Apollo Solutions Machine Learning Developer Test

Context

You are being hired by a fictional biotech company specializing in genetic research. The task involves analyzing **embeddings** derived from images to classify genetic syndromes. These embeddings are outputs from a pre-trained classification model. The company wants to improve its understanding of the data distribution and enhance the classification accuracy of genetic syndromes based on these embeddings.

Your objective is to implement a comprehensive pipeline that includes data preprocessing, visualization, classification, manual implementation of key metrics, and insightful analysis.

Dataset Description

You are provided with a **pickle file** (`mini_gm_public_v0.1.p`) that contains all the necessary data. The data structure is as follows:

- **Embeddings**: 320-dimensional vectors representing images.

Hierarchy:

```
 {
  'syndrome_id': {
    'subject_id': {
      'image_id': [320-dimensional embedding]
    }
  }
}
```

- **Goal**: Use these embeddings to classify the **syndrome_id** associated with each image.

Note: If you encounter the `"numpy.core._multiarray_umath"` error when loading the pickle file, please upgrade your `numpy` package.

Task Description

Development

1. **Data Processing**:
    - **Load and preprocess** the data from the pickle file.
        - Flatten the hierarchical data structure into a suitable format for analysis.
        - Ensure data integrity and handle any missing or inconsistent data.
    - **Exploratory Data Analysis**:
        - Provide statistics about the dataset (e.g., number of syndromes, images per syndrome).
        - Discuss any data imbalances or patterns observed.

2. **Data Visualization**:
    - Use **t-SNE** to reduce the dimensionality of the embeddings to 2D.
        - Generate a plot that visualizes the embeddings colored by their **syndrome_id**.
        - Identify and interpret clusters or patterns in the visualization.
        - Discuss how these patterns might relate to the classification task.

3. **Classification Task**:
    - Implement the **K-Nearest Neighbors (KNN)** algorithm to classify the embeddings into their respective **syndrome_id**.
        - Use both **Cosine** and **Euclidean** distance metrics.
        - Perform **10-fold cross-validation** to evaluate the model performance.
        - **Determine the optimal value of kk** (from 1 to 15) for KNN using cross-validation.
        - **Implement**:
            - Calculation of **AUC (Area Under the ROC Curve)**.
            - **F1-Score**.
            - **Top-k Accuracy**.
        - Compare the classification results between the two distance metrics.
        - Discuss any differences in performance and possible reasons.

4. **Metrics and Evaluation**:
    - **Generate ROC AUC curves** for both Cosine and Euclidean distance metrics.
        - Average the results across the cross-validation folds.
        - Plot both curves on the same graph for comparison.
    - **Create tables** summarizing the performance metrics for both algorithms.
        - Include metrics such as Top-k Accuracy, AUC, F1-Score, etc.
        - Ensure that the tables are clear and can be automatically generated (e.g., from code).

5. **Report**:
    - Write a detailed report that includes:
        - **Methodology**: Explain the steps taken, including data preprocessing, choice of algorithms, and parameter selection.
        - **Results**: Present the findings with supporting graphs and tables.
        - **Analysis**: Interpret the results, compare the performance between distance metrics, and discuss any insights gained.
        - **Challenges and Solutions**: Describe any difficulties encountered and how you addressed them.
        - **Recommendations**: Propose potential improvements or next steps for further analysis.

Guidelines

- **Deliverables**:
    1. Python scripts (`.py` files) as specified, with:
        a. Requirements
        b. READ ME
    2. A PDF report summarizing your work, including:
        a. methodologies,
        b. results,
        c. insights,
    3. A PDF answering the “Interpretation” questions.

- **Submission**:
    - You have 5 **days** to complete the test.
    - Submit your deliverables via email to **guilherme.marchini@apollosolutions.dev**.

- **Communication**:
    - For any questions or clarifications during the exercise, feel free to email **guilherme.marchini@apollosolutions.dev**.
- **Important Notes**:
    - Do **not** submit the code in a Jupyter Notebook format.
    - The use of Jupyter Notebooks is **not suitable** for this project.
    - Ensure your code can be run as standalone Python scripts.

Interpretation

1 - In Figure 1 we have a data distribution, the dots represent the sparse data for the axis X and Y, and the lines represent the fit of a hypothetical classification model. Based on the distributions of Figure 1:

- Which distribution has the best balance between bias and variance?
- Describe your thoughts about your selection.

