

Introductory
Data Analysis
&
Data Science
with
Python

Kazi Sakib Hasan
Computer Science Program, Brac University

Author's Manuscript Edition 2024

Brief Contents

SL	Contents	Page
00	Preface	01
01	Introduction to Statistics <ul style="list-style-type: none"> ● 1.1 Statistical Data Types 02 ● 1.2 Basic Terminologies 05 ● 1.3 Descriptive Statistics 07 <ul style="list-style-type: none"> - Univariate and multivariate dataset 07 - Measures of Central Tendency 08 - Measures of Dispersion 09 ● 1.4 Descriptive Statistics with Python 11 <ul style="list-style-type: none"> - Data Analysis 12 - Data Representation with Pandas DataFrame 12 - Necessary DataFrame Features 15 - The NumPy Library 16 - Data Visualization 18 <ul style="list-style-type: none"> - Line Plot 19 - Scatter Plot 26 - Bar Chart 29 ● 1.5 Probability 32 <ul style="list-style-type: none"> - Probability Terminologies 32 - Binomial Distribution of Probability 33 - Geometric Distribution of Probability 35 	
02	Statistical Analysis <ul style="list-style-type: none"> ● 2.1 Student's T-Test 37 ● 2.2 ANOVA 39 <ul style="list-style-type: none"> - Tukey HSD (Honestly Significant Difference) Test 40 ● 2.3 Pearson Correlation 42 <ul style="list-style-type: none"> - Strength of Linear Relationship 43 Project 44	
03	Survival Analysis <ul style="list-style-type: none"> ● 3.1 Estimating Survival Function 46 <ul style="list-style-type: none"> - Kaplan-Meier Estimator 46 	

	<ul style="list-style-type: none"> ● 3.2 Hazard Function Analysis <ul style="list-style-type: none"> - Nelson-Aalen Estimator 	48 49
04	Bivariate Predictive Modelling <ul style="list-style-type: none"> ● 4.1 Exponential Growth-Decay Model ● 4.2 Simple Linear Regression ● 4.3 Polynomial Regression ● 4.4 Model Evaluation <ul style="list-style-type: none"> - Mean Squared Error Project	52 52 55 59 62 62 64
05	Multivariate Predictive Modelling <ul style="list-style-type: none"> ● 5.1 Supervised Machine Learning Models <ul style="list-style-type: none"> - ML Terminologies - Model Structure - Project Walkthrough Project	64 66 66 68 68 77
	Appendix <ul style="list-style-type: none"> ● Data Portfolio 	78

Preface

Some academic degrees that had lost much of their importance recently are making a comeback due to the rise of artificial intelligence. Among them, two special branches are Statistics and Neuroscience. Both these branches are related to Generative AI. Because Machine Learning Algorithms are written based on statistical modeling, then large language models are modified to create Generative AI Models (for example: ChatGPT, Microsoft Copilot, ChatSonic etc.). On the other hand, neuroscience research is needed to scrutinize how these models influence human decision-making. Anyway, in this course we will only discuss Statistics and Machine Learning.

Things I hope that you will learn at the end of this course:

- Students have little idea of where to apply the knowledge of traditional school-college or university mathematics or statistics in real life. That's why even if he passes with good grades, he can't use his knowledge at work. Therefore, this course will show how to solve math-related problems using programming instead of emphasizing what formula to use.
- Some of the topics of Machine Learning which are useful for doing research (thesis, masters), will be discussed in this course.
- Basics of machine learning will be discussed. Students will learn to build and deploy models on their own.
- Above all, students will get a basic understanding of Data Science and Data Analysis.

Note that this is a self-learning reflection text. In this text, I only wrote about how I learned the concepts, and I am not yet a professional in this field. I conduct research independently and sometimes with the university faculties regarding machine learning, AI, and statistics.

1. Introduction to Statistics

In simple words, statistics is the analysis of data to find certain information from it. For example, Let's say we have mathematics exam marks of 30 students. The exam marks of these 30 students are a dataset. In a small sense, this dataset is no information. But when the mean of all the marks in this dataset is calculated, then this average mark is called the data obtained from that dataset. Now, data can be of different types.

1.1 Statistical Data Types

Qualitative Data: Its other name is Categorical Data. The type of data that is not made up of numbers is categorical data. For example, the name of a group of people, furniture types, car models, aircraft models, food types etc. But the number of people, the number of furniture, the number of cars, the number of airplanes, the number of food items, etc. are not included in categorical data because they are data made up of numbers.

Qualitative data is divided into two parts.

- **Ordinal Data:** Some qualitative data can be expressed numerically and arranged sequentially. That is, this type of data has a type of "Order". For example, the rating of an application. Consumers are usually given four to five options during rating. Let's say the options are "poor", "fair", "good", "very good". These data could be expressed numerically if desired. That is, "poor" was rated 0, "fair" was rated 1, "good" was rated 2, and "very good" was rated 3. Then, the entire rating system would be numerical and then the quality of the application could be verified more easily. This type of data is called Ordinal Data. Education level (Primary, High, Undergraduate, Post-Graduate) these are also included in Ordinal Data.
- **Nominal Data:** Any kind of qualitative data other than ordinal data is nominal data. For example, the name of any object, toy, electronic device, structures etc.

Quantitative Data: The opposite of categorical data is quantitative data. These data are made up of numbers. In simple words, all real numbers are Quantitative Data. For example: 1,2,3,4,5,0.1,0.5,-1,-2,-3 etc.

Like Qualitative Data, Quantitative Data is also divided into two branches.

- **Discrete Data:** The type of data that can only be positive integers is generally called Discrete Data. For example: number of cars in a garage, age of a person, runs of a batsman in a cricket match etc. If we notice, we can see that the number of cars can never be decimal. There may be

5 cars, but never 5.5 cars. Similarly human age and cricketer's runs are never decimal numbers. So these are Discrete Data.

- Continuous Data: Such data types can be positive integers, decimal numbers and negative numbers or negative decimal numbers. For example, temperature, altitude, velocity etc. The temperature of a place can be 10 degrees celsius, and it can be 12.5 degrees. It can even be 0 degrees, -2 degrees or -11.6 degrees celsius. So temperature is a continuous data. But human height is not negative, yet it is continuous data because it can be a decimal number. The same is the case with the speed of the vehicle. In simple words, all Qualitative Data except Discrete Data are Continuous Data.

Note that Discrete and Continuous Data are often debated that Discrete Data can sometimes be decimal numbers. However, we won't go into those now.

Evaluation

1. Qualitative Data are categorized into..

- (a) 4 category
- (b) 3 category
- (c) 2 category
- (d) 6 category

2. What is an example of ordinal data?

- (a) Natural numbers
- (b) Height in centimeters
- (c) Education phase
- (d) Name of fruits

3. Which data is discrete?

- (a) Velocity
- (b) Number of bones
- (c) Height
- (d) Temperature

4. Which data is continuous?

- (a) Acceleration
- (b) Gas particles
- (c) Number of books
- (d) Trees

5. What is the synonym of qualitative data?

- (a) Nominal data
- (b) Categorical data
- (c) Ordinal data
- (d) Descriptive data

6. The number of students in a class is an example of..

- (a) Ordinal data
- (b) Continuous data
- (c) Discrete data
- (d) Nominal data

7. Choose the right statement

- (a) Continuous data only can be decimal numbers
- (b) Nominal data can be ordered and replaced with numbers
- (c) Analyzed data forms information
- (d) [7,8,2,5] represents information

8. Choose the FALSE statement

- (a) "Gravitational force" is continuous data
- (b) "Age" is numerical data
- (c) "Temperature" is quantitative data
- (d) The first four consecutive numbers are ordinal data

9. Choose the nominal data

- (a) Null
- (b) Apple
- (c) 3
- (d) Very high

10. "Infinity" is a..

- (a) Continuous Data
- (b) Nominal Data
- (c) Discrete Data
- (d) Not a data type

Answers: 1.c, 2.c, 3.b, 4.a, 5.b, 6.c, 7.c, 8.d, 9.b, 10.d

1.2 Basic Terminologies

Population: In the general sense, Population means the population of a place. Population means much the same in statistics, but slightly different. Suppose we need to find the average age of people in a city. Here, what is created by every person in the city is Population. Similarly, if we have to work with every student of a school then Population is created with every student of that school, or if we want to check the effectiveness of medicine produced by a medical institution, then every medicine is Population. That is, each element of a domain is made up of Population.

Sample: A sample is created by combining some elements from each element of a domain. In simple words, if a few people or a few elements are selected from a population then it will be a sample. For example, in the above example, if we want to calculate the average age of the people in the city, then we have to ask each person their age separately, which is quite time-consuming. But if a sample of 30 people is collected from each area of that city and their average age is determined, then the work will be completed quickly. The average age of the people in the city can be estimated.

However, while selecting the sample, it should be observed that it is representative. That is, one cannot be biased while selecting. While collecting the sample, if we deliberately take the average of the sample of young children or the elderly only, then the average age will not represent the average age of everyone in the city. So the sample should be collected through divine selection.

Below are some techniques to select the sample:

Simple Random Sampling: Everyone in the population has an equal chance of being selected. Such sampling can be done by writing numbers on paper in one of the boxes. Only those who score a certain number will be included in the sample.

Stratified Random Sampling: Population is divided into certain groups (strata) according to their characteristics. Then random sampling is done from each strata. For example, in the case of biological research, different groups (strata) are created with tall, short, crooked people and then taking samples from there.

Systematic Sampling: Sampling is done with only the k-th element starting from a starting point of the population is systematic sampling. For example: by lining up 100 people in the population and creating a sample with every ten participants. Then there will be a sample of 10
(10,20,30,40,50,60,70,80,90,100)

Snowball Sampling: When existing participants bring new participants separately, it is snowball sampling.

There are many other sampling techniques. We don't have to know them all. Only one thing to remember while sampling is that whatever technique we use, it may be our own technique but biased sampling cannot be done. It should also be noted that not all types of sampling are equally useful in all types of research. In most of the cases, we use **simple random sampling**.

Parameter: The characteristics of a population are called parameters. In the above example, if we have calculated the average age of each person in the city, then the average value is a Parameter. Similarly, standard deviation, variance, mean, mode etc. of that population will also be considered as parameters.

Statistic: The characteristics of a sample are called statistics. In the above example, if we only find out the average age of the sample instead of finding out the average age of the entire city, then the average age of this sample will be Statistic. Similarly, the standard deviation, variance, mean, mode etc. of that sample will also be considered as statistics.

Evaluation

1. Total 100 bots are manufactured by Pacifica Corporation. Are these bots referring to population or sample?
2. 70 bots from them are chosen for quality testing. Are these bots referring to population or sample?
3. The average time of finishing a specific task for the 100 bots is 30.5 seconds. Is this average time a statistic or parameter?

Answers: 1. Population, 2. Sample, 3. Parameter

1.3 Descriptive Statistics

Descriptive statistics usually provide a general description of a univariate dataset. For example: mean, median, mode, standard deviation, variance of the values of a dataset are part of descriptive statistics. Data visualization also falls under descriptive statistics. However, in the case of Multivariate Dataset, there is a need for descriptive statistics.

1.3.1 Univariate and Multivariate Dataset

Descriptive Statistics is the statistical approach that is followed to get some basic information from a Univariate Dataset. As a matter of fact, univariate datasets are those datasets that contain only one variable. This variable can be exam marks, age, name, any number etc. That is, if the dataset is expressed as Rectangular, it will have one column. Otherwise, if the dataset has more than one column or variable then it is Multivariate Dataset. Note that all the datasets we see in Microsoft Excel or Google Excel are in rectangular shape. In this case, the datasets with one column are Univariate Dataset, the dataset with multiple columns is Multivariate Dataset.

Age
10

12
11
19

Table 1.1: Univariate Dataset

Age	Gender	Favorite Color
10	Male	Yellow
12	Male	Black
11	Female	White
19	Female	Black

Table 1.2: Multivariate Dataset

In Descriptive Statistics we will mainly work with univariate datasets. The following topics are discussed in this branch.

1.3.2 Measures of Central Tendency

Through the measure of central tendency, it is known that the data points of a Univariate Dataset are moving towards "a particular value". By doing this, it is possible to gain basic knowledge about the dataset by this Central Tendency. There are three ways to determine the Central Tendency of a dataset.

Mean (Average): The value obtained by dividing the sum of the data points of a Univariate Dataset rich in Quantitative Data by the number of those data points is the Mean (Average) of the dataset. It is also called Average or Arithmetic Mean. The mean of a dataset expresses the central tendency of that dataset.

Median: Median is the middle data point of the dataset. However, before extracting the median, the data points must be sorted in ascending or descending order. Otherwise, the central tendency of the data points cannot be understood. For example: If a dataset is arranged by arranging numbers from 1 to 6 in ascending or descending order, its median will be $(3+4)/2$ or 3.5. But if the dataset is sorted like this, [2,1,6,5,3,4], then its median will be 5.5. It is obvious that there is a mistake here. Because, where the highest number in the dataset is 6, the central value of that dataset is 5.5.

Mode: The data point that occurs in excess of a Univariate Dataset is Mode. If a univariate dataset has one mode then it is Unimodal Univariate Dataset, if two modes Bimodal Univariate Dataset and if more than two modes then it is Multimodal Univariate Dataset. For example [2,2,4,3,6], it is a Unimodal Univariate Dataset because it has only one mode and that is 2. Again, [2,2,3,4,5,5], both data points 2 and 5 occur twice in this dataset. So the mode of this dataset is 2 and 5. That is why it is a Bimodal Univariate Dataset. Besides, [3,3,3,4,4,4,5,5,5,6,8,9] it is a Multimodal Univariate Dataset. Because data points 3, 4, and 5 occur 3 times each in this dataset.

Outlier

Another thing we need to keep in mind while dealing with the mean value as Central Tendency. That is Outlier or Extreme Value. Consider an opener's run of ten consecutive matches [25,27,21,28,29,31,30,150,23,25]. That is, he averaged around 39 runs. But, notice that in one match he played very well and scored 150 runs, but in others he did not play so well, and his average went up because of that one 150 run. Here this 150 runs is an outlier or extreme value. This 150 runs does not represent the normal score of the opener. So, if we want to find out the central tendency of the opener's run by averaging then we first need to remove this outlier from the dataset. Then the mean or average run should be calculated. Then the average of the opener will be around 27 runs.

1.3.3 Measures of Dispersion

Dispersion refers to how scattered the data points of a dataset are from each other. Range, Standard Deviation (SD), and Variance provide an accurate idea of the dispersion of a dataset.

Range: The value obtained by subtracting the smaller value from the larger value of a dataset is the Range of the dataset. Suppose, in a class the maximum mark is 90 and the minimum mark is 50, then the range here is $(90-50)$ or 40.

Uncertainty: How much the mean value of a dataset differs from the maximum or minimum value of the dataset is called Uncertainty. $\text{Uncertainty} = \text{range}/2$

For example, [5,6,4,2,1,8] this dataset has mean 4.33 and uncertainty $(8-1)/2$ or 3.5. Uncertainty is usually written as an integer, so we round 3.5 to 3. That is, the range of data points in the dataset is 4.33 ± 3 . Notice that $4.33+3$ is 7.33 and $4.33-3$ is 1.33, which represent the largest and smallest data points in the dataset, respectively.

Standard Deviation: Standard Deviation is a measure of how far the data points of the dataset are from the mean value. A higher value of SD means that the data points are farther away from the mean value. Suppose the dataset of marks of three students in a small coaching center is [5,10,7]. Another section is the dataset of three other students' marks [8,9,10].

Below is the mathematical calculation of the SD of the first dataset (D1) and the second dataset (D2).

First dataset:

Given, D1 = [5,10,7]

Mean = $(5+10+7)/3 = 7.33$

Deviance from mean = $(5-7.33) + (10-7.33) + (7-7.33)$

Squared deviation = $(5-7.33)^2 + (10-7.33)^2 + (7-7.33)^2 = 12.667$

Standard Deviation (SD) = $\sqrt{(12.667/n)}$ [n = number of total data points]

= $\sqrt{(12.667/3)} = 2.05$ (Ans)

Second dataset:

Given, D2 = [8,9,10]

Mean = $(8+9+10)/3 = 9$

Deviance from the mean = $(8-9) + (9-9) + (10-9)$

Squared Deviation = $(8-9)^2 + (9-9)^2 + (10-9)^2 = 2$

Standard Deviation (SD) = $\sqrt{(2/3)} = 0.81$ (Ans)

Now, we have extracted the SD of the two datasets. What can be understood from this? It is seen that the SD of D2 is lower than the SD of D1. That is, the data points of D2 are closer to each other than the data points of D1. Now, we can draw many conclusions from this. We may say that everyone in the D2 section studies in groups because all their marks are close to each other. But D1 students don't do that. I can also say that maybe the teacher does not teach with much attention in D1 class, but he is biased in D2 class. Thus the SD of a dataset helps us to make various decisions. This is why SD is an important topic in research and applied statistics.

Note, in the above example when we get the value of SD from the squared deviation, we have taken the value of n as the total number of data points because we have worked with all the students in the entire classroom, which indicates the population. But if we had selected a Sample rather than all of the class, then the value of n would have to be subtracted by 1 from the total data points, $(n-1)$. This matter should be taken care of.

Variance: The value obtained by squaring the SD is Variance. It also determines how scattered the data points are. In the example above, the variance of D1 is $(2.05)^2$ or 4.20 and the variance of D2 is $(0.81)^2$ or 0.65.

