UNIT-4

LIBRARIES AND DATASETS

4.1. Jupyter Installation and Use

- Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
- Jupyter has support for over 40 different programming languages and Python is one of them. Python is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook itself.

Jupyter Notebook can be installed by using either of the two ways described below:

- **Using Anaconda:**

  Install Python and Jupyter using the Anaconda Distribution, which includes Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science. To install Anaconda, go through How to install Anaconda on windows? and follow the instructions provided.

- **Using PIP:**

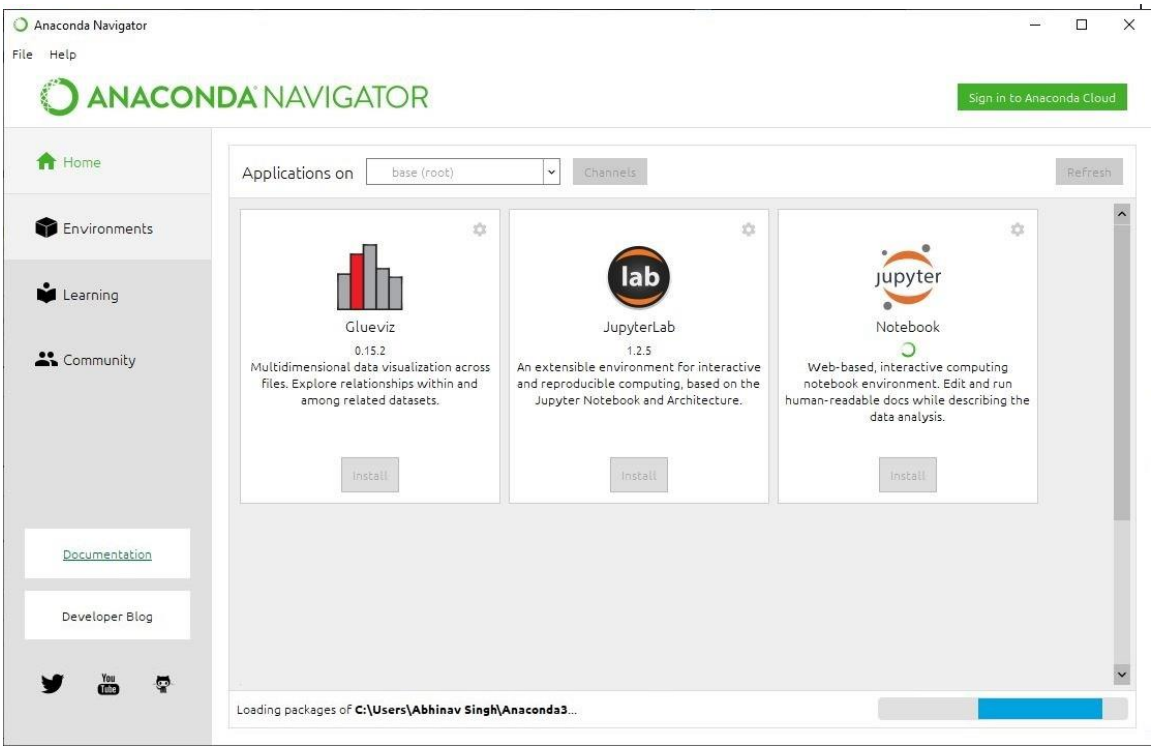
  Install Jupyter using the PIP package manager used to install and manage software packages/libraries written in Python. To install pip, go through How to install PIP on Windows? and follow the instructions provided.

