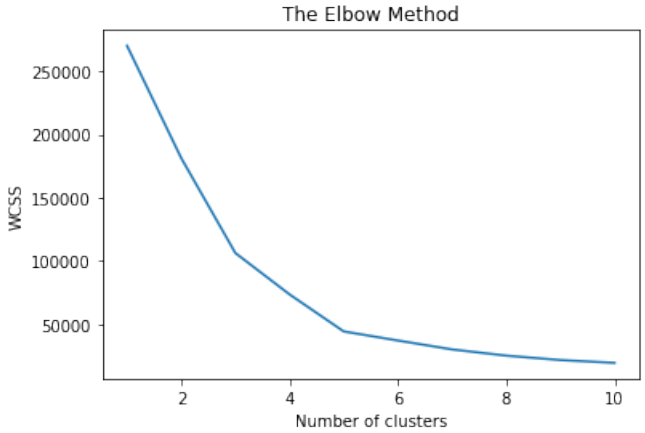
in independent work, in 2006[3] by Rafail Ostrovsky, Yuval Rabani, Leonard Schulman and Chaitanya Swamy. (The distribution of the first seed is different.)

17 K-Means Clustering code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Mall_Customers.csv')
```

```
X = dataset.iloc[:, [3, 4]].values
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()
```

17.1 Elbow Method(Graph):



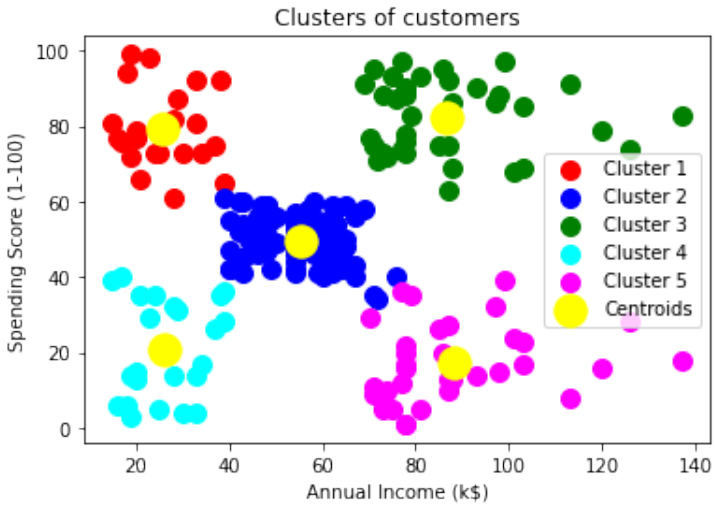

```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

17.2 Visualising the clusters :

```
plt.scatter(X[y_kmeans == 0, 0],
X[y_kmeans == 0, 1],
s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1],
s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1],
```

```
s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1],
s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1],
s = 100, c = 'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

17.3 Output:



18 References:

1. Artificial intelligence/Ivivity.com
2. Arificial Intelligence/Wikipedia.com
3. Artificial Intelligence/Tutorialpoint.com
4. JAVAPOINT.com