Mean Absolute Deviation: Another way to understand Dispersion is to determine the Mean Absolute Deviation (MAD). But this is a less important point. But it's good to know. Below is a mathematical calculation of MAD for a Univariate Dataset:

$$D3 = [6, 8, 7]$$

$$\text{Mean} = (6+8+7)/3 = 7$$

$$\text{Deviation from mean} = (7-6) + (7-8) + (7-7)$$

$$\text{Absolute deviation from mean} = |(7-6)| + |(7-8)| + |(7-7)| = 2$$

$$\text{Mean Absolute Deviation (MAD)} = 2/n = 2/3 = 0.66 \text{ (Ans)}$$

1.4 Descriptive Statistics with Python

Python is a General Purpose Programming Language, i.e. it can be used to solve various types of problems. Python has several libraries. Different types of libraries are capable of different types of tasks. For example, for statistics there are numpy, pandas, stats etc. The topics of Descriptive Statistics discussed in the last lecture will be discussed here again through Python programming. Note that statistical tasks or supervised machine learning are fairly straightforward with this programming language thanks to Python's large number of libraries. Even those with no prior knowledge of Python can easily learn the concepts and use them in their daily life. However, if you are new, you can take a look at Python's built-in data structures (string, list, tuple, dictionary). Introduction to Python (w3schools.com)

1.4.1 Data Analysis

Before starting Applied Statistics with Python we need to discuss a few things about data analysis. In this chapter we will discuss some fundamentals of data analysis with Python. Note that we will be doing this course using Google Collaboratory.

1.4.1.1 Data Representation with Pandas DataFrame

In order to analyze any type of data and extract information from it, the data must first be represented. Typically these days we present this data via Microsoft Excel or Google Spreadsheet .xlsx files. In these applications data is represented in rectangular shape. That is, there is a column and the data is represented in the rows below this column. The data in all columns of each row is called sample or record, and a characteristic of each sample or record is called a data point.

Name	AI Model	Manufacturer	Malfunction
Syn-D1	Synthia	X-Force	High
Nex-53	Nexus	Pacifia	Medium
Xeagle-97	Helix	Nocturne MX	Low
Nex-32	Nexus	Pacifia	Medium
Xeagle-101	Helix	Nocturne MX	Low
Syn-D5	Synthia	X-Force	High
Syn-D6	Synthia	Pacifia	Medium
Xeagle-65	Helix	Nocturne MX	Low
Nex-05	Nexus	Pacifia	Medium
LiM-01	Lumine	Nocturne MX	Low

Here, Name, AI Model, Manufacturer, and Malfunction are columns. Below these columns, the data is presented in the form of rows. Each row is called a sample or record. Rows are composed of datapoints. The data written in blue color in the above picture are data points. Columns are highlighted in green and sample / record is highlighted in orange.

But we cannot use these .xlsx files if we want to do data analysis with the help of Python. Because Python can't read it. Python can read .csv files. So, we need to convert all .xlsx files to .csv files first. This can be easily done in Google Spreadsheets.

Clicking on the File tab, placing the cursor on the Download option will give you the option to download the spreadsheet file in 6 formats. From there, clicking on Comma Separated Values (.csv) will download the .csv file of the spreadsheet to the computer.

Now, we need to import this CSV file into the code cell of the G-colab notebook. If you run the following codes, a prompt will appear and if you click on it and select the CSV file from the PC, our project will get access to the CSV file.

```
from google.colab import files
import io
import pandas as pd
uploaded = files.upload()
file_name = list(uploaded.keys())[0]
df = pd.read_csv(io.BytesIO(uploaded[file_name]), encoding='latin-1')
# convert df into data dictionary
data_dict = df.to_dict(orient='list')
```

Code Analysis

The above code stores the .csv file in the form of dataframe and dictionary in the df variable according to the input given by the user. To do that, first download the files module from the google.colab package, which is able to communicate with the files inside the computer. Because of this module we can upload a .csv file to the Collab environment. If no one understands this code, no problem. This code is not used for any data analysis, only for file upload. So you don't even need to memorize or understand this code. Just copy and paste.

Pandas is a huge Python library specifically designed for data analysis. This library contains many functions which can be used as per requirement. For example, DataFrame is a pandas function that converts raw data into a rectangular dataset. Suppose, we have data stored in a variable called “random_dataset” and we want to view it using this DataFrame function. Then we have to write in the code cell

```
df = pandas.DataFrame(random_dataset)
```

Here, df is a variable name. You can replace df with something else if you want. No problem. As a result of writing the code, the rectangular shape of our previous dataset is stored in the df variable. Now if we write and run the following code,


```
display(df)
```

Then our program will display our sorted dataset.

It should be noted that Python can be used by changing the name of the library if desired. For example, if someone finds it difficult to write the name pandas repeatedly, he can rename it while importing this library.

```
import pandas as pd
```

Then we need to use pd in the code as well.

```
df = pd.DataFrame(random_dataset)
display(df)
```

Note that once a package, library or module is downloaded, there is no need to include it in the code again. That is, there is no need to repeatedly write “import pandas as pd”, “from scipy.stats import ttest_ind”, “import numpy as np” etc. It takes a bit longer to run the code. Also, you don’t have to declare a variable repeatedly in Python. If you have declared a variable or function in the above code cell, you may not mention them again in the next code cells.

The dataframe can also be represented as a dictionary if desired

```
data_dict = df.to_dict(orient='list')
```

The advantage of doing this is that those who have less knowledge about data frames but have a good idea about dictionaries can easily do data analysis.

Similarly, a dictionary can also be converted to a dataframe if desired.

```
df = pd.DataFrame(data_dict)
```

A DataFrame can optionally be converted to an .xlsx file and saved to Google Drive.

```
from google.colab import drive
drive.mount('/content/drive')

file_path = '/content/drive/MyDrive/name_your_file.xlsx'

df.to_excel(file_path, index=False)
```

In the above program, the DataFrame called `df` is saved in the user's Google Drive account. Write the name of your file by which you want to save it in your google drive erasing “name_your_file” in the above code snippet.

1.4.1.2 Necessary DataFrame Features

This section is for those who don't know much about dictionaries and don't know much about dataframes. Here are some special features of dataframes that are often used in data analysis. Not sure, the variable in which we have stored our dataframe is called `df`.

```
df.columns
```

Running this code will display the names of each column in the dataset.

```
df.shape
```

How many rows and how many columns are there in the dataset can be seen.

```
df.head(n)
```

Displays the first n number of records in the dataset.

```
df.tail(n)
```

Displays the last n number of records in the dataset.

```
df.info()
```

Describes the data types, memory usage, non-null values, etc. within the dataset.

```
df.describe()
```

Describes the descriptive statistics of the numeric columns of the dataset. For example: mean, standard deviation, min, max etc.

```
df.isnull()
```

Shows the missing values of the dataframe or dataset. Some records do not contain data points. These are called missing values. These records of missing values are often removed before data analysis.

```
df = df.dropna()
```

Discards records that have null or missing values from the dataset. Then store the dataset via a new dataframe variable.

```
df = df.fillna(0)
```

Replaces missing values with 0. If anything else is written in place of 0, they will be replaced.

```
df['column_name'].unique()
```

Shows what unique values or basic values a column has. Duplicate values are not shown.

```
df = pd.concat([df1, df2]), axis = 0, ignore_index = True)
```

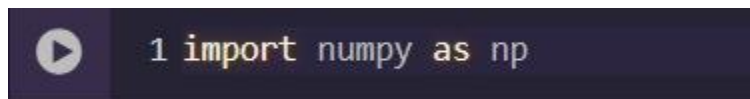
The code is used to merge two datasets (df1, df2) with the same column.

```
df = pd.concat([df1, df2]), axis =1)
```

The code is used to merge two datasets with different columns.

1.4.1.3 The NumPy Library

NumPy (Numerical Python) is an important mathematical and statistical library for Python used in data science, machine learning, signal processing and many other fields. Using NumPy, various tasks of Descriptive Statistics etc. can be performed very easily. To use Numpy, first import this library in the following way. Good thing, we'll be doing all the coding for this course in the Google Collaboratory.



```
1 import numpy as np
```

In the following code snippet, the mean, median, SD, variance, MAD of a dataset has been determined with the help of NumPy.

```
[9] 1 dataset = [15,20,18,19,14,8,17,18,16,16]
    2 mean_value = np.mean(dataset)
    3 median_value = np.median(dataset)
    4 standard_deviation = np.std(dataset)
    5 variance = np.var(dataset)
    6 mad = np.mean(np.abs(np.array(dataset) - mean_value))
    7
    8 print(f"Mean: {mean_value}")
    9 print(f"Median: {median_value}")
   10 print(f"SD: {standard_deviation}")
   11 print(f"Variance: {variance}")
   12 print(f"Mean Absolute Deviation: {mad}")

Mean: 16.1
Median: 16.5
SD: 3.2078029864690882
Variance: 10.290000000000001
Mean Absolute Deviation: 2.3
```

In the code snippet, the texts on the left side of the “=” sign are Python variables (dataset, mean_value, median_value, standard_deviation, variance, mad) and the texts on the right side are the values of those variables. For example, in the first line, the Univariate Dataset is represented through Python's built-in data structure “List” and the Python variable called “dataset” is storing this dataset. The mean value of the dataset is determined by writing np.mean(dataset) in the second line, and this value is stored in a Python variable called mean_value. Thus when all the variables are done, print() function is used to display those values from line 8.

However, figuring out the mode with Numpy is a little tricky. Because this library cannot work with a multimodal dataset. So we will use the “Counter” class of the “collections” module. If a dataset, list, string or tuple is sent to the Counter class, then the Counter class as an object returns the frequency distribution of that dataset through the dictionary, that is, how many times a value is repeated. From there we can understand the mode of our dataset.

```

✓ 0s 1 from collections import Counter

✓ [14] 1 frequency_distribution = Counter(dataset)
0s    2 print(f"Frequency distribution: {frequency_distribution}")

Frequency distribution: Counter({18: 2, 16: 2, 15: 1, 20: 1, 19: 1, 14: 1, 8: 1, 17: 1})

```

From the above code it can be seen that both the numbers 18 and 16 occur 2 times. The rest came once. So, the modes of the dataset are 18 and 16. Since there are two modes, it is a Unimodal Dataset.

Along with the counter() function, the count() function is also quite necessary. To be used often. This function is used to find the number of times a particular element occurs in an array/list/ or tuple.

```

data = [2,3,4,4,4,5,6]
count_element = data.count(4)
print(count_element)

```

Running this code will display 3. Because, the number 4 appears 3 times in the list called data.

Another such useful function is index(). It is used to know the index number of a particular element in a list / array / tuple.

```

data = ['S', 'Y', 'N', 'T', 'H' ]
position = data.index('N')
print(position)

```

The output of this program will be 2. Because the letter 'N' is at the 2nd index position of the data variable. This index calculation starts from zero. That is, the 0th index position has 'S', 1st position has 'Y' and so on.

1.4.1.4 Data Visualization

Data Visualization is an important concept in data analysis and descriptive statistics. Representing data only with numbers is not understandable to all types of people, moreover by looking at pictures one gets a clear idea about the entire data very soon. In this chapter we will learn how to visualize data in two ways. These are: Line plotting, and bar graph. In addition to this, I wanted to keep a scatter plot in

this course, but many bugs are often seen while doing scatter plot in Python. So I didn't show the scatter plot. But we will learn the alternative of scatter plot with line plotting.

The Matplotlib library will be used for visualization.

```
1 import matplotlib.pyplot as plt
2
```

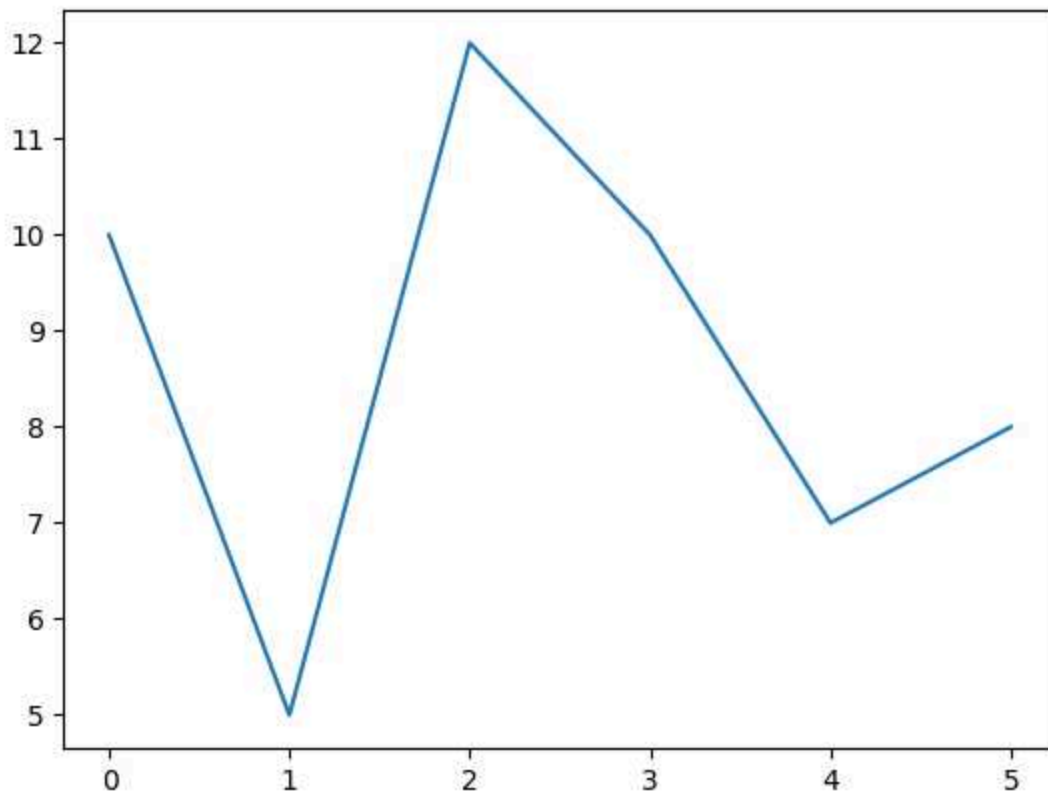
Here the matplotlib.pyplot module is imported from the matplotlib library. In short it is named as plt. Data is visualized using various functions of this module.

Line Plot

Below is the very basic line plotting.

```
1 team_a = [10, 5, 12, 10, 7, 8]
2 plt.plot(team_a)
3 plt.show()
```

In the above code, the runs of any six overs between the innings of a cricket team are stored in the team_a variable. The plt.plot(team_a) syntax actually means matplotlib.pyplot.plot(team_a), which means line plotting graphs of the integers in the team_a variable. Then plt.show() syntax instructs the program to show the generated line plot to the programmer. But even if this line was not written, it would have continued. If we ran the code by writing plt.plot(team_a), the program would show us the same line plot as below.

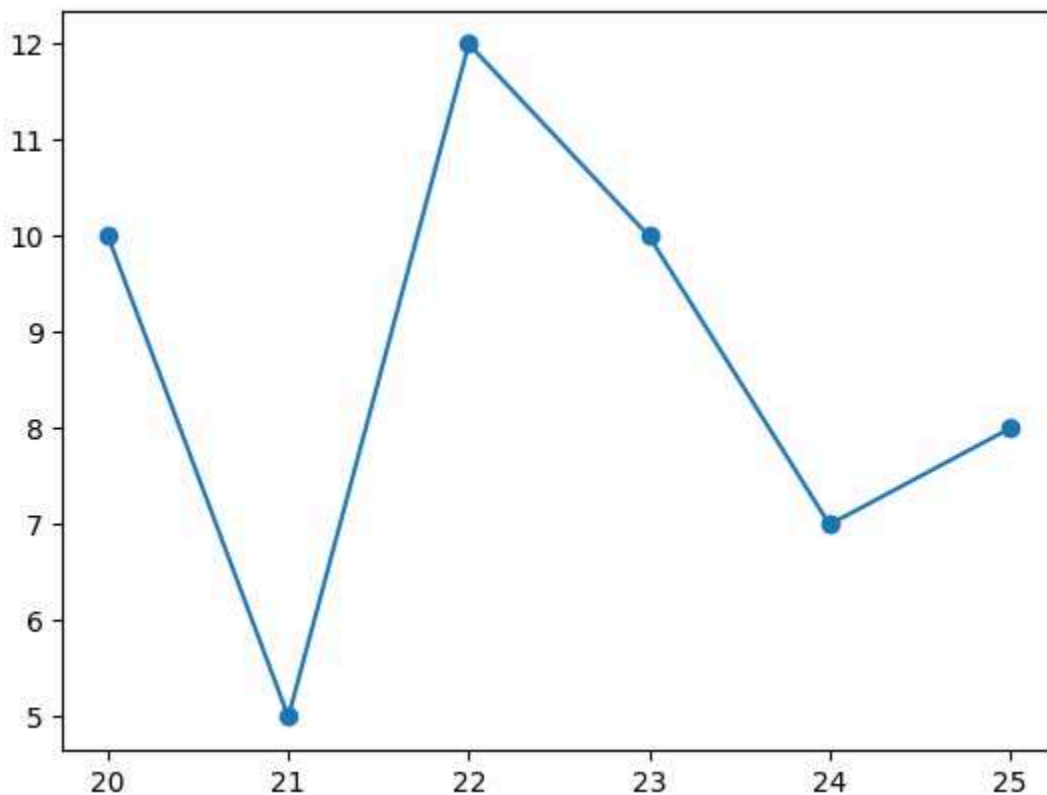


Here, it appears, the points along the x-axis are assigned 0 to 5 by default. But as we know, the over can never be 0 in the game. So we need to fix the x-axis values. Let's say the overs are 20 to 25 overs. That is, the points [20, 21, 22, 23, 24, 25] will lie along the x-axis. Also, no markers are visible in the runs on the y-axis of the plotting. Due to which it is not well understood how many runs were scored in any over.

```
1 team_a = [10, 5, 12, 10, 7, 8]
2 overs = [20, 21, 22, 23, 24, 25]
3 plt.plot(overs, team_a, marker='o')
4 plt.show()
```

Here, we have used two more parameters of the plot() function, which we left blank in the previous code. The points I give along the x-axis go to the first parameter, the points I give along the y-axis go to

the second parameter, and the marker goes to the very end. This series should be maintained. If the values cannot be reversed, the graph will not come out correctly.