Installing Jupyter Notebook using Anaconda:

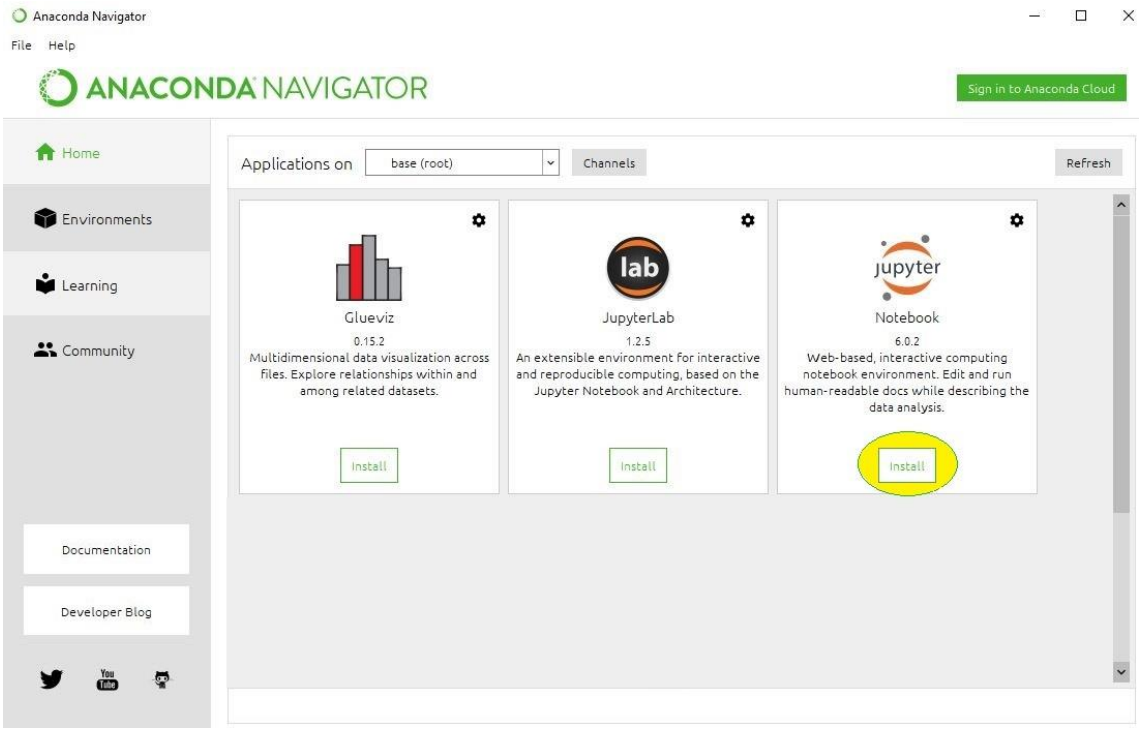
Anaconda is an open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing. Anaconda works for R and python programming language. Spyder(sub-application of Anaconda) is used for python. Opencv for python will work in spyder. Package versions are managed by the package management system called conda.

To install Jupyter using Anaconda, just go through the following instructions:

- **Launch Anaconda Navigator:**

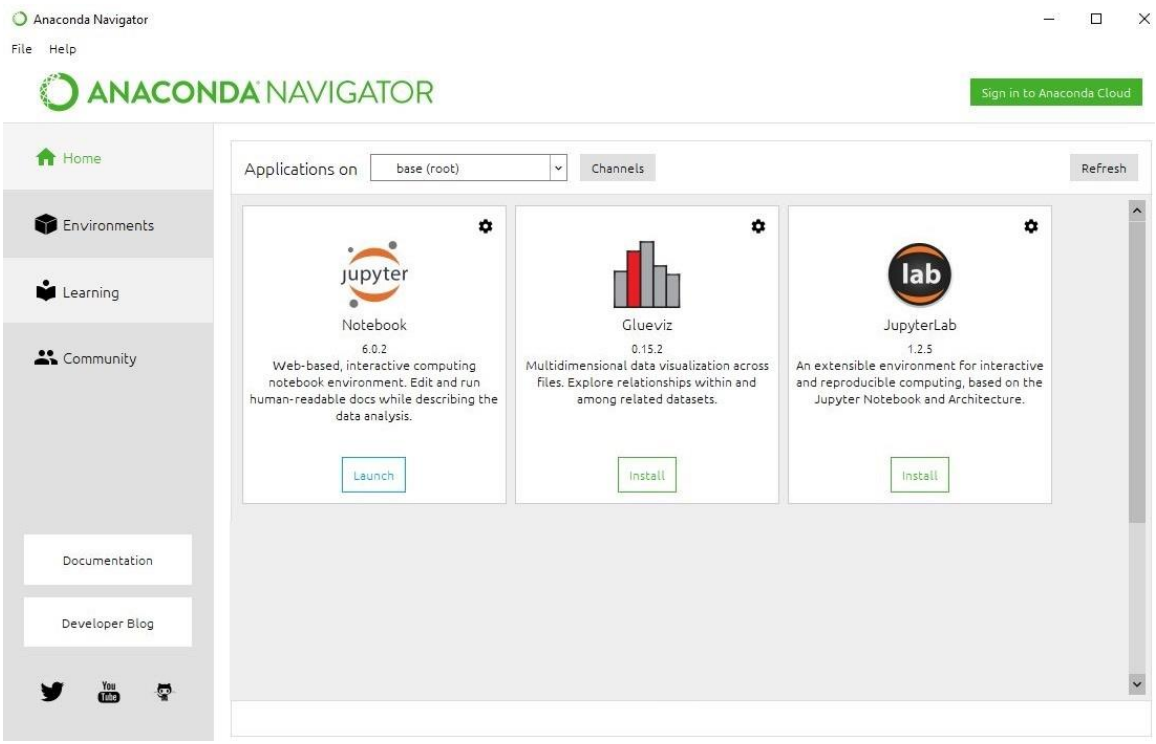


- **Click on the Install Jupyter Notebook Button:**



- **Loading Packages:**
- **Finished Installation:**

- **Launching Jupyter:**



- **Installing Jupyter Notebook using pip:**

  PIP is a package management system used to install and manage software packages/libraries written in Python. These files are stored in a large "on-line repository" termed as Python Package Index (PyPI).

  pip uses PyPI as the default source for packages and their dependencies.

  To install Jupyter using pip, we need to first check if pip is updated in our system. Use the following command to update pip:

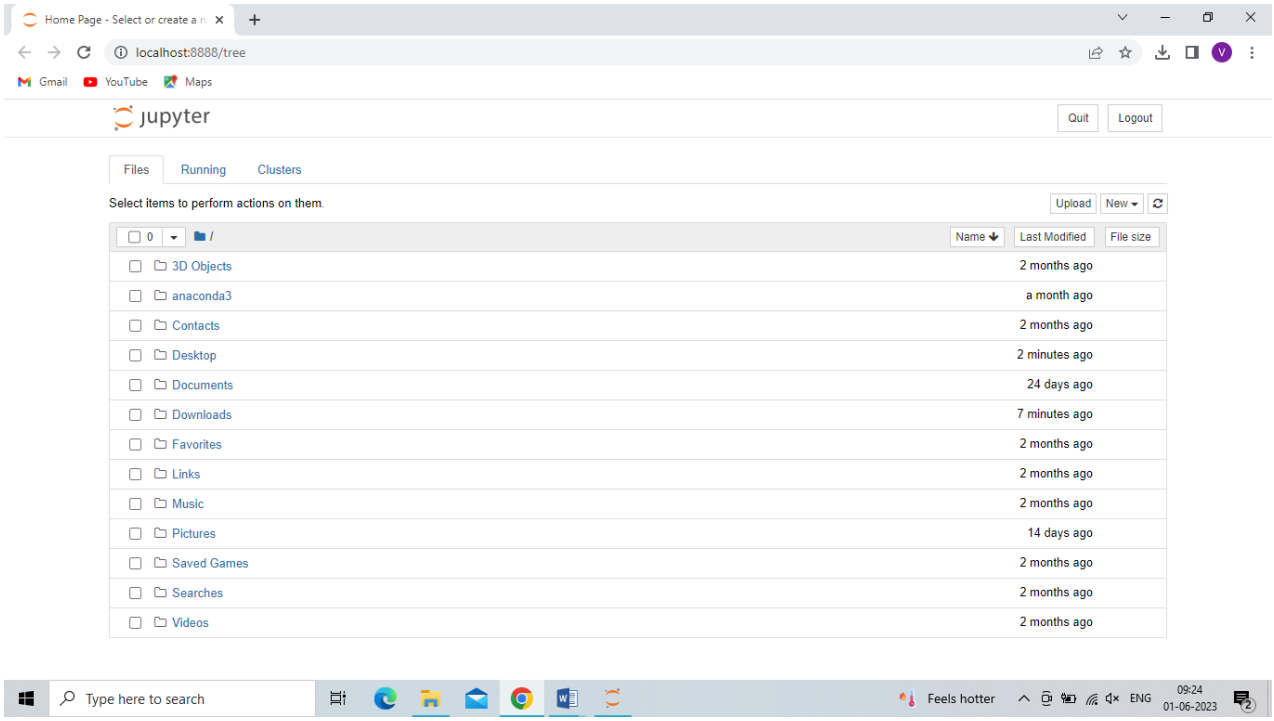
  python -m pip install --upgrade pip

- **Launching Jupyter:**
  Use the following command to launch Jupyter using command-line:
  jupyter notebook

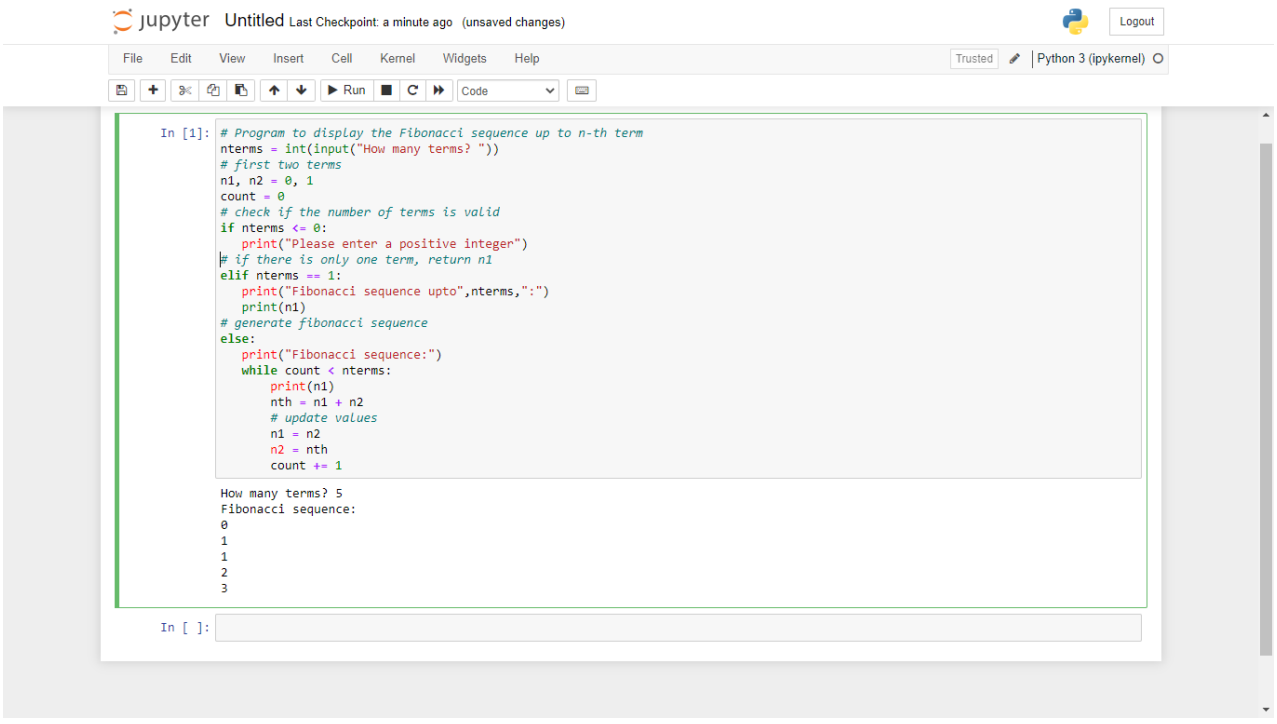
Jupyter Uses:

- Programming Practice.
- Collaborating Across Projects and Tools.
- Data Organization and Cleaning.
- Data Visualization and Sharing.

