**Programming Assignment: Dimensionality Reduction**

**שימו לב: יש תרגום לעברית בסוף הקובץ**

**Objective**: In this assignment, you will gain hands-on experience with dimensionality reduction techniques by implementing your own functions and building a basic user interface using Streamlit.

This will help you understand the fundamental concepts behind **dimensionality reduction** and practice developing a minimal UI to interact with the data.

You will analyze the dataset of Knesset elections in Israel, where each row represents a ballot box (קלפי). The columns include: city name, ballot box number, and the number of votes given to each party.

**GOAL :**

Your task is to determine how different cities cluster based on voting patterns and how different parties tend to receive votes from similar or distinct areas. You will identify and interpret clustering patterns that emerge from your analysis.

**Assignment Overview**

There are two main parts to this assignment:

1. Data Processing and Analysis (90 points)
2. User Interface Using Streamlit (Bonus - 20 points)

All function implementations must be written in a Python file (.py), and all demonstrations of the functions’ usage should be shown in a Jupyter Notebook.

**Instructions**

**Part 1: Data Processing and Analysis (90 points)**

1. **Load the Dataset (10 points)**
   * Write a function: def load\_data(filepath: str) -> pd.DataFrame:
     + **Input**:
       - filepath: The file path to a CSV or Excel dataset.
     + **Output**:
       - A pandas DataFrame containing the loaded data.
     + **Note**: This function must be implemented in a Python file (.py).
   * Use libraries such as **numpy** or **pandas** to load the data and return it as a pandas DataFrame.
2. **Group and Aggregate Data (20 points)**
   * Write a function:

def group\_and\_aggregate\_data(df: pd.DataFrame, group\_by\_column: str, agg\_func) -> pd.DataFrame:

* + - **Input**:
      * df: A pandas DataFrame containing the dataset.
      * group\_by\_column: The column to group data by (e.g., 'city name').
      * agg\_func: The aggregation function to apply to each group (e.g., mean, sum, or count).
    - **Output**:
      * A pandas DataFrame with aggregated results.
    - **Note**: This function must be implemented in a Python file (.py).
  + Demonstrate this function in a Jupyter Notebook by:
    - Loading the election dataset.
    - Aggregating it by city name.
    - Displaying the total number of votes each party received in each city.

1. **Remove Sparse Columns (10 points)**
   * Write a function:

def remove\_sparse\_columns(df: pd.DataFrame, threshold: int) -> pd.DataFrame:

* + - **Input**:
      * df: A pandas DataFrame.
      * threshold: The minimum total sum for a column to be retained in the DataFrame.
    - **Output**:
      * A pandas DataFrame with sparse columns removed.
    - **Note**: This function must be implemented in a Python file (.py).
  + Demonstrate this function in a Jupyter Notebook by removing columns representing parties that received fewer votes than a specified threshold.

1. **Dimensionality Reduction with PCA (30 points)**
   * Write a function:

def dimensionality\_reduction(df: pd.DataFrame, num\_components: int, meta\_columns: list[str]) -> pd.DataFrame:

* + - **Input**:
      * df: A pandas DataFrame containing the data to be reduced.
      * num\_components: The number of principal components to retain.
      * meta\_columns: A list of metadata columns to exclude from dimensionality reduction (these should be included in the final output without changes).
    - **Output**:
      * A pandas DataFrame with the reduced dimensions and the metadata columns.
    - **Note**: This function must be implemented in a Python file (.py).
  + Implement **PCA (Principal Component Analysis)** from scratch using **numpy** and **pandas**.

Do not use high-level libraries such as **scikit-learn**.

1. **Visualize the Reduced Data (20 points)**
   * **Compare Cities**:
     + Aggregate the data so that each row represents a city, and each column represents the total number of votes obtained by each party.
     + Remove the data for parties that received less than 1000 total votes.
     + Create a scatter plot of the reduced data (for num\_components = 2) to visualize dimensionality reduction.
     + Estimate the number of clusters visually.
   * **Compare Parties**:
     + Transpose the city-wide data so that each row represents a party, and each column represents a city.
     + Remove data for cities with fewer than 1000 total votes.
     + Create a scatter plot of the reduced data (for num\_components = 2) to visualize dimensionality reduction.
     + Estimate the number of clusters visually.
     + **Note**: Instead of using matplotlib, consider using **Plotly** for interactive visualizations. Research “Plotly hover” to add tooltips for identifying specific cities or parties. You may also explore creating 3D visualizations to better understand data relationships.

**Part 2: User Interface Using Streamlit (Bonus - 20 Points)**

1. **Create a Basic Streamlit UI**
   * Use **Streamlit** to build a user interface that allows users to:
     + Upload their dataset.
     + Choose a column to group by and select an aggregation function.
     + Set the number of components for dimensionality reduction.
     + Select whether to show city- or party-wise processing.
     + Display the reduced dataset and visualize it interactively.

**Deliverables**

1. **Python Script(s)**:
   * Include all required function implementations in one or more .py files.
2. **Jupyter Notebook**:
   * Use a Jupyter Notebook to demonstrate how each function works with the dataset.
3. **Streamlit App**:
   * Provide the Streamlit app code as a .py file, along with instructions to run it locally.
4. **README File**:
   * Include a README file explaining:
     + How to run the Python script(s).
     + How to interact with the Streamlit UI.

**What to Submit**

* You need to submit all the Python files you wrote, all the Jupyter Notebooks (most probably, one notebook).
* Everything you need to run Streamlit.
* Also, record a video of not more than three minutes in which you explain what you did and how you did it. If you implemented the Streamlit interface, you can add two more minutes to demonstrate the Streamlit interface. Points will be reduced if your video is longer than the time limit. Speeding up the recording is not allowed - points will be reduced.

**Requirements**

* **Libraries Allowed**:
  + **numpy**
  + **pandas**
  + **matplotlib**
  + **plotly**
  + **streamlit**
* **Avoid using**:
  + **scikit-learn** or any other high-level machine learning libraries for dimensionality reduction.
* **Ensure**:
  + Code is modular and clear.
  + Every function has a clear docstring.

**Tips**

* Study how PCA works and how eigenvectors reduce data dimensions.
* Modularize your code for clarity and reusability.
* Keep the Streamlit UI intuitive and user-friendly.