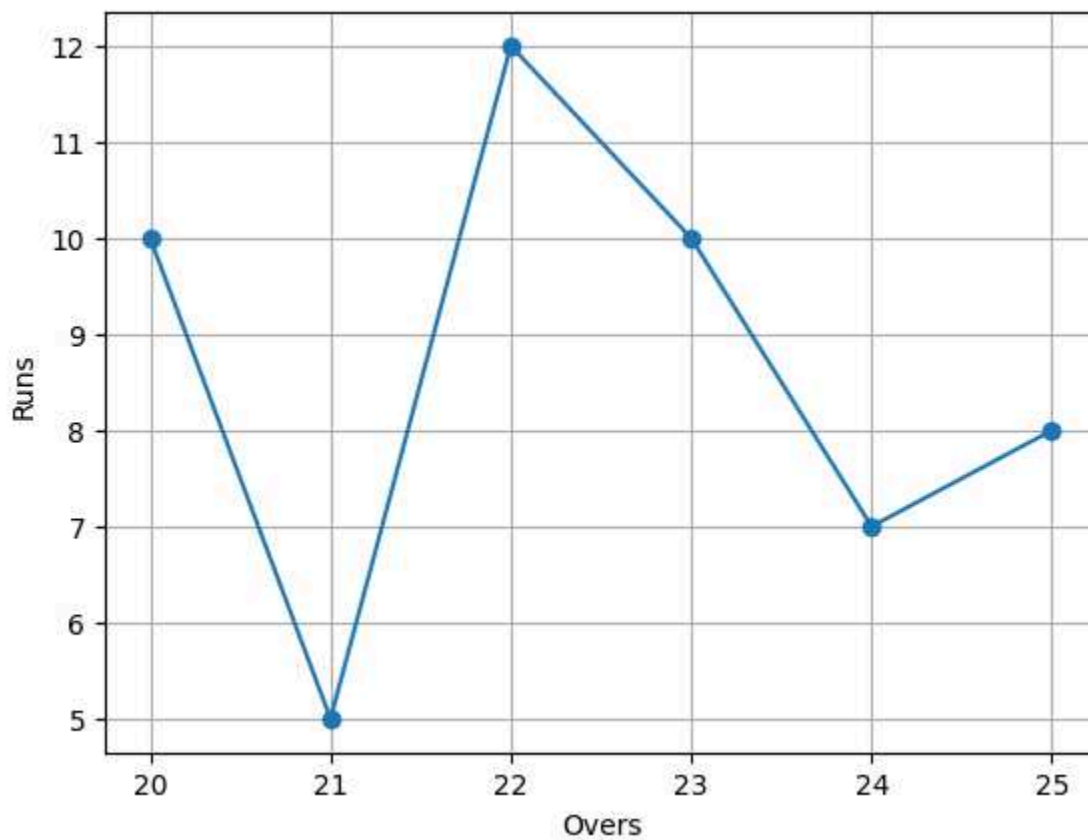


Now the plotting looks pretty good. However, there are still some things left out. For example, it would be better to write the labels of x-axis and y-axis. Otherwise, the audience will understand how over is given on which axis and run on which axis. Also, our visualization has no grid. So many may not clearly understand the datapoint despite the markers.

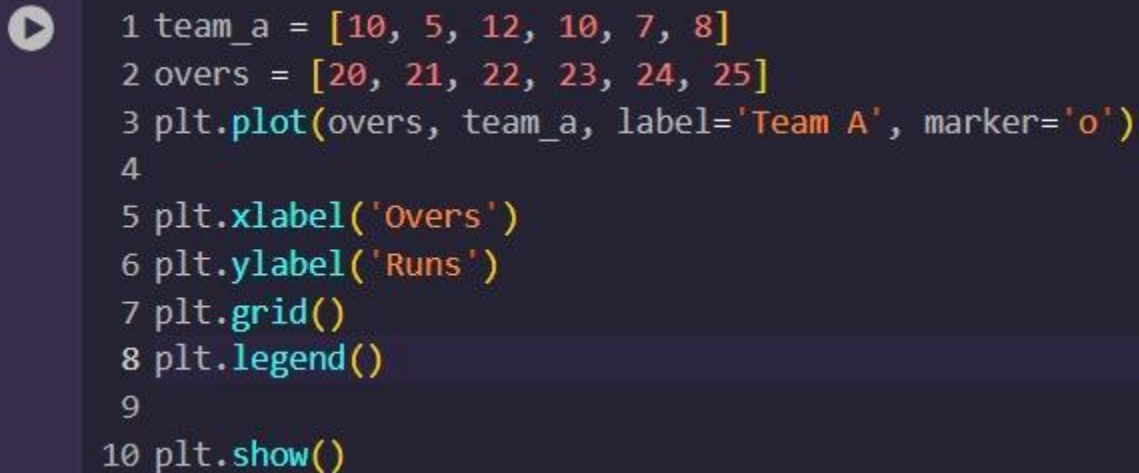

```
1 team_a = [10, 5, 12, 10, 7, 8]
2 overs = [20, 21, 22, 23, 24, 25]
3 plt.plot(overs, team_a, marker='o')
4
5 plt.xlabel('Overs')
6 plt.ylabel('Runs')
7 plt.grid()
8
9 plt.show()
```

In lines 5 and 6 we have labeled these two axes using `xlabel()` and `ylabel()` functions respectively.

Gridding of the plot is done by `grid()` function in line 7.

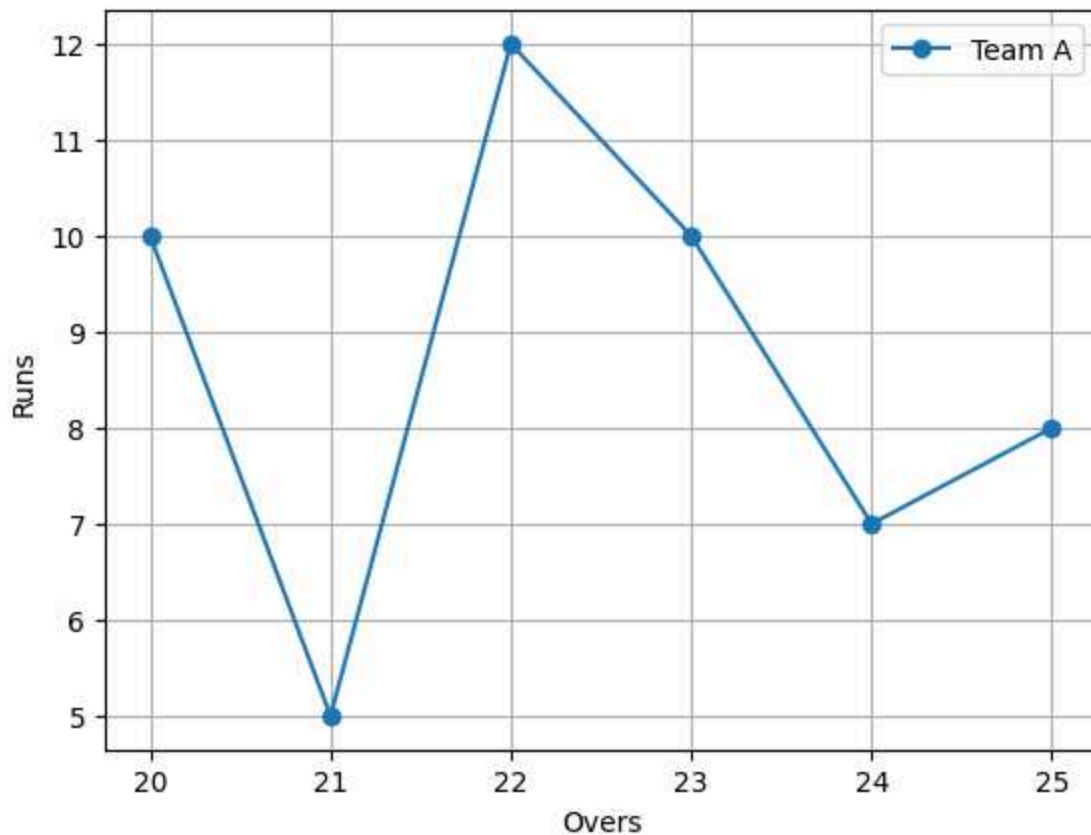


Due to the gridding and labeling any spectator can now clearly understand how many runs have been scored in any over. But still there is one thing left out. Can you say what it is? It is called Legend. That is, the viewer will not know which group's data the graph is representing. So we need to use the `legend()` function.

A code editor window with a dark background and a play button icon on the left. It contains ten lines of Python code for plotting data. The code defines two lists, 'team_a' and 'overs', and uses 'plt' to create a plot with labels, a grid, a legend, and a show command.

```
1 team_a = [10, 5, 12, 10, 7, 8]
2 overs = [20, 21, 22, 23, 24, 25]
3 plt.plot(overs, team_a, label='Team A', marker='o')
4
5 plt.xlabel('Overs')
6 plt.ylabel('Runs')
7 plt.grid()
8 plt.legend()
9
10 plt.show()
```

Line three of the program for the legend is slightly modified. Another parameter of the `plot()` function that we did not use, is now used. In the `label` parameter we have given the name of the team (Team A). Then the `legend()` function is provoked in line 8.

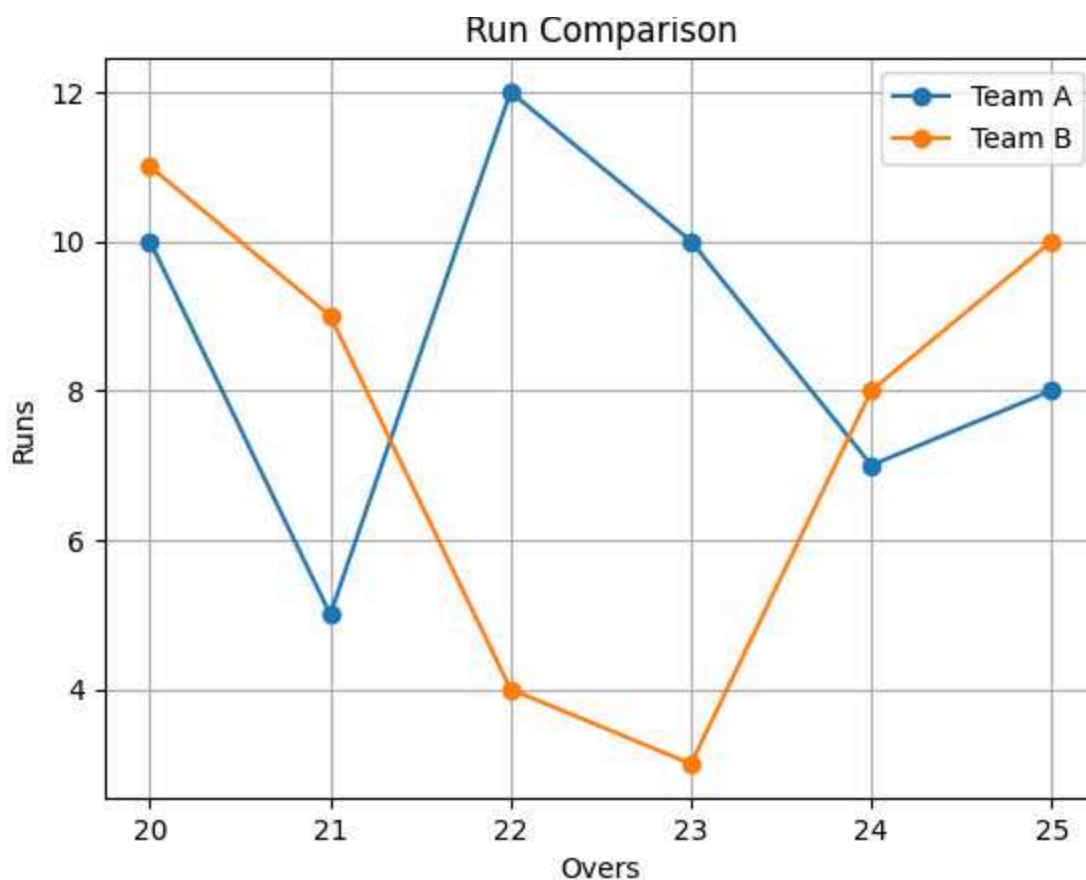


Now our graph can be said to be fairly complete. Any viewer can understand what is visualized in this graph.

If we want we can also visualize the opponent's runs from 20 to 25 overs in this same graph. Then the audience will understand the comparison between the two teams. You may all have seen that in the middle of the game in cricket, the runs comparison between the two teams in certain overs is often shown like this.

```
1 team_a = [10, 5, 12, 10, 7, 8]
2 team_b = [11, 9, 4, 3, 8, 10 ]
3 overs = [20, 21, 22, 23, 24, 25]
4 plt.plot(overs, team_a, label='Team A', marker='o')
5 plt.plot(overs, team_b, label='Team B',marker='o')
6
7 plt.xlabel('Overs')
8 plt.ylabel('Runs')
9 plt.grid()
10 plt.legend()
11 plt.title('Run Comparison')
12
13 plt.show()
```

In the second line, the runs of the opposing team are also stored in the same list as the first team. The fifth line contains the `plot()` function for the second group. Just by writing these two lines of code we can see the opponent team plotting in our previous graph. However, line 11 uses the `title()` function to give the visualization a name.

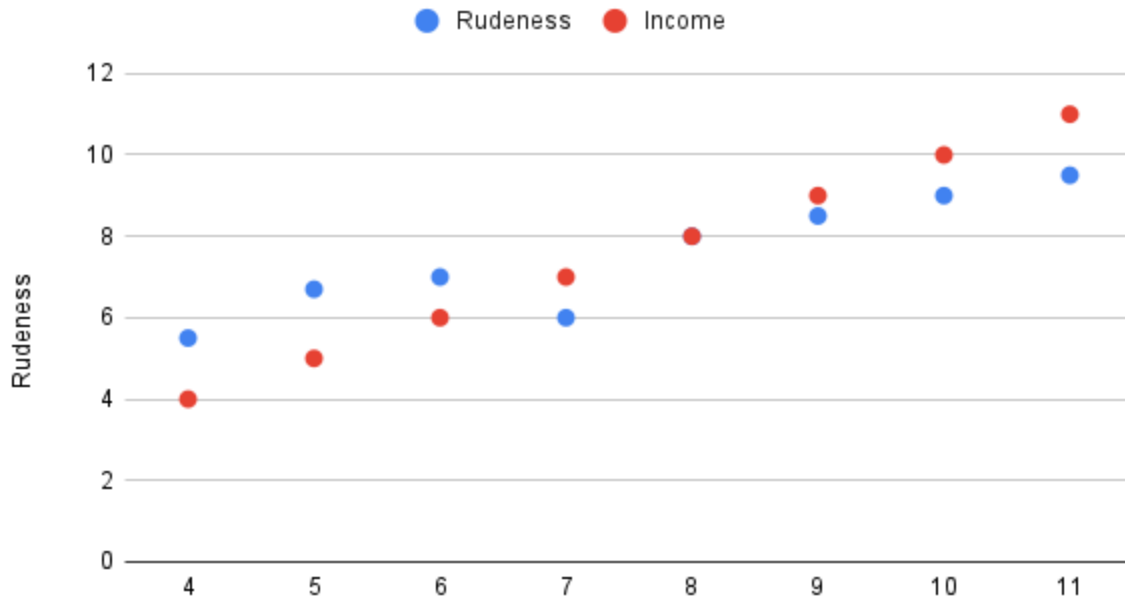


In this way the spectators will be able to see the comparison of the runs of both the teams in 20 to 25 overs.

Scatter Plot

In scatter plotting two variables are plotted along x-axis and y-axis. Decisions are then made by looking at the location of the variable's data points. Scatter plots are often used in statistics, research, or data analysis. For example, whether there is correlation between two variables, whether there are outliers in the sample, what is the spread of the data points, where is the trend of the data points, scatter plot is used to know. Below is an example of a scatter plot that I created with Google Spreadsheet.