- Teaching Data Science Skills.



Run code in Jupyter



4.2. Datasets: Kaggle

- A dataset is a collection of structured or unstructured data that is organized and stored for analysis, processing, and information retrieval. It can be thought of as a set of observations or records, each representing a distinct entity or item.

- Datasets can come in various formats, including spreadsheets, databases, text files, images, videos, or any other form of digital information. They can be created for specific purposes, such as scientific research, machine learning, data analysis, or business intelligence.
- In the context of machine learning, datasets play a crucial role. They are used to train, validate, and test machine learning models. A typical machine learning dataset consists of input features or variables and their corresponding target or output values. By analyzing and learning from the patterns and relationships within the dataset, machine learning models can make predictions or perform tasks based on new, unseen data.
- Datasets can be obtained from various sources, including public repositories, research institutions, government agencies, private companies, or by collecting data directly through surveys, experiments, or sensors. It is important to ensure that datasets are representative, reliable, and appropriately processed to yield meaningful and accurate results in any data analysis or machine learning project.
- Inside Kaggle you'll find all the code & data you need to do your data science work. Use over 50,000 public datasets and 400,000 public notebooks to conquer any analysis in no time.

4.3. Python Libraries:

Python has a rich ecosystem of libraries and packages that provide additional functionality and extend the capabilities of the language. Here are some popular Python libraries across different domains:

- **NumPy** (Numerical Python) is a fundamental library for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. Here are some key features and concepts related to NumPy:
    - **ndarray:** NumPy's main object is the ndarray (n-dimensional array). It represents a grid of values, all of the same data type, indexed by a tuple of non-negative integers. The dimensions of an array are called axes, and the number of axes is known as the array's rank.
    - **Array Creation**: NumPy provides various functions to create arrays. For example, you can create an array from a Python list using the numpy.array() function. Other functions like numpy.zeros(), numpy.ones(), and numpy.arange() can be used to create arrays initialized with zeros, ones, or a range of values.
    - **Array Operations**: NumPy enables efficient element-wise operations on arrays, such as arithmetic operations (addition, subtraction, multiplication, division), exponentiation, and trigonometric functions. These operations are vectorized, meaning they are applied to the entire array rather than individual elements, resulting in improved performance.
    - **Universal Functions (ufuncs):** NumPy provides a large collection of universal functions (ufuncs) that operate element-wise on arrays, including mathematical functions (e.g.,

numpy.sin(), numpy.cos()), statistical functions (e.g., numpy.mean(), numpy.std()), logical operations (e.g., numpy.logical_and(), numpy.logical_or()), and more.

- **Indexing and Slicing:** NumPy allows indexing and slicing of arrays to access specific elements, rows, columns, or subarrays. This enables efficient data extraction and manipulation. Indexing starts at 0, and negative indices can be used to access elements from the end of the array.
- **Shape Manipulation:** NumPy provides functions to change the shape or dimensions of arrays. Functions like numpy.reshape(), numpy.flatten(), and numpy.transpose() allow you to manipulate the shape and layout of arrays according to your needs.
- **Broadcasting:** Broadcasting is a powerful feature of NumPy that allows operations between arrays of different shapes, as long as they are compatible. Broadcasting enables implicit element-wise operations between arrays with different sizes, reducing the need for explicit loops and improving code readability.
- **Linear Algebra:** NumPy provides a rich set of functions for linear algebra operations, such as matrix multiplication (numpy.matmul() or @ operator), matrix inversion (numpy.linalg.inv()), eigenvalue computation (numpy.linalg.eig()), and more.
- **Random Number Generation:** NumPy includes functions to generate random numbers from various probability distributions. The numpy.random module provides functions like numpy.random.rand(), numpy.random.randn(), and numpy.random.randint() for generating random arrays or numbers.
- **Performance Optimization:** NumPy's underlying implementation is in C, which makes it highly optimized for performance. It provides efficient and fast numerical operations, making it a preferred choice for scientific computing and numerical computations.
- NumPy is widely used in various domains, including scientific research, data analysis, machine learning, and computational mathematics. It serves as a foundational library for many other libraries and packages in the Python ecosystem, enabling efficient data manipulation, numerical computations, and advanced mathematical operations.

