COMP5434 Project

1. Overview

There are three datasets (Data1, Data2, and Data3) in this project. For each dataset, you need to develop a solution to predict the class labels of the testing records in the dataset. A record in the datasets contains several attributes. The real-world meaning of the class labels and attributes are removed, which will let you sense the power of big data analytics to generate accurate results *without* expert knowledge of a certain application domain.

2. Submission Deadline

a. Predictions on Kaggle: **11:55 PM, April 6th, 2025**
b. Presentation slides, report and code: **11:59 PM, April 7th, 2025**

3. Team Requirements

Students should participate in this project in teams. Each team should have a voluntary coordinator for administrative purposes. **A team coordinator** should fill a form (PolyU Connect accounts required, https://forms.office.com/r/f4L2aMFU9r) to register the team **before 11:59 PM March 6th, 2025**.
You may use the discussion board in Blackboard to find your teammates.
To avoid high workload and free riders, each team should have **3 or 4** students.
If there are any students who are not in a team after the deadline, they will be organized into teams randomly. We will try to keep the size of such teams within 3~4 students. However, in extreme cases, it may not follow the regular guidelines.

4. Data Description and Instructions

For each dataset, we provide its training data (train.csv), testing data (test.csv), and submission sample (sample_submission.csv).

Dataset 1

Overview: This dataset records academic and training performance of some students. Your goal is to train a model to predict **whether each student will be placed** (i.e., can find a job) based on the training set.

Training data: *train.csv*

- Each row is a data record with its StudentID, 10 attributes, and a label.
- The last cell of each row ("*label*" column) is "*class label"* in integer, which is our classification target.
    - This label indicates whether this student is placed.
        - *1* indicates *placed*.
        - *0* indicates *unplaced*.
- Training data includes training records with ground-truth class labels.
- Use the training data to train your solution.

Testing data: *test.csv*

- Each row is a data record with its StudentID and 10 attributes. The attributes are the

same as those of *train.csv*, but the *label* is missing.
- Use your method to get your predicted labels of the testing records in *test.csv* and generate submission file which includes your predictions in the format of *sample_submission.csv*.
- We will obtain your solutions' performance based on your predicted labels for testing data.
- Our evaluation is based on the Macro-F1 metric.
    - A detailed explanation of the calculation of the metrics:
        - https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification
    - Example: A third-party library to calculate the Macro-F1 scores:
        - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
- After generating your predicted labels, you need to submit it to our online platform.

Sample Submission: *sample_submission.csv*
- This file is a sample of your submission about the predictions of *test.csv*.
- This file includes two columns.
    - *StudentID*: the identifier of each record in *test.csv*.
    - *label*: the prediction of this record.

Dataset 2

Overview: This dataset records some features with labels. Your goal is to train a model to predict the labels based on the training set. The missed values are labelled as "?" in this dataset.

Training data: *train.csv*
- Each row is a data record with its id, 18 attributes, and a label.
- The last cell of each row ("*label*" column) is "*class label"* in integer, which is our classification target.
    - This label indicates a class from 1 to 5.
- Training data includes training records with ground-truth class labels.
- Use the training data to train your solution.

Testing data: *test.csv*
- Each row is a data record with its id and 18 attributes. The attributes are the same as those of *train.csv*, but the *label* is missing.
- Use your method to get your predicted labels of the testing records in *test.csv* and generate submission file which includes your predictions in the format of *sample_submission.csv*.
- We will obtain your solutions' performance based on your predicted labels for testing data.
- Our evaluation is based on the Macro-F1 metric.
    - A detailed explanation of the calculation of the metrics:
        - https://scikit-

learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification

    - Example: A third-party library to calculate the Macro-F1 scores:
        - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
- After generating your predicted labels, you need to submit it to our online platform.

Sample Submission: *sample_submission.csv*
- This file is a sample of your submission about the predictions of *test.csv*.
- This file includes two columns.
    - *id*: the identifier of each record in *test.csv*.
    - *label*: the prediction of this record.

Dataset 3

Overview: This dataset records the information about credit card transactions with fraud indicator. Your goal is to train a model to classify **whether this transaction is fraudulent or not** based on the training set.

Training data: *train.csv*
- Each row is a data record with its id, 28 attributes, and a label.
- The last cell of each row ("*label*" column) is "*class label"* in integer, which is our classification target.
    - This label indicates the transaction is fraudulent or not.
        - *1* indicates *fraudulent*.
        - *0* indicates *not*.
- Training data includes training records with ground-truth class labels.
- Use the training data to train your solution.