Rudeness vs. Income

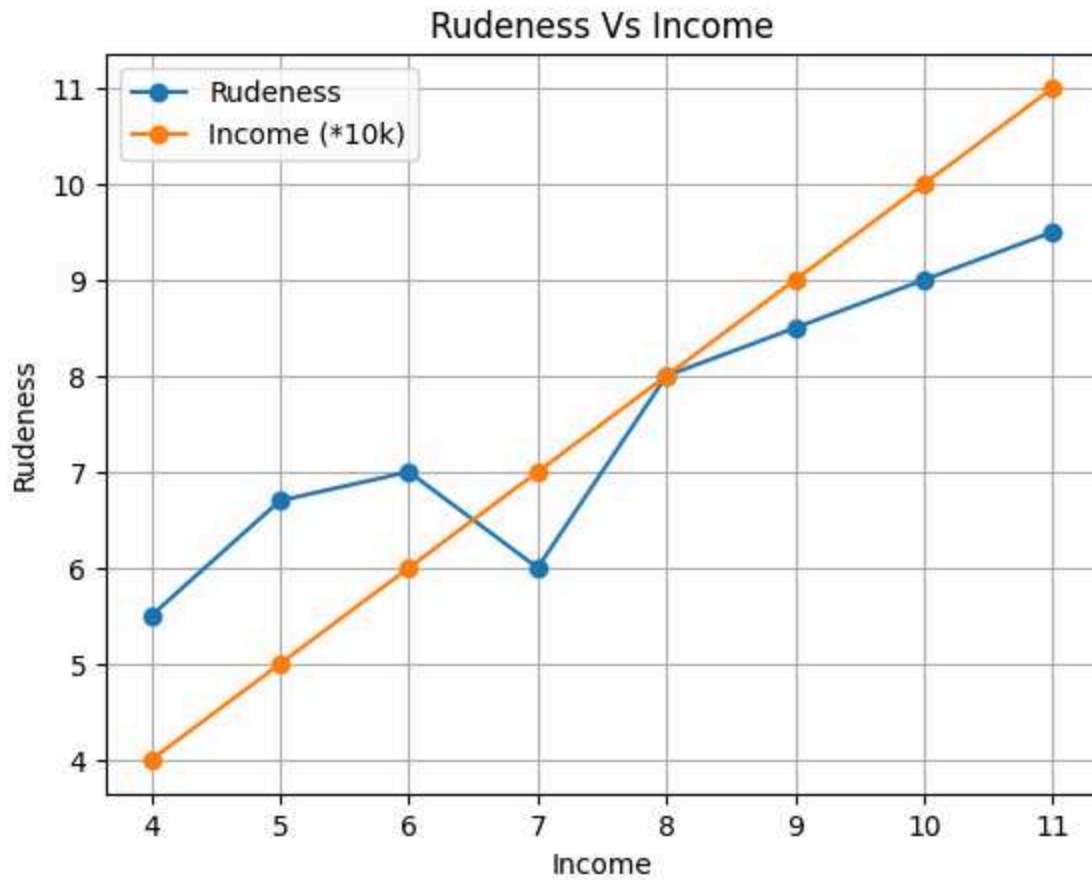


Here is a scatter plot of the monthly income of eight people and their rudeness level. If you notice, you can see that in the plot, both the income and rudeness variables are increasing with each other. That is, by looking at this scatter plot, we can say that income and rudeness correlate positively. Again, the data points appear close to each other, meaning there are no outliers and their variance is low. Note that the scatter plot shown does not represent any real data. I made this synthetic plot to give an example.

Now let's see how Line Plot can work with Scatter Plot. The matter is very simple. By plotting the two variables separately, we get a visualization similar to a scatter plot.

```
1 income_x10k = [4,5,6,7,8,9,10,11]
2 rudeness = [5.5, 6.7, 7, 6, 8, 8.5, 9, 9.5]
3 x_values = income_x10k.copy()
4 plt.plot(x_values,rudeness, label='Rudeness',marker='o')
5 plt.plot(x_values,income_x10k,label='Income (*10k)',marker='o')
6 plt.legend()
7 plt.grid()
8 plt.xlabel('Income')
9 plt.ylabel('Rudeness')
10 plt.title('Rudeness Vs Income')
```

Each line of this code is the same as our previous line of plot code. But in line number three I have made a little change. To keep the income variable on the x-axis, I first stored it in the variable called `x_values`, then in the fourth and fifth lines, the argument was passed to the parameters of the `plot()` function accordingly.



Now, comparing the previous scatter plot and this line plot, you will realize that both plots represent the same data.

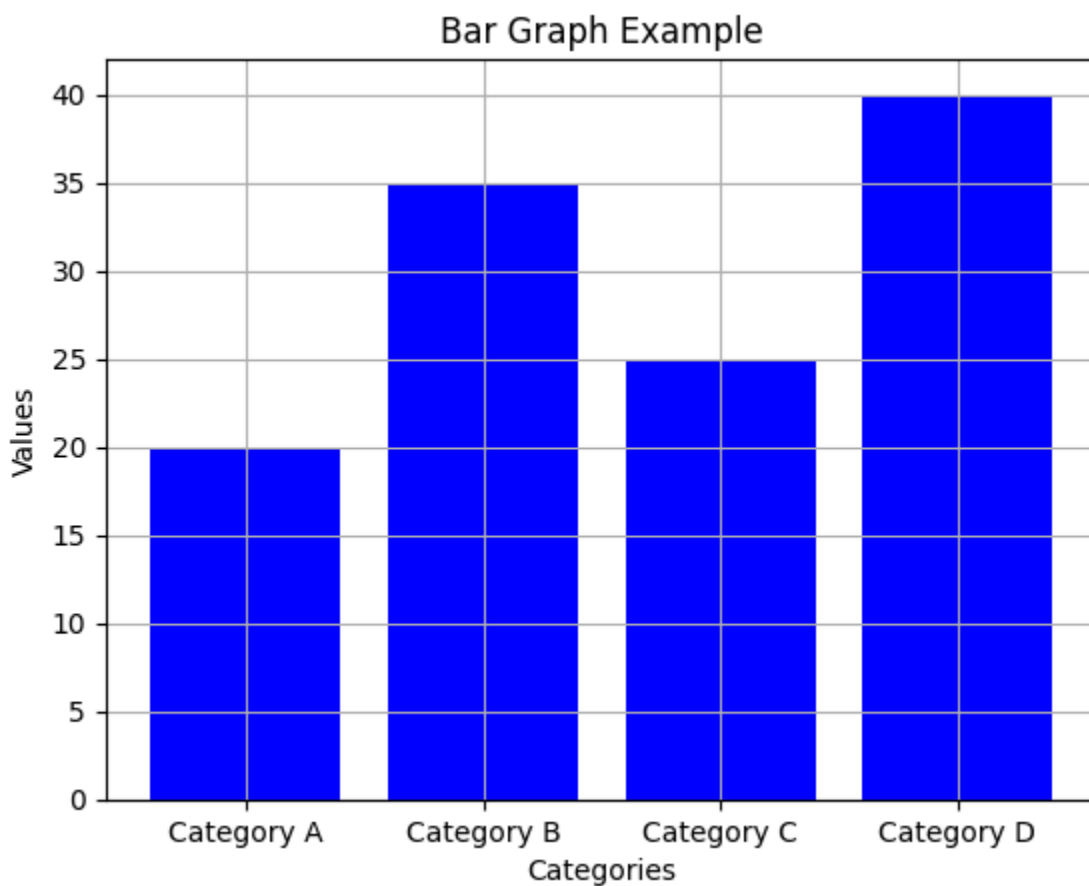
Bar Chart

Bar charts are used to represent categorical data. Each bar length represents the value of a particular variable. Below is a sample program and its output for creating a bar graph:


```

1 categories = ['Category A', 'Category B', 'Category C', 'Category D']
2 values = [20, 35, 25, 40]
3
4 plt.bar(categories, values, color='blue')
5 plt.xlabel('Categories')
6 plt.ylabel('Values')
7 plt.title('Bar Graph Example')
8 plt.grid()
9 plt.show()
10

```



Evaluation

You should use Python to analyze, solve or answer the following questions

1.A rectangular synthetic dataset is given below.

Mech ID	AI Model	Manufacturer	Task Rate
---------	----------	--------------	-----------

Syn-01	Synthia	Pacifia Corporation	6.5
Nct-01	Nocturnal	NCT Mechas	8
XF-01	Hellfire	X-Force Russia	9
Syn-02	Synthia	NCT Mechas	8.5
Nct-05	Nocturnal	NCT Mechas	7.75

- Write python programs to store these tables as a dictionary, convert it into pandas data frame and display the data frame.
- Print every column name.
- Print the unique values of the “Manufacturer” column.
- Add this column (which I am representing as a dictionary) with the dataset: {'Disability': [Medium, 'Low', 'High', 'Medium', 'Low']} and print the data frame.
- Write a Python program to find out the bot with the maximum task rate.

2. Two war mech AI model's shooting accuracy status is given below. They were deployed in 7 test battles and their average accuracy to shoot the hostiles were recorded. Decide which model is better. (Note: Maximum accuracy = 1)

synthia = [0.8,0.83,0.79,0.83,0.82,0.83,0.81]

darkwave = [0.99, 0.99, 0.95, 0.68, 0.50, 1, 0.6]

Think about the problem for a while. If you cannot solve, you can take help from the hint or solution sections.

3. A business company started to sell five new products in the market. Your task is to find out which product the price can be increased a little to eventually profit more, that means you need to find out which product is sold the most.

The dataset containing selling status can be found in this .csv file: <https://shorturl.at/iwyN5>

Download it and open it using Google Colab to analyze the dataset. Analyze and find out which product is sold the most.

4. A .csv file containing the yearly temperature of 1850-2023 is given. <https://shorturl.at/gjkuJ>

Your task is to visualize the column related to the 'temperature' and interpret the visualization.

This trend will let us know whether the world is being warmer day by day, or its relatively constant. Your visualization should include title, x and y axis labeling, and grid. Note that the

.csv file given here contains real-life data so you are inspired to add this task as a small project in your data portfolio. Check the appendix section of this coursebook to learn how to create a data portfolio.

Solution Hints

2. Calculate the average accuracy of two models. Higher accuracy means better models. Nevertheless, if the averages are equal or very close to each other, then calculate their standard deviations. Because in a war, we need weapons that can consistently hit the enemies with constant accuracy. Therefore, the model that has larger SD denotes that the model's accuracy is not consistent, rather sometimes it is high and sometimes it is low.
3. Get the frequency distribution using the Counter() function. It will tell you which product is sold the most. Now, if the price of the product is increased a little bit, due to the high buying rate of the product the company's profit will swiftly increase.
4. Plot the monthly anomaly variable.

Solution link:

<https://colab.research.google.com/drive/11qfDEzxe84twAT2zfRJPEuE3GgTHr7eT?usp=sharing>

1.5 Probability

The likelihood of an event occurring is called probability in statistics. For example: the probability of rain on a particular day, the probability of a wicket falling on a ball, the probability of giving a gift to a lover, the probability that she will be more happy, etc. Since outcome prediction has a lot to do with data science, probability is an important topic in data science. However, since probability concepts we all have read in school or college so this concept will not be discussed much. Here are just a few things to remind you.

1.5.1 Probability Terminologies

Some important terms of probability are explained below.

Experiment: The type of action (trial/operation) that brings an outcome is an experiment. For example: an experiment of rolling a dice on the board of a ludo game.

Sample Space: All possible outcomes of an experiment are Sample Space. For example, if you roll the dice on the board $\{1,2,3,4,5,6\}$, any of these outcomes can come up. The set of these outcomes is the Sample Space.

Sample Size: The total number of outcomes in the Sample Space set is called Sample Size. For example, the sample size of the above example is 6.

Event: The outcome that comes after the experiment is an event. For example, suppose a dice is rolled and a 4 is obtained. This is an event. Events are basically a subset of Sample Space.

Desired Event: The outcomes that are prioritized before starting an experiment are desired events. For example, if a player wants his dice to land on 5 or 6 before rolling the dice, then $\{5,6\}$ is a desired event. It is also a subset of Sample Space.

Probability of occurrence of an event (E), $P(E) = \text{Sample size of desired events} / \text{Sample size}$

The value of probability is never less than 0 and never greater than 1. If the value is 0, it means that there is no possibility of the event happening, and if it is 1, it means that the event will surely happen.

Example 1: Out of 30 students in a class 10 like mathematics and the rest do not. If a student is selected through random selection, what is the probability that the student does not like mathematics?

Solution: The class does not like math $(30-10)$ or 20 people.

So, probability of not liking math $= 20/30 = 0.67$

That is, if a student is selected through random selection, the probability that he will not choose Mathematics is 0.67 or 67% (Ans).

1.5.2 Binomial Distribution

When to use binomial distribution?

- In a random experiment where there can be only two outcomes: success and failure.
- Two outcomes have their own **constant** probability of occurring.
- If you want to find out the probability of **k** successes in **n** trials, you can use binomial distribution.

Example 1: (For real!) I once conveniently sampled 200 girls from Brac University and sent them friend requests. 55 of them accepted the request within two weeks (follow up period). Now, answer the following questions.

- (a) Calculate the probability of acceptance and assume that the probability is constant. Now calculate what is the probability that I will be accepted by 4 girls within two weeks if I send friend requests to 10 girls?
- (b) What is the probability that 4 to 6 girls will accept?
- (c) If I send friend requests to 150 girls, how many girls do you think they will accept (Calculate expected value)?
- (d) The approximation will vary. Determine the interval of these variations and interpret (calculate standard deviation).
- (e) Is it unusual that from that sample of 150 girls, 60 girls accepted my friend request?

Solution Manual

Step 1: Define Success and Failure in the experiment.

Step 2: Write down the values of **n** and **k** (**note** : n will be always greater than k)

Step 3: What is the probability of success (p) and failure (1-p)?

Step 4: Apply the formula $P(x=k) = {}^nC_k \cdot p^k \cdot (1-p)^{n-k}$

Solution (a)

Here, success = friend request getting accepted.

Failure = friend request getting rejected.

$$n = 10$$

$$k = 4$$

Probability of acceptance / success (p) = $55/200 = 0.275$ (Ans)

Probability of failure (1-p) = $(1-0.275) = 0.725$

Therefore, $P(x = 4) = {}^{10}C_4 \cdot 0.275^4 \cdot 0.725^{10-4} = 0.17$ (Ans)

Solution (b)

$P(4 \leq x \leq 6) = {}^{10}C_4 \cdot 0.275^4 \cdot 0.725^{10-4} + {}^{10}C_5 \cdot 0.275^5 \cdot 0.725^{10-5} + {}^{10}C_6 \cdot 0.275^6 \cdot 0.725^{10-6} = 0.17 + 0.07 + 0.02 = 0.26$ (Ans)

Solution (c)

Expected value $\mu = E(x) = np = 150 \cdot 0.275 = 41.25$

Approximately 41 girls will accept the friend request.

Solution (d)

Standard deviation $\sigma = SD(x) = \sqrt{np(1 - p)} = \sqrt{150 \cdot 0.275 (1 - 0.275)} = 5.46$

Interval of these variations = $(41-5.46, 41+5.46) = 35.53, 46.46$

41 girls are expected to accept the friend request with a standard deviation of 5.46. It means, in some cases 5 girls can vary to accept or reject the request.

Solution (e)

We expected that 41 girls would accept the request.

We are observing that 60 girls accepted the request.

$$\text{So, z-score of the observation} = \frac{\text{observed} - \text{expected}}{SD} = \frac{60 - 41}{5.46} = 3.47 > 2$$

From the empirical rule we know that 95% of the data falls between 2 SD from the mean. The rest of the data is unusual. Therefore, if the absolute z-score of the observation surpasses 2, we can say that the observation is unusual. So, it is unusual that 60 girls would accept my friend request.

1.5.3 Geometric Distribution

When to use geometric distribution?

- To count the probability of getting first success in nth trial
- Probability of success and failure is constant.

Example 1: Let us consider a new synthetic scenario. The probability (assuming its constant) of my friend request getting accepted by girls is 0.275. Now, answer the following questions.

- (a) What is the probability of getting accepted at 7th request?
- (b) What is the probability of getting accepted in the first three trials?
- (c) How many friend requests should I expect to send to get the first acceptance (calculate expected value)?
- (d) Calculate variance and standard deviation. Interpret standard deviation in this context.

Solution Manual

Step 1: Define Success and Failure.

Step 2: Write down the total number of trials **n**.

Step 3: Apply the formula $P(n) = (1-p)^{n-1} \cdot p$

Solution (a)

Here, success = getting accepted. Failure = getting rejected.

$$n = 7$$

$$\text{Given, } p = 0.275$$

$$P(7) = (1-0.275)^{7-1} \cdot 0.275 = 0.03 \text{ (Ans)}$$

Solution (b)

$$\begin{aligned} P(x \leq 3) &= P(1) + P(2) + P(3) = (1-0.275)^{1-1} \cdot 0.275 + (1-0.275)^{2-1} \cdot 0.275 + (1-0.275)^{3-1} \cdot 0.275 \\ &= 0.275 + 0.19 + 0.14 = 0.605 \text{ (Ans)} \end{aligned}$$

Solution (c)

$$E(x) = 1/p = 1/0.275 = 3.63$$

It is expected that I should send approximately 4 friend requests to get the first acceptance.

Solution (d)

$$\text{Var}(x) = (1-p) / p^2 = (1-0.275) / (0.275)^2 = 9.58$$

$$\text{SD}(x) = \sqrt{\text{Var}(x)} = \sqrt{9.58} = 3.09$$

Approximately 4 trials are required to get the first success with an sd of 3.09, which means the first success will be achieved within 1 to 7 trials.

2. Statistical Analysis

Decision-making is an important focus of statistics or data science. Statistics always work to facilitate human decision making. By analyzing big data, meaningful information is extracted from it through various types of statistical analysis. In this chapter we will discuss three types of statistical analysis.

They are Student's T-Test, ANOVA (Analysis of variances) and Pearson Correlation. I will discuss with this a post-hoc test named Tukey HSD which is associated with ANOVA. These tests are also referred to as Hypothesis Testing.

2.1 Student's T-Test

T-Test is done to find out whether there is a significant difference between the mean values of two groups or Univariate Dataset. In 1908, William Sealy Gosset discovered the method while working at St. James's Gate Distillery and published a research paper on the method under the pseudonym "Student". That's why T-Test is also called Student's T-Test. Note that ANOVA (Analysis of Variance) is used to find out the difference between the mean values of multiple groups. These are a type of statistical analysis or hypothesis testing. Apart from these there are some other tests like the z-test, chi-square test etc. However, we will discuss T-Test and ANOVA in this course.

