ARYA VIDYA MANDIR
UPDATED DATA ANALYSIS
LEVEL 1 & LEVEL 2

LEVEL 1- COURSE - AI/ML WITH DATA ANALYSIS - 12 SESSIONS
LEVEL 2- COURSE - AI/ML WITH DATA ANALYSIS - 12 SESSIONS

LEVEL 1 - AI/ML WITH DATA ANALYSIS

	SESSION NO	Topic name	Concepts learnt
1	**SESSION 1:**	Undertsanding Python analysis packages - NUMPY	1. Introduction to Arrays 2. Arrays vs Lists 3. Creating Arrays using NumPy 4. NumPy Array Methods 5. Dimensions in Arrays 6. Indexing 7. Slicing
2	**SESSION 2:**	Undertsanding Python analysis packages - ADVANCED NUMPY	1. NumPy Data Types 2. Converting datatype of an array 3. Array Reshaping 4. Joining Arrays 5. Array Stacking 6. Filtering NumPy Array
3	**SESSION 3:**	Exploring the World of Images and Data with NumPy and OpenCV	1. NumPy Topics - Copy & View 2. Reshaping & Flattening the NumPy arrays 3. Sorting the NumPy Arrays 4. Introduction to openCV-Python 5. OpenCV Topics - vstack, hstack, indexing, cropping, slicing,filtering
4	**SESSION 4:**	Mastering the Basics of Pandas Series: Handling Data Types, Appending, and Concatenation	1. What is Pandas? 2. Pandas Series 3. Datatype handling in Panda Series 4. Appending in Panda Series 5. Concatenation in Panda Series
5	**SESSION 5:**	Mastering Data Manipulation with Pandas: From Creating DataFrames to Loading and Analyzing CSV and Excel Files	1. Dataset, Series & Dataframe 2. Creating a DataFrame 3. Loading files into a DataFrame 4. Read CSV in Pandas 5. Read Excel in Pandas
6	**SESSION 6:**	Exploring Advanced DataFrame Operations in Pandas: Combining, Merging, Grouping, and Aggregating Data for Better Analysis	1. Filtering data in DataFrame 2. Combining in DataFrame (Joins) 3. Merging in DataFrame 4. Counting in DataFrame 5. Functions in DataFrame (groubpy & filter)
7	**SESSION 7:**	Matplotlib: A Comprehensive Guide to Visualizing Data and Analyzing Relationships A Comprehensive Guide to Plotting Multiple Charts and Adding Grid Lines, Subplots, Titles, and Legends	1. What is matplotlib? 2. Display Multiple Plots 3. Matplotlib Pie Charts 4. Matplotlib Adding Grid Lines 5. Matplotlib Subplot 6. Title 7. Legend function in Matplotlib
8	**SESSION 8:**	Mastering Data Visualization with Matplotlib: Mastering Scatter Plots, Heatmaps, and Confusion Matrices	1. Scatter Plots in matplotlib 2. Scatter Plots using CSV in matplotlib 3. Change marker and marker size using CSV 4. Scatter plot colored by category 5. Markers' Size in Scatter Plot 6. DataFrame in Heatmap using matplotlib 7. Confusion Matrix using matplotlib
9	**SESSION 9:**	Linear Regression and Supervised Machine Learning: Understanding Types, Assumptions, and Real-World Applications with Python Examples	1. Machine Learning - Introduction 2. Types of Machine Learning 3. Types of Supervised Learning: 4. Linear Regression - Introduction 5. Simple Linear Regression 6. Multiple Linear Regression 7. Assumptions in Linear Regression Model
10	**SESSION 10:**	Beyond the Straight Line: Unleashing the Power of Polynomial Regression in Python	1. Polynomial Regression - Introduction 2. Need for Polynomial Regression 3. Python implementation of polynomial regression 4. Construction of polynomial regression model 5. Displaying the polynomial regression result 6. Polynomial Smooth Regression Using CSV with Python 7. Polynomial Regression with Various Polynomial degree ranges

LEVEL 2 - AI/ML WITH DATA ANALYSIS

	SESSION NO	Topic name	Concepts learnt
1	**SESSION 1:**	Understanding Image processing refers to the manipulation and analysis of digital images using various algorithms and techniques. Understanding OpenCV (Open Source Computer Vision Library) is a popular open-source library designed for computer vision and machine learning tasks.	1) Image processing 2) OpenCV
2	**SESSION 2:**	Understanding MediaPipe is a cutting-edge, cross-platform framework for building multimodal perceptual AI pipelines.	1) MediaPipe 3) MediaPipe Face Detection and Tracking 4) MediaPipe Pose Detection and Tracking 5) MediaPipe Object Detection and Tracking 6) MediaPipe Customization and Development 7) Facial landmark detection
3	**SESSION 3:**	Implentation using MediaPipe is a cutting-edge, cross-platform framework for building multimodal perceptual AI pipelines	Project 1 Using the Session 1 & 2 Topics: Project using Mediapipe for instagram filters examples - Develop a Python-based project with Mediapipe for implementation.
4	**SESSION 4:**	Understanding topics in the field of data science for data manipulation, analysis, and visualization.	1) Face recognition library 2) Revisting of NumPy 3) Revisting of Pandas 4) Revisting of Matplotlib
5	**SESSION 5:**	Understanding three of the most widely used and popular libraries for machine learning and deep learning	1) TensorFlow 2) Keras 3) Scikit learn
6	**SESSION 6:**	Implentation using Computer Vision & Face Recognition Library	Intermediate Project 2 : Project using Computer Vision & Face Recognition Library for Student Attendance System - Using Computer Vision & Face Recognition Library for capturing Student Image & Updating Attendance System using the option of "auto click" when video image of student is captured.
7	**SESSION 7:**	Understanding topics in the field of automation and scripting	1) Pyautogui 2) Cursor movement 3) Error handling 4) Input device interface
8	**SESSION 8:**	Implentation using Pyautogui	Intermediate Project 3 : Project using Pyautogui library to move the cursor based on the finger movement detected by an input device. - Using Pyautogui for Color on the Finger and the Cursor moves when the finger moves.
9	**SESSION 9:**	Understanding tools and techniques necessary to build powerful and effective machine learning models.	1) Convolutional Neural Networks (CNNs) 2) Transfer Learning 3) Data Augmentation 4) Image Preprocessing 5) Loss Functions 6) Optimization Algorithms 7) Performance Metrics 8) API Integration with TensorFlow Serving
10	**SESSION 10:**	Implentation using CLASSIFICATION	Intermediate Project 4 : Project using Google Collab use CLASSIFICATION APIs for Rose or Sunflower or Tulip Classification - Create Python project that uses the Google Collab platform to access the pre-trained models and online Classification Model APIs. - Using online CLASSIFICATION Model APIs for Flowers Classification and predicting with test images to identify whether Rose or Sunflower or Tulips.

11	**SESSION 11:**	Initial Project 1 : Google Dino Game of - Automation - Develop a Python-based project with Pyautogui and PIL (Python Imaging Library) for implementation.	Google Dino Game of - Automation - Develop a Python-based project with Pyautogui and PIL (Python Imaging Library) for implementation.
12	**SESSION 12:**	Final Project 2 : Greenhouse gas forecasting - Predict future levels of greenhouse gases using NUMPY, SCIKIT LEARN libraries	Greenhouse gas forecasting - Predict future levels of greenhouse gases using NUMPY, SCIKIT LEARN libraries

11	**SESSION 11:**	Implementation using OBJECT DETECTION using YOLO Model & SORT	Final Project 5 : Project on Developing a machine learning model for OBJECT DETECTION - Using basic concepts in MODEL TRAINING including i) Data Collection ii) Model Selection iii)Training Model iv) Testing v) Prediction of Image with sample.
12	**SESSION 12:**		