- **Pandas:** pandas is a powerful library for data manipulation and analysis. It offers data structures such as dataframes and Series, which enable easy handling, cleaning, and exploration of structured data. Here are some key features and concepts related to Pandas:
  - **DataFrame:** The central data structure in Pandas is the DataFrame. It represents a two-dimensional table of data with labeled rows and columns. Each column in a DataFrame is called a Series, which is a one-dimensional labeled array. DataFrames can contain heterogeneous data types (e.g., numbers, strings, dates) and handle missing values.
  - **Data Input and Output:** Pandas provides functions to read data from various file formats, such as CSV, Excel, SQL databases, JSON, and more. The pandas.read_csv() function, for

example, is commonly used to load data from a CSV file into a DataFrame. Similarly, Pandas offers functions like to_csv(), to_excel(), and to_sql() to export data from a DataFrame to different formats.

- **Data Selection and Manipulation:** Pandas offers powerful indexing and selection capabilities to extract, filter, and manipulate data in a DataFrame. You can use column names or labels, as well as numerical indexing, to access specific data. Functions like loc[] and iloc[] allow for label-based and integer-based indexing, respectively. Operations like filtering rows based on conditions, selecting specific columns, or applying functions across the data are also straightforward with Pandas.
- **Data Cleaning and Preprocessing:** Pandas provides a variety of functions to handle missing data, duplicate values, and outliers. Functions like dropna(), fillna(), and duplicated() assist in removing or imputing missing values, identifying duplicates, and handling outliers. Additionally, Pandas offers functions for data transformation, such as sorting, merging, reshaping, and aggregating data, to prepare it for analysis.
- **Descriptive Statistics:** Pandas enables the calculation of various descriptive statistics on numerical data in a DataFrame. Functions like mean(), median(), std(), min(), and max() provide summary statistics for columns or across the entire DataFrame. The describe() function generates a comprehensive summary of the distribution of each column, including count, mean, standard deviation, minimum, quartiles, and maximum values.
- **Data Visualization:** Pandas integrates well with other Python visualization libraries, such as Matplotlib and Seaborn, allowing for easy data visualization. Pandas provides built-in plotting functions, including line plots, bar plots, histograms, scatter plots, and more. These functions simplify the process of creating basic visualizations directly from a DataFrame or Series.
- **Time Series Functionality:** Pandas has extensive support for time series data, making it suitable for analyzing and manipulating temporal data. It provides specialized data structures, such as the DatetimeIndex and PeriodIndex, along with functions to resample, interpolate, shift, and aggregate time series data. Pandas also offers date range generation and supports time zone handling.
- **Grouping and Aggregation:** Pandas allows for grouping data based on one or more variables and performing aggregations on those groups. The groupby() function is used to create groups, and then aggregate functions like sum(), mean(), count(), and apply() can be applied to calculate statistics on each group. This functionality is particularly useful for analyzing data by categories or performing group-level operations.
- **Integration with NumPy and Scikit-Learn:** Pandas seamlessly integrates with other libraries, such as NumPy and scikit-learn. It can convert between Pandas data structures and NumPy

arrays efficiently. This integration enables smooth data transformations and preprocessing before feeding the data into machine learning models built with scikit-learn.

