Mahima Panchal

NUMPY

```
In [1]: import numpy as np
```

Q.1 Convert a 1D array to a 2D array with 2 rows

```
In [2]: one_dimensional_array = np.array([1, 2, 3, 4, 5, 6])
```

```
In [3]: two_dimensional_array = one_dimensional_array.reshape(2, -1)
```

```
In [4]: print(two_dimensional_array)
```

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

Q.2 Get the common items between a and b

```
In [5]: a = np.array([1, 2, 3, 2, 3, 4, 3, 4, 5, 6])
```

```
In [6]: b = np.array([7, 2, 10, 2, 7, 4, 9, 4, 9, 8])
```

```
In [7]: common_items = np.intersect1d(a, b)
```

```
In [8]: common_items
```

```
Out[8]: array([2, 4])
```

Q.3 Get all items between 5 and 10 from a.

```
In [9]: a = np.array([2, 6, 1, 9, 10, 3, 27])
```

```
In [10]: items = (a >= 5) & (a <= 10)
```

```
In [11]: result = a[items]
```

```
In [12]: result
```

```
Out[12]: array([ 6,  9, 10])
```

Q.4 Limit the number of items printed in python NumPy array a to a maximum of 6 elements.

```
In [13]: a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])
```

```
In [14]: limit = a[:6]
```

```
In [15]: limit
```

```
Out[15]: array([0, 1, 2, 3, 4, 5])
```

```
In [ ]:
```

PANDAS

1. Compute the minimum, 25th percentile, median, 75th, and maximum of ser.

```
In [16]: import pandas as pd
```

```
In [17]: ser = pd.Series([34,56,80,39,67,89,56,88,99,100])
```

```
In [18]: summary = ser.describe(percentiles=[.25, .5, .75])
```

```
In [19]: minimum = summary['min']
percentile_25 = summary['25%']
median = summary['50%']
percentile_75 = summary['75%']
maximum = summary['max']
```

```
In [20]: print("Minimum:", minimum)
print("25th Percentile:", percentile_25)
print("Median:", median)
print("75th Percentile:", percentile_75)
print("Maximum:", maximum)
```

```
Minimum: 34.0
25th Percentile: 56.0
Median: 73.5
75th Percentile: 88.75
Maximum: 100.0
```

2. Creating A Pandas Data Frame and Using Sample Data Sets

```
In [21]: data = [
    ['Alice', 25, 'New York'],
    ['Bob', 30, 'San Francisco'],
    ['Charlie', 35, 'Los Angeles'],
    ['David', 28, 'Chicago']]
```

```
In [22]: columns = ['Name', 'Age', 'City']
```

```
In [23]: df = pd.DataFrame(data, columns=columns)
```

```
In [24]: df
```

Out[24]:

	Name	Age	City
0	Alice	25	New York
1	Bob	30	San Francisco
2	Charlie	35	Los Angeles
3	David	28	Chicago

```
In [25]: data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, 30, 35, 28],
    'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago']
}
```

```
In [26]: df = pd.DataFrame(data)
```

```
In [27]: df
```

Out[27]:

	Name	Age	City
0	Alice	25	New York
1	Bob	30	San Francisco
2	Charlie	35	Los Angeles
3	David	28	Chicago

3. Using NumPy, create a Pandas Data Frame with five rows and three columns.

```
In [28]: data = np.random.rand(5, 3)
```

```
In [29]: df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3'])
```

```
In [30]: df
```

Out[30]:

	Column1	Column2	Column3
0	0.192295	0.527028	0.075631
1	0.271191	0.294547	0.647943
2	0.094032	0.860319	0.155306
3	0.020488	0.342489	0.472517
4	0.405668	0.300916	0.009525

4. For a Pandas Data Frame created from a NumPy array, what is the default behavior for the labels for the columns? For the rows?

```
In [31]: #Columns: The default column labels (column names) are integer numbers starting from 0 and incrementing consecutively.
#For example, the first column will have the label '0', the second column '1', and so on.

#Rows: The default row labels (row index) are also integer numbers starting from 0 and incrementing consecutively.
#The rows are labeled with integer indices unless you specify custom row labels (index) when creating the DataFrame.
```

```
In [32]: data = np.array([[1, 2], [3, 4], [5, 6]])
```

```
In [33]: df = pd.DataFrame(data)
```

```
In [34]: df
```

Out[34]:

	0	1
0	1	2
1	3	4
2	5	6

5. take csv file contains at least 10,000 rows and 12 columns which numerical and text values according to that continue following steps.

```
In [35]: DF = pd.read_csv('Downloads/IMDB_10000.csv')
```

```
In [36]: DF
```

Out[36]:

	title	year	certificate	runtime	genre	desc	rating	votes
0	Freddy	2022	UA 16+	124 min	Drama, Mystery, Thriller	The lines between love and obsession blur in t...	7.9	16,441
1	An Action Hero	2022	U	130 min	Action	Youth Icon. Superstar. Action Hero. At the age...	8.1	15,690
2	Kantara	2022	UA	148 min	Action, Adventure, Drama	It involves culture of Kambala and Bhootha Kol...	8.7	78,358
3	Khakee: The Bihar Chapter	2022–	UA 13+	45 min	Action, Crime, Drama	As a righteous cop pursues a merciless crimina...	8.3	4,464
4	Drishyam 2	2022	UA	140 min	Crime, Drama, Mystery	A gripping tale of an investigation and a fami...	8.6	18,743
...	...	...	...	...	...	...	...	...
9995	Kisan Aur Bhagwan	1974	NaN	142 min	Action, Comedy, Drama	NaN	NaN	NaN
9996	Aadmi Sadak Ka	1977	NaN	138 min	Drama, Family	NaN	NaN	NaN
9997	Nadodi Mannan	1958	NaN	220 min	Action, Adventure, Comedy	NaN	NaN	NaN
9998	Njan Marykutty	2018	U	126 min	Drama	NaN	NaN	NaN
9999	Paap Ko Jalaa Kar Raakh Kar Doonga	1988	UA	153 min	Crime	NaN	NaN	NaN

10000 rows × 8 columns

6. Write the code to show the number of rows and columns in data frame.

```
In [37]: num_rows, num_columns = DF.shape
```

```
In [38]: print(num_rows)
print(num_columns)
```

```
10000
8
```

7. How might you show the first few rows of data frame?

```
In [39]: DF.head()
```

Out[39]:

	title	year	certificate	runtime	genre	desc	rating	votes
0	Freddy	2022	UA 16+	124 min	Drama, Mystery, Thriller	The lines between love and obsession blur in t...	7.9	16,441
1	An Action Hero	2022	U	130 min	Action	Youth Icon. Superstar. Action Hero. At the age...	8.1	15,690
2	Kantara	2022	UA	148 min	Action, Adventure, Drama	It involves culture of Kambala and Bhootha Kol...	8.7	78,358
3	Khakee: The Bihar Chapter	2022–	UA 13+	45 min	Action, Crime, Drama	As a righteous cop pursues a merciless crimina...	8.3	4,464
4	Drishyam 2	2022	UA	140 min	Crime, Drama, Mystery	A gripping tale of an investigation and a fami...	8.6	18,743

```
In [40]: genre_column = DF['genre']
```

