Assignments:

Python for Data Analysis and Machine Learning

1. Build a Python application that reads a **dataset** (CSV) and stores it in multiple data structures (**lists, tuples, sets, dictionaries**). Demonstrate efficient retrieval, filtering, and aggregation using each structure. Compare their performance.

2. Select a real-world, **messy dataset** (e.g., a large **CSV** or **JSON** file) that has at least five features and over 1,000 records. Write a Python script to import the data, perform **data cleaning** (handle missing values using imputation or removal , remove duplicates), and perform **data transformation** (e.g., creating a new calculated feature). Finally, export the cleaned data to a new CSV file.

3. Use **NumPy** arrays to analyze **weather data** (temperature, humidity, rainfall). Perform operations such as **reshaping**, **broadcasting**, and calculating **statistical measures** (**mean, median, variance, standard deviation**). Compare results with built-in Python functions.

4. Generate two large, random **NumPy matrices** (at least 100x100) representing two different data measurements. Write functions to perform **element-wise operations** , calculate the **statistical functions** (mean, median, variance, standard deviation) for each matrix , and perform a **matrix multiplication** (if dimensions allow) or use **broadcasting** to combine them. Document your code with explanations of the array creation and reshaping process.

5. Take a real-world dataset (e.g., **Titanic, House Prices, or a custom dataset**). Use **Pandas** to handle missing values, duplicates, and inconsistent formats. Implement **Min-Max scaling, Z-score normalization, and log transformation** on a dataset. Compare how different scaling techniques affect model performance.

6. Use statistical methods (**Z-score, IQR**) and visualization to detect and handle **outliers** in a dataset (e.g., salaries, product prices). Compare model accuracy before and after handling outliers.

7. Obtain a dataset of sales records. Use **Pandas** to load, filter records based on a condition (e.g., sales greater than $500) , and use the **groupby()** method to aggregate total sales by region or product category. Create a **descriptive report** that includes **describe()** and **info()** outputs and a summary of the aggregated statistics.

8. Using **Pandas**, analyze a **stock price dataset**. Perform resampling, rolling mean, and differencing. Train a regression or clustering model on time-based features.

9. Select a dataset (e.g., **COVID-19 statistics, stock market data**). Perform exploratory data analysis using **Pandas and NumPy**. Generate descriptive statistics, visualize trends, and summarize insights.

10. Given a dataset with raw attributes (e.g., **sales transactions**), create new features (e.g., total revenue, customer frequency, category-based sales). Apply discretization, transformation, and outlier detection to improve dataset quality.

11. Choose a numerical dataset and implement at least two different methods for **outlier detection** (e.g., Z-score, IQR). Write a script that identifies these outliers and then **handles them** either by removal or by value capping/replacement. Compare and contrast the distribution of the key feature before and after handling the outliers.

12. Create a function that takes a Pandas DataFrame as input. This function must identify columns with incorrect data types (e.g., numbers stored as strings) and apply **type conversion**. Furthermore, use **conditional statements** and **looping** to validate data entries against a set of business rules (e.g., age must be positive) and flag the invalid records.

13. Use a dataset suitable for **Linear Regression** (e.g., house prices). Before training, apply **feature scaling** or **standardization** to the independent variables using NumPy and/or Pandas and then Scikit-learn's preprocessing tools. Implement the Linear Regression model and Evaluate models using R^2, RMSE, and MAE. Discuss overfitting and apply regularization techniques if needed.

14. Select a binary classification dataset (e.g., predicting customer churn). Implement a **Logistic Regression** model. Preprocess the data, including handling categorical variables (if any). **Evaluate the model's performance** using metrics like accuracy, precision, and recall.

15. Choose a **classification dataset** (e.g., **Iris, Breast Cancer**). Implement data preprocessing (cleaning, scaling, encoding), train models **(Logistic Regression, Decision Trees, KNN, SVM, ANN),** and Evaluate models using confusion matrix,

accuracy, precision, recall, F1-score, etc. and ROC curves. Identify the best-performing model.