- **Matplotlib:** Matplotlib is a plotting library that provides a wide range of options for creating static, animated, and interactive visualizations. It is widely used for generating charts, histograms, scatter plots, and other types of plots.
    - **Pyplot Interface:** Matplotlib's pyplot module provides a MATLAB-like interface for creating and customizing plots. It enables users to quickly generate basic plots by using functions like plot(), scatter(), bar(), hist(), and imshow(). Pyplot functions allow for easy customization of plot elements such as labels, titles, legends, colors, and line styles.
    - **Figure and Axes:** A Figure is the top-level container that holds all the elements of a plot, including one or more Axes objects. Axes represent an individual plot with a specific set of coordinates. Multiple Axes can be arranged within a single Figure, allowing for the creation of subplots or complex layouts. The Figure and Axes structure provides fine-grained control over plot composition and appearance.
    - **Line Plots:** Matplotlib offers various types of line plots, including line graphs, step plots, and scatter plots. Line plots are commonly used for visualizing trends, time series data, and continuous variables. Users can specify line styles, markers, colors, and annotations to enhance the visual representation of the data.
    - **Bar Plots:** Bar plots in Matplotlib are useful for comparing different categories or groups. They can represent both categorical and numerical data. Matplotlib supports vertical and horizontal bar plots, grouped and stacked bar plots, and customization of bar widths, colors, and labels.
    - **Histograms:** Histograms are effective for visualizing the distribution of a continuous variable. Matplotlib allows users to create histograms with customizable bin sizes, bin edges, and normalization options. Histograms can provide insights into the underlying data distribution, identify outliers, and highlight key statistical properties.
    - **Scatter Plots:** Scatter plots are ideal for visualizing relationships between two continuous variables. Matplotlib provides functions to create scatter plots with options for marker styles, sizes, colors, and transparency. Scatter plots are useful for identifying patterns, clusters, or correlations in data.
    - **Pie Charts:** Matplotlib supports the creation of pie charts to display proportions or percentages of categorical data. Pie charts can be customized with colors, labels, and explode options to emphasize specific slices. Matplotlib also allows for exploded pie charts and donut charts.
    - **Annotations and Labels:** Matplotlib enables users to add annotations, text, and labels to plots. Annotations can be used to highlight specific data points, provide additional context, or add

descriptions. Matplotlib supports custom text positioning, font styles, arrow annotations, and LaTeX rendering for mathematical expressions.

- **Customization and Styling:** Matplotlib provides extensive customization options to fine-tune the appearance of plots. Users can control axis limits, ticks, labels, grid lines, legends, color maps, line styles, and plot backgrounds. Matplotlib supports the use of style sheets to apply predefined visual themes or create custom styles.
- **Saving and Exporting:** Matplotlib allows users to save plots in various formats, including PNG, JPEG, PDF, SVG, and more. Plots can be saved programmatically or interactively using the GUI provided by Matplotlib. This feature facilitates the integration of Matplotlib-generated plots into reports, presentations, or web applications.

- **Scikit-learn:** scikit-learn is a popular machine learning library that provides various algorithms and tools for classification, regression, clustering, dimensionality reduction, and model selection. It also offers utilities for data preprocessing, model evaluation, and cross-validation.
- **Tensorflow:** tensorflow is an open-source library primarily used for deep learning and neural network-based computations. It provides a flexible platform for building and training machine learning models, especially those involving large-scale datasets and complex architectures.
- **Keras:** Keras is a high-level deep learning library that runs on top of tensorflow. It offers a user-friendly interface for defining and training neural networks, making it easier to experiment and iterate on different models.
- **Pytorch:** pytorch is another popular deep learning library that emphasizes flexibility and dynamic computation graphs. It provides efficient tensor operations and automatic differentiation, making it suitable for research and development in the field of deep learning.
- **Opencv:** opencv (Open Source Computer Vision Library) is a powerful library for computer vision and image processing tasks. It offers a wide range of functions and algorithms for image and video analysis, object detection, feature extraction, and more.
- **NLTK:** NLTK (Natural Language Toolkit) is a library for natural language processing. It provides tools and resources for tasks such as tokenization, stemming, tagging, parsing, and sentiment analysis, making it valuable for text-based applications.
- **Django:** Django is a high-level web framework that simplifies the process of building web applications in Python. It follows the Model-View-Controller (MVC) architectural pattern and provides features like URL routing, template rendering, database ORM (Object-Relational Mapping), and user authentication.

These are just a few examples of the numerous Python libraries available. Depending on your specific needs and interests, there are many other libraries and packages that can cater to different domains, such as image processing, natural language processing, data visualization, network programming, and more.