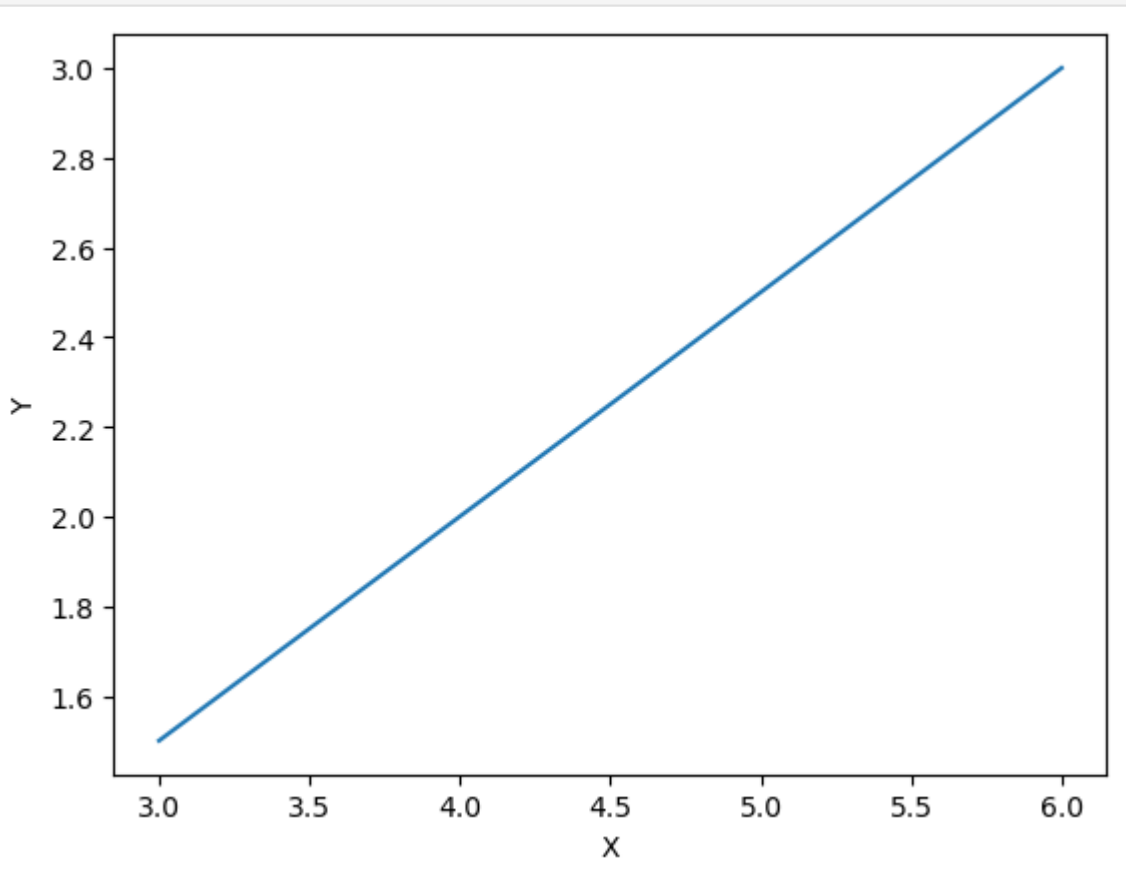
```
In [41]: print(type(genre_column))
```

```
<class 'pandas.core.series.Series'>
```

9. Create a line plot using the x and y values provided below. Label the y-axis as “Y” and label the x-axis as “X”.

```
In [42]: import matplotlib.pyplot as plt
```

```
In [43]: x = [3, 4, 5, 6]
y = [1.5, 2, 2.5, 3]
plt.plot(x,y)
plt.ylabel('Y')
plt.xlabel('X')
plt.show()
```

