Machine Learning Model Development Framework Guide
(C++ Version)

Introduction

This guide provides an overview of a C++ framework designed for machine learning algorithm development and evaluation. The framework includes data preprocessing, model training, and evaluation using various performance metrics.

Framework Components

1. Preprocessor Class:

- Responsible for preprocessing the input data.
- Methods for preprocessing numerical, categorical, and ordinal features.

2. Classifier Abstract Class:

- Base class for all classifiers.
- Abstract methods: fit (to train the model), predict (to make predictions) and predict_proba (to make probability of the outcome).

3. Classifier Implementations:

- ‘**NaiveBayesClassifier**’: Implements Naive Bayes algorithm.
- ‘**KNearestNeighbors**’: Implements K-Nearest Neighbors algorithm.
- ‘**MultilayerPerceptron**’: Implements a basic structure for Multilayer Perceptron.

4. Model Evaluation:

- Function ‘**evaluate_model**’ to evaluate model performance.
- Metrics: Accuracy, F1 Score, Precision, Recall, MCC (Matthews Correlation Coefficient), AUC (Area Under Curve).

5. Main Function:

- Workflow for loading data, preprocessing, model training, cross-validation, and evaluation.
- K-Fold Cross-Validation and test set evaluation.
- **Predict the outcomes for 'testWithoutLabel.csv' and save them as 'test_results.csv' to allow the TAs to compute PR scores, which are crucial for the scoring process.**
- Saving results to Excel for further analysis.
  - Represent models with numbers (1 for Naïve Bayes, 2 for KNN, and 3 for MLP) in the model name column. (Already completed, just let you know. You can see it in line: 144-146, 164)

Usage

- Include ‘helper.h’ and c++ standard library for every function you need.
- Free to add any member function or variable within your model implementation as needed.
- Instantiate Preprocessor class with your dataset for preprocessing.
- Define your models in `unordered_map<float, Classifier*> models`.
- Use the evaluate_model function to get the performance metrics.
- Compile your code using command “`g++ helper.cpp main.cpp -o main`” to generate the executable file named main.exe.

Best Practices

- Customize preprocessing methods based on the dataset's characteristics.
- Extend the Classifier classes according to the algorithm's specific needs.
- Handle exceptions and validate inputs for robust model training and evaluation.

Conclusion

This framework serves as a starting point for developing and evaluating various machine learning models. Users are encouraged to extend and adapt the framework according to their specific project requirements.