Prerequisite for AI/ML Bootcamp

1. Download Anaconda Navigator

   Link :https://www.anaconda.com/download

- Python Basics

Students are suggested to go through these library for at least once before coming to the bootcamp.

- Numpy
- Pandas
- Matplotlib

Some youtube channels you can go to for the hands on tutorials.

1. Campusx : https://www.youtube.com/@campusx-official/playlists
2. Codebasics : https://www.youtube.com/@codebasics/playlists
3. Krish naik : https://www.youtube.com/@krishnaik06/playlists

Cheatsheet for Numpy

NumPy, a fundamental package for scientific computing in Python:

import NumPy

- `import numpy as np`

Creating Arrays

- `np.array([1, 2, 3])` - Create a 1D array
- `np.zeros(3)` - Array of zeros
- `np.ones(3)` - Array of ones
- `np.empty(3)` - Uninitialized array
- `np.arange(4)` - Array of consecutive integers
- `np.linspace(0, 10, 5)` - Array of 5 evenly spaced values from 0 to 10

Array Attributes

- `arr.ndim` - Number of array dimensions
- `arr.shape` - Array dimensions
- `arr.size` - Total number of elements
- `arr.dtype` - Data type of the elements

Array Indexing and Slicing

- `arr[2]` - Access element
- `arr[1:4]` - Slice from index 1 to 3
- `arr[:2]` - Slice from start to index 1
- `arr[1:4:2]` - Slice from index 1 to 3 with step 2
- `arr[::-1]` - Reverse the array

Array Operations

- `arr + arr` - Element-wise addition
- `arr - arr` - Element-wise subtraction
- `arr * arr` - Element-wise multiplication
- `arr / arr` - Element-wise division
- `np.sqrt(arr)` - Element-wise square root
- `np.exp(arr)` - Element-wise exponentiation

Aggregation Functions

- `arr.sum()` - Sum of all elements

- `arr.min()` - Minimum element
- `arr.max()` - Maximum element
- `arr.mean()` - Mean of elements
- `arr.std()` - Standard deviation

Reshaping and Transposing

- `arr.reshape(3, 2)` - Reshape array to 3x2
- `arr.T` - Transpose the array

Boolean Indexing

- `arr[arr > 5]` - Elements greater than 5

Linear Algebra

- `np.dot(arr1, arr2)` - Dot product
- `np.linalg.inv(matrix)` - Inverse of a matrix
- `np.linalg.eig(matrix)` - Eigenvalues and eigenvectors

Random Number Generation

- `np.random.rand(3)` - Three random numbers from a uniform distribution [0, 1)
- `np.random.randn(3)` - Three random numbers from a normal distribution
- `np.random.randint(1, 10, 3)` - Three random integers between 1 and 10

Cheatsheet for Pandas

Pandas, a powerful data manipulation library in Python:

Import Pandas

- `import pandas as pd`

Reading Data

- `pd.read_csv('file.csv')` - Read data from a CSV file
- `pd.read_excel('file.xlsx')` - Read data from an Excel file
- `pd.read_json('file.json')` - Read data from a JSON file

Creating DataFrames

- `pd.DataFrame(data)` - Create a DataFrame from a dictionary or array

Viewing Data

- `df.head()` - View the first 5 rows
- `df.tail()` - View the last 5 rows

- `df.sample(n)` - Randomly select n rows

Data Inspection

- `df.info()` - Summary of the DataFrame
- `df.describe()` - Statistical summary
- `df.shape` - Number of rows and columns
- `df.columns` - Column names
- `df.dtypes` - Data types of columns

Selecting Data

- `df['column']` - Select one column
- `df[['col1', 'col2']]` - Select multiple columns
- `df.iloc[rows, columns]` - Select data by row and column numbers ● `df.loc[rows, 'column']` - Select data by row labels and column names

Filtering Data

- `df[df['column'] > value]` - Rows where the column is greater than a value
- `df[(df['col1'] > value) & (df['col2'] == 'text')]` - Using multiple conditions