Guess that the result of each quiz of Ashiq and Promi has been released. Their obtained numbers are given below as a dataset in Python List.

```
ashiq = [15, 12, 14, 14]
promi = [12, 11, 14, 13]
```

It is seen that Ashiq's total marks are 55 and Promi's total marks are 50. If the mean of their marks is calculated then it will be seen that Ashiq's average marks is 13.75 and Promi's total marks is 12.5. So, apparently we can say that in the quiz Ashiq got better marks than Promi so he is a better student than Promi.

But statistically we cannot say that. Because, maybe Promi's health was bad during the exam, or for some other random reason Promi's test got a little bad due to which their marks dropped.

Hypothesis testing is done to solve this problem. If we can statistically prove that there is a significant difference between the mean values of Promi and Ashiq, then we can say that Ashiq performed better than Promi. Whether there is a wide gap or not can be understood from the p-value obtained from T-Test.

Usually, if the p-value is greater than 0.05, it means that there is no significant difference between the mean values of the two datasets. This phenomenon is called Null Hypothesis (H_0) remains. At the outset of Hypothesis Testing we assume that the mean of the two datasets is not significantly different. This assumption is called Null Hypothesis (H_0). For Aashiq-Promi's quiz marks to hold H_0 means, "Aashik performed better than Promi in the quiz" We do not have strong statistical evidence to say this. Because p-value is greater than 0.05, we could not reject the Null Hypothesis. That is, there is no significant gap between the mean marks of Ashiq and Promi. So, I cannot say that Aashiq performed better than Promi. Rather, for some random reason, Promi's performance got a little worse and Aashiq did better.

But if the p-value is less than 0.05, it means we have strong statistical evidence to say that Ashiq performed better than Promi. There is nothing random here. Then, we reject the Null Hypothesis (H_0). Conversely, we accept the Alternative Hypothesis (H_1). Then we can't say that there is a big gap between the average marks of Promi and Ashiq. On the one hand Ashiq's total marks are higher, on the other hand there is a wide gap in their average marks, so it can be safely said that Ashiq's performance in quizzes was actually better than Promi's.

Hypothesis testing using pen and paper is quite a complicated process. But Hypothesis Testing is quite easy with Python. We will import the `scipy.stats` library. Then using the `ttest_ind()` function there, we can easily find out the value of t score and p-value. If the value of this p-value is less than 0.05, Null Hypothesis (H_0) will be rejected and Alternative Hypothesis (H_1) will be accepted. Which means, there is a wide gap between the mean values of the two datasets. Otherwise, no.

```
from scipy.stats import ttest_ind
ds1 = [3,6,5]
ds2 = [20,30,25]
t_score, p_value = ttest_ind(ds1,ds2)
if p_value < 0.05 :
    print(f'P-value: {p_value}')
    print('Reject Null Hypothesis (H0). Accept Alternative Hypothesis (H1).')
    print('There is statistically significant mean difference among the two groups.')
else :
    print(f"Failed to reject Null Hypothesis (H0)")
```

Output

```
P-value: 0.002530490792582146
Reject Null Hypothesis (H0). Accept Alternative Hypothesis (H1).
There is statistically significant mean difference among the two groups.
```

Hypothesis testing: The above process you learned above is also known as hypothesis testing for two means for independent samples. When to use this?

- Assume you want to assess two teaching methods. You will pick two sections and conduct the independent t-test to measure which section has significantly higher marks on average.
- You can also conduct this to analyze which marketing strategy technique leads to better profit.

In short, whenever you need to compare the mean difference from two independent groups (where every participant is unique), you conduct an independent t-test.

There is another t-test that is known as paired t-test. When to use this?

- Assume you want to test whether the headache reduces after taking a medicine. You will collect the time to reduce headache for several members before taking the medicine and after taking the medicine. If there is a significant difference in the test, you can say that the medicine actually works.

In paired t-tests, the participants in both groups are the same (not unique) and mostly this testing is used when you need to know the mean difference of before and after experiments.

Here's how you can conduct the paired t-test.

```
import scipy.stats as stats

# Sample data
pre_treatment = [130, 128, 132, 129, 135]
post_treatment = [120, 115, 130, 125, 135]

# Paired t-test
t_stat, p_value = stats.ttest_rel(pre_treatment, post_treatment)
print(f't-statistic: {t_stat}, p-value: {p_value}')
```

You know what to do if p-value is less than 0.05, or greater than 0.05. Even if it's 0.05, it's better to reject the null hypothesis (H_0).

2.2 ANOVA

When there is a need to determine whether the means of data points in more than two datasets are significantly different, the T-Test cannot be applied. Because T-Test can only work with two datasets. In such cases, it can be determined using ANOVA.

The concept of ANOVA is very similar to T-Test. After doing the ANOVA testing the value of f-statistics and p-value comes out.

If the value of p-value is less than 0.05 then we understand that there is at least one dataset whose mean value is significantly different from the mean value of any other dataset. That is, we will then reject H_0 and accept H_1 .

For your ease you can state the hypotheses in the following manner.

H_0 : The mean of three or more groups are equal.

H_1 : The mean of three or more groups are not equal (significant difference exists)

On the other hand, if the p-value is equal to or greater than 0.05 then we will not reject H_0 . Which means, there is no significant difference between the average values of the datasets.

2.2.1 Tukey HSD (Honestly Significant Differences) Test

Tukey HSD test is performed after ANOVA testing, if p-value < 0.05 (Acceptance of H_1) only. Because the value of p-value smaller than 0.05 means that there is a large difference between the mean values of the groups. Now, the Tukey HSD post-hoc test is performed to show which group's mean values are significantly different from those of the other groups. But if we cannot reject the Null Hypothesis (H_0), then there is no need to do this test. Note that the test performed to find out whether there is a large difference in the average value of any dataset between more than two datasets is called post-hoc analysis. Tukey HSD Test is also a post-hoc analysis.

Statisticians discourage all complex statistical analysis such as T-Test, ANOVA, post-hoc analysis in pen and paper. Because it wastes a lot of time and the chances of making mistakes are high. So, without explaining their formula, we will discuss how these analyzes are done through statistical software (e.g. Python).

A code snippet is shown below. The program first performs an ANOVA test on more than two datasets, and if H_0 is rejected, Tukey's HSD test is performed to see if the means between the datasets are significantly different.

```
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# Example data for three groups
```

```

data = {'A': [11, 9, 10], 'B': [10, 12, 11], 'C': [50, 51, 52]}

# Creating a DataFrame with a 'group' column
df = pd.DataFrame({'value': [value for values in data.values() for value
in values],
                    'group': [group for group in data.keys() for _ in
range(len(data[group]))]})

# Performing one-way ANOVA
model = ols('value ~ group', data=df).fit()
anova_table = sm.stats.anova_lm(model)

# If the ANOVA is statistically significant (p < 0.05), proceed with
post-hoc tests
if anova_table['PR(>F)'][0] < 0.05:
    # Perform Tukey's HSD test
    tukey_result = pairwise_tukeyhsd(df['value'], df['group'])

    # Display the results
    print(tukey_result)
else:
    print("ANOVA not statistically significant; post-hoc tests not
performed.")

```

Output

```

Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====
group1 group2 meandiff p-adj  lower  upper  reject
-----
      A      B      1.0 0.4827 -1.5052  3.5052  False
      A      C     41.0   0.0 38.4948 43.5052   True
      B      C     40.0   0.0 37.4948 42.5052   True
-----

```

In the above program, if there is a significant difference in the mean difference of the data points of variables A, B, and C, the program performs the Tukey HSD test. According to the test results, there is a significant difference in the mean of the data points of A,C and B,C datasets. That's why the output shows False in the reject column for A,B (i.e, there is no significant/huge difference between them) and True for A,C and B,C (there is a significant difference between them).

2.3 Pearson Correlation

Sometimes it may be the case that as the value of one variable increases, the value of another variable increases. Or, the value of one is increasing while the value of the other is decreasing. Such a relationship between two variables is called Correlation. A widely used method of calculating this correlation is Pearson's Correlation Coefficient, whose value can range from -1 to 1 and is expressed by r . If the value of two variables (e.g. age, height, number) increases with each other then the value of the coefficient will move towards the positive direction, and if it decreases then it will move towards the negative direction. That is, if the value of Correlation Coefficient is 1, it means that the data points of two datasets are increasing at the same rate. And if -1, it means that one of the data points in the two datasets is increasing and the other is decreasing. Let's make this clear with an example.

Here is a Synthetic Dataset (a synthetic dataset is a type of fake dataset that does not represent any real data, only created for the purpose of running an example or special experiment) showing the ages of 4 girls and how many times they cry per week.

Age	Cry per week
10	0
14	3
18	6
22	9

Note, however, that as age increases, so does the amount of crying per week. Since, is increasing at an “equal rate”, we can say without finding the value of r that there is a positive linear relationship between the age of the girls in the sample and the tendency to cry. This Positive Linear Relationship is also referred to as “Positive Correlation”. That is, from this dataset we can conclude that girls tend to cry more as they get older. Nevertheless, an important thing, Correlation does not denote causation. We can't say that girls cry when they get older. That is, two variables exhibiting a linear relationship with each other does not mean that they cause each other. Some hidden factors (*lurking variables*) are at work here, which researchers have to find out, or inform those who will study this later. There may be some other reasons behind the crying of girls, which are unknown to us. It could be family problems, hormonal problems, aging and tendency to hang out with bad boys etc. So, if we are one of

the psychological research teams, we will mention in our research paper that since the tendency of girls to cry increases with age, hence family and society should take extra care of growing girls so that depression, anxiety, family problems or any other problem cannot affect them.

Explanation: Correlation Does Not Denote Causation!

Correlation is an interesting subject. Because sometimes all such variables are correlated with each other which makes us think enough. A famous topic is that those who eat too much ice cream, many of them drown in water. Isn't that mysterious?

The fact is, people eat more ice cream in the summer and people swim more in the summer. So naturally the number of drownings is higher in summer. On the other hand, the number of people eating ice cream is also high. That is, these two variables are positively correlated with each other. But the point here is not that they are drowning because of eating ice cream! Rather, many new swimmers are drowning because of going swimming in the extreme heat. That is why it is said Correlation does not denote causation!

2.3.1 Strength of Linear Relationship

The value of r indicates how strong the linear relationship between two variables is.

Correlation Coefficient r	Strength	Direction
0.7 - 1	Strong	Positive
-0.7 - 1	Strong	Negative
0.3 - 0.7	Moderate	Positive
-0.3 - -0.7	Moderate	Negative
0 - 0.3	Weak	Positive
-0.3 - 0	Weak	Negative
0	No correlation	No direction

The following code snippet shows the calculation of the Pearson correlation coefficient using the `scipy.stats` library. Here, X and Y are two variables (dataset).

```
from scipy.stats import pearsonr

# Example data
x = [1, 2, 3, 4, 5]
y = [2, 3, 4, 5, 6]

# Calculate Pearson correlation coefficient and p-value
correlation_coefficient, p_value = pearsonr(x, y)

# Print the results
print("Pearson correlation coefficient:", correlation_coefficient)
print("P-value:", p_value)
```

Output

```
Pearson correlation coefficient: 1.0
P-value: 0.0
```

Project

Finish the projects and add to your data portfolio

1. A real-life dataset is given here in .csv format.

https://drive.google.com/file/d/1P-OyaL9pnCWlqQdO2MF_nnY0VdeotEV/view?usp=sharing

It represents the sleeping disorder of certain people. status Download and import it in your google colab IDE.

Now, determine the correlation coefficient of the following variables and interpret your findings.

- Sleep duration and physical activity level
- Sleep duration and quality of sleep
- Sleep duration and stress level
- Sleep duration and heart rate
- Heart rate and daily steps
- Heart rate and physical activity level

2. The following real-life dataset consists of two columns. One is the mean readability index (MRI) of ChatGPT and the other one is the MRI of Microsoft Copilot.

https://drive.google.com/file/d/1aCTPMhiAm0WTbSPjykiZNjFILuaBxc15/view?usp=drive_link

Readability Index is an approach to determine the difficulty of a text. A Higher RI score means higher difficulty. Your task is to figure out whether there exist significant mean differences in the MRI of ChatGPT and Copilot. If yes, then compare the mean MRI of the two AI chatbots. Chatbot with higher mean denotes that it responds with more difficult sentences than the other bot.

3. Sentiment Analysis is one of the most famous Natural Language Processing (NLP) techniques and it can detect neutralness, positiveness, and negativeness of an opinion or sentiment and compoundly represent them with numbers. The following dataset consists of this information where ChatGPT's response to moral questions were recorded.

<https://drive.google.com/file/d/1CYxtcR1VbK-tTQ98dABepoY0B9bxqCAz/view?usp=sharing>

Your task is to determine whether significant mean differences exist between the three variables 'Negative', 'Positive' and 'Neutral'.

Solution link:

<https://colab.research.google.com/drive/1HHnbbfDVSGMliAhkMUyq2B44w7fB5u7?usp=sharing>

3. Survival Analysis

Survival analysis is a special branch of statistics that analyzes the time of occurrence of an event. For example: the lifespan of a cell, how long an electrical device is functional, how long a patient lives after contracting a disease etc. There are three important terms in survival analysis.

- i. Exposure
- ii. Event
- iii. Censoring Data

That is, first there will be an exposure, then the event will happen. The time from this exposure to the event is discussed in Survival Analysis. In the above examples,

- Cell birth = exposure, cell death = event
- Electrical equipment manufacturing = exposure, electrical equipment failure = event
- Disease = exposure, death = event

Let's consider the last example. Consider that 5 patients were diagnosed with a serious disease a year ago. We wanted to analyze who died in the first 12 months from the date of diagnosis. Thus, the first patient died within 7 months, the second died after 5 months, the third was still alive after 12 months, the fourth died after 6 months and the fifth died after 9 months.

Here, the third patient was still alive after 12 months, so his data will be censored. That is, if the event does not occur despite the exposure, then we will censor that data piece.

In this chapter we will focus on two important aspects of survival analysis. The topics are: Estimation of Survival Function, and Hazard Function Analysis.

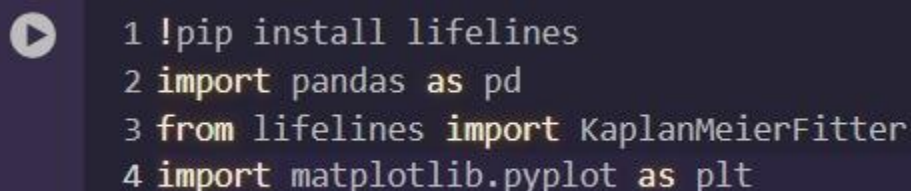
3.1 Estimating Survival Function

The probability of an event not occurring at a particular time point is calculated by the survival function. If the survival probability of 5 patients is low before giving the drug, and the survival probability is high after giving the drug, then the result of the drug can be said to be good. And if there is a significant difference (rejection of H_0) in the average values of this survival probability, then the medicine is generally called effective.

Survival probability can be estimated by Kaplan-Meier Estimator.

3.1.2 Kaplan-Meier Estimator

Through this estimator, Survival Function or Survival Probability is estimated from Lifetime Data. This survival probability is expressed by $S(t)$ and is essentially the probability that the event does not occur at time point t . For example, probability of cell not being destroyed at time t , probability of cancer patient not dying, probability of machine efficiency not being lost etc. The value of $S(t)$ can be easily extracted from a Lifetime Data Table through the following Python script.



```
1 !pip install lifelines
2 import pandas as pd
3 from lifelines import KaplanMeierFitter
4 import matplotlib.pyplot as plt
```