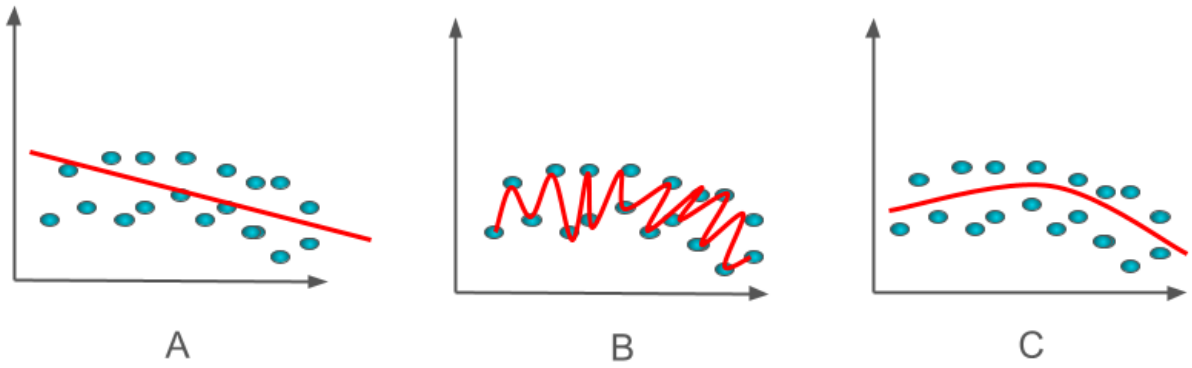


Figure 1 - Data distribution samples

2 - Figure 2 presents a simple graph with 2 curves and 1 line. In model selection and evaluation:

- What is the purpose of this graph and its name?
- What kind of model result does the dashed line represent?
- Which curve represents a better fit, the red or the green? Why?
- Describe your thoughts about your selection.

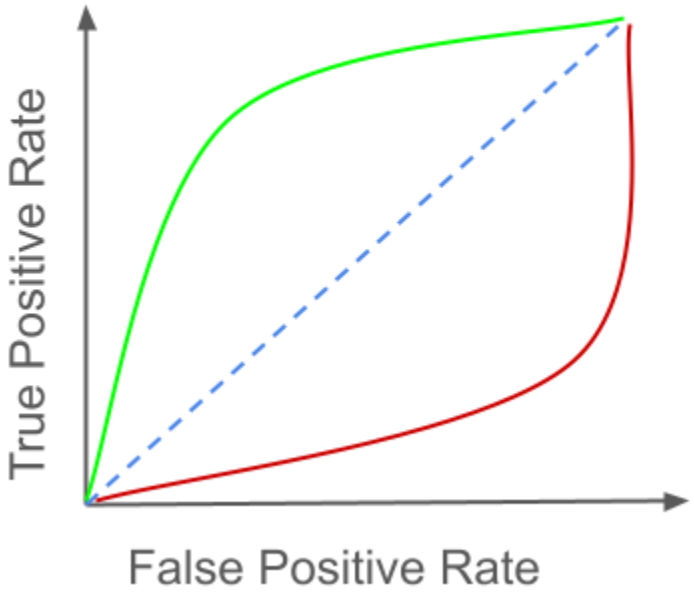
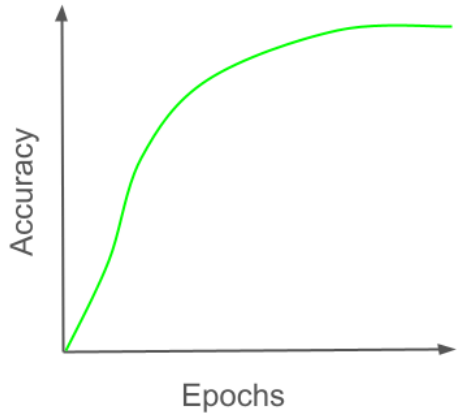


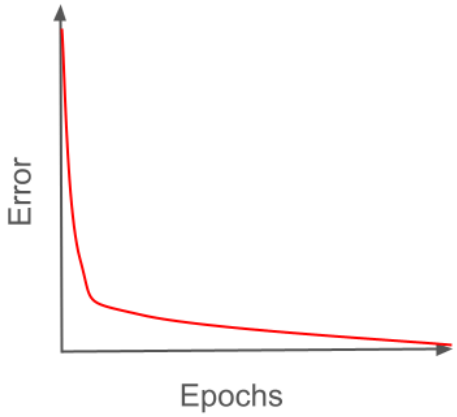
Figure 2 - Simple graph

3 - Figure 3 presents a classification model training and the evaluation. This model classifies 3 classes (A, B, C). Graph A represents the training accuracy over the epochs, Graph B represents the training loss over the epochs, and the table represents the evaluation of the model using some test samples, we used a confusion matrix to evaluate the classes trained.

- Can we say that the model has a good performance in the test evaluation?
- What phenomenon happened during the test evaluation?
- Describe your thoughts about your selection.



A



B

	A	B	C
A	0.5	0.25	0.25
B	0.15	0.45	0.35
C	0.1	0.4	0.5

Figure 3 - Model train and evaluation pipeline

Additional Requirements

To ensure the test assesses your individual capability:

- **Originality**:
  - Write the code yourself. Do not copy from external sources.
  - If you refer to any resources, cite them appropriately in your report.
- **AI Assistance**:
  - You may use AI tools for debugging or minor assistance.
  - However, the core implementation and analysis should be your own work.
  - Over Reliance on AI-generated code may impact your evaluation.

Good luck, and we look forward to reviewing your submission!