# Introduction to ML Term Project - Fall 2023

**Due Date: January 7, 2024, 23:59:59**

## Objectives:

This term project, undertaken individually, entails the development of two distinct classifiers: a **Multilayer Perceptron (MLP)** and either **Naïve Bayes** or **k-Nearest Neighbors (kNN)**. Each classifier will be evaluated independently, using metrics such as **F1-score**, **Matthews Correlation Coefficient (MCC)**, and **Area Under the Curve (AUC)**. This term project is an individual project, not a team project, with the aim to hone your skills in developing ML algorithms from scratch, including data preprocess, model structure design, parameter tuning, and model evaluation.

## Dataset Description:

The data for this project are real-world data of candidemia patients. Each patient is described by 77 features, F1~F77, plus a class (i.e., target), Outcome. This is a binary class concept learning problem, that is, Outcome is either 1 or 0. A dataset (trainWithLabel.csv) is already provided for you to train and validate your models, and your learned models **will be finally tested on an independent test set (testWithoutLabel.csv), which will be available on December 15, 2023**. For additional information of the patient features, please refer to **Data Description.docx**.

## Language: Python or C++

## Submission:

1. **Project Report**: Submit a report that explains both your classifiers, saved as **studentID.pdf** (for example, **311551001.pdf**).
2. **Source Code**: Submit the source code for both the **MLP** and your chosen classifier (**Naïve Bayes** or **kNN**), saved as **studentID.py** (for example, **311551001.py or 311551001.cpp**).
3. **Results**: Include both **cv_results.xlsx** and **test_results.xlsx**.

- **Note:** Combine all the required files into one zip file, named **studentID.zip** (for example, **311551001.zip**).

## Project Requirements:

1. **Model Implementation**: Implement an **MLP** classifier and opt for either a **Naïve Bayes** or a **kNN** classifier for integrated development. Follow the structure outlined in **main_framework.py for Python development**, or **main.cpp for C++**

**development**. Ensure that both models are compatible with the **Preprocessor** class and the **evaluate_model** function, enabling efficient data processing and performance assessment.

2. **Data Preprocessing**: Apply suitable preprocessing techniques such as imputation, standardization, or transformation, customizing these methods to fit the specific requirements of each model and the characteristics of the data.

3. **Model Construction:** the file **trainWithLabel.csv** is provided on **E3**. Feel free to use it to train your models or tune your parameters. For more details, please refer to the **Model Development Framework Guide (Python/C++).docx**.

   (1) kNN: Select an appropriate definition for '**distance**' and determine the optimal number of neighbors (k).

   (2) Naïve Bayes: Effectively address the '**zero probability problem**'.

   (3) MLP: Design the network structure with care and select appropriate loss and activation functions.

   *Concentrate on optimizing both the MLP and your selected classifier, Naïve Bayes or kNN, by addressing the specific issues unique to each model.

4. **Performance measure**

   To assess the performance of ML models in predicting candidemia, we have provided a brief summary of essential performance metrics. These measures and their definitions are as follows:

| Performance Metric | Definition |
|---|---|
| Recall[a] | TP/(TP+ FN) |
| Precision[b] | TP/(TP+ FP) |
| ACC | (TP+ TN)/(TP+ TN+ FP+ FN) |
| F1-score | $\frac{2\times Recall\times Precision}{Recall+Precision}$ |
| MCC | $\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}}$ |
| AUC | Area under the ROC curve |

TP, true positive; TN, true negative; FP, false positive; FN, false negative; MCC, Matthews correlation coefficient; ACC, accuracy; AUC, area under the curve; ROC, receiver operating characteristic.

[a] Recall is equivalent to sensitivity in its definition.

[b] Precision is equivalent to positive predictive value in its definition.

## Grading Policy

1. **Source Code (30%)**: Evaluation will be based on the correct implementation and functionality of both classifiers.

2. **Performance Measurements (30%):**
   - Performance Rating (PR) will be determined by the best metrics (F1-score, MCC, AUC) achieved by either classifier, considering both K-Fold Cross-Validation (70%) and an independent test set (30%).
   - PR Scores:
     - I. PR 99: 30/30 points
     - II. PR 90: 27/30 points
     - III. PR 85: 25.5/30 points
     - IV. PR 80: 24/30 points
     - V. PR 75: 22.5/30 points
     - VI. Below PR 75: Scores decrease progressively (e.g., PR 70 might score 21/30 points)

3. **Report (40%):**
   - Implementation Details: Include a description of preprocessing techniques, data handling, and the architectures of both classifiers, detailing hyperparameters and other relevant specifics.
   - Discussion: Provide an analysis of implementation challenges, prediction results, and key insights, including a comparative assessment of the models.

## Plagiarism Policy:

Ten students will be randomly selected for oral exam after the submission of term project. Any plagiarism will result in a zero credits for the implicated project. These oral examinations are scheduled between January 8, 2024, and January 9, 2024.