DATA ANALYSIS LEVEL 1 :
INTRODUCTION TO MACHINE LEARNING

SESSION NO	LMS SESSION DESCRIPTION - UPDATED	GITHUB SESSION DESCRIPTION	OUTCOME FROM SESSION	TOPIC INTRODUCTION
SESSION 1: Mastering Arrays: A Comprehensive Guide to NumPy and Beyond	1. Introduction to Arrays 2. Arrays vs Lists 3. Creating Arrays using NumPy 4. NumPy Array Methods 5. Dimensions in Arrays 6. Indexing 6.1 Negative Indexing 7. Slicing 7.1. Negative Slicing	**1. Introduction to Arrays 2. Arrays vs Lists 3. Creating Arrays using NumPy (Different Functions to create arrays) 4. Dimensions in Arrays 5. Reshaping arrays 6. Indexing, Slicing 7. Practice Probelm**	**Outcome for creating arrays using NumPy:** Student will learn how to create arrays using NumPy. NumPy is a Python library used for numerical computations. Student will learn how to create arrays of different dimensions and types using NumPy. **Outcome for NumPy array methods:** **Stu**dent will learn how to perform mathematical operations on arrays using NumPy array methods. NumPy provides a wide range of mathematical functions that can be applied to arrays. Student will learn how to use these methods to manipulate and transform arrays. **Outcome for dimensions in arrays:** **Stu**dent will learn about the concept of dimensions in arrays. Arrays in NumPy can have different dimensions, ranging from 1 to n dimensions. Student will learn how to create arrays of different dimensions, and how to manipulate them using NumPy array methods. **Outcome for indexing:** **Stu**dent will learn how to access individual elements in arrays using indexing. Indexing is the process of retrieving an individual element from an array by its position. Student will learn how to use indexing to extract specific values from arrays. Outc**ome for slicing:** **Stu**dent will learn how to extract a portion of an array using slicing. Slicing is the process of selecting a portion of an array by specifying a range of indices. Student will learn how to use slicing to extract a subset of an array, and how to manipulate that subset using NumPy array methods.	Arrays have played a crucial role in many scientific discoveries throughout history, from the detection of pulsars to the estimation of railway construction costs. Pulsars are rapidly rotating neutron stars that emit a regular, repeating signal. In the 1960s, astronomer Jocelyn Bell Burnell used a phased array to analyze this mysterious signal, ultimately leading to the discovery of pulsars and earning her team a Nobel Prize in Physics. Meanwhile, in the mid-1800s, the British government turned to mathematician William Farr to quickly and accurately estimate the cost of railway construction using data points, such as the length of the line and the type of terrain it crossed, stored in an array. Today, arrays continue to be a powerful tool in many fields, including scientific research, image processing, and video game development. In this session, we'll dive into the basics of arrays and explore NumPy, a powerful Python library for working with arrays. We'll cover topics such as the difference between arrays and lists, creating arrays using NumPy, and using NumPy array methods. We'll also explore dimensions in arrays, indexing, and slicing, including negative indexing and negative slicing. Whether you're interested in scientific computing or simply want to improve your programming skills, this session on arrays is sure to be an exciting and informative experience. So let's dive in and explore the power of arrays!
SESSION 1 Practicals	**NumPy based Indexing : Fancy Indexing**	**NumPy based Indexing : Fancy Indexing**	**Problem Statement:** **Fancy Indexing** Fancy indexing allows to select entire rows or columns out of order on a numpy array	**LINKS:** https://github.com/aakankshaws/numpy-exercise/blob/master/Numpy%20Indexing%20and%20Selection.ipynb
SESSION 2: Mastering Data Operations with NumPy: From Reshaping to Filtering	1. NumPy Data Types 2. Converting datatype of an array 3. Array Reshaping 4. Joining Arrays 5. Array Stacking 5.1 Horizontal Stack - Stacking Along Rows 5.2 Vertical Stack - Stacking Along Columns 6. Filtering NumPy Array	**1. NumPy Data Types 2. Converting datatype of an array 3. Array Reshaping 4. Joining Arrays 5. Array Stacking 5.1 Horizontal Stack - Stacking Along Rows 5.2 Vertical Stack - Stacking Along Columns 6. Filtering NumPy Array**	**Outcome for NumPy data types:** Student will learn about the different data types supported by NumPy. NumPy provides a wide range of data types for numerical computations, ranging from integers to floating-point numbers, complex numbers, and booleans. Student will learn how to specify the data type of an array using NumPy. **Outcome for converting datatype of an array:** **Stu**dent will learn how to convert the data type of an array using NumPy. Converting the data type of an array can be useful when you want to perform mathematical operations on arrays with different data types. Student will learn how to convert an array to a different data type using NumPy. **Outcome for array reshaping:** **Stu**dent will learn how to reshape arrays using NumPy. Reshaping arrays involves changing the dimensions of an array without changing its content. Student will learn how to reshape arrays of different dimensions using NumPy array methods. **Outcome for joining arrays:** **Stu**dent will learn how to join arrays using NumPy. Joining arrays involves combining two or more arrays into a single array. Student will learn how to join arrays horizontally and vertically using NumPy array methods. **Outcome for array stacking:** **Stu**dent will learn how to stack arrays using NumPy. Stacking arrays involves combining two or more arrays into a single array along a new axis. Student will learn how to stack arrays vertically, horizontally, and depth-wise using NumPy array methods. **Outcome for filtering NumPy array:** **Stu**dent will learn how to filter arrays using NumPy. Filtering arrays involves selecting specific elements from an array based on a condition. Student will learn how to use NumPy array methods to create a Boolean mask and filter an array based on that mask.	In 2015, a team of researchers at NASA's Jet Propulsion Laboratory (JPL) in California made a remarkable discovery. Using data collected by the Kepler space telescope, they identified a distant planet with conditions that made it potentially habitable for life. The planet, dubbed Kepler-438b, is located about 640 light-years away from Earth and is roughly the same size as our own planet. What makes it so interesting is that it orbits its star at just the right distance to potentially support liquid water and, therefore, life. But how did the researchers make this discovery? One critical tool in their arsenal was NumPy, a Python library for scientific computing. NumPy allowed them to efficiently work with large datasets and perform complex calculations, enabling them to identify the telltale signs of the planet's orbit. In this session, we'll explore the basics of NumPy, starting with data types and how to convert between them. We'll then dive into array reshaping, which allows us to change the shape of an array without changing its data. Next, we'll cover joining and stacking arrays, which are powerful techniques for combining data from multiple sources. Finally, we'll learn about filtering arrays, which allows us to extract only the data we need from a larger dataset. By the end of this session, you'll have a solid understanding of NumPy and its capabilities, as well as some practical skills that you can apply to your own data analysis projects. So let's get started and discover the power of NumPy!
SESSION 2 Practicals	**NumPy based : Real-World Multidimensional Array Filtering**	**NumPy based : Real-World Multidimensional Array Filtering**	**Problem Statement** Define a 3D, 3x2x4 numpy array. We defined the indexes of this array as floor, apartment and room. Then, with filtering, we extracted the apartment numbers according to the magnets in the kitchens of the apartments in this building.	**LINKS:** **https://github.com/ozlemekici/multidimensional_array_filtering**
SESSION 3: Exploring the World of Images and Data with NumPy and OpenCV	1. NumPy Topics - Copy & View 2. Reshaping & Flattening the NumPy arrays 3. Sorting the NumPy Arrays 4. Introduction to openCV-Python 5. OpenCV Topics - vstack, hstack, indexing, cropping, slicing,filtering. 5.1 OpenCV and NumPy Operations: vstack 5.2 OpenCV and NumPy Operations: hstack 5.3 OpenCV and NumPy Operations: indexing 5.4 OpenCV and NumPy Operations: Crop using indexing 5.5 OpenCV and NumPy Operations: Cropping Numpy Zeros 5.6 OpenCV and NumPy Operations: Extracting Channel using indexing 5.7 OpenCV and NumPy Operations: Cropping & Patching 5.7 OpenCV and NumPy Operations: Gray Slicing 5.8 OpenCV and NumPy Operations: Masking using Filtering	**1. NumPy Topics - Copy & View 2. Reshaping & Flattening the NumPy arrays 3. Sorting the NumPy Arrays 4. Introduction to openCV-Python 5. OpenCV Topics - vstack, hstack, indexing, cropping, slicing,filtering. 5.1 OpenCV and NumPy Operations: vstack 5.2 OpenCV and NumPy Operations: hstack 5.3 OpenCV and NumPy Operations: indexing 5.4 OpenCV and NumPy Operations: Crop using indexing 5.5 OpenCV and NumPy Operations: Cropping Numpy Zeros 5.6 OpenCV and NumPy Operations: Extracting Channel using indexing 5.7 OpenCV and NumPy Operations: Cropping & Patching 5.7 OpenCV and NumPy Operations: Gray Slicing 5.8 OpenCV and NumPy Operations: Masking using Filtering**	**NumPy Copy & View:** Understand the difference between a copy and a view of a NumPy array Create a copy of a NumPy array using the copy() method Create a view of a NumPy array using the view() method Modify elements of a view and observe changes in the original array Use the base attribute to check if an array is a view or a copy of another array **NumPy Reshaping & Flattening:** Reshape a NumPy array using the reshape() method Flatten a NumPy array using the flatten() method Use the ravel() method to flatten a NumPy array in-place Use the newaxis keyword to add a new dimension to a NumPy array **NumPy Arrays Sorting:** Sort a NumPy array using the sort() method Sort a NumPy array along a specific axis using the axis parameter Use the argsort() method to get the indices that would sort a NumPy array Use the lexsort() method to perform a multi-dimensional sort on a NumPy array **OpenCV Python: vstack & hstack:** Understand how to use the vstack and hstack functions in OpenCV to concatenate multiple images vertically and horizontally Use the cv2.split() function to split a multi-channel image into its individual channels Use the cv2.merge() function to merge multiple channels into a single multi-channel image **OpenCV Python: Indexing & Cropping:** Understand how to access individual pixels in an image using indexing Crop a region of interest from an image using indexing and slicing Use the cv2.rectangle() function to draw a rectangle around a region of interest in an image **OpenCV Python: Slicing & Filtering:** Use slicing to extract a sub-image from an image Apply a filter to an image using the cv2.filter2D() function Use the cv2.GaussianBlur() function to apply a Gaussian blur to an image Use the cv2.Canny() function to detect edges in an image.	Have you ever wondered how images are processed and analyzed in real life? Have you ever been curious about how self-driving cars recognize the objects around them? Or how Instagram applies different filters to your pictures? The answer lies in the powerful combination of NumPy and OpenCV. But first, let me tell you a story about a wildlife photographer named Emma. Emma loves capturing the beauty of nature through her camera lens. One day, she went on a safari trip to Africa and took thousands of photographs of different animals in their natural habitats. However, when she got back home and tried to organize her photos, she realized that it was nearly impossible to sort through all of them manually. That's when Emma discovered NumPy, a library in Python that can manipulate large arrays and matrices of numerical data. With NumPy, Emma was able to reshape and flatten her photo arrays, making them easier to sort and process. She could also sort her photos by their metadata, like the date and location they were taken. But Emma's work was not complete yet. She wanted to analyze the images and extract specific features, like the animals' shapes and colors. That's where OpenCV comes in. OpenCV is a library that allows you to process and analyze images and videos using computer vision techniques. Emma used OpenCV to stack and crop her images, and even filter out certain colors to highlight the features she wanted to focus on. This is just one example of how NumPy and OpenCV are used in real life. From self-driving cars to medical imaging, these libraries are revolutionizing the way we process and analyze data. So, if you want to learn how to use these powerful tools, join us in this session where we'll explore NumPy and OpenCV and their real-world applications.
SESSION 3 Practicals	**OpenCV based :**	**OpenCV based : People Counting [Object-Detection]**	**Problem Statement** People Counter Based on OpenCV with concept of program counting number of people incomming and outgoing a particular door.	**LINKS:** https://github.com/narayananramu/opencv-people-counter
SESSION 4: Mastering the Basics of Pandas Series: Handling Data Types, Appending, and Concatenation	1. What is Pandas? 2. Pandas Series 3. Datatype handling in Panda Series 4. Appending in Panda Series 5. Concatenation in Panda Series	**1. What is Pandas? 2. Pandas Series 3. Datatype handling in Panda Series 4. Appending in Panda Series 5. Concatenation in Panda Series**	**Pandas Series:** Pandas Series is a one-dimensional labeled array that can hold any data type such as integers, strings, floats, etc. It is similar to a column in a spreadsheet or a database table. **Datatype handling in Panda Series:** The data type of a Pandas Series can be changed using the astype() method. For example, if you have a Pandas Series with integer values and you want to convert it to float values, you can use the astype(float) method. **Appending in Panda Series:** The append() method is used to append one Pandas Series to another. For example, if you have two Pandas Series and you want to combine them into one, you can use the append() method. **Concatenation in Panda Series:** The concatenate() function is used to concatenate multiple Pandas Series into one. For example, if you have three Pandas Series and you want to concatenate them into one, you can use the concatenate() function.	Have you ever gone grocery shopping and wondered how supermarkets keep track of all the products they sell? They use a database to store and manage information about their inventory, prices, and sales. One popular tool for working with databases is Pandas, a Python library that allows you to manipulate and analyze data in various ways. For example, let's say you want to compare the prices of different brands of cereal in a supermarket. Student can use Pandas to create a "series" of data, which is like a column of information in a spreadsheet. Each row represents a different type of cereal, and the columns show the brand, price, and other details. Student can also use Pandas to change the data types of the values in the series, for instance, converting the price column from strings to numerical values. Furthermore, you can append new rows of data to the existing series or concatenate multiple series together to create a larger dataset. These operations are essential for working with large, complex data sets, such as those used by businesses and organizations for decision-making. In this session, we will learn about the basics of Pandas and focus on creating and manipulating series. We will also explore how to handle data types in series, append and concatenate data, and more. So get ready to dive into the world of Pandas and discover how it can help you manage data in your own projects!
SESSION 4 Practicals	**PANDAS based : Missing Values Treatment Problem**	**PANDAS based : Missing Values Treatment Problem**	**Problem Statement** **Problem on data analysis or data engineering where we should never lose any data, ideally there are two ways with which you can deal with missing values one by filling it with mean median and mode and second by removing the rows contating missing values.**	**LINKS:** **https://github.com/rejin-omotec/AVM-DD/tree/master/pandas**