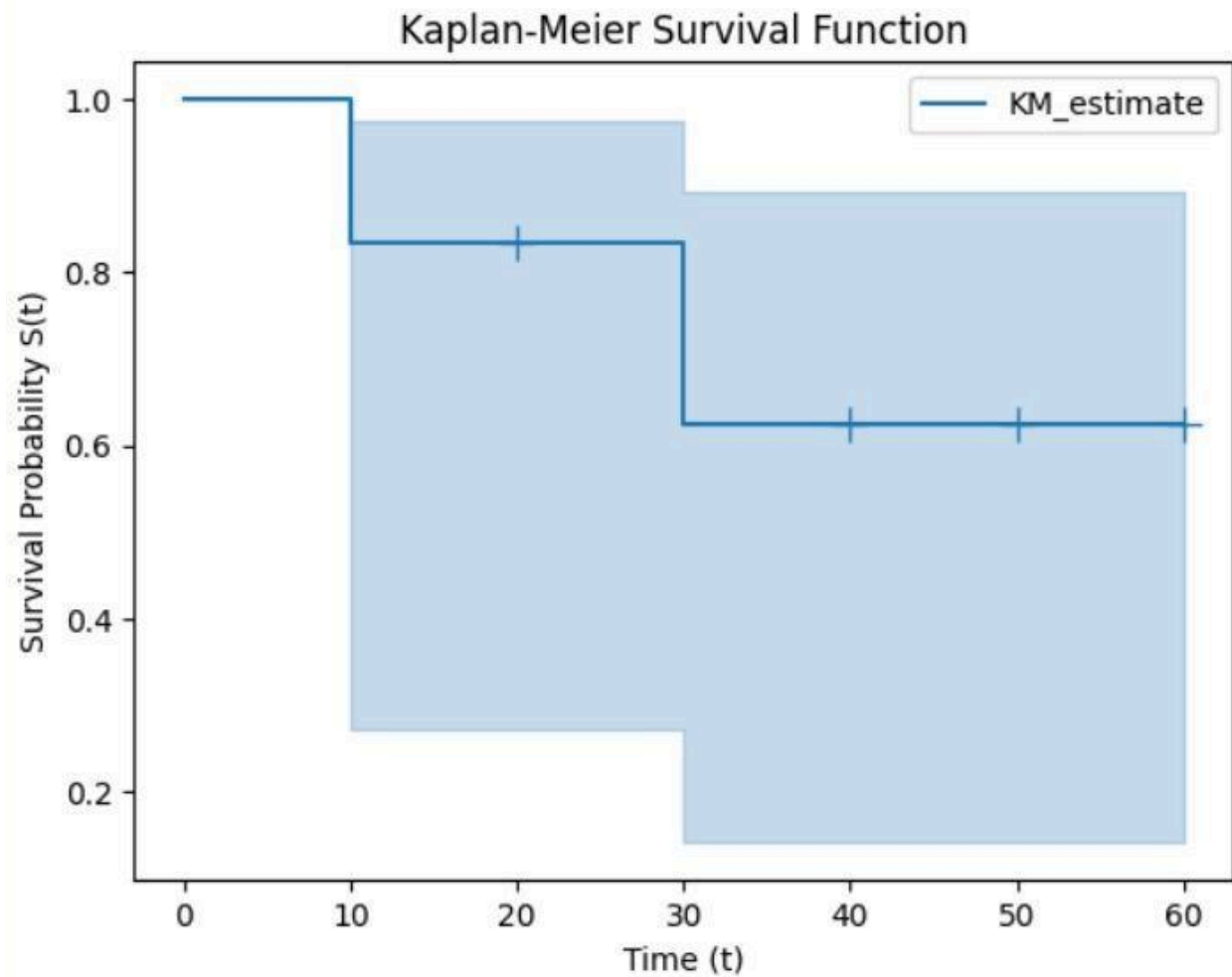
Here, in the first line we have installed the Python lifelines package. We can use all functions for survival analysis from this package. In the second and fourth lines we install the pandas and matplotlib libraries respectively. Their usefulness has been discussed in previous chapters. In the third line we fetch a KaplanMeierFitter class from the installed lifelines library. We will perform our time-to-event analysis through various methods of this class.

```

1 # Sample Life data table
2 data = {'time': [10, 20, 30, 40, 50, 60],
3         'event': [1, 0, 1, 0, 0, 0]} # 1 indicates event occurred, 0 indicates censored
4
5 df = pd.DataFrame(data)
6
7 # Creating a Kaplan-Meier Estimator object (model)
8 kmf = KaplanMeierFitter()
9
10 # Fitting the data to the kmf model
11 kmf.fit(durations=df['time'], event_observed=df['event'])
12
13 # Plotting the survival function
14 kmf.plot_survival_function(show_censors=True)
15 plt.title('Kaplan-Meier Survival Function')
16 plt.xlabel('Time')
17 plt.ylabel('Survival Probability')
18 plt.show()
19
20 # Estimate the probability of an event not occurring at the given time point
21 time_point = 50
22 probability_at_time_point = kmf.predict(time_point)
23 print(f"The estimated survival probability at time {time_point} is: {probability_at_time_point}")
24

```

The above code snippet shows how survival probability is estimated at time t through the KaplanMeierFitter class. The three notable methods of this class are `fit()`, `plot_survival_function()`, and `predict()`. The first method fits the Life Data Table into a KaplanMeierFitter model with two parameters (durations, and event_observed). Then, it plots the survival probability ($S(t)$) against time (t) via `plot_survival_function()`. Here, passing the “True” argument to the `show_censors` parameter will also display the censored observations in the graph as a fine long line. Above all, the KaplanMeierFitter class or model calculates the Survival Probability through several mathematical formulas and algorithms within the `predict()` method. Output of the program is shown below:



The estimated survival probability at time 50 is: 0.625

That is, according to our sample life data table (code snippet line 2), the probability of no event occurring in 50 seconds or the value of $S(t)$ in 50 seconds is 0.625.

3.2 Hazard Function Analysis

The rate of how many cumulative events have occurred at a particular time point t is calculated through Hazard Function Analysis.

A good function for work on the Nelson-Aalen Estimator.

3.1.2 Nelson-Aalen Estimator

This estimator extracts the value of the cumulative hazard function or $H(t)$ at a specific time point t from the Life Data Table. This value indicates the event occurrence rate up to that time point. The higher the rate in datasets of patient death or device malfunction, the worse the outcome.

Note that $H(t)$ and $S(t)$ are directly proportional to each other. Because, if the value of Survival Probability continues to increase, Cumulative Hazard will naturally decrease. Conversely, increasing Cumulative Hazard means decreasing the value of Survival Probability. The relationship between them is expressed through the following formula.

$$S(t) = \exp(-H(t))$$

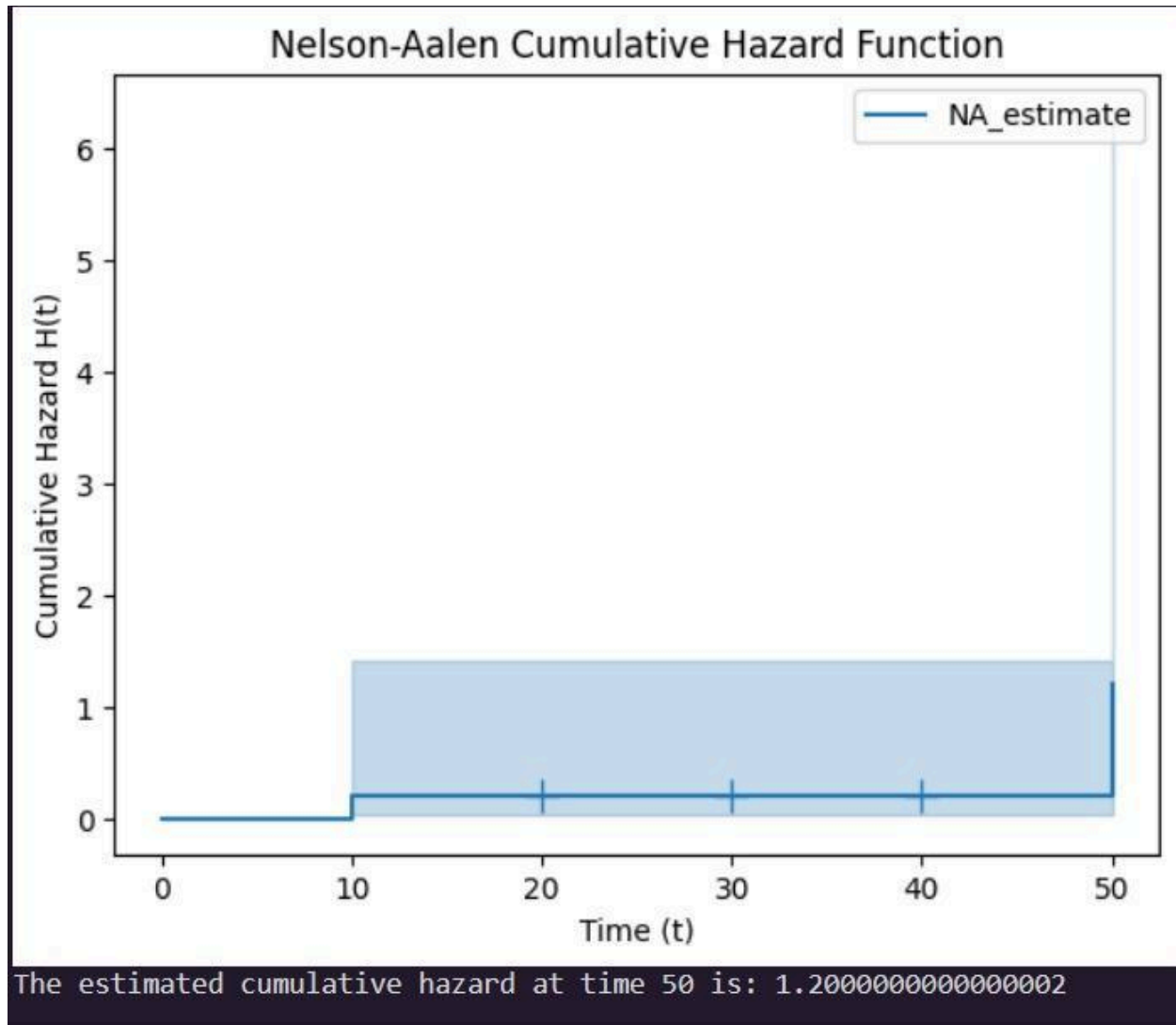
Below is the hazard function analysis using Nelson-Aalen Estimator through Python scripting.

```
1 from lifelines import NelsonAalenFitter
2 # Sample life data table
3 data = {'time': [10, 20, 30, 40, 50],
4         'event': [1, 0, 0, 0, 1]} # 1 indicates event occurred, 0 indicates censored
5
6 # Converting the dictionary into a dataframe
7 df = pd.DataFrame(data)
8
9 # Creating Nelson-Aalen estimator object
10 naf = NelsonAalenFitter()
11
12 # Using the object model
13 naf.fit(durations=df['time'], event_observed=df['event'])
14
15 # Plotting the cumulative hazard function
16 naf.plot_cumulative_hazard(show_censors=True)
17 plt.title('Nelson-Aalen Cumulative Hazard Function')
18 plt.xlabel('Time')
19 plt.ylabel('Cumulative Hazard')
20 plt.show()
21
22 # Estimating the cumulative hazard at given time point
23 time_point = 50
24 cumulative_hazard_at_time_point = naf.cumulative_hazard_.loc[time_point].iloc[0]
25 print(f"The estimated cumulative hazard at time {time_point} is: {cumulative_hazard_at_time_point}")
26
27
```

Like the Kaplan-MeierFitter class, the NelsonAalenFitter class is imported in the first line. Each method in this class works exactly like the KaplanMeierFitter class. So I did not describe it again here.

We will see the same thing in the code of the Supervised Machine Learning model. There too the code of each model is identical except for one place.

Below is the output of the above program:



That is, according to our sample life data table (code snippet line 3), the value of Cumulative Hazard ($H(t)$) at 50 seconds is 1.20. Or, the event rate for the first 50 seconds is 1.20.

Evaluation

Part A. A zombie virus has been spreading in a city. You need to estimate what is the possibility of not turning into a zombie after 30 seconds of being bitten. The life data table is given below.

```
data = {'time': [10, 10, 10, 10, 10, 20, 20, 20, 20, 20, 20, 30, 30, 30, 30, 30, 40, 40, 40, 40, 40, 50, 50, 50, 50, 50],
        'event': [0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

Part B. The medical scientists soon invented a medicine that can slow down the zombie-turning process. The medicines were applied to the victims just after they were bitten by the zombies. The life dataset of the victims on whom the medicines are applied is given below.

```
data = {'time': [10, 10, 10, 10, 10, 20, 20, 20, 20, 20, 20, 30, 30, 30, 30, 30, 40, 40, 40, 40, 40, 50, 50, 50, 50, 50],
        'event': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0]}
```

Assess the medicines by statistical analysis.

Solution Hint

- For part A, simply calculate the value of $S(t)$ at 30s.
- Part B is a bit tricky. Calculate the value of $S(t)$ at 10s, 20s, 30s, 40s and 50s before medication. Then, calculate the same after medication and list it as well. After this, perform the student's t-test on these lists and get the p-value. If the p-value is equal to or less than 0.05, that means there is significant difference in the mean values of the probabilities. Therefore, the medicine can be applied.

Solution link:

https://colab.research.google.com/drive/1eVi82IpVZzyVcrsxem_rP-pbRcYdNIqL?usp=sharing

4. Bivariate Predictive Modelling

In statistics, we sometimes have to predict what will happen in the future. These estimates are made through different types of statistics or mathematical models. For example, every machine learning model is a predictive model. Because these models find patterns in large amounts of data and make a decision based on that. In this chapter we will learn about three popular mathematical predictive models. These are: Exponential Growth-Decay Model, Linear Regression, and Polynomial Regression. As these models are deployed on two variable datasets, they are also called Bivariate Predictive Models.

4.1 Exponential Growth-Decay Model

Exponential Growth-Decay Model A type of mathematical model that deals with how something will grow or decline over time. It is expressed by the following equation:

$$p(t) = p_0 \cdot e^{kt}$$

Where, $p(t)$ = Number of population at t times.

p_0 = Number of population in the beginning when $t = 0$

e = constant with a value of 2.71828183

k = relative growth rate

t = time

Three questions can be answered with this model.

1. At what rate is the number growing? (relative growth rate)
2. What can the number be at any given time? (population estimation)
3. How long will it take to reach a certain number by increasing or decreasing the number?

An example will make the point clear.

The following dataset contains the year and the population of Bangladesh in that year.

Year	Population (million)
2014	156
2015	158

2016	160
2017	162
2018	163.7

- What is the relative growth rate of the population?
- What is the estimated population of Bangladesh in 2024?
- When will the population of Bangladesh double ?

Solution

a. We know, $p(t) = p_0 \cdot e^{kt}$

Given that, $p_0 = 156$

From the dataset, if $t = 1$, then $p(t=1) = 158$ (Note that, we can take $t = 1, 2, 3, \dots, n$ anything we want. If $t = 2$ then $p(t=2) = 160$)

Now, We know, $p(t) = p_0 \cdot e^{kt}$

$$\Rightarrow 158 = 156 e^{k \cdot 1}$$

$$\Rightarrow 1.01 = e^k$$

$$\Rightarrow \ln(1.01) = \ln(e^k)$$

$$\Rightarrow \ln(1.01) = \ln e \cdot k$$

$$\Rightarrow k = \ln(1.01) \quad [\text{since } \ln e = 1]$$

$$\Rightarrow k = 9.95 \times 10^{-3} \text{ (Answer)}$$

b. $t = 2024 - t_0 = 2024 - 2014 = 10$

$$\text{Hence, } p(10) = 156 * e^{k \cdot 10} \quad [k = 9.95 \times 10^{-3}]$$

$$= 172.32 \text{ million (Answer)}$$

(If you google it, you will see that the population of Bangladesh is about 174 million in 2024. That is, our estimation is much closer. If the dataset was a little bigger, the estimation might be more accurate. But the big thing is that the population of Bangladesh is still growing at about 1.01 relative growth rate which is quite dangerous for our country, I think.)

c. Given that, $p(t) = 156 * 2 = 312$

$$p_0 = 156$$

$$k = 9.95 \times 10^{-3}$$

From the growth-decay equation, we get,

$$312 = 156 e^{k.t}$$

$$\Rightarrow 2 = e^{k.t}$$

$$\Rightarrow \ln 2 = 9.95 \times 10^{-3} t$$

$$\Rightarrow t = 69.6 \text{ years}$$

So, the population will be doubled in year $(2014+69) = 2083$ (Answer)

That is, if the population of Bangladesh increases at a 1.01 relative growth rate, the population of this country will double in 2083 after about 59 years from today.

Below is an implementation of the EGD (Exponential Growth Decay) Model using Python. Note that I have programmed this model by myself without any library and only put the value of k to two places after the decimal point. Due to this, the value of $p(t)$ or predicted value may change considerably. So if you want you can take the value of k whole or a few more places after the decimal. In that case, change my code to $k = \text{temp}/t$.

