PASS TASK (WEEK 2)

About this task

Step-1

At the completion of week 2 modules, you are required to compete a lesson review to indicate what you have learnt and how you learnt it by submitting evidence requested at the end of this file.

Step-2

Your tutor will then review your submission and will give you feedback. If your submission is incomplete the tutor will ask you to include missing parts. Tutor can also ask follow-up questions, either to clarify something that you have submitted or to assess your understanding of certain topics.

Feedback and submission deadlines

Feedback deadline: Monday 1 April (No submission before this date means no feedback!)
Submission deadline: Before creating and submitting portfolio.

Evidence of Learning

1. Submit a summary report (pdf format) in Ontrack (https://ontrack.deakin.edu.au)
    1.1. Summarise the main points that is covered in the week 1 and 2.
    1.2. Provide summary of your reading list – external resources, websites, book chapters, code libraries, etc.
    1.3. Reflect on the knowledge that you have gained by reading contents of the week 1 and 2 with respect to machine learning.
    1.4. Attempt the quiz given in weekly content (1.28 and 2.14) and add screenshot of your score (>85% is considered completion of this task) in this report.
2. Complete the following problem-solving task given in weekly content, and submit your code file (.ipynb) separately to OnTrack (https://ontrack.deakin.edu.au).

Problem Solving Task

Instructions:
Ensure you provide well-structured code for each question. Clearly explain your reasoning where required and submit all necessary files, including the modified dataset after performing the required transformations.

---

1. Create three variables: **Sensor_IDs**, **Temperature_Readings**, and **Humidity_Levels**.
   - Assign five **unique random** values to each variable.
   - Ensure that each Sensor_ID is a unique alphanumeric string (use Hashing for unique sensor IDs).
   - Assign **random temperature and humidity readings** within valid ranges for five sensors.
   - Ensure that:
     - Each Sensor_ID is unique.
     - Temperature readings are between **-10°C and 50°C**.
     - Humidity levels are between **10% and 100%**.
   - Print and validate the assigned values.

2. Create a **dictionary** where:
   - The keys are **Sensor_IDs**.
   - The values are **dictionaries containing Temperature_Reading and Humidity_Level**.
   - Implement a function that:
     - **Filters sensors exceeding a user-defined temperature threshold**.
     - **Sorts the dictionary** in descending order by temperature.
     - **Retrieves and prints the top 3 highest temperatures**.
   - Ensure robustness by handling **missing or invalid sensor readings**.

3. Define a variable **air_quality** that can take one of the following values: **'Good'**, **'Moderate'**, **'Unhealthy'**, or **'Hazardous'**.

   - Implement a **function** that:
     - Recommends an activity based on air quality.
     - Handles **unexpected inputs** and prompts the user for correction.
   - **Simulate air quality readings over 10 instances**:
     - Randomly assign one of the four air quality categories to each instance.
     - Count how many times each category appears.
     - Calculate and print the percentage of occurrences for each category.
     - Display the results in a structured format (e.g., table or dictionary).

4. Write a while loop that continuously prompts the user to enter a number:

   - **Break the loop** when a negative number is entered.
   - **Classify even and odd numbers**, storing them in separate lists.
   - **Log both lists to a text file (loop_log.txt)** upon loop termination.
   - Ensure **error handling for non-numeric inputs**.

- Validate input to **prevent duplicate entries** in storage.

5. Write a Python program that:
    - Accepts a **scientific term** as input.
    - Prints all **characters at even positions** in one line and **odd positions** in another.
    - Do this **without using list slicing ([::])**.
6. Read a dataset **'WeatherData.csv'** using pandas (download the dataset from the Resources).
    - Display the **number of weather stations (rows) and recorded parameters (columns)**.
    - Compute the **mean wind speed** and subtract it from all wind speed values.
    - Print both the original and modified values.
7. Read **'WeatherData.csv'** and identify missing values.

    - Choose **two different imputation techniques** to handle missing values.
    - Justify why you selected each technique for the respective feature.
    - **Compare the distribution of the imputed column before and after imputation.**
    - **Save the cleaned dataset (WeatherData_Cleaned.csv).**

8. Encode the **'Weather_Condition'** column using appropriate encoding approach and display the encoded value.
    - Justify your selection of encoding technique.
    - Save the modified dataset as a new CSV file.
9. Apply **Min-Max Scaling** to **all numerical columns**, excluding Station_ID. Generate **two visualizations**:

    - **Histogram of temperatures before and after scaling**.
    - **Boxplot of wind speed before and after scaling**.
    - Analyze and explain the **effect of Min-Max Scaling** on the dataset's distribution.