SESSION 5: Mastering Data Manipulation with Pandas: From Creating DataFrames to Loading and Analyzing CSV and Excel Files	1. Dataset, Series & Dataframe 2. Creating a DataFrame 3. Loading files into a DataFrame 3.1 Reading in DataFrame 3.2 DataFrame head() Function 3.3 DataFrame tail() Function 3.4 Information About the DataSet 4. Read CSV in Pandas 4.1 Read Sample CSV: 4.2 To check maximum number of rows 4.3 Read a CSV file and print the first few rows using the head() method: 4.4 Read a CSV file and print the Last few rows using the tail() method: 4.5 Pandas describe() method on CSV Files 4.6 Writing a Pandas DataFrame to CSV file 5. Read Excel in Pandas 5.1 Install xlrd 5.2 Load multiple sheets 5.3 Display List of Columns Headers of the Excel Sheet 5.4 Pandas read_excel() usecols example	**1. Dataset, Series & Dataframe 2. Creating a DataFrame 3. Loading files into a DataFrame 3.1 Reading in DataFrame 3.2 DataFrame head() Function 3.3 DataFrame tail() Function 3.4 Information About the DataSet 4. Read CSV in Pandas 4.1 Read Sample CSV: 4.2 To check maximum number of rows 4.3 Read a CSV file and print the first few rows using the head() method: 4.4 Read a CSV file and print the Last few rows using the tail() method: 4.5 Pandas describe() method on CSV Files 4.6 Writing a Pandas DataFrame to CSV file 5. Read Excel in Pandas 5.1 Install xlrd 5.3 Display List of Columns Headers of the Excel Sheet 5.4 Pandas read_excel() usecols example**	**Pandas:** Pandas is one of the most popular libraries for working with data in Python. Pandas provides a wide range of data structures and tools for data analysis, including the dataset, series, and dataframe. **Dataset::** A dataset is a collection of data points that can be analyzed or processed. In Pandas, a dataset is generally represented as a dataframe. **Series:** A series is a one-dimensional array that can hold any data type, such as integers, floats, or strings. A series can be created from a list, tuple, or dictionary. **DataFrame:** A dataframe is a two-dimensional labeled data structure that contains rows and columns. A dataframe can be thought of as a spreadsheet or a SQL table. It is the most commonly used data structure in Pandas. **Creating a DataFrame:** To create a dataframe in Pandas, we can use the DataFrame() function. **Read CSV in Pandas:** To read data from a CSV file using Pandas, we can use the read_csv() function.We import the Pandas library and use the read_csv() function to read the data from a CSV file named "data.csv". The resulting DataFrame is then printed using the print() function. **Read Excel in Pandas:** To read data from an Excel file using Pandas, we can use the read_excel() function. We import the Pandas library and use the read_excel() function to read the data from an Excel file named "data.xlsx". The resulting DataFrame is then printed using the print() function. In this tutorial, we learn how to read data from CSV and Excel files in Python using the Pandas library. We see examples of how to use the read_csv() and read_excel() functions to load data into a Pandas DataFrame from external files. With this knowledge, you should be able to start working with tabular data in Python and Pandas using data from CSV and Excel files.	Have you ever wondered how social media platforms like Facebook and Instagram manage to show you only the posts and pictures that are relevant to your interests? The answer lies in data analysis and manipulation, which is the core of the Python library called Pandas. Imagine you work for Facebook, and your boss has asked you to analyze user data to find out what kind of posts people engage with the most. Student have thousands of rows of data, including the user's name, age, location, the type of post, and the number of likes and comments. How do you make sense of all this information? This is where Pandas comes in. With its powerful tools for handling datasets, creating data frames, and loading files, you can quickly and easily organize and manipulate data to draw meaningful insights. Student can use the head() and tail() functions to preview data, the describe() function to get a summary of the data's statistical measures, and even write the output to a CSV or Excel file. So, whether you're a data analyst or just someone interested in learning how to work with data, understanding Pandas can be an invaluable skill in today's data-driven world.
SESSION 5 Practicals	**Dataframe based : Case study:** Air quality data of European monitoring stations AirBase (The European Air quality dataBase)	**Dataframe based : Case study:** Air quality data of European monitoring stations AirBase (The European Air quality dataBase)	**Problem Statement** **Activity:** Perform all Dataframe operations on the Dataset of Airbase - The European Air quality dataBase	**LINKS:** https://github.com/jorisvandenbossche/pandas-tutorial/blob/master/solved%20-%2007%20-%20Case%20study%20-%20air%20quality%20data.ipynb https://github.com/jorisvandenbossche/pandas-tutorial/blob/master/pandas_introduction.ipynb
SESSION 6: Exploring Advanced DataFrame Operations in Pandas: Combining, Merging, Grouping, and Aggregating Data for Better Analysis	1. Filtering data in DataFrame 2. Combining in DataFrame 2.1 Inner Join: 2.2 Left Outer Join: 2.3 Right Outer Join: 2.4 Full Outer Join: 3. Merging in DataFrame 4. Counting in DataFrame 5. Functions in DataFrame 5.1 Aggregations 5.2 View Groups 5.3 Select a Group 5.4 Filtration 5.5 Fill NA on CSV using fillna() 5.6 Group and Filter on CSV using groupby() & filter()	**1. Filtering data in DataFrame 2. Combining in DataFrame 2.1 Inner Join: 2.2 Left Outer Join: 2.3 Right Outer Join: 2.4 Full Outer Join: 3. Merging in DataFrame 4. Counting in DataFrame 5. Functions in DataFrame 5.1 Aggregations 5.2 View Groups 5.3 Select a Group 5.4 Filtration 5.5 Fill NA on CSV using fillna() 5.6 Group and Filter on CSV using groupby() & filter()**	**Filtering data in DataFrame:** Filtering data in a DataFrame involves selecting specific rows that meet certain criteria. This can be done using boolean indexing or the .query() method. **Combining in DataFrame:** Combining data in a DataFrame involves stacking multiple DataFrames on top of one another or combining them side by side. This can be done using the .concat() method. **Merging in DataFrame:** Merging data in a DataFrame involves combining two or more DataFrames based on a common column or index. This can be done using the .merge() method. **Counting in DataFrame:** Counting data in a DataFrame involves counting the number of occurrences of certain values in one or more columns. This can be done using the .value_counts() method or the .groupby() method in combination with the .count() method. **Aggregations in DataFrame:** Aggregating data in a DataFrame involves computing summary statistics for groups of rows or columns. This can be done using the .groupby() method in combination with various aggregation functions such as .sum(), .mean(), .max(), and .min(). **View Groups in DataFrame:** Viewing groups in a DataFrame involves displaying the groups that were created using the .groupby() method. This can be done using the .groups attribute. **Select a Group in DataFrame:** Selecting a group in a DataFrame involves selecting a specific group that was created using the .groupby() method. This can be done using the .get_group() method. **Filtration in DataFrame:** Filtering data in a DataFrame involves selecting specific rows that meet certain criteria. This can be done using boolean indexing or the .query() method. **Fill NA in DataFrame:** Filling missing values in a DataFrame involves replacing NaN or null values with a specified value or method. This can be done using the .fillna() method. **Group and Filter on CSV using groupby() & filter() in DataFrame:** Grouping and filtering data in a CSV file involves grouping the data based on one or more columns and then filtering the resulting groups based on certain criteria. This can be done using the .groupby() method in combination with the .filter() method.	Dataframes are a powerful data structure that are commonly used in data analysis and manipulation tasks. Some real-world applications of dataframes include: Healthcare analysis, Sports analysis,Financial analysis,, Marketing analysis,Social media analysis. Overall, dataframes are a versatile data structure that can be used in a wide range of real-world applications to extract insights from data and inform decision-making. Amazon is a company that generates massive amounts of data every day from online shopping transactions, customer reviews, and other sources. To manage and analyze this data, Amazon uses a variety of tools, including Pandas. One example of how Amazon uses Pandas is to analyze customer reviews. Customer feedback is crucial to the success of Amazon's business, and they need to understand what customers like and don't like about their products. Using Pandas, Amazon can filter and group customer reviews by product, category, and other attributes, making it easier to identify patterns and trends. Amazon also uses Pandas for supply chain management. By merging data from different sources such as warehouses, suppliers, and logistics partners, they can optimize their supply chain and ensure that products are delivered to customers quickly and efficiently. By learning how to use Pandas to manipulate and analyze data, you'll be gaining skills that are highly valued by companies like Amazon and are essential for success in many fields. Whether you're interested in business, technology, or science, the ability to work with data is becoming increasingly important in today's world.
SESSION 6 Practicals	**Practice Problem** Dataframe - Advanced groupby & Filter operations Case study: air quality data of European monitoring stations	**Practice Problem** Dataframe - Advanced groupby & Filter operations Case study: air quality data of European monitoring stations	**Problem Statement** **Activity :** Advanced groupby & filter operations using Dataframe CSV **Activity :** Case Study implementation	**LINKS:** https://github.com/jorisvandenbossche/pandas-tutorial/blob/master/solved%20-%2004b%20-%20Advanced%20groupby%20operations.ipynb
SESSION 7: Mastering Data Visualization with Matplotlib: A Comprehensive Guide to Plotting Multiple Charts and Adding Grid Lines, Subplots, Titles, and Legends	1. What is matplotlib? 2. Display Multiple Plots 3. Matplotlib Pie Charts 4. Matplotlib Adding Grid Lines 5. Matplotlib Subplot 5.1 Draw Multiple plots 5.2 Subplot function Arguments 5.3 matplotlib add_subplot() function 6. Title 6.1. Super Title 7. Legend function in Matplotlib	**1. What is matplotlib? 2. Display Multiple Plots 3. Matplotlib Pie Charts 4. Matplotlib Adding Grid Lines 5. Matplotlib Subplot 5.1 Draw Multiple plots 5.2 Subplot function Arguments 5.3 matplotlib add_subplot() function 6. Title 6.1. Super Title 7. Legend function in Matplotlib**	**Matplotlib:** Matplotlib is a Python library used for data visualization. It provides a variety of functions for creating different types of plots, including line plots, scatter plots, bar plots, and more. **Display Multiple Plots:** Displaying multiple plots in Matplotlib involves creating multiple subplots within a single figure. This can be done using the . subplot() method or the .subplots() method. **Matplotlib Pie Charts:** Pie charts in Matplotlib are used to show the proportion of each category in a dataset. This can be done using the .pie() method. **Matplotlib Adding Grid Lines:** Adding grid lines in Matplotlib involves adding horizontal and vertical lines to the plot to help with visualizing the data. This can be done using the .grid() method. **Matplotlib Subplot:** Subplots in Matplotlib involve dividing a single figure into multiple smaller plots. This can be done using the .subplot() method or the .subplots() method. **Matplotlib Title:** Adding a title to a plot in Matplotlib involves providing a descriptive title that summarizes the contents of the plot. This can be done using the .title() method. **Matplotlib Legend function:** Adding a legend to a plot in Matplotlib involves providing a label for each line or set of data in the plot. This can be done using the .legend() method. The .legend() method allows you to specify the location of the legend and other properties such as font size and color.	Have you ever wondered how data scientists and researchers are able to effectively communicate their findings using graphs and charts? Well, the secret is in a powerful data visualization library called Matplotlib. From the stock market to medical research, data visualization plays a critical role in a wide range of industries. For instance, consider the case of a pharmaceutical company that needs to analyze the effectiveness of a new drug. In order to determine whether the drug is working, the company would collect data from clinical trials and use Matplotlib to create graphs and charts that show how the drug is affecting patients over time. This can help the researchers identify any trends or patterns in the data that may indicate whether the drug is working as intended. In this session, we'll explore the power of Matplotlib and how it can be used to create compelling visualizations that help us better understand data. We'll start by understanding the basics of Matplotlib and how to display multiple plots on the same chart. We'll then dive into pie charts, which are useful for showing proportions of data in a circular format. Additionally, we'll explore how to add grid lines to our charts, which can make them more readable and easier to understand. We'll also learn how to create subplots, which allow us to display multiple plots in the same figure. This can be particularly useful when we want to compare different sets of data side-by-side. Finally, we'll discuss how to add titles and super titles to our visualizations and use the legend function to label different plots. By the end of this session, you'll have a better understanding of the power of data visualization and the skills to create your own compelling charts and graphs with Matplotlib.
SESSION 7 Practicals	**Matplotlib based : CSV based Data Visualization**	**Matplotlib based : CSV based Data Visualization**	**Problem Statement** **Activity:** Students are provided with this Dataset 'students.csv' In this data set you need to visualize whether we have more male or female students in school. **Activity:** Get Correlation Matrix for 'students.csv' data.	**LINKS:** https://github.com/rejin-omotec/AVM-DD/tree/master/MatPlotLib
SESSION 8: Mastering Scatter Plots, Heatmaps, and Confusion Matrices with Matplotlib: A Comprehensive Guide to Visualizing Data and Analyzing Relationships	1. Scatter Plots in matplotlib 2. Scatter Plots using CSV in matplotlib 3. Change marker and marker size using CSV 4. Scatter plot colored by category 5. Markers' Size in Scatter Plot 6. DataFrame in Heatmap using matplotlib 6.1 Creating Heatmap using matplotlib library 7. Confusion Matrix using matplotlib 7.1 Creating a Confusion Matrix	**1. Scatter Plots in matplotlib 2. Scatter Plots using CSV in matplotlib 3. Change marker and marker size using CSV 4. Scatter plot colored by category 5. Markers' Size in Scatter Plot 6. DataFrame in Heatmap using matplotlib 6.1 Creating Heatmap using matplotlib library 7. Confusion Matrix using matplotlib 7.1 Creating a Confusion Matrix**	**Scatter Plots in Matplotlib:** Scatter plots in Matplotlib are used to visualize the relationship between two variables. This can be done using the . scatter() method. **Scatter Plots using CSV in Matplotlib:** Scatter plots using CSV data in Matplotlib involve loading the data from a CSV file and then creating a scatter plot using the .scatter() method. **Change marker and marker size using CSV:** Changing the marker and marker size using CSV data involves specifying the marker type and size for each data point in the CSV file. This can be done using the marker and s parameters of the .scatter() method. **Scatter plot colored by category:** Coloring a scatter plot by category involves assigning a different color to each category or group of data points. This can be done using the c parameter of the .scatter() method. **Markers' Size in Scatter Plot:** Markers' size in a scatter plot refers to the size of each data point in the plot. This can be adjusted using the s parameter of the .scatter() method. **DataFrame in Heatmap using Matplotlib:** Creating a heatmap in Matplotlib involves visualizing the values in a DataFrame using a color scale. This can be done using the .heatmap() method. **Confusion Matrix using Matplotlib:** A confusion matrix in Matplotlib is a table used to evaluate the performance of a classification algorithm. This can be created using the .confusion_matrix() function from the sklearn.metrics module and then visualized using Matplotlib.	Have you ever wondered how scientists and researchers use data to study complex systems, such as climate change or social networks? One way they do it is by creating scatter plots to display their data in a clear and informative way. For example, let's say you are interested in studying the relationship between a student's study time and their exam scores. Student could collect data on the study time and exam scores of several students and plot it on a scatter plot using matplotlib, a popular data visualization library in Python. By looking at the scatter plot, you might be able to see if there is a correlation between study time and exam scores, and how strong that correlation is. But scatter plots can do more than just show the relationship between two variables. With matplotlib, you can also add markers, change their size and color, and create heatmaps and confusion matrices to visualize even more complex data. So, are you ready to learn how to create your own scatter plots using matplotlib and explore the world of data visualization? Let's get started!
SESSION 8 Practicals	**Matplotlib based Data Visualization in Python:** Data Visualization using Heat Maps using Matplotlib and Seaborn. Data Visualization of the performance of an algorithm,a confusion matrix, also known as an error matrix	**Matplotlib based Data Visualization in Python:** Data Visualization using Heat Maps using Matplotlib and Seaborn. Data Visualization of the performance of an algorithm,a confusion matrix, also known as an error matrix	**Problem Statement:** Using CSV file create Heat Maps using Matplotlib and Seaborn In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one	**LINKS:** https://github.com/mGalarnyk/Python_Tutorials/tree/master/Request **https://github.com/Tarun-Kamboj/Data_Visualization_with_Python/tree/master/Code/Confusion%20Matrix**