```
import math
def EGD_Model(X, Y, x_value) :
    p0 = Y[0]
    pt = Y[-1]
    quotient = pt/p0
    temp = math.log(quotient)
    t = len(X) - 1
    k = round(temp/t, 2)
    t1 = x_value - X[0]
    pt1 = p0 * math.e**(k*t1)
    estimation = round(pt1, 2)
    return estimation
EGD_Model([2014, 2015, 2016, 2017, 2018], [156, 158, 160, 162, 163.7], 2024)
```

আউটপুট

172.41

4.2 Simple Linear Regression

If there is a linear relationship (linear relationship/linear association) between two variables, then what will happen in the future can be estimated using the Simple Linear Regression Model. This model has a dependent and an independent variable. A change in the value of the Independent Variable results in a change in the value of the Dependent Variable. The predicted future value is actually the value of the Dependent Variable that is substituted for the Independent Variable. The relationship between these two variables is expressed by the following equation. It is through this equation that the future data is predicted by finding the pattern between the dependent and independent variables.

$$Y = mX + b$$

Here, Y = Dependent variable

m = slope of line of best fitting

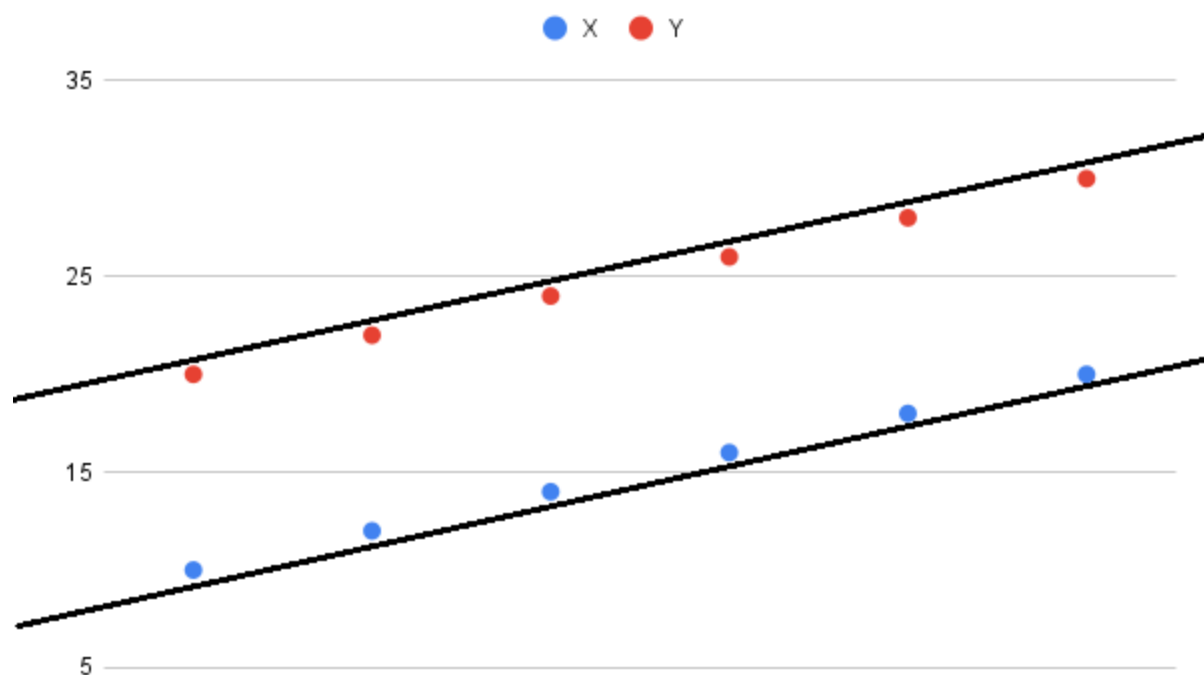
X = Independent variable

b = y-intercept

The equation sounds very familiar, doesn't it? Because it is the equation of the same straight line.

There are some reasons behind this.

When building a predictive model with SLR, the data points for the dependent and independent variables are plotted in a scatter plot. Lines are then drawn from the y and x axes such that most of the scatter plotted data points fall between those two lines. Drawing this line is called best fitting. The better we can fit the data points, the better our SLR model will perform. Now if the linear relationship between the two variables is not that good, then we can't do the best fit as well. Then our model will not give a more accurate prediction. An example below will give a better understanding of the issues.



This scatter plot shows an increase in the value of Y along with the value of X.

X	Y
10	20
12	22
14	24
16	26
18	28
20	30

That is, where X is the independent variable and Y is the dependent variable. The increase in quality is also completely linear. The value is increasing by 2 each time. This is exactly why the line drawn covering the X and Y data points in the scatter plot is the best fit. No data points were omitted. Now if

we want to know that if the value of X is 22, what will be the value of Y then we can use the Simple Linear Regression Model.

$$Y = mX + b$$

$$m = \frac{n(\Sigma XY) - (\Sigma X)(\Sigma Y)}{n(\Sigma X^2) - (\Sigma X)^2}$$

$$b = \frac{(\Sigma Y) - m(\Sigma X)}{n}$$

$$\text{Then, } \Sigma X = 10 + 12 + 14 + 16 + 18 + 20 = 90$$

$$\Sigma Y = 20 + 22 + 24 + 26 + 28 + 30 = 150$$

$$\Sigma XY = (10 \cdot 20) + (12 \cdot 22) + (14 \cdot 24) + (16 \cdot 26) + (18 \cdot 28) + (20 \cdot 30) = 2320$$

$$\Sigma X^2 = 10^2 + 12^2 + 14^2 + 16^2 + 18^2 + 20^2 = 1420$$

$$\text{So, } m = \frac{6(2320) - 90 \cdot 150}{6(1420) - (90)^2} = 1$$

$$b = \frac{150 - 1(90)}{6} = 10$$

Now, if $x = 22$, then,

$$Y = 1 \cdot 22 + 10 = 32 \text{ (Ans)}$$

Consider another example.

X	Y
2	4
4	8
6	16
8	32
10	64

If $x = 20$, then $y = ?$

Solution

$$Y = mX + b$$

$$m = \frac{n(\sum XY) - (\sum X)(\sum Y)}{n(\sum X^2) - (\sum X)^2}$$

$$b = \frac{(\sum Y) - m(\sum X)}{n}$$

Then, $\sum X = 2+4+6+8+10 = 30$

$\sum Y = 4+8+16+32+64 = 124$

$\sum XY = (2*4) + (4*8) + (6*16) + (8*32) + (10*64) = 1032$

$\sum X^2 = 2^2 + 4^2 + 6^2 + 8^2 + 10^2 = 220$

$$\text{So, } m = \frac{5(1032) - 30*124}{5(220) - (30)^2} = 7.2$$

$$b = \frac{124 - 7.2(30)}{5} = -18.4$$

Now, if $x = 22$, then,

$$Y = 7.2 * 20 - 18.4 = 125.6 \text{ (Ans)}$$

But here we have gone wrong. Although the value of Y increases with the value of X, the difference between the data points between X and Y is not constant. In a linear relationship, this difference must be constant. As in the first example our difference was 10. But here the difference is first 2, then 4, then 8 and so forth.

So here X and Y are not actually in a perfectly linear relationship, although the value of Y increases with the value of X.

Therefore, before using a model, it is necessary to understand whether it should be used at all. Because the prediction model will always give, but which model gives more accurate prediction is the subject for consideration.

Also, sometimes we may need to use the Simple Linear Regression model even though X and Y are not completely linearly related. For example, the population example we looked at in the last chapter's Exponential Growth Decay Model, SLR can be used even though it is not a perfectly linear relationship. In real life we will rarely see a perfectly linear relationship, in most cases the difference between the data points of X and Y will not be completely constant. There may be slight variations in some points. Then we have to consider wisely whether to use the SLR model or not.

If we also calculate the population prediction with the SLR Model then the population in 2024 will be 175.46 M. On the other hand, the EGD Model predicted 172.32 M. In reality the population in 2024 is 174M. That is, the error of the EGD Model is 1.68 and the error of the SLR Model is 1.46. So, since the SLR Model has less error, it would be better to use the SLR Model rather than the EGD Model for population estimation (that is, for solving our particular problem), even though there were slight

differences in the Year and Population data points and the Exponential Growth-Decay Model is a widely used model in population estimation. Again, sometimes there may be a situation where the error of SLR Model is coming more and the error of EGD Model is coming less. Therefore, we cannot judge which model is more effective by looking at two figures. Below is how to create and use SLR Model with the help of programming:

```
import numpy as np
from sklearn.linear_model import LinearRegression

# Given data
years = np.array([2014, 2015, 2016, 2017, 2018])
population = np.array([156, 158, 160, 162, 163.7])

# Reshape the data to meet the requirements of the Linear Regression model
years = years.reshape(-1, 1)

# Create and fit the Linear Regression model
model = LinearRegression().fit(years, population)

# Predict the population for a future year (e.g., 2025)
future_year = np.array([[2024]])
predicted_population = model.predict(future_year)

print("Predicted population in 2024:", predicted_population[0])
```

Output

```
Predicted population in 2024: 175.46000000000004
```

4.3 Polynomial Regression

If the SLR fails to best fit, then such a situation is called under-fitting. Thus the Polynomial Regression model can be used to best fit under-fitting. Linear Regression Model can give good and accurate predictions only about the variables which are in linear relationship. But when the relationship becomes non-linear it gives wrong predictions. To solve such problems we will use Polynomial Regression which can handle both non-linear and linear relationships well. So, it is better to use PR Model instead of SLR Model while programming. In fact, by looking at the program data, it will be understood whether there is a PR model or SLR model. When we read SLR we saw that the equation

of that model is $Y = mX + b$, where the power of X is 1. That is why it is called the Simple Linear Regression Model or Linear Polynomial Regression Model. In fact, it is part of Linear Regression and Polynomial Regression. If the power of x was 2, then we would call it the Quadratic Polynomial Regression Model whose equation would be $Y = aX^2 + bX + c$. Thus, the equation will also change with the value of X . The general equation of Polynomial Regression is:

$$y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$$

We cannot do the problems related to Polynomial Regression easily in pen and paper. We will do this using Python. For that, first of all we need to import the required libraries.

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

Here Polynomial Feature from sklearn.preprocessing library and Linear Regression from sklearn.linear_model library are imported which will be useful to create Polynomial Regression Model.

```
# Given data
years = np.array([2014, 2015, 2016, 2017, 2018])
population = np.array([156, 158, 160, 162, 163.7])

# Reshape the data to meet the requirements of the Linear Regression model
years = years.reshape(-1, 1)

# Transform the features to include polynomial terms up to degree 2
poly_features = PolynomialFeatures(degree=2)
years_poly = poly_features.fit_transform(years)

# Create and fit the Polynomial Regression model
poly_model = LinearRegression().fit(years_poly, population)

# Predict the population for a future year (e.g., 2028)
future_year = np.array([[2024]])
future_year_poly = poly_features.transform(future_year)
predicted_population = poly_model.predict(future_year_poly)
```

```
print("Predicted population in 2024 (Polynomial Regression):",
predicted_population[0])
```

The above program predicts the population in 2024 using the Polynomial Regression Model.

Output:

```
Predicted population in 2024 (Polynomial Regression): 172.80285714013735
```

So, our model error is (174-172.80) or 1.19. Above we have seen that the error of the EGD Model is 1.68 and the error of the SLR Model is 1.46. Therefore, the best model for this specific problem of population prediction is the PR Model. As its error is less than the other two models. In this way we will apply 2/3 models before making any kind of prediction in real life and see which model has more accuracy or less error. After this model evaluation process, we will use the model that has less error or more accuracy.

Anyway, a more generalized version of the PR Model program above is given below. It will predict the value of Y based on any value of X related to Polynomial Regression.

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

def polynomial_regression(X, Y, x_value) :
    X = np.array(X)
    Y = np.array(Y)
    X = X.reshape(-1,1)
    poly_features = PolynomialFeatures(degree=2)
    years_poly = poly_features.fit_transform(years)
    poly_model = LinearRegression().fit(years_poly, population)
    future = np.array([[x_value]])
    future_poly = poly_features.transform(future)
    prediction = poly_model.predict(future_poly)
    return prediction[0]

polynomial_regression([2014,2015,2016,2017,2018],[156,158,160,162,163.7],2024)
```


4.4 Model Evaluation

During predictive modeling we have to make judgments about which model to use at which time. Above we have learned about three types of predictive models that indicate the relationship of one variable to another. Now, all these three models can predict future outcomes. So do we just learn any model and deploy it again and again? The answer is, no. Because, not all models always predict accurately. Then it can be seen that another model beat the previous model in the game of accuracy. In such cases we may need to use the second model. For example, the MSE of the EGD Model is lower than other models in terms of population prediction, while the MSE of the EGD Model may be higher than SLR or PR Model when predicting the profit of a company. Note that the full form of MSE is Mean Squared Error. It is a useful method to compare the accuracy of predictive models. Among multiple models, the model with lower MSE is considered more accurate.

4.4.1 Mean Squared Error

MSE is one of the most important methods in determining the error of a predictive model. At the same time, comparison between multiple models is also possible with this method. Suppose, we use a predictive model (e.g. EGD Model) to estimate the profits of a company in February 2023 as 10000, 15500, 16000, and 18000 for the months of May, June, July, and August. In the month of September we noticed that the actual profits were 9500, 15000, 16400 and 18050. that is,

```
predicted_values = [10000, 15500, 16000, 18000]
original_values = [9500, 15000, 16400, 18050]
```

Notice that the original values differ from what we predicted. The difference between the original value and the predicted value is called the residual error.

There will always be some residual error in our model and it is normal to have this error, nothing to be alarmed about. There are many ways to measure these errors. For example, in the previous examples we calculated the residual error. Now we will learn a new method to determine the error, which is called Mean Squared Error (MSE). By analyzing the average value of Residual Error it gives an idea about the accuracy of a model.

Figuring out MSE is very easy. The absolute value of the difference between each predicted value and the original value should be averaged. In simple words, the average of the absolute value of each

residual error is Mean Squared Error. It is also called Mean Squared Deviation. The reason for taking the absolute value is that if the difference becomes negative, then if we take the mean, its value will also be less. Then even an incorrect model will have lower MSE.

The MSE of the EGD Model for the above problem is calculated below.

Given, predicted_values = [10000, 15500, 16000, 18000]
 original_values = [9500, 15000, 16400, 18050]
 so, differences = [(10000-9500), (15500-15000), (16000-16400), (18000-18050)]
 absolute_differences = [500, 500, 400, 50]
 mean_squared_error = (500+500+400+50)/4 = 1450/4 = 362.5

So, the MSE of the company's EGD Model in determining profitability is 362.5

Now, let's say another data scientist estimates this same profit using the PR Model. His prediction was like this:

predicted_values = [9500, 14900, 16400, 18060]
 original_values = [9500, 15000, 16400, 18050]
 So, MSE of PR Model = (0+100+0+100)/4 = 200/4 = 50

It is seen that the MSE of the Polynomial Regression Model is less than the MSE of the Exponential Growth-Decay Model. Therefore, the company should use the PR Model in determining profitability. Because the residual error of this model is less.

If we are ever asked to compare between two or more models, we can use MSE. The model with the lowest MSE predicts better than the other models.

MSE can be calculated with the help of the mean_squared_error function of the sklearn.metrics library of Python.

```
import numpy as np
from sklearn.metrics import mean_squared_error

# Example data
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]

# Calculate MSE
```

```
mse = mean_squared_error(y_true, y_pred)

print("Mean Squared Error:", mse)
```

Output

```
Mean Squared Error: 0.375
```

Project

You are inspired to add your projects in your data portfolio

1. The following drive link accesses you to a .csv file containing the dataset of year of experience and salary. Your tasks are-

Part B: You need to build a predictive model for predicting future salaries based on the inputs of years of experience.

Part A: To do Part A, you need to first of all figure out which model has the smallest MSE. Deploy EGD, SLR, and PR Model to find out the predicted salaries for the year of experiences [4.0, 4.1, 4.2, 4.6]. Then, compare the predictions with the original values to calculate MSE. Model with the lowest MSE is considered as the best model. Therefore, use that model to complete Part B.

Dataset link :

https://drive.google.com/file/d/1nxxisBXEkaXEqZcKrsfLvyOKB6WMgChN/view?usp=drive_link

For any sort of confusions, feel free to take help from the solutions.

Solution link :

https://colab.research.google.com/drive/1QT6pyhIDQ-sMHCnpp88AaG_OzFAawybw?usp=sharing

5. Multivariate Predictive Modelling

In the last chapter we read about three types of predictive models. These are Exponential Growth-Decay Model, Simple Linear Regression Model and Polynomial Regression Model. Notice that these models had two variables: Dependent and Independent variables. That is, the datasets we used for those models were Bivariate datasets. In this chapter we will learn about the Supervised Machine Learning Models where there will not be only two variables but there can be countless variables here. Supervised Machine Learning Models are deployed on multivariate dataset. Variables are

called “features”. A Supervised ML Model predicts future outcomes by analyzing the data points of many features. We will discuss these issues in detail soon.

Machine learning models are a subset of artificial intelligence. In other words, the foundation of artificial intelligence was built on machine learning. The behavior of today's “human-like” chatbots can also be explained with machine learning. Before discussing machine learning in detail I will talk about philosophy for a moment. Philosophy of how people think.

Recently University of Pennsylvania’s cognitive science professor Noam Chomsky wrote that the difference between humans and AI is that humans have creativity but chatbots do not. Chatbots consume huge amounts of data and retain it, and based on the user’s questions, it provides the necessary data through predictive modeling. That is, ChatGPT is a statistical model that copies other people's data into itself and then displays it to the user. So, according to him ChatGPT is not really artificial intelligence, but it is a pleasure machine.

But one thing he missed was how people learn or become creative. When we reach the age of five or six, and sit down with our parents to do math for the first time, they teach us some rules of addition and subtraction. After we memorize or learn these rules, we have no difficulty in adding and subtracting. Here, the approach parents teach us to add and subtract is “data”. These data are processed and stored by our brain, and used later as needed. Not only this addition and subtraction, everything we do in every moment of our life, whatever experience or knowledge we gain is all data that our brain stores and we make all our decisions based on them. Our every moment decision, activity is determined by our pre-experience which is nothing but data. You are even reading this article of mine under the influence of some data. For example: “Data science is an important topic nowadays and there is a lot of demand for this topic” - you know this data and you decided to read this self-learning text on data science and data analysis.

That is, if humans can process thousands of data and make decisions, then if the same power is given to a machine, it can think like a human, is it unreasonable? No, and the same is done with ChatGPT or other large language models. First, they are fed thousands of data and with the help of predictive modeling or machine learning or deep learning algorithms, they predict the data, i.e., answer a user's query. In this way, generative AI models are developed in accordance with human learning patterns, which are based on machine learning.

5.1 Supervised Machine Learning Models

Models that predict outcomes based on specific dataset features are called Supervised ML Models. Supervised ML models can be of different types depending on the algorithm. For example: kNN, Logistic Regression, Gaussian Naive Bayes etc.

5.1.1 ML Terminologies

Feature: We know that multivariate dataset has more than one column, which is also called variable sometimes. These "columns" or "variables" are features. But there are some exceptions. The following example will make it clear.

Student ID	Faculty Initial	BP (systolic)	Q. Difficulty	Mark
634	XYZ	120	Very Hard	70
725	ABC	90	Easy	90
124	KLM	110	Medium	83
990	KLM	106	Medium	80
675	ABC	90	Easy	87
780	KLM	114	Hard	74
220	XYZ	118	Hard	72
477	XYZ	120	Very Hard	67
410	ABC	94	Easy	89

Here, basically "Student ID", "Faculty Initial", "BP (systolic)", "Q. Difficulty", "Mark" each column can work as a feature. We have to see what our model predicts. If we want to predict the "Mark" of a new student based on the data points in the "Student ID", "Faculty Initial", "BP (systolic)", "Q. Difficulty" columns whose information is not in the above dataset, then the model we I will create there "Student ID", "Faculty Initial", "BP (systolic)", "Q. Difficulty" these will be features (X). On the other hand, the "Mark" column will be the output column (y). That is, we are deriving the value of y based on the value of X.