16. Use a classification dataset. Implement the **KNN** algorithm. Experiment with at least three different values for 'K' and determine the optimal 'K' based on a rigorous evaluation using cross-validation.

17. Use a customer **segmentation dataset**. Apply **K-Means and Hierarchical Clustering**. Compare clustering results, visualize cluster assignments, and interpret insights for business decision-making.

18. Apply **K-Means Clustering** to an unlabeled dataset (e.g., customer segmentation). Use the "Elbow Method" to determine a reasonable number of clusters (K). Visualize the final clusters and provide a descriptive interpretation of the characteristics of each cluster.

19. Use the same dataset from the K-Means assignment. Implement **Hierarchical Clustering**. Generate and interpret the **dendrogram** to determine the optimal number of clusters. Compare and contrast the results with the K-Means analysis.

20. Apply an **SVM** model for classification. Experiment with at least two different kernel functions (e.g., **linear, RBF**) and analyze how the choice of kernel impacts the model's accuracy and training time.

21. Select a single, complex dataset and execute a full **end-to-end data analysis workflow**. This must include: data loading, **preprocessing** (handling missing values, feature scaling/normalization) , selecting and training a **supervised learning model**, and finally, **evaluating and reporting** the model's performance.

22. Take a dataset with a continuous numerical feature. Apply three different **transformation techniques** (e.g., log, square root, Box-Cox) to this feature, and also apply **discretization** (binning). Analyze and report how each transformation/discretization affects the distribution of the data and its correlation with a target variable.

23. Create a collection of functions, including a **lambda function** and at least one custom function, to systematically clean and process a dataset. Apply these functions to a

Pandas DataFrame to demonstrate the efficiency of **functional programming concepts** in data manipulation.

24. Design a data structure (e.g., a **nested dictionary** or **list of lists**) to represent hierarchical data (like a file system or organizational chart). Write a **recursive function** to traverse this structure and extract specific pieces of information, such as all the 'leaves' or elements at a certain depth.

25. Create a large, **sparse matrix** using NumPy (one that is mostly zeros). Implement a custom function to efficiently store, manage, and perform basic statistical calculations (e.g., finding the non-zero mean) on this sparse matrix, focusing on minimizing memory usage.

26. Take a dataset such as **housing prices**. Build regression models (Li**near and Polynomial**). Evaluate models using R^2, RMSE, and MAE. Discuss overfitting and apply regularization techniques if needed.

27. Select one regression and one classification problem (from separate datasets). For each problem, implement two different models (e.g., Linear vs. Non-linear Regression; KNN vs. Logistic Regression). **Compare their performance** (e.g., using R^2 vs. Accuracy) and analyze the strengths and weaknesses of each model based on the dataset characteristics.

28. Write a single script that can dynamically import and combine data from three different file types: **CSV, Excel, and JSON**. Ensure the final combined dataset is properly indexed and cleaned using Pandas methods before calculating a simple descriptive statistic (e.g., mean) across all combined data.

29. Work with a large CSV/JSON dataset (>100MB). Optimize memory usage using appropriate data types, chunk loading, and indexing. Document the performance improvements achieved.

30. Implement **k-fold cross-validation** on a dataset. Use **GridSearchCV** or **RandomizedSearchCV** to tune hyperparameters for models like SVM or Decision Trees. Document improvements.

31. Combine **Pandas** with **Matplotlib/Seaborn** to create insightful visualizations (**heatmaps, scatter plots, pair plots**). Use them to explain correlations and feature importance in a dataset.

32. Work with an imbalanced dataset (e.g., **fraud detection, disease detection**). Apply oversampling, undersampling, or SMOTE. Evaluate how balancing affects classifier performance.

33. Build a predictive model for a practical scenario (**student performance, loan approval**, **disease prediction**). Emphasize data wrangling, preprocessing, feature selection, and result interpretation.

34. Choose a dataset from Kaggle or UCI Repository. Perform a full pipeline:
    - Data import & cleaning
    - EDA & preprocessing
    - Model training & evaluation (supervised + unsupervised)
    - Visualization & reporting insights