SESSION 9: Introduction to Linear Regression and Supervised Machine Learning: Understanding Types, Assumptions, and Real-World Applications with Python Examples	1. Machine Learning - Introduction 2. Types of Machine Learning 3. Types of Supervised Learning: 4. Linear Regression - Introduction 5. Simple Linear Regression 5.1 Linear Regression using sklearn - Python Example 5.2 Realtime Linear Regression: Prediction Algorithm for Forecasting Stock Price 6. Multiple Linear Regression 7. Assumptions in Linear Regression Model	**1. Machine Learning - Introduction 2. Types of Machine Learning 3. Types of Supervised Learning: 4. Linear Regression - Introduction 5. Simple Linear Regression 5.1 Linear Regression using sklearn - Python Example 5.2 Realtime Linear Regression: Prediction Algorithm for Forecasting Stock Price 6. Multiple Linear Regression 7. Assumptions in Linear Regression Model**	**Machine Learning Introduction:** Machine Learning is a field of study that focuses on the development of algorithms that can learn from and make predictions on data. It involves the use of statistical and mathematical models to analyze and make predictions based on data. **Types of Machine Learning:** There are three main types of Machine Learning: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. **Types of Supervised Learning:** Supervised Learning is a type of Machine Learning where the algorithm is trained on a labeled dataset, which means that the desired output is known for each input. There are two main types of Supervised Learning: Regression and Classification. **Linear Regression Introduction:** Linear Regression is a statistical method used to analyze and model the relationship between two variables. It involves fitting a straight line to the data and using this line to make predictions on new data. **Simple Linear Regression:** Simple Linear Regression is a type of linear regression where there is only one independent variable and one dependent variable. The goal of Simple Linear Regression is to find the best-fit line that describes the relationship between the two variables. **Multiple Linear Regression:** Multiple Linear Regression is a type of linear regression where there are multiple independent variables and one dependent variable. The goal of Multiple Linear Regression is to find the best-fit line that describes the relationship between the independent variables and the dependent variable. **Assumptions in Linear Regression Model:** Linear Regression models are based on a number of assumptions that must be met in order for the model to be valid. Some of these assumptions include linearity, independence, normality, and homoscedasticity. Violations of these assumptions can lead to inaccurate predictions and unreliable models.	Did you know that Machine Learning is being used in a wide range of applications in our daily lives, from voice assistants like Siri and Alexa to self-driving cars? In fact, a real-life example of Machine Learning in action is Google's search engine algorithm, which uses Machine Learning to predict which search results are most relevant to your query. But that's not all - Machine Learning is also being used in healthcare to predict patient outcomes and diagnose diseases, in finance to detect fraud and make investment decisions, and in sports to analyze player performance and predict game outcomes. In this session, we'll be exploring one of the most popular Machine Learning algorithms - Linear Regression - and its real-world applications. We'll be focusing on how Linear Regression can be used to predict stock prices, an essential component of financial decision-making. Imagine being able to predict stock prices accurately and make informed investment decisions based on Machine Learning algorithms! We'll also cover Multiple Linear Regression, which can be used to predict outcomes based on multiple variables. For instance, Multiple Linear Regression can be used to predict the price of a house based on factors like location, square footage, and number of bedrooms. By the end of this session, you'll not only have a solid understanding of Machine Learning and Linear Regression but also how these algorithms can be applied to solve real-world problems and make informed decisions. So, let's begin this exciting journey & explore the fascinating world of Machine Learning!
SESSION 9 Practicals	**Linear Regression based : Using dataset "Advertising.csv"**	**Linear Regression based : Using dataset "Advertising.csv"**	**Problem Statement** Problem using Linear Regression - Run your First Machine Learning Algorithm Perform Evaluation Using: RMSE & R Squared	**LINKS:** **https://github.com/rejin-omotec/AVM-DD/tree/master/Linear%20regression**
SESSION 10: Beyond the Straight Line: Unleashing the Power of Polynomial Regression in Python	1. Polynomial Regression - Introduction 2. Need for Polynomial Regression 3. Python implementation of polynomial regression 4. Construction of polynomial regression model 5. Displaying the polynomial regression result 6. Polynomial Smooth Regression Using CSV with Python 7. Polynomial Regression with Various Polynomial degree ranges	**1. Polynomial Regression - Introduction 2. Need for Polynomial Regression 3. Python implementation of polynomial regression 4. Construction of polynomial regression model 5. Displaying the polynomial regression result 6. Polynomial Smooth Regression Using CSV with Python 7. Polynomial Regression with Various Polynomial degree ranges**	**Polynomial Regression Introduction:** Polynomial Regression is a statistical method used to analyze and model the relationship between two variables when the relationship is not linear. It involves fitting a polynomial function to the data and using this function to make predictions on new data. **Need for Polynomial Regression:** Polynomial Regression is needed when the relationship between the independent and dependent variables is not linear. In such cases, linear regression may not be able to accurately describe the relationship between the variables. **Python implementation of Polynomial Regression:** Polynomial Regression can be implemented in Python using the numpy and sklearn libraries. Numpy is used to create arrays for the independent and dependent variables, while sklearn is used to create the polynomial regression model. **Construction of Polynomial Regression Model:** The polynomial regression model is constructed by fitting a polynomial function to the data. This involves choosing the degree of the polynomial and then using this degree to create the polynomial function. **Displaying the Polynomial Regression Result:** The result of the polynomial regression model can be displayed using a scatter plot of the data points and the fitted polynomial curve. This can be done using the matplotlib library. **Polynomial Smooth Regression Using CSV with Python:** Polynomial Smooth Regression is a variation of Polynomial Regression that is used to smooth out the curve of a dataset. This can be implemented in Python by loading the data from a CSV file and then using the numpy and sklearn libraries to create the polynomial smooth regression model. **Polynomial Regression with Various Polynomial Degree Ranges:** The degree of the polynomial used in Polynomial Regression can greatly affect the accuracy of the model. Therefore, it is important to experiment with different polynomial degree ranges in order to find the degree that best fits the data. This can be done using cross-validation and other techniques.	Do you ever wonder how Netflix recommends movies and TV shows that you might like? Or how Facebook suggests friends that you might know? These are examples of machine learning algorithms at work. One of the most powerful machine learning techniques is Polynomial Regression, which is used to predict future trends and make informed decisions. Polynomial Regression is like having a crystal ball that can help us see into the future. It's a bit like a magic trick, but it's based on math and data analysis. In today's session, we will learn how to use Polynomial Regression to make predictions using Python. We'll start by understanding why we need Polynomial Regression, and then we'll move on to constructing a model and displaying the results. But that's not all! We will also learn how to use Polynomial Regression to predict the outcome of sports games, like soccer and basketball. We'll also explore how farmers can use Polynomial Regression to predict crop yields and make informed decisions about planting and harvesting. So get ready to be a data wizard, and let's learn how to use Polynomial Regression to make predictions like a pro! By the end of this session, you'll be able to impress your friends and family with your data analysis skills, and you'll be ready to apply them to real-world problems.
SESSION 10 Practicals	**Polynomial Regression based : Using dataset "Position_Salaries.csv"**	**Polynomial Regression based : Using dataset "Position_Salaries.csv"**	**Problem Statement** Problem using Polynomial Regression - Predicting new results with Polynomial Regression. Note that the input variable must be in a numpy 2D array.	**LINKS:** **https://github.com/rejin-omotec/AVM-DD/tree/master/Polynomial%20regression**
Initial Project	**Initial Project 1 : Google Dino Game of - Automation** - Develop a Python-based project with Pyautogui and PIL (Python Imaging Library) for implementation.	**Initial Project 1 : Google Dino Game of - Automation - Develop a Python-based project with Pyautogui and PIL (Python Imaging Library) for implementation. - This project is very basic and consists of only about 50 lines of code.**	**PROBLEM STATEMENT:** - Click on the restart button using Pyautogui library using "replaybutton" coordinates. - Calculate the sum of all white pixels values present in the box in front of Dinosaur. - If the sum of pixels values present at any time in the box becomes less than the sum of white pixels values, it means either "bush" or "bird" is coming. So either we have to make our Dino jump or bend down. - In order to protect Dino from "Bush", we make a jump. - In order to protect Dino from "Bird", we always keep our Dino down.	We will work with Pyautogui and PIL (Python Imaging Library) for implementation. This project is very basic and consists of only about 50 lines of code but its result will make you surprise. LINK: https://www.geeksforgeeks.org/google-chrome-dino-bot-using-image-recognition-python/ https://github.com/ayushgemini/automate-chrome-dino-game-python
~~**Final Project**~~	~~**Final Project 2 :**~~ ~~**Greenhouse gas forecasting**~~ ~~- Predict future levels of greenhouse gases using NUMPY, SCIKIT LEARN libraries~~	~~**Final Project 2 :**~~ ~~**Greenhouse gas forecasting**~~ ~~- Predict future levels of greenhouse gases using NUMPY, SCIKIT LEARN libraries~~	~~**PROBLEM STATEMENT:**~~ ~~- This Python project is to develop a greenhouse gas forecasting model using data from previous years, leveraging the power of numerical computing with NumPy and machine learning with Scikit-learn.~~ ~~- The goal is to accurately predict the amount of greenhouse gas emissions in the future based on historical data, allowing for better planning and mitigation strategies to combat climate change.~~ ~~- The project will involve data preprocessing, exploratory data analysis, feature engineering, model selection and tuning, and evaluation of the model's performance.~~ ~~- The end result will be a robust and accurate model that can help stakeholders make informed decisions about reducing greenhouse gas emissions.~~	~~LINKS:~~ ~~**CO2 Emission Forecast with Python with SARIMA Model (Seasonal ARIMA)**~~ ~~https://www.kaggle.com/code/vijaikm/co2-emission-forecast-with-python-seasonal-arima/notebook~~ ~~[1]~~