So, how do supervised machine learning models predict output from features?

Much like how people predict output. The above table shows the data of 9 students. Now if I talk about a new student whose ID is 881, the student's faculty initial is XYZ, his blood pressure during the exam was 116 and his question standard was Hard. But I will not tell how many marks he got. You have to guess that. Can you guess approximately how many marks he can get?

How do you find the answer to this question? You will see from the given dataset of 9 students whose questions were difficult, how they scored, then you will see who their faculty was and finally you will see what their blood pressure was in the exam hall. After these analyses you can understand from the dataset that marks of such students are 70, 72, and 67 respectively. Having recognized this pattern in the dataset, now if you average these three marks, you can estimate that the marks of the student of ID 881 will be close to that average.

Models work much the same way. Recognizes patterns of features in a dataset and predicts an output based on that pattern. These predictions can be divided into two categories based on the type of statistical data.

- Classification: If the predicted output is binary type or ordinal data then that prediction is called classification. If the model gives a “Yes” or “No” type of prediction, that is, if there are no other data points except two data points in the output column of the dataset, the prediction obtained from the output column, or the prediction obtained from the model, is called **Binary classification**. On the other hand, if the model gives predictions like ordinal data type, i.e. if the output column is composed of ordinal data, then the prediction obtained from that model is called **multi-class classification**. For example, a multi-class classification model is used to predict human height details (e.g. short, medium, tall) based on certain features.
- Regression: When the model predicts continuous data type it is called Regression. Eg: Prediction of temperature, numerical height, blood pressure etc. are examples of Regression.

Now, how do the models predict? The model splits the entire dataset into two parts. The first part is the training dataset, the second part is the testing dataset. The training dataset is a large part of the entire dataset from which the model learns the patterns of features and output for each of those feature vectors. The features of the training dataset and the output column are called *X_train dataset*, *y_train dataset* respectively. On the other hand, the features and output columns of the Testing dataset are called *X_test dataset* and *y_test dataset* respectively. (You can change these names if you want while

coding). For example: 634, XYZ, 120, Very Hard in the first sample or record of our above dataset are feature vectors, and 70 is output. Conventionally, Training dataset is created with 80% samples of the dataset and Testing dataset with remaining 20% samples. In that case, the features of the first 7 students in our dataset are $X_{train\ dataset}$, the outputs associated with these features are $y_{train\ dataset}$. The features of the dataset of eighth and ninth grade students are $X_{test\ dataset}$ and the outputs are $y_{test\ dataset}$.

So, by recognizing patterns from the training dataset, the model predicts the output of the testing dataset. Then we check the accuracy of the model by comparing the predicted output with the original output. If the accuracy is good then the model is usable.

But since it is not possible for machines to think as easily as humans, machines recognize this pattern with the help of different statistical or mathematical models based on which different Supervised ML Algorithms have been created. We will learn about some popular algorithms in this chapter. These are: K-Nearest Neighbor (KNN), Naive Bayes, Logistic Regression, Support Vector Machines (SVM), Decision Tree, and Random Forest.

5.1.2 Model Structure

Supervised machine learning models are basically objects of certain classes. That is, the models are created following the Object-oriented programming paradigm. Therefore, these models have some methods with which the models can predict future outcomes from existing data. These methods are: **fit()** : The X (features) and y (output variable) variables are separated from the DataFrame and sent inside this method. This method trains, or prepares, the model by running different code inside the x and y variables separately.

predict() : predicts the future outcome following the specified algorithm.

Project Walkthrough

Since machine learning is a bit more complex than the topics in the previous chapters and there are many things to discuss here, we will try to learn the details of machine learning through projects here. It will be easy to understand. You can add this project to your data portfolio.

Project : Lung Cancer Detection using Supervised ML Models of 80%+ accuracy

Dataset:

<https://drive.google.com/file/d/11KtEiJHGFZdr3f64TZOUSNs4YZBxaKAZ/view?usp=sharing>

Dataset Information: This dataset contains records of people with different characteristics who have cancer and who do not. This dataset has 16 columns. These are: GENDER, AGE, SMOKING, YELLOW_FINGERS, ANXIETY, PEER PRESSURE, CHRONIC DISEASE, FATIGUE, ALLERGY, WHEEZING, ALCOHOL CONSUMING, COUGHING, SHORTNESS OF BREATH, SWALLOWING DIFFICULTY, CHEST PAIN, AND LUNG_CANCER. That is, there is information that people with certain characteristics have been diagnosed with lung cancer. Among them, the columns except LUNG_CANCER will be taken as features, and LUNG_CANCER will be taken as the output column. So, the job of the model we'll build is to take the first 15 columns of data for a person and then predict whether or not they have cancer.

Plan: In this project we will diagnose lung cancer (y) using different features (X). That is, our output will be a classification, since we will not get anything other than “Yes” or “No”. Also, we will not only rely on one particular model, but deploy multiple models to see which model gives more accurate results.

Please note: How the models mathematically predict the output will not be discussed in detail here to avoid time constraints and complications. Our main focus will be on programming and deploying models.

So let's begin. We will first detect cancer using the K-Nearest Neighbor (KNN) Classifier model. It is a famous classification model that plots the feature values initially and predicts the value closest to the feature input as the output using the Euclidean distance formula.

At the beginning of deploying any ML Model, we need to import two modules and their two functions first. Modules are scikit learn library `model_selection` and scikit learn library `metrics`. `train_test_split` and `accuracy_score` functions should be imported from these modules respectively. By the first function we can divide the dataset into training and testing dataset. By checking the original value and predicted value, the accuracy score can be calculated by the second one. Alternatively, mean squared error, R-squared, silhouette score can also be used. But here we will use the `accuracy_score()` function.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```


After uploading and displaying the dataset, we can see that every column has a quantitative datapoint, except for two columns. They are the GENDER column and LUNG_CANCER column. The data points for GENDER are M and F. We humans can see M and F and understand that they refer to male and female respectively. But the computer will not understand that. Because computers only understand numbers. So what we have to do before deploying the model is to preprocess the data. Numeric the data points in each column and correct any errors in the dataset.

`LabelEncoder()` is used to encode the characters in the dataset. Also an easy way is to express the numbers manually. Running the following code will replace M with 1 and F with 0 in the GENDER column of our dataframe (df). On the other hand, “YES” of LUNG_CANCER will be replaced by 1 and “NO” by 0.

```
df['LUNG_CANCER'] = df['LUNG_CANCER'].replace({'YES':1, 'NO':0})
df['GENDER'] = df['GENDER'].replace({'M':1, 'F':0})
```

Now, our dataset may contain duplicate samples, and some data points may contain null values. If we don't clean them, our model will not always give correct output. Accuracy will be very less.

```
df = df.drop_duplicates()
```

Now it is possible to build a machine learning model with our dataset. We will start with the KNN Classifier Model.

```
from sklearn.neighbors import KNeighborsClassifier

# Select feature and output
X = df.drop(columns='LUNG_CANCER')
y = df['LUNG_CANCER']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a KNN classifier model
knn = KNeighborsClassifier()

# Train the KNN classifier
```

```
knn.fit(X_train, y_train)

# Make predictions on the test set
y_pred = knn.predict(X_test)

# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Code Analysis

```
from sklearn.neighbors import KNeighborsClassifier
```

The neighbors module is imported from the scikit learn library through this line. Then a class called KNeighborsClassifier has been imported from that module. When we create an object of this class, it will be our KNN Model.

```
X = df.drop(columns='LUNG_CANCER')
y = df['LUNG_CANCER']
```

Here, X is the features and y is the output column. Since we will be predicting cancer based on features, we split our dataset into two parts. That is, we removed the LUNG_CANCER column from our df dataset, created a new dataset and stored it in the X variable. This dataset of X variables will serve as our feature dataset. On the other hand, I stored only the LUNG_CANCER column in the y variable. This is our output column.

Note that if we need to exclude more columns for the X variable, then columns= should be followed by the names of those columns. For example:

```
X = df.drop(columns= ['col A', 'col B', 'col C'])
```

But we don't need it for this project.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

We then further divided our x and y datasets, i.e. the feature dataset and the output dataset into four parts and stored the means of the following variables.

- (i) X_train = dataset constructed from the first 80% samples of the feature dataset.
- (ii) y_train = dataset constructed from the first 80% samples of the output dataset.

- (iii) `X_test` = dataset constructed from the remaining 20% samples of the feature dataset.
- (iv) `y_test` = dataset constructed from the remaining 20% sample of the output dataset.

If you pass 0.3 instead of 0.2 argument to the `train_test_split()` function's `test_size` parameter, then the training-testing ratio would not be 80:20, but 70:30. But by convention the ratio should be 80:20.

```
knn = KNeighborsClassifier()
```

An object is created from the `KNeighborsClassifier` class. This object is stored in a variable called `knn`. Now this `knn` object is our KNN CLASSIFIER MODEL.

Congratulations! By writing this line we have created our first supervised machine learning model. Here's what happened,

I wrote earlier that `knn` is an object created from `KNeighborsClassifier`. We know that every object has some methods with the help of which the object performs different types of operations. One such method is `fit()`. We have passed the `x_train` and `y_train` dataset to the parameters of this `fit()` method. `X_train` contained 80% of the feature dataset and `y_train` contained the first 80% of the output dataset. Now passing these datasets in the `fit()` method, various KNN Algorithm related codes will be executed in the `fit()` method. This will cause our model (`knn` object) to capture the pattern of the dataset. Using this pattern he can predict future outcomes.

```
y_pred = knn.predict(X_test)
```

Do you remember? Did we store the last 20% of the feature dataset in the `X_test` variable? This `X_test` dataset was not used in model building. Consequently, this dataset is unknown to the model. So we will now pass the `X_test` dataset into the model's `predict()` method. Once this is sent, the model will predict the corresponding output based on the data pattern (that it learned a while ago). The output of each sample of `X_test` will be stored in the `y_pred` variable as an array.

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Our model output is predicted and stored in `y_pred`. Meanwhile, we stored those same outputs (original values) in the `y_test` variable before building the model. So, now if we pass `y_test` and

`y_pred` as arguments to the `accuracy_score()` function imported from `sklearn.metrics`, it will compare the elements of these two arrays. Suppose, `y_test` (original values) is in `[0,0,1,0,1,1,0]` and `y_pred` (predicted values by model) is in `[1,0,1,0,1,1,0]`. So, the accuracy of the model will be 0.85 or 85%. Because, out of 6 outputs, the model predicted 5 outputs correctly and 1 incorrectly.

So, the code of our KNN Classification Model is finished. Now if we run it then we can see the accuracy of this model. If the accuracy is 0.8 or more then we can use the model for lung cancer detection. Otherwise, the matter will become more risky. Especially in the medical sector it is more sensitive to use low accuracy models.

```
Accuracy: 0.7857142857142857
```

After running the code the accuracy is 78%. Less than 80%. So using this model is a bit risky. That's why we will deploy other models and see how the accuracy comes. The source code of the other models will also be exactly the same, just a few changes in some places. You will understand this by yourself if you see the programs.

Below is the accuracy test of Naive Bayes, Logistic Regression, SVM, Decision Tree Classifier and Random Forest models.

```
from sklearn.naive_bayes import GaussianNB

X = df.drop(columns=['LUNG_CANCER'])
y = df['LUNG_CANCER']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

NB = GaussianNB()

NB.fit(X_train, y_train)

predictions = NB.predict(X_test)

accuracy = accuracy_score(y_test, predictions)

print(f'Accuracy: {accuracy:.2f}')
```

Output

Accuracy: 0.95

```
from sklearn.linear_model import LogisticRegression

X = df.drop(columns=['LUNG_CANCER'])
y = df['LUNG_CANCER']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

LogR = LogisticRegression()

LogR.fit(X_train, y_train)

predictions = LogR.predict(X_test)

accuracy = accuracy_score(y_test, predictions)

print(f'Accuracy: {accuracy:.2f}')
```

Output

Accuracy: 0.89

```
from sklearn.svm import SVC

X = df.drop(columns=['LUNG_CANCER'])
y = df['LUNG_CANCER']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

svm_classifier = SVC()

svm_classifier.fit(X_train, y_train)

y_pred = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

Output

Accuracy: 0.8214285714285714

```

from sklearn.tree import DecisionTreeClassifier

X = df.drop(columns=['LUNG_CANCER'])
y = df['LUNG_CANCER']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

DTC = DecisionTreeClassifier()

DTC.fit(X_train, y_train)

predictions = DTC.predict(X_test)

accuracy = accuracy_score(y_test, predictions)

print(f'Accuracy: {accuracy:.2f}')

```

Output

Accuracy: 0.79

```

from sklearn.ensemble import RandomForestClassifier

X = df.drop(columns=['LUNG_CANCER'])
y = df['LUNG_CANCER']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

RF = RandomForestClassifier()

RF.fit(X_train, y_train)

predictions = RF.predict(X_test)

accuracy = accuracy_score(y_test, predictions)

print(f'Accuracy: {accuracy:.2f}')

```

Output

Accuracy: 0.89

It appears that the Naive Bayes model has the highest accuracy, 0.95. This is more than our desired accuracy (80%). So we can take this model for lung cancer detection. It uses 15 pieces of information about a patient or person to tell whether the person is at risk of cancer or not. As a result, the person can become aware earlier and can protect himself from getting cancer. Since the chance of survival after cancer is very low, the only way is to detect the cancer before it occurs or to detect the possibility. Since the accuracy of our model is 0.91, if our model gives a positive result to a person in cancer detection, it means that the probability of that person being affected by cancer is 0.91. Then he can be aware of his disease and save himself from getting cancer through proper treatment. Thus, our model can play an important role in the medical field. Note that this project is similar to a replication study of some machine learning research papers. That is, a lot of research has already been done on this. So no one should be inspired by this project idea and write a research paper on it.

But there is one more thing about accuracy, it fluctuates. A model that gives 0.95 accuracy the first time, the accuracy may change again the next time the same program is run. However, this change is not much so there is not a lot to worry about.

Well, now let's see how we will use our model. Let's show an example. Since we have selected the NB model, we will show the example with it.

Assume, our model is being used in the Medical Unit of Pacifica Corporation. Now here a person comes to get tested to see if he is at risk of cancer. We got the following feature vectors from her.

Gender = Female (0)

Age = 59

Smoking = Yes (2)

Yellow fingers = Yes (2)

Anxiety = Yes (2)

Peer pressure = No (1)

Fatigue = Yes (2)

Allergy = Yes (2)

Wheezing = Yes (2)

Alcohol consuming = Yes (2)

Coughing = No (1)

Shortness of breath = Yes (2)

Swallowing difficulty = Yes (2)

Chest pain = Yes (2)

Now, we will store these responses in a list or array. Then, inside the `predict()` method of the NB object or model, I will send that array through another array so that the two dimensional array goes inside the `predict()` method. Otherwise, an error will appear in the code. Then running the `cancer_prediction` variable will print [1] or [0]. If [1] occurs, it means that he is at risk of cancer. Conversely, if it comes to [0], it means that he is not at risk of cancer.

```
info = [0, 59, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2]
cancer_prediction = NB.predict([info])
print(cancer_prediction)
```

If a scientist wants to detect the cancer of multiple people or patients at the same time instead of one by one, then we need to create a dataset with the information of all of them. After that, if you pass it to the `predict()` method, it will print who is at risk of cancer and who is not at risk of cancer in a new array.

```
predictions = NB.predict(patients_dataframe)
print(predictions)
```

Project

A dataset is given here:

https://drive.google.com/file/d/1P-OyaL9pnCWlqQdO2MF_nnY0VdeotEV/view?usp=sharing

Create a predictive model that is able to predict a person's sleep apnea with more than 70%+ accuracy. Also, show how to use that model.

Warnings:

- There are duplicate values.
- Some columns contain categorical data.

Preprocessing guidelines :

- Drop the person ID column first, then remove the duplicate rows.

Solution link:

https://colab.research.google.com/drive/1QVphntW6Mr0ij_pPig-K3m3hd7D7Czx?usp=sharing

You can find similar datasets from Kaggle to do more projects, research, and practices. Here's the link : <https://www.kaggle.com/>

Kaggle is a platform for ML engineers and data scientists.

Appendix

Data Portfolio

A data portfolio, also known as a data science portfolio is your personal website dedicated to store your data science projects only. Here you are inspired to put the results and summary of your data projects or research projects. For courses where you do not get any certifications, these courses include several projects to finish, just like this course. After finishing the projects, you should add this to your portfolio. This portfolio is a replacement for certificates, and during your job applications at data analysis, research or data science posts, the hirers might ask to see your data portfolio.

Here's how you can make one.

- Go to the link: [Build your data portfolio. Everything you need to create high quality data projects and showcase them on your own beautiful portfolio website. \(datascienceportfol.io\)](https://datascienceportfol.io/)
- Just sign in and put your projects according to the site. It is that easy! If you need any clues regarding how to arrange your portfolio, feel free to have a visit to my portfolio. [Kazi Sakib Hasan | Independent Researcher and Data Science Enthusiast | Data portfolio \(datascienceportfol.io\)](https://datascienceportfol.io/)

Cleaning Outliers

The following code snippet shows how to remove outliers from a dataset using the John Tukey's IQR formula to remove outliers.

```
import numpy as np

data = [10, 12, 14, 15, 100, 16, 18, 19, 200]
```

```
data = np.array(data)

# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

# Define the lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
filtered_data = data[(data >= lower_bound) & (data <= upper_bound)]

print(filtered_data)
```

filtered_data is the dataset that is the modified version free from the outliers that were existing in data.

The End. I hope you learned something new and useful. :D