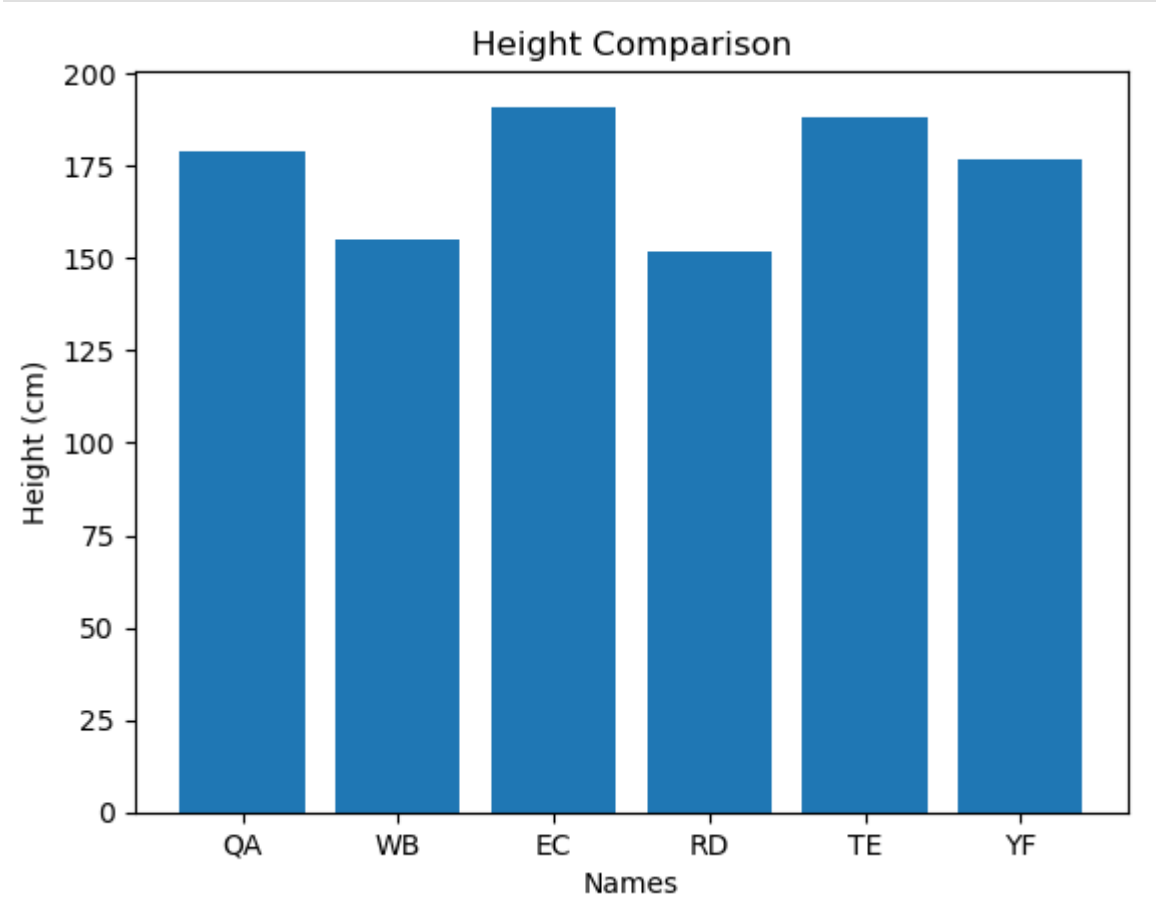


10. Create an array of numbers between 0 and 6 with increments of 0.3 and name its “x”.Then on the same plot, plot x, x^2, x^3, and x^4. For consistency, use the following style lines respectively, “ro”, “bs”, “g”, and “:”. Lastly, make sure that the x-axis covers 0 to 6, while the y-axis spans from 0 to 125. Do not worry if you are not familiar with the style lines —you will recognize them as soon as you see the plot.

11. Heights and initials of a group of individuals are provided below. Create a bar plot titled “Height Comparison” to compare the heights among this group. height = [179, 155, 191, 152, 188, 177] names = ['QA', 'WB', 'EC', 'RD', 'TE', 'YF']

```
In [44]: height = [179, 155, 191, 152, 188, 177]
names = ['QA', 'WB', 'EC', 'RD', 'TE', 'YF']
```

```
In [47]: plt.bar(names, height)
plt.title("Height Comparison")
plt.xlabel("Names")
plt.ylabel("Height (cm)")
plt.show()
```



12. Plot a histogram of x, where x consists of 100,000 randomly-selected points with a normal distribution (hint: you can use numpy.random.randn() to generate the random points). The histogram should have 10 bins. Look at how the histogram changes when we try 20 and 50 bins.

```
In [48]: x = np.random.randn(100000)
```

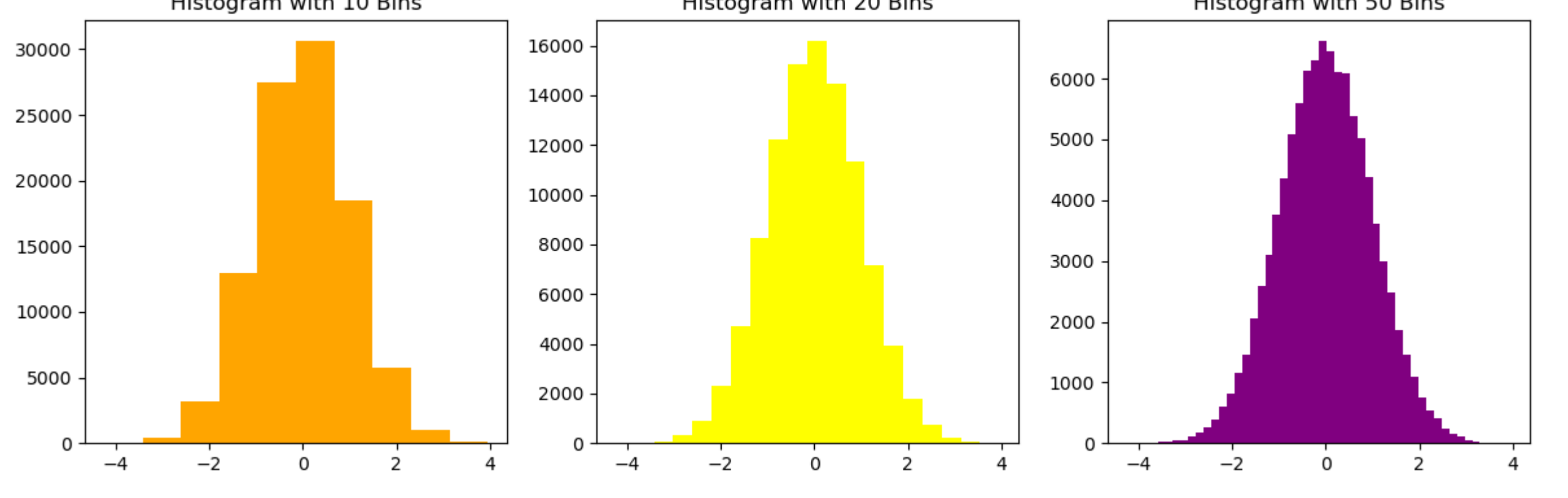
```
In [56]: plt.figure(figsize=(12, 4))
plt.subplot(131)
plt.hist(x, bins=10, color='orange')
plt.title("Histogram with 10 Bins")

plt.subplot(132)
plt.hist(x, bins=20, color='yellow')
plt.title("Histogram with 20 Bins")

plt.subplot(133)
plt.hist(x, bins=50, color='purple')
plt.title("Histogram with 50 Bins")

plt.tight_layout()

plt.show()
```