DATA ANALYSIS LEVEL 2 :
INTRODUCTION TO MACHINE LEARNING

SESSION NO	LMS SESSION DESCRIPTION	GITHUB SESSION DESCRIPTION	OUTCOME FROM SESSION	TOPIC INTRODUCTION
SESSION 1:	**1) Image processing** *Image acquisition and representation* *Image enhancement techniques (histogram equalization, contrast stretching)* *Image restoration techniques (noise reduction, image deblurring)* *Image segmentation techniques (thresholding, edge detection)* *Image compression techniques (JPEG, PNG)* **2) OpenCV:** *Installation and Setup of OpenCV* *Image processing and manipulation with OpenCV* *Basic Image Handling Operations with OpenCV* *Image Transformation and Enhancement with OpenCV* *Image Filtering with OpenCV* *Image Blending and Compositing with OpenCV* *Color Spaces and Color Management with OpenCV* *Text and Font Handling with OpenCV*	**1. Introduction** **2. Installation and Setup** **3. Image processing** **- Image acquisition and representation** **- Image enhancement techniques** **- Image restoration techniques** **- Image segmentation techniques** **- Image compression techniques** **4. OpenCV** **- Image Transformation and Enhancement** **- Image Filtering** **- Image Blending and Compositing** **- Color Spaces and Color Management** **- Text and Font Handling**	**Outcome of Learning Image processing:** By using various image processing techniques, it's possible to improve the quality of an image, remove noise, enhance colors, and add various filters and effects. **Outcome of Learning MediaPipe:** Understanding of MediaPipe and its capabilities , Ability to use MediaPipe for computer vision tasks, Familiarity with MediaPipe's pre-built components and APIs,Knowledge of how to integrate MediaPipe with other software and hardware. Understanding of face detection and tracking algorithms,Ability to use MediaPipe's pre-built face detection and tracking components,Familiarity with the evaluation metrics used to measure face detection and tracking performance,Ability to apply face detection and tracking to real-world applications (e.g. face recognition, emotion detection). Ability to customize MediaPipe for specific use cases,Ability to develop new components for MediaPipe,Knowledge of best practices for MediaPipe development and deployment,Ability to integrate MediaPipe with other software and hardware.	Image processing using MediaPipe is best utilized in the Snapchat app. Snapchat's filters are implemented using MediaPipe's computer vision and machine learning tools. Also this concept is best applied in the Google Meet video conferencing platform, which uses MediaPipe to enhance the video call experience. **Image processing** involves various techniques such as resizing, cropping, color correction, and filtering to enhance the image quality. **MediaPipe Module:** MediaPipe is a powerful open-source cross-platform framework that provides developers with a flexible way to build real-time multi-modal machine learning applications. MediaPipe is a versatile framework that can be used in a wide range of applications that require real-time multi-modal machine learning capabilities. **Here are some of the applications of MediaPipe:** Augmented Reality: MediaPipe can be used to create AR applications that can track objects and human faces in real-time. This can be used to overlay digital objects on top of the real-world environment. Other Applications include Robotics,Gaming,Healthcare,Automotive,Security and Surveillance & Entertainment MediaPipe, MediaPipe Face Detection and Tracking, MediaPipe Pose Detection and Tracking, MediaPipe Object Detection and Tracking, MediaPipe Customization and Development, and Facial Landmark Detection use several concepts related to computer vision, deep learning, and machine learning. Some of the key concepts used in these tools and modules are: **1) Object Detection:** Object detection is a computer vision technique that involves identifying objects within images or videos. This is often achieved using deep learning models such as CNNs. **2) Pose Detection:** Pose detection is a computer vision technique that involves detecting human body poses, including the position and orientation of different body parts. This is often achieved using deep learning models such as CNNs. **3) Tracking:** Tracking is a computer vision technique that involves following objects or people within a video stream over time. This can be done using various techniques such as optical flow, Kalman filters, and deep learning-based tracking algorithms. **4) Customization and development:** MediaPipe provides a flexible framework for developing and customizing machine learning models and computer vision applications. This allows developers to create custom pipelines and models that can be adapted to specific use cases and domains. **5) Landmark Detection:** Landmark detection is a computer vision technique that involves identifying and localizing specific points of interest within an image or video. This is commonly used for facial landmark detection, where specific facial features such as eyes, nose, and mouth are identified and tracked over time. **6) Facial landmark detection:** MediaPipe provides a pre-trained facial landmark detection model that can be used to locate various facial features such as eyes, nose, mouth, and chin. Overall, the project likely involves a combination of these concepts to create a set of Instagram filters using MediaPipe.
SESSION 2 :	**1) MediaPipe** Overview of MediaPipe and its capabilities Comparison with other computer vision libraries and frameworks Installation and setup 3) MediaPipe Face Detection and Tracking Techniques used for face detection and tracking (neural networks, Kalman filtering) Evaluation of face detection and tracking performance Applications of MediaPipe face detection and tracking (face recognition, emotion detection) 4) MediaPipe Pose Detection and Tracking Techniques used for pose detection and tracking (convolutional neural networks) Evaluation of pose detection and tracking performance Applications of MediaPipe pose detection and tracking (fitness tracking, sports analysis) 5) MediaPipe Object Detection and Tracking Techniques used for object detection and tracking (Single Shot Detector) Evaluation of object detection and tracking performance Applications of MediaPipe object detection and tracking (surveillance, traffic analysis) 6) MediaPipe Customization and Development Customizing MediaPipe for specific use cases Developing new components for MediaPipe Integrating MediaPipe with other software and hardware Best practices for MediaPipe development and deployment 7) Facial landmark detection Types of facial landmarks and their importance Facial landmark detection algorithms and techniques Evaluation of facial landmark detection models Applications of facial landmark detection	**1) Overview of MediaPipe and its Capabilities** **2) Comparison with other computer vision libraries and frameworks** **3) Installation and Setup** **4) MediaPipe Hand tracking and Gesture recognition** **5) MediaPipe Face Detection and Tracking**	**Outcome of Learning Convolutional Neural Networks (CNNs):** CNNs are well-suited for tasks such as object recognition, image classification, and facial expression recognition. By using CNNs, it's possible to accurately identify various features within an image, such as facial expressions, which can then be used for various applications. **Outcome of Learning Generative Adversarial Networks (GANs):** GANs can be used to create new images or modify existing ones, which can be useful for creating new filters, effects, and even entire images. By training a GAN on a set of images, it's possible to generate new images that are similar in style, content, or both. **Outcome of Learning Transfer learning:** By reusing pre-trained models, it's possible to save time and resources in training new models. Transfer learning can be used to fine-tune existing models for specific tasks, which can be useful for creating custom filters or effects that are tailored to a specific use case. **Outcome of Learning OpenCV- Image processing and manipulation:** Student will learn how to load and save images, resize and crop images, and adjust brightness, contrast, and other image parameters. Student will learn how to access and modify individual pixels in an image, as well as extract information such as image size and format. Student will learn how to add text to images and manipulate text properties such as font, size, and color.	CNN with MediaPipe alongwith computer vision, and deep learning is best utilized in autonomous driving. In autonomous driving, cameras and other sensors are used to collect data about the surrounding environment, including other cars, pedestrians, traffic lights, and signs. CNNs and MediaPipe are essential tools for developing computer vision and deep learning applications in a wide range of fields, including robotics, gaming, security, healthcare, and entertainment. **Convolutional Neural Networks (CNNs) and MediaPipe** are powerful tools for developing applications that require computer vision and deep learning capabilities. Here are some of the applications of CNNs and MediaPipe: *1)Object Detection and Recognition* *2)Pose Estimation* *3)Facial Landmark Detection* *4)Augmented Reality* *5)Autonomous Vehicles* *6)Healthcare* **Convolutional Neural Networks (CNNs):** CNNs are commonly used in computer vision tasks such as object recognition and image classification. In this project, CNNs may be used for tasks such as facial expression recognition. **Generative Adversarial Networks (GANs):** GANs are a type of neural network architecture that can be used to generate new images based on a given set of training data. In this project, GANs may be used to create new filters or modify existing ones. **Transfer learning:** Transfer learning involves reusing a pre-trained model for a new task. In this project, transfer learning may be used to fine-tune existing models for specific filter effects. Transfer Learning: a technique where a pre-trained model is used as a starting point for a new task. This can help improve the accuracy of the model and reduce the amount of data needed for training. **OpenCV Python :** Learning these topics will give a strong foundation in image processing and manipulation using the Python OpenCV, which can be applied to a wide range of real-world applications, such as image editing, computer vision, and machine learning.
SESSION 3: PROJECT	**Initial Project 1 Using the Session 1 & 2 Topics:** **Project using Mediapipe for instagram filters examples** *- Develop a Python-based project with Mediapipe for implementation.*	**Initial Project 1 Using the Session 1 & 2 Topics:** **Project using Mediapipe for instagram filters examples** ***- Develop a Python-based project with Mediapipe for implementation.***	**PROBLEM STATEMENT:** - In this project, we aim to develop custom Instagram filters using MediaPipe, machine learning, and computer vision techniques. **Approach:** The approach involves using MediaPipe to detect and track facial landmarks in real-time. Then use machine learning and computer vision techniques to apply various digital effects to these facial landmarks. These effects can include changing the color of the user's hair, adding digital makeup, or applying various other visual effects. We will use CNNs to train models that can recognize different facial features and apply the corresponding digital effects. **Deliverables:** At the end of the project, aim to develop a set of custom Instagram filters that can be used by Instagram users. These filters will be developed using MediaPipe, machine learning, and computer vision techniques, and will be capable of applying various digital effects to user's facial features in real-time. **Expected Outcome:** Outcome of this project is to develop custom Instagram filters that can recognize and track facial landmarks and apply various digital effects in real-time. These filters will provide an engaging and interactive user experience to Instagram users, allowing them to create unique and personalized visual content. The project will also serve as a demonstration of the power and potential of MediaPipe, machine learning, and computer vision techniques in developing real-time computer vision applications.	Instagram filters are a popular feature that allows users to apply digital effects to their photos and videos. By leveraging the power of MediaPipe, machine learning, and computer vision, we aim to create filters that can track and recognize human facial landmarks and apply various digital effects in real-time. **MediaPipe:** It can be used to detect and track facial landmarks in real-time, making it an ideal tool for developing custom Instagram filters. **OpenCV:** OpenCV is a popular computer vision library that can be used for various image and video processing tasks, including facial landmark detection and tracking. **Convolutional Neural Networks (CNNs):** CNNs can be used to train models that can recognize different facial features and apply the corresponding digital effects. Convolutional Neural Networks (CNNs): a type of neural network that is commonly used for image classification tasks due to its ability to automatically learn features from raw image data. **Python OpenCV:** OpenCV library can be used for various image processing tasks, including resizing, cropping, and filtering images.
SESSION 4:	**1) Face recognition library** *Face Detection* *Face Encoding* *Face Recognition* **2) Revision of NumPy** *Array creation and manipulation* *Linear algebra operations* *Fourier transforms* *Random number generation* *Broadcasting and vectorization* *Masking and filtering* *Array indexing and slicing* *Saving and loading data* *Mathematical functions* *Statistics operations* **3) Revision of Pandas** *Data manipulation and cleaning* *Data aggregation and grouping* *Time series analysis* *Merging and joining data* *Data visualization* *Data input and output* *Missing data handling* *Reshaping and pivoting data* *Statistical analysis* *Categorical data handling* **4) Revision of Matplotlib** *Line and scatter plots* *Histograms and bar charts* *Box plots and violin plots* *Heatmaps and contour plots* *Polar plots and pie charts* *3D plots and surfaces* *Annotations and text* *Subplots and insets* *Colormap and colorbars* *Styling and customization*		**Outcome of using the Python face recognition library** depends on the specific task at hand. Here are some examples: Face detection: The face detection module of the library can detect the presence and location of faces in an image or video. The outcome of face detection is usually a bounding box around each detected face. Face recognition: The face recognition module of the library can recognize faces in an image or video by comparing them to a set of known faces. The outcome of face recognition is usually a label or name associated with each recognized face. Face clustering: The face clustering module of the library can group similar faces together based on their visual similarity. The outcome of face clustering is usually a set of clusters, each containing faces that are visually similar to each other. **Outcome of Learning NumPy:** This is a library for numerical computing in Python that provides tools for working with arrays, matrices, and linear algebra **Outcome of Learning Pandas:** This is a library for data manipulation and analysis in Python that provides tools for working with structured data. Gain knowledge and skills to work with data manipulation and cleaning **Outcome of Learning Matplotlib:** This is a library for data visualization in Python that provides tools for creating charts, graphs, and other visualizations. Acquire knowledge and skills to create line and scatter plots	**There are many real-world applications of OpenCV, NumPy, Pandas, and Matplotlib, some of which are:** - Facial recognition and tracking systems in security and surveillance - License plate recognition in traffic monitoring and control - Object detection and tracking in autonomous vehicles and drones - Medical image analysis for disease detection and diagnosis **Face recognition technology** has numerous real-world applications in various fields. Some of the most common applications include:Security and surveillance, Identification and authentication, Healthcare, Marketing and advertising.Law enforcement. Overall, face recognition technology has numerous real-world applications and is becoming increasingly popular in various industries. **NumPy** is a Python library used for numerical computing, which is essential for many scientific and engineering applications. It provides a powerful array manipulation and linear algebra functionality, which makes it an important tool for scientific computing. **Pandas** is a Python library used for data manipulation and analysis, which is essential for managing and processing large datasets. It provides tools for data cleaning, data transformation, data aggregation, and merging, making it a powerful tool for data analysis. **Matplotlib** is a Python library used for data visualization, which is important for analyzing and interpreting data. It provides a wide range of visualization tools, including line plots, scatter plots, histograms, bar plots, and heatmaps, making it a powerful tool for data analysis and presentation.