Handling Missing Data

- `df.dropna()` - Drop rows with missing values
- `df.fillna(value)` - Fill missing values with a specified value

Data Manipulation

- `df['column'].map(func)` - Apply a function to a column
- `df.apply(func, axis)` - Apply a function along an axis
- `df.groupby('column')` - Group data
- `df.pivot_table(values, index, columns)` - Create a pivot table

Sorting Data

- `df.sort_values(by='column')` - Sort by the values of a column

Merging and Concatenating

- `pd.concat([df1, df2])` - Concatenate DataFrames
- `pd.merge(df1, df2, on='column')` - Merge DataFrames on a key

Exporting Data

- `df.to_csv('file.csv')` - Write to a CSV file
- `df.to_excel('file.xlsx')` - Write to an Excel file
- `df.to_json('file.json')` - Write to a JSON file

DateTime Operations

- `pd.to_datetime(df['column'])` - Convert a column to DateTime ● `df.set_index('DateTimeColumn')` - Set a DateTime column as the index

String Operations

- `df['column'].str.lower()` - Convert strings to lowercase
- `df['column'].str.upper()` - Convert strings to uppercase
- `df['column'].str.contains('text')` - Check if string contains a pattern

Cheatsheet for Matplotlib

Matplotlib, a popular Python library for creating static, interactive, and animated visualizations:

Import Matplotlib

- `import matplotlib.pyplot as plt`

Basic Plot

- `plt.plot(x, y)` - Plot y versus x as lines and/or markers
- `plt.show()` - Display the plot

Figure and Axes

- `fig, ax = plt.subplots()` - Create a figure and a set of subplots

Plotting Data

- `ax.plot(x, y)` - Plot data on the axes
- `ax.scatter(x, y)` - Scatter plot
- `ax.bar(x, height)` - Bar chart
- `ax.hist(data)` - Histogram
- `ax.boxplot(data)` - Box plot

Customizing Plots

- `plt.title('Title')` - Add a title
- `plt.xlabel('X-axis Label')` - Label the x-axis

- `plt.ylabel('Y-axis Label')` - Label the y-axis
- `plt.xlim([xmin, xmax])` - Set x-axis limits
- `plt.ylim([ymin, ymax])` - Set y-axis limits
- `plt.xticks(ticks, labels)` - Set x-axis tick marks
- `plt.yticks(ticks, labels)` - Set y-axis tick marks
- `plt.legend()` - Add a legend
- `plt.grid(True)` - Add a grid

Multiple Plots

- `fig, axs = plt.subplots(nrows, ncols)` - Create a grid of subplots
- `axs[row, col].plot(x, y)` - Plot on specific subplot

Plot Styles and Colors

- `plt.style.use('style_name')` - Use a specific plot style
- `plt.plot(x, y, color='color_name')` - Specify color of the plot
- `plt.plot(x, y, linestyle='line_style')` - Specify line style

Saving Figures

- `plt.savefig('filename.png')` - Save the current figure

Error Bars

- `plt.errorbar(x, y, yerr=error)` - Plot y with error bars

Pie Charts

- `plt.pie(sizes, labels=labels)` - Pie chart of sizes

3D Plots

- `from mpl_toolkits.mplot3d import Axes3D`
- `fig = plt.figure()`
- `ax = fig.add_subplot(111, projection='3d')` - 3D axes
- `ax.plot_surface(X, Y, Z)` - 3D surface plot

Heatmaps and Image Plots

- `plt.imshow(data)` - Display data as an image (e.g., heatmap)

Contour Plots

- `plt.contour(X, Y, Z)` - Contour plot
- `plt.contourf(X, Y, Z)` - Filled contour plot

Customizing Axes

- `ax.set_xlim([xmin, xmax])` - Set x-axis limits for axes object
- `ax.set_ylim([ymin, ymax])` - Set y-axis limits for axes object
- `ax.set_title('Title')` - Set title for axes object