13. What are the features of TensorFlow?

```
In [ ]: TensorFlow is an open-source machine learning framework developed by Google that is widely used for various machine learning
and deep learning tasks. It offers a wide range of features and capabilities that make it a popular choice among researchers
and practitioners.
Here are some of the key features of TensorFlow:

Flexibility: TensorFlow provides a flexible platform for developing and deploying machine learning models across a variety
of platforms and devices, including CPUs, GPUs, and TPUs (Tensor Processing Units).

Deep Learning Support: TensorFlow has built-in support for deep learning, including deep neural networks and
various neural network layers. It also supports popular deep learning frameworks like Keras, which can be used as
a high-level API for building neural networks.

Highly Scalable: TensorFlow is designed to scale easily, making it suitable for both small-scale and
large-scale machine learning projects. It can be used on a single machine or in distributed computing environments.

Support for Multiple Languages: While TensorFlow is primarily used with Python, it also provides support for
other programming languages, including C++, Java, and Go, through its TensorFlow Serving and TensorFlow Lite libraries.

Community and Ecosystem: TensorFlow has a large and active community of developers, researchers, and users.
This has led to the creation of numerous open-source projects, libraries, and pre-trained models that
extend TensorFlow's capabilities.

Mobile and Embedded Devices: TensorFlow Lite is designed for mobile and embedded devices, allowing you to run machine
learning models on smartphones, IoT devices, and more.
```

14. List a few limitations of TensorFlow.

```
In [ ]: While TensorFlow is a powerful and widely-used machine learning framework, it does have some limitations and challenges:

Steep Learning Curve: TensorFlow can be challenging for beginners due to its complexity.
The framework's extensive capabilities and the need to understand concepts like computation graphs and sessions can make
it less approachable for newcomers to machine learning.

Verbose Code: Writing TensorFlow code can be verbose, especially when compared to high-level libraries like Keras.
This verbosity can make the code harder to read and maintain.

Complexity: TensorFlow's flexibility and fine-grained control can be a double-edged sword.
While it offers immense control over model architecture and training, it can lead to more complex code for simple tasks.

Performance Optimization: Achieving optimal performance in TensorFlow often requires careful optimization and tuning,
especially when dealing with large-scale models and datasets. This can be time-consuming and may require a deep
understanding of hardware and software configurations.

Limited Mobile Support: While TensorFlow Lite allows deployment on mobile and embedded devices,
mobile support can still be challenging, and the framework may not be as lightweight as some alternatives for these platforms.

Large Model Sizes: Deep learning models created with TensorFlow can have large file sizes,
which may be a concern when deploying models on resource-constrained devices or over the internet.
```

15. What do you know about supervised and unsupervised machine learning?

```
In [ ]: Supervised learning:
Supervised learning, as the name indicates, has the presence of a supervisor as a teacher.
Basically supervised learning is when we teach or train the machine using data that is well-labelled.
Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of
examples(data) so that the supervised learning algorithm analyses the training data(set of training examples)
and produces a correct outcome from labeled data.

Unsupervised learning:
Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing
the algorithm to act on that information without guidance.
The task of the machine is to group unsorted information according to similarities, patterns,
and differences without any prior training of data.
Unlike supervised learning, no teacher is provided that means no training will be given to the machine.
Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.
```