SESSION 5:	**1) TensorFlow** *Deep learning* *Convolutional neural networks (CNN)* *Recurrent neural networks (RNN)* *Transfer learning* *Natural language processing (NLP)* *Reinforcement learning* *Generative adversarial networks (GAN)* *Autoencoders* *Object detection and tracking* *Image classification* *Time series analysis* **2) Keras** *Neural network models* *Sequential models* *Functional models* *Model optimization and tuning* *Regularization techniques* *Advanced activation functions* *Transfer learning* *Image classification* *Object detection and tracking* *Text classification and NLP* **3) Scikit learn** *Regression models* *Classification models* *Clustering algorithms* *Dimensionality reduction techniques* *Model selection and evaluation* *Preprocessing and feature extraction* *Ensemble methods* *Gradient boosting* *Decision trees* *Support vector machines (SVM)*	**1) TensorFlow** ***Deep learning*** ***Convolutional neural networks (CNN)*** ***Recurrent neural networks (RNN)*** ***Transfer learning*** ***Natural language processing (NLP)*** ***Reinforcement learning*** ***Generative adversarial networks (GAN)*** ***Autoencoders*** ***Object detection and tracking*** ***Image classification*** ***Time series analysis*** **2) Keras** ***Neural network models*** ***Sequential models*** ***Functional models*** ***Model optimization and tuning*** ***Regularization techniques*** ***Advanced activation functions*** ***Transfer learning*** ***Image classification*** ***Object detection and tracking*** ***Text classification and NLP*** **3) Scikit learn** ***Regression models*** ***Classification models*** ***Clustering algorithms*** ***Dimensionality reduction techniques*** ***Model selection and evaluation*** ***Preprocessing and feature extraction*** ***Ensemble methods*** ***Gradient boosting*** ***Decision trees*** ***Support vector machines (SVM)***	**Outcome of Learning TensorFlow:** This is an open source machine learning library that is commonly used for deep learning applications, including image classification and object detection. Learn to work with artificial neural networks and deep learning models Acquire knowledge and skills to build and train neural networks Understand how to perform image and speech recognition using deep learning Learn to use TensorFlow for natural language processing Acquire skills to create and optimize deep learning models Understand how to use TensorFlow to deploy models to production Learn how to use TensorFlow to perform transfer learning Know how to work with TensorFlow's data and control flow graphs **Outcome of Learning Keras:** This is a high-level neural networks API that is built on top of TensorFlow and is designed to simplify the process of building deep learning models. Gain knowledge and skills to build and train deep learning models Understand how to perform image classification and object detection Learn to create and train neural networks for natural language processing Acquire skills to use Keras for audio and speech processing Know how to use Keras for regression and classification problems Learn to create and train convolutional neural networks (CNNs) and recurrent neural networks (RNNs) Acquire knowledge to perform transfer learning with Keras Understand how to use Keras to deploy models to production **Outcome of Learning Scikit-learn:** This is a machine learning library that provides tools for data preprocessing, feature selection, and model evaluation. Gain knowledge and skills to work with various machine learning algorithms Learn to preprocess data and perform feature scaling Acquire skills to perform model selection and validation Understand how to perform regression and classification using Scikit-learn Learn to use clustering and dimensionality reduction techniques Acquire knowledge to work with decision trees and random forests Understand how to perform model evaluation and tuning Learn to use Scikit-learn for natural language processing	**Real-world applications of these Python technologies:** - Google's AlphaGo, a program that beat the world champion at the board game Go, which was trained using TensorFlow. - Keras implementation for image classification in the medical field, where it has been used to identify tumors and other abnormalities in medical images. - Scikit-learn is primarily used in the finance industry, where it has been used to predict stock prices and identify patterns in financial data. **TensorFlow** is an open-source machine learning framework developed by Google that is used for building and training machine learning models. It provides a wide range of tools and resources for building neural networks, including support for distributed computing, automatic differentiation, and GPU acceleration. **Keras** is a high-level neural networks API written in Python that runs on top of TensorFlow, designed to simplify and streamline the process of building and training neural networks. It provides a user-friendly interface for designing and training neural networks, allowing researchers and developers to focus on the model architecture rather than the low-level implementation details. **Scikit-learn** is a Python library used for machine learning and data mining. It provides a wide range of tools for data preprocessing, feature selection, model selection, and model evaluation, making it a powerful tool for developing and evaluating machine learning models.
SESSION 6:	**Intermediate Project 2 :** **Project using Computer Vision & Face Recognition Library for Student Attendance System** - *Using Computer Vision & Face Recognition Library for capturing Student Image & Updating Attendance System using the option of "auto click" when video image of student is captured.*		**PROBLEM STATEMENT:** - The aim of this project is to develop an Attendance System using Computer Vision that can automatically detect and recognize the faces of students and mark their attendance based on their presence. The system will be developed using various machine learning and computer vision concepts such as face detection, face recognition, object tracking, image classification, and deep learning, with the help of a convolutional neural network. - The system will use a webcam or any other camera to capture the video stream of students entering the classroom, and then perform face detection to identify the faces in the video stream. Once the faces are detected, the system will use face recognition to recognize the faces and match them with the faces of students in the database. Object tracking will be used to track the movements of the students in the video stream, ensuring that their attendance is marked correctly. - Image classification using a convolutional neural network will be used to classify the students based on their facial features and mark their attendance accordingly. The system will also feature a user interface (UI) that will display the video stream and the attendance records in real-time, allowing the teacher or administrator to monitor the attendance status of the students. - Finally, the system will use data storage and management to store the attendance records securely and ensure easy access and retrieval of attendance data. The attendance records can be exported to Excel or any other format for further analysis or processing.	Developing a computer vision-based attendance system in Python involves using a combination of machine learning, computer vision, UI development, and data storage and management concepts to create a system that can automatically detect and record student attendance using video streams. Python project for using computer vision for an attendance system would involve using several machine learning and computer vision concepts, including: Face detection: This involves using computer vision techniques to detect and locate human faces within an image or video frame. Face recognition: This involves using machine learning algorithms to recognize individual faces and match them against a database of known faces. Object tracking: This involves using computer vision techniques to track the movement of objects within a video stream, such as tracking the movement of a student's face as they move in and out of the camera's field of view. Image classification: This involves using machine learning algorithms to classify images based on their content, such as classifying images of students as either present or absent. Deep learning: This involves using deep neural networks to analyze and process images, such as using a convolutional neural network (CNN) to detect and classify faces. User interface (UI) development: This involves developing a graphical user interface (GUI) for the attendance system, which allows users to interact with the system and view attendance records.
SESSION 7:	**1) Pyautogui** *Introduction to Pyautogui library* *Installing Pyautogui* *Pyautogui functions and methods* *Pyautogui keyboard and mouse control* *Pyautogui screenshot and image recognition* **2) Cursor movement** *Cursor movement basics* *Moving the cursor with Pyautogui* *Cursor movement optimization* *Cursor speed and acceleration control* **3) Error handling** *Introduction to error handling* *Types of errors in Python* *Error handling with try and except* *Debugging with Pyautogui* **4) Input device interface** *Types of input devices* *Input device interface basics* *Touchpad and touchscreen interface with Pyautogui* *Sensor interface with Pyautogui*	**1. Pyautogui** **2. OpenCV Library (opencv-python)** **3. Python Imaging Library** **4. Cursor movement** **5. Error handling** **6. Input device interface**	**Outcome of Learning Pyautogui:** Pyautogui provides cross-platform GUI automation capabilities, allowing developers to automate mouse and keyboard actions, take screenshots, manipulate windows, and more. Pyautogui library is a Python module that enables developers to automate GUI tasks and perform on-screen operations using Python code. Cursor movement refers to the action of moving the mouse pointer on the screen using Pyautogui's mouse control functions. **Outcome of Learning Input device interface:** The input device interface refers to the mechanism through which the program receives input data from the device that detects the finger movement. Depending on the device, this may involve using a specific library or driver to interface with the device and retrieve the finger position data.	**Best real-world applications of PyAutoGUI is in the field of software testing and quality assurance. Also it can be used to automate repetitive tasks such as data entry or form filling.** **PyAutoGUI** is a Python library that enables users to control the mouse and keyboard to automate tasks and interact with graphical user interfaces (GUIs). PyAutoGUI library provides many functions for automating mouse clicks and keystrokes, moving the mouse cursor, taking screenshots, and interacting with on-screen GUI elements. The library also has support for keyboard shortcuts, hotkeys, and virtual keystrokes. **The Cursor Movement feature in PyAutoGUI** allows users to control the position of the mouse cursor on the screen. This is done by specifying the x and y coordinates of the cursor on the screen, and the library moves the cursor to that location. Cursor Movement is one of the core features of PyAutoGUI that enables users to automate tasks and interact with on-screen elements. **Error handling** is an important aspect of any programming language, and PyAutoGUI provides robust error handling mechanisms. When an error occurs in PyAutoGUI, it raises an exception that can be caught and handled by the user's code. The library also provides detailed error messages that can help in debugging and resolving issues. **The Input Device Interface feature in PyAutoGUI** allows users to interact with various input devices such as touchpads, touchscreens, and other sensors that can provide position data. This feature enables users to control the cursor movement based on the input device data. The Input Device Interface is an important feature of PyAutoGUI for creating applications that involve interacting with various input devices.
SESSION 8: PROJECT	**Intermediate Project 3 :** **Project using Pyautogui library to move the cursor based on the finger movement detected by an input device.** - *Using Pyautogui for Color on the Finger and the Cursor moves when the finger moves.*	**Intermediate Project 3 :** **Project using Pyautogui library to move the cursor based on the finger movement detected by an input device.** **- Using Pyautogui for Color on the Finger and the Cursor moves when the finger moves.**	**PROBLEM STATEMENT:** - The objective of this Python project is to develop a system that uses Pyautogui library to move the cursor based on the finger movement detected by an input device. - The project aims to provide a new way of interacting with computers, particularly for people with limited mobility who cannot use traditional input devices such as keyboard and mouse. - The input device used for detecting the finger movement can be a touchpad, touchscreen, or any other sensor that can provide the finger position data. - The project involves developing a program that can interface with the input device, process the finger position data, and use the Pyautogui library to move the cursor on the screen accordingly. - The project will require expertise in Pyautogui programming, input device interfacing, and data processing. - The success of the project will be measured by the accuracy and responsiveness of the cursor movement to the finger movement, as well as the ease of use and accessibility of the system.	The Python project that uses Pyautogui for using Finger and the Cursor moves when the finger moves does not involve any machine learning or computer vision concepts. Pyautogui is a Python library used for automating keyboard and mouse actions. In this project, the finger movement is detected by some input device and the Pyautogui library is used to move the cursor based on the finger movement. The input device used for detecting the finger movement can be a touchpad, touchscreen, or any other sensor that can provide the finger position data. Therefore, this project mainly involves input device interfacing and Pyautogui programming, rather than machine learning or computer vision.
SESSION 9:	**1) Convolutional Neural Networks (CNNs)** **Revision of the topics** *Convolutional Layers,Pooling Layers,Strides,Padding, Filters,Activations,Backpropagation,Dropout,Batch Normalization* **2) Transfer Learning** **Revision of the topics** *Pre-trained models, Fine-tuning,Feature Extraction* **3) Data Augmentation** *Image rotation* *Image flipping* *Image scaling* *Image cropping* *Random transformations* *Random brightness adjustments* *Random contrast adjustments* **4) Image Preprocessing** *Image resizing* *Image normalization* *Image centering* *Image standardization* *Image denoising* *Image sharpening* **5) Loss Functions** *Mean Squared Error (MSE)* *Mean Absolute Error (MAE)* **6) Optimization Algorithms** *Gradient Descent* *Stochastic Gradient Descent (SGD)* *Adam* **7) Performance Metrics** *Accuracy* *Precision* *Recall* *Confusion Matrix* **8) API Integration with TensorFlow Serving** *Introduction to TensorFlow Serving.* *Setting up TensorFlow Serving,* *Creating a TensorFlow model for serving,* *Defining the API interface*	**1. Convolutional Neural Networks (CNNs)** **2. Transfer learning** **3. Data Augmentation** **4. Image preprocessing** **5. Loss Functions** **6. Optimization Algorithms** **7. Performance metrics** **7.1 Accuracy** **7.2 Precision, Recall, and F1 Score** **7.3 Confusion Matrix** **7.4 ROC Curve** **8. API Integration with TensorFlow Serving**	**Outcome of Learning Data Augmentation:** This technique where new training data is created by applying transformations to existing data. This can help increase the size and diversity of the training dataset, leading to better generalization and reduced overfitting. **Outcome of Learning Image Preprocessing:** This set of techniques used to prepare raw image data for use in a model. This may include resizing, normalization, and data augmentation. **Outcome of Learning API Integration:** This process of integrating the trained model into an application or service using an API (Application Programming Interface). This allows other developers to easily use the model in their own applications without needing to know the details of the underlying implementation.. **TensorFlow Serving:** This could cover the basic concepts of TensorFlow Serving, such as what it is, what problems it solves, and how it works. Setting up TensorFlow Serving: This could cover the installation and configuration of TensorFlow Serving, including the different deployment options and how to choose the best one for your use case Creating a TensorFlow model for serving: This could cover the process of creating a TensorFlow model that can be served by TensorFlow Serving, including how to define the model architecture, train it, and export it in a format that can be used by TensorFlow Serving. Defining the API interface: This could cover how to define the API interface for the TensorFlow model, including the input and output formats, and how to specify the data types, shapes, and other parameters.	The combination of these Python topics can help in building and improving the performance of image classification models. Specifically: **Data Augmentation** - Data augmentation is a technique that involves generating additional training data by applying various transformations to the existing training images. This can improve the performance of the classification model by increasing its ability to generalize to new images. **Image Preprocessing** - Image preprocessing techniques are used to prepare the training data for the classification model. This can include techniques such as normalization, resizing, and cropping to ensure that the images are in a suitable format for the model. **Loss Functions -** Loss functions are used to measure the difference between the predicted class probabilities and the true class labels. This is used to update the weights of the neural network during training and improve its ability to correctly classify new images. **Optimization Algorithms -** Optimization algorithms are used to update the weights of the neural network during training in order to minimize the loss function. Common optimization algorithms include Stochastic Gradient Descent (SGD), Adam, and Adagrad. **Performance Metrics** - Performance metrics such as accuracy, precision, recall, and F1 score are used to evaluate the performance of the classification model on a validation dataset. These metrics can be used to compare different models and identify areas for improvement. **API Integration -** Integration with classification APIs such as Google Cloud Vision or AWS Rekognition can simplify the process of developing an image classification system by providing pre-trained models and easy-to-use APIs for deploying and accessing the classification model.
SESSION 10: PROJECT	**Intermediate Project 4 :** **Project using Google Collab use CLASSIFICATION APIs for Rose or Sunflower or Tulip Classification** - *Create Python project that uses the Google Collab platform to access the pre-trained models and online Classification Model APIs.* - *Using online CLASSIFICATION Model APIs for Flowers Classification and predicting with test images to identify whether Rose or Sunflower or Tulips.*	**1. Image classification** **2. Setup** **3. Download and explore the dataset** **4. Load data using a Keras utility** **5. Visualize the data** **6. Configure the dataset for performance** **7. Standardize the data** **8. A basic Keras model** **8.1 Create the model** **8.2 Compile the model** **8.3 Model summary** **8.4 Train the model** **9. Visualize training results** **10. Overfitting** **11. Data augmentation** **12. Dropout** **13. Compile and train the model** **14. Visualize training results** **15. Predict on new data**	**PROBLEM STATEMENT :** Develop a Python-based project that utilizes image classification APIs to classify an image as a Rose or Sunflower or Tulip. - Create Python project that uses the Google Colab platform to access the pre-trained models and Classification APIs. **The project will involve the following steps:** **Data Collection:** Collecting the required images for the classification task. **Data Storage:** Store the required images in a compressed format in cloud storage websites. **Data Preparation:** Preparing the collected images for model training and testing. **Model Selection:** Choosing the pre-trained model from the available options that best suits our image classification task. **Model Training and Fine-tuning:** Fine-tuning the selected pre-trained model using the collected data. **Model Evaluation:** Evaluating the model performance on a test dataset to assess the accuracy and other performance metrics. **API Integration:** Integrating the trained model with the Google Colab Classification APIs to classify new images.	The main objective of this project is to create a reliable and accurate image classification system that can accurately identify Rose or Sunflower or Tulips from an input image. Create Python project that uses the Google Colab platform to access the pre-trained models and online Classification Model APIs.
SESSION 11 , 12 : FINAL PROJECT	**Final Project 5 :** **Project on Developing a machine learning model for OBJECT DETECTION** - *Using basic concepts in MODEL TRAINING including* *i) Data Collection* *ii) Model Selection* *iii)Training Model* *iv) Testing* *v) Prediction of Image with sample.*	**Final Project 5 :** **Project on Developing a machine learning model for OBJECT DETECTION** ***- Using basic concepts in MODEL TRAINING including*** ***i) Data Collection*** ***ii) Model Selection*** ***iii)Training Model*** ***iv) Testing*** ***v) Prediction of Image with sample.***	**PROBLEM STATEMENT:** The objective of this project is to develop a machine learning model for object detection. The project will include the following concepts: **Data Collection:** Collecting a dataset of images for the objects that we want to detect. The dataset should be diverse and representative of the objects in different scenarios. **Model Selection:** Selecting a suitable machine learning model for object detection. Popular models for object detection include YOLO, Faster R-CNN, and SSD. **Training Model:** Training the selected model on the collected dataset. This involves fine-tuning the model on the collected data, adjusting the hyperparameters, and optimizing the model's performance. **Testing:** Evaluating the performance of the trained model on a testing dataset. This involves calculating various performance metrics such as precision, recall, and F1 score. **Prediction of Image with Sample:** After testing, the model will be used to predict objects in new images. This will be demonstrated using a sample image.	Overall, the goal of the project is to build an accurate and robust object detection model that can be used in various applications, such as surveillance, autonomous vehicles, and object tracking. The project aims to develop a machine learning model that can accurately detect objects in images, which can have various applications in fields such as surveillance, autonomous vehicles, and robotics.

	SESSION DESCRIPTION	CHANGES	OUTCOME FROM SESSION
SESSION # 1:	**REVISION OF PYTHON LEVEL 1 :** **-programming,Data types** **-Variables** **-Conditional Statement , if , if else , if elif** **-Loops** **-List Dictionary** **-Functions**	No Changes in Session # 1	**Outcome for Data types** Student could write a program that prompts the user to input a number and then checks whether the number is an integer or a float using the type() function. Student could write a program that creates variables of different data types (e.g. string, integer, float) and then prints out their values and data types using the print() and type() functions. **Outcome for Variables** **Stu**dent could write a program that calculates the area of a rectangle using variables for the length and width of the rectangle. Student could write a program that prompts the user to enter their name and then greets them using a variable to store their name and the print() function. **Outcome for Conditional Statements** **Stu**dent could write a program that prompts the user to enter their age and then tells them whether they are old enough to vote or not using an if statement. Student could write a program that prompts the user to enter a number and then tells them whether the number is positive, negative, or zero using if, elif, and else statements. **Outcome for Loops** **Stu**dent could write a program that uses a while loop to count from 1 to 10 and prints each number on a new line. Student could write a program that uses a for loop to iterate over a list of names and prints out a personalized greeting for each name. Outc**ome for Lists** **Stu**dent could write a program that creates a list of numbers and then calculates the sum and average of the numbers using loops and variables. Student could write a program that prompts the user to enter several words and then adds them to a list, sorts the list alphabetically, and prints out the sorted list. Outc**ome for Dictionaries** **Stu**dent could write a program that creates a dictionary of prices for different items and then calculates the total cost of several items based on their prices using loops and variables. Student could write a program that prompts the user to enter a word and then checks whether the word is in a dictionary of English words using an if statement and a dictionary. Outc**ome for Functions** **Stu**dent could write a program that defines a function to calculate the area of a circle given its radius and then prompts the user to enter a radius and prints out the area using the function. Student could write a program that defines a function to reverse a string and then prompts the user to enter a string and prints out the reversed string using the function.
SESSION # 2:	**Error Handling and Paths in python :** **- Exception handling** **-try and exception block** **-Many exception block** **-finally and raise** **Paths :** **-cd command and dir command** **-Absolute path and Relative path**	Error Handling and Paths in python	**Outcome of Error & Exceprion Handling:** Understand what exceptions are and why they occur in Python programs Use try and except blocks to handle exceptions Use multiple except blocks to handle different types of exceptions Use the finally block to execute code after an exception has occurred Raise your own exceptions using the raise statement Understand the concept of paths in Python Use absolute and relative paths to navigate the file system in Python **Outcome of Paths:** In Python, a path is a string that represents the location of a file or directory on the file system. There are two types of paths in Python: absolute and relative paths. **Outcome of Absolute Path:** An absolute path specifies the location of a file or directory from the root directory of the file system. For example, on a Unix-based system, the absolute path /home/user/file.txt specifies the file.txt file in the user's home directory. **Outcome of Relative Path:** A relative path specifies the location of a file or directory relative to the current working directory. For example, if the current working directory is /home/user, the relative path file.txt specifies the file.txt file in the user's home directory.

SESSION # 3:	**Introduction of Pygame:** **-intoduction of pygame** **-Event** **-Screen and Surface** **-Drawing shapes** **-Setting FPS**	Introduction in Pygame	**Outcome of Introduction to Pygame:** Pygame is a Python module used for game development, multimedia applications, and other graphical user interface (GUI) programs. It provides functions and tools to help create games and other visual applications. **Outcome of Pygame Events:** Pygame events are actions or inputs from the user or other sources, such as keyboard, mouse, or joystick inputs. Pygame events can be handled by defining event handlers or callbacks, which are functions that respond to specific events. **Outcome of Pygame Screen and Surface:** Pygame provides the display module to create a screen surface, which is a rectangular area where you can draw graphics or display images. The Surface object is used to represent an image or a portion of a screen surface. **Outcome of Pygame Drawing Shapes:** Pygame provides functions to draw basic shapes such as rectangles, circles, and lines. These functions take a Surface object and a set of parameters such as color, size, and position to draw the desired shape. **Outcome of Pygame Setting FPS:** FPS stands for frames per second, which is the number of times the screen is updated per second. Pygame provides the Clock class to control the game loop and set the FPS. The tick method of the Clock class can be used to control the FPS by delaying the loop for a certain amount of time.
SESSION # 4:	**Input text box and images (Scaling and rotating):** **-To create text input box** **-To display images** **-To rotate and scalling image**	Input text box and images (Scaling and rotating)	**Outcome of Scaling Input text box:** Student can create an Input text box in Pygame and scale it according to your requirements by using the Pygame Rect object. The Rect object will allow you to set the position and size of the text box on the screen. Student can also use the Pygame Font module to set the font and size of the text inside the text box. To handle user input, you can use the Pygame Key module and update the text box accordingly. **Outcome of Scaling images:** **Stu**dent can scale images in Pygame by using the Pygame Surface object and the Pygame transform module. The Surface object represents the image that you want to scale and the transform module provides methods to scale the image. Student can use the pygame.transform.scale function to scale the image to a new size. To preserve the aspect ratio of the image while scaling, you can use the pygame. transform.smoothscale function. **Outcome of Rotating images:** **Stu**dent can rotate images in Pygame by using the Pygame Surface object and the Pygame transform module. The Surface object represents the image that you want to rotate and the transform module provides methods to rotate the image. Student can use the pygame.transform.rotate function to rotate the image by a specified angle. To rotate an image around a specific point, you can use the pygame.transform.rotate function along with the pygame.Rect.center attribute. **Outcome of Rotating Input text box:** To rotate an Input text box in Pygame, you can create a Surface object with the same size as the text box and then blit the text box onto it. Then you can use the Pygame transform module to rotate the Surface object and the text box will be rotated along with it. To preserve the position of the text box while rotating, you can calculate the new position of the rotated text box using trigonometry.
SESSION # 5:	**Pygame Continued and Introduction to Oops:** **- Animanation in Pygame** **- OOPS based approach** **- Class and object** **- __init__ & self**	Animation in PyGame	**Outcome of PyGame & an object-oriented programming approach:** Understand the basics of creating an animation using PyGame Apply OOP principles to encapsulate behavior and state within a class Understand how to use PyGame to create visual effects and handle user input Understand how to create a class that represents a game object, with methods for updating its position, bouncing off the walls, and changing its appearance Understand how to create a loop that continually updates the position and appearance of game objects, and how to handle user input within the loop Understand how to use PyGame's built-in functions and constants to control the animation, such as setting the frame rate, detecting collisions, and changing colors

SESSION # 6:	**Inheritance and Polymorphism :** **-Inheritance** **-Polymorphisms** **-Super()**	Inheritance and Polymorphism in Pygame	**Outcome of PyGame Inheritance, and PyGame Polymorphism:** Describe the basic principles of object-oriented programming (OOP) and explain how they relate to Python and PyGame. Explain the concept of inheritance in Python and PyGame, and demonstrate how to create a subclass that inherits from a superclass. Discuss the advantages and disadvantages of using inheritance in PyGame, and explain how to avoid common pitfalls when designing class hierarchies. Describe the principles of polymorphism in Python and PyGame, and demonstrate how to use method overriding and overloading to achieve different behaviors in related classes. Develop a basic PyGame program that uses inheritance and polymorphism to create a customizable game character with multiple behaviors and attributes.
SESSION # 7:	**Wallpaper animation** **what we have done so far in the Ball animation**	Wallpaper animation what we have done so far in the Ball animation	TBU
SESSION # 8:	**Pygame - Game 1 as decided in Review Meeting**	Interactive Game using PyGame	TBU
SESSION # 9:	**Pygame - Game 2 as decided in Review Meeting**	Interactive Game using PyGame	TBU
SESSION # 10:	**Pygame - Game 2 as decided in Review Meeting** **Session 10 - banking Application using OOP's** **Create a banking app using OOP's based programming:**	REPLACE - Banking App with Game / SocialMedia App using OOP's based programming.	TBU

SR. No	Week no	Topic name - To be covered	Topic name - Actually Covered
1		Python revision	Basics of python, installing vs code
2		Introduction to ARRAYS library - NUMPY	Functions, indexing, slicing
3		Opeartions on Numpy Arrays	Functions, indexing, slicing (Continued), numpy setup
4		Introduction to IMAGE PROCESSING library - OPENCV	Numpy continued, Pratice problems, reshape, Hstack, Vstack
5		OPEN CV - Dawing shapes and Mouse call back	revison of Numpy, Application of numpy interms of image processing. Introduction to PyAutoGui. In pyautogui completed all the keyboard control things. Next time complete the mouse control for all class
6		OPEN CV - Trackbars and thresholding	overview of what is done till date started with open cv explained what is RGB pixel values introduced to openCV intalling opencv and reading an image using opencv
7		Introduction to PyGame	Pygame is an open-source library that is designed for making video games. It has inbuilt graphics and sound libraries. It is also beginner-friendly, and cross-platform.
8		Pygame Project	PYGAME - Build a Snake Game. Use the steps in the given link to Build Snake game. https://www.geeksforgeeks.org/snake-game-in-python-using-pygame-module/
9		Emotion & Gesture Recognition	Image processing , Pygame , Easy ML
10		Introduction to openCV, Basics of image procesing, reading images. Importance of waitKey	Smart Selfie Using Computer Vision
11		AI Virtual Painter 1	AI Virtual Painter | OpenCV Python | Computer Vision Without MEDIAPIPE
12		AI Virtual Painter 2	

SR. No	Project Name	Project Details / Project Link
1	**Pygame Project - Snake Game**	PYGAME - Build a Snake Game. Use the steps in the given link to Build Snake game. https://www.geeksforgeeks.org/snake-game-in-python-using-pygame-module/
2	**Emotion & Gesture Recognition**	Human-Emotion-and-Gesture-Detector Understanding how to build a human emotion and gesture detector with Deep Learning **GITHUB LINK:** https://github.com/Bharath-K3/Human-Emotion-and-Gesture-Detector
3	**Smart Selfie Using Computer Vision**	Smile-Detector OpenCv-Python to develop a computer vision based smart selfie application that automatically takes pictures when you smile **GITHUB LINK:** https://github.com/Tinzyl/Smile-Detector
4	**AI Virtual Painter | OpenCV Python | Computer Vision**	AI-Virtual-Painter : implement real-time hand tracking and enable drawing on the screen with the index finger. **GITHUB LINK:** https://github.com/PedroRavaglia/AI-Virtual-Painter
5	**Pygame - PONG Game**	GITHUB LINK: https://github.com/OSSpk/Awesome-Python-Games
6	**Pygame - SPACE SHIP & Asteroids Game**	GITHUB LINK: https://github.com/OSSpk/Awesome-Python-Games
7	**OPENCV - Chrome Dinosaur Game controlled by real-time eye blinks**	GIHUB LINK https://github.com/coding-ai/eyeBlinkedTREX
8	**Pygame -- space shooter - Cosmic Heat**	GIHUB LINK: https://github.com/Dave-YP/cosmic-heat-pygame
9	**Blur-Detection-Web-App (Using Flask)**	GITHUB LINK:: https://github.com/Furkan-Gulsen/Blur-Detection-Web-App
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		

Introduction to openCV, Basics of image procesing, reading images.
Importance of waitKey

ARYA VIDYA MANDIR - HLP / SAP CURRICULUM 2022-2023

TOTAL 24 SESSIONS - ONCE A WEEK
SEM 1 - COURSE - AI/ML WITH DATA ANALYTICS - 10 SESSIONS
SEM 1 - COURSE - BLOCKCHAIN / NFT - 4 SESSIONS
SEM 2 - COURSE - CIRCUIT APP INTERFACE - 10 SESSIONS

SEM 1 - AI/ML WITH DATA ANALYTICS

Month	Week Range	S no	Topic name	Concepts learnt
June	13-17	1	Undertsanding Python anaylsis packages - ADVANCED NUMPY	- Libraries - ADVANCED NUMPY - Array creation, modification - vectors, matricies, 1 and 2 D arrays - Array operations - slicing, fucntions, arithmetic operations - Write a program to create Chess pattern of 1 and 0 using NUMPY
June	20-24	2	Undertsanding Python anaylsis packages - MATPLOTLIB	- Graphical represantations using MATPLOTLIB - Understanding types of graphs - line, scatter, bar, pie, histogram - Plotting graphs - line, scatter, barchart, histogram - Plotting combined signals, Histogram of random nos - Write a program to create 3-phase plot using MATPLOTLIB
June	27-30	3	Machine learning using Python program - Packages and Data sets	- Explore ML package - SCIKIT LEARN - Classes and fucntions related to ML algorithm - Exploring Data sets - KAGGLE, UCI repository - Sample data anaylsis
July	4-8	4	Machine learning using Python program - REGRESSION MODEL	- REGRESSION based ML on OCEAN SALINITY DATASET - data import anaylsis, creation of REGRESSION model - perform TRAIN TEST SPLIT, train the model with INPUTS and OUTPUTS , performance anaylsis
July	11-15	5	Machine learning using Python program - CLASSIFICATION MODEL	- CLASSIFICATION based ML on WHEAT dataset - import data- perform filtering, check corelation, create REGRESSION model, performance anaylsis - check, repeat with variations and monior performance
July	18-22	6	Machine Learning Process Flow	- Advanced configurability achieved in programming mode - 4 blocks of progarmming - upload data, set train set ratio, create and test ML model - Understand requirements - variable name compulsion, optional algorithm, additional per processing
July	25-29	7	AI TOOL - programming mode 1	- REGRESSION bases ML based on WEATHER DATA using programming mode - Write code to upload OG dataset - Write code to select INPUT/OUTPUT parameters, train test split ratio - Write a code to create and train ML model - Test the AI model using Validation data - Change parameters and observe differences in model performance - RANDOM STATE, MODEL PARAMETERS, DIFFERENT ALGORITHMS
Aug	1-5	8	AI TOOL - programming mode 2	- CLASSIFICATION based ML on IRIS FLOWER data using programming mode - Write code to upload OG dataset - Write code to select INPUT/OUTPUT parameters, train test split ratio - Write a code to create and train ML model - Test the AI model using Validation data - Change parameters and observe differences in model performance - RANDOM STATE, MODEL PARAMETERS, DIFFERENT ALGORITHMS
Aug	**8-12**	**9**	**Capstone project 1 - Greenhouse gas forecasting**	**- Predict future levels of greenhouse gases using NUMPY, SCIKIT LEARN libraries**
Aug	17,29,30	**10**	**Capstone project 2 - Web Scraping**	**- How to Collect and process data from multiple sources**
Aug	**22-26**		**BUFFER**	

SEM 2 - BLOCKCHAIN / NFT / METAVERSE

Month	Week Range	S no	Topic name	Concepts learnt
Sept	19-23	1	Introduction to BlockChain technology	- Intro and Discussion on Distributed LEDGER used as a storage in public databases
Sept	26-30	2	Structure / Industry Applications	- How blockchain works - Recording transactions as block of data - Forming CHAIN OF DATA - Industry applications - Healthcare, Government, Media and Advertising, Telecommunications
Oct	3-7	3	NFT - Introduction	- Representation of digital / non digital assets
Oct	10-14	4	Structure / Industry Applications	- Platforms used to create NFT - NFT applications - art, gaming, collectibles, personal identity management

SEM 2-CIRCUIT APP INTERFACE

Month	Week Range	S No	Topic Name	Concepts Learnt
Nov	7-11	1	LED control	-Introduction to Bluetooth technology -LED control with Bluetooth techonology
Nov	14-18	2	MOTOR CONTROL	-Introduction To BOT -Learn About Motor -Learn About Bluetooth technology -Control The Motor With android phone
Nov	21-25	3	Assembly With Chasis	-Construction Of Basic Robot -Connect The Motor And Wheels Caster - Connet Wheels To Chasis
Nov	28-30	4	Wire connection	-Motor Connection To Autobot PCB module -Connection of motor and bluetooth with Arduino UNo -coding part of Robot
Dec	5-9	5	Android controlled robot	-control the robot with help of Moblie application - move the bot in forward. backward,left , right and stop
Dec	12-16	6	Gesture controlled robot	-Execute same robot with help of gyro control - learn gyro sensor in moblie applications
Dec	19-23	7	Line Following Robot-1 attachment of IR	-make attachment of IR sensor for robot -Connection of IR and bluetooth with Arduino UNo
Jan	2-6	8	Line Following Robot-2 Testing and Execution	-Coding part of Line follower robot -Execute the line follower robot
Jan	9-13	**9**	**CAPSTONE PROJECT 1 - Obstacle Avoidance Robot-1 Attachment of Ultrasonic sensor**	**-make attachment of Distance sensor for robot -Connection of ultrasonic sensor and bluetooth with Arduino UNo**
Jan	16-20	**10**	**Obstacle Avoidance Robot-2 Testing and Executions**	**-Coding part of obstale avoidance robot -learn the logic of IF and ELSE condiition in coding**
Jan	23-27		**BUFFER**	

Sr no.	Topic name	Concept Learnt	
1	Light Sensor	-Introduction to Arduino with Python -add/display images in python -Take the data arduino serial monitor to Python platform -according to LDR Values display images	Ready
2	LED control	-Control the LED with Python terminal	Ready
3	Tone generator	-Make the GUI in python to control the buzzer as Sa re ga ma	Ready
4	RGB	-Make the GUI in python to control the LED light	Ready
5	Neopixel	Control the neopixel with help of python	Ready
6	Controlling servo motor	-Control the Servo motor with help of Slider	Ready
7	Temperture display	Display the Emoji as per value of temperture	Ready
8	Radar 1	-control motion of servo motor using arduino -calculate distance using ultrasound sensor-passing value on serial monitor	Ready
9	Radar 2	-Read value from serial monitor and display corrosponding point on map using python	
10	Capstone 1 - part-1		
11	Capstone 1 -part-2		

[1] Link 1:

```
---------
import numpy as np
import pandas as pd
import matplotlib.pylab
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
import warnings
import itertools
import statsmodels
import statsmodels.api as sm
from statsmodels.tsa.stattools import coint, adfuller
```