Testing data: *test.csv*
- Each row is a data record with its id and 28 attributes. The attributes are the same as those of *train.csv*, but the *label* is missing.
- Use your method to get your predicted labels of the testing records in *test.csv* and generate submission file which includes your predictions in the format of *sample_submission.csv*.
- We will obtain your solutions' performance based on your predicted labels for testing data.
- Our evaluation is based on the Macro-F1 metric.
    - A detailed explanation of the calculation of the metrics:
        - https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification
    - Example: A third-party library to calculate the Macro-F1 scores:
        - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
- After generating your predicted labels, you need to submit it to our online platform.

Sample Submission: *sample_submission.csv*

- This file is a sample of your submission about the predictions of *test.csv*.
- This file includes two columns.
    - *id*: the identifier of each record in *test.csv*.
    - *label*: the prediction of this record.

5. Project Platform

We use Kaggle (https://kaggle.com) to evaluate your predictions.
We hold three competitions for these three tasks. You should attend them as a team. **Your team name in Kaggle must be the same as the name in our team list**, which will be released after the registration period.
You may refer to https://www.kaggle.com/docs/competitions and the video (Kaggle-instruction) to know how to use Kaggle.

Competition Links:

1. Task 1: https://www.kaggle.com/t/82001a058a1d471aac76ab861f380eba
2. Task 2: https://www.kaggle.com/t/4995a1e12c04433ba1b36a40118fb45e
3. Task 3: https://www.kaggle.com/t/e21ec5fd3049444b8658d5c5cc71a9e3

Important Notes:

1. When you submit your prediction file, you can check your Macro-F1 score and rank in all teams immediately.
    a. However, it is a TEMPORARY performance because it is calculated from A PART OF (about 50%) testing data (called Public Leaderboard).
    b. When the competition ends, it will evaluate your submission from THE OTHER PART OF (another about 50%) testing data and update your ranks (called Private Leaderboard).
    c. This performance in Private Leaderboard will be used to grade your methods.
    d. **You should select two submissions for each competition BEFORE THE COMPETITION END.** Your final score and placement at the end of the competition will be whichever selected submission performed best on the private leaderboard. If you do not select submission(s) to be scored before the competition closes, the platform will automatically select those which performed the highest on the public leaderboard.
2. The competitions will end at **11:55 PM, April 6th, 2025**
3. The finalized rank will be released at **8:00 AM, April 7th, 2025**
4. You can only submit **10 times per day** to check your temporary performance, due to the resource limitation.

6. Tasks and Requirements

- Develop three solutions, one for each dataset, to predict the class label of each data record.
- You can develop any solutions, based on *either the algorithms introduced in this subject or the methods beyond the course content.*
- Macro-F1 will be used as the evaluation metric.
- Any programming language
    - Your code should be clean and well-documented (e.g., with sufficient

comments)

- You can use low-level third-party packages to facilitate your implementation.
- **Your implementation should involve sufficient technical details developed by yourselves.**
    - DO NOT simply call ready-to-use classification models provided in existing packages, as a **Blackbox**, to finish the project.
        - For example, Spark provides an implementation of Naïve Bayes classifier (https://spark.apache.org/docs/latest/mllib-naive-bayes.html), which, after preparing the data, can be used to train the data with just a simple function call, as shown in the following (Cases like this example will lead to point deduction)

```
import org.apache.spark.mllib.classification.{NaiveBayes, NaiveBayesModel}
import org.apache.spark.mllib.util.MLUtils

// Load and parse the data file.
val data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")

// Split data into training (60%) and test (40%).
val Array(training, test) = data.randomSplit(Array(0.6, 0.4))
      Calling a read-to-use classifier as a blackbox leads to point deduction
val model = NaiveBayes.train(training, lambda = 1.0, modelType = "multinomial")

val predictionAndLabel = test.map(p => (model.predict(p.features), p.label))
val accuracy = 1.0 * predictionAndLabel.filter(x => x._1 == x._2).count() / test.count()

// Save and load model
model.save(sc, "target/tmp/myNaiveBayesModel")
val sameModel = NaiveBayesModel.load(sc, "target/tmp/myNaiveBayesModel")
```

- **You MAY use third-parity libraries**. As long as your implementation involves reasonable algorithmic details for solving the problem, then it is fine. Unless it is too obvious, we will be very moderate when deciding if an implementation is solely based on Blackbox algorithms.
    - **i.e., let us see your effort, instead of purely calling black box algorithms.**

7. Hints

- *Note that data may be dirty.*
- Before running your solutions, you may need to follow the 5-step data analytical process (See Lecture 1 Slides) to prepare the data, in order to achieve higher performance.
- Try your best to improve your solutions' performance.

8. Report File

- Include your team name, your names, your student ids in the report
- *Suggested report structure:*
    - Problem definition
    - Data Analysis and Model design
    - Solution and implementation details

        - Probably follow the major steps in the 5-step data analytical process
        - For instance
            - Your unique technical designs to improve performance
            - Improve your basic design by adopting advanced machine learning model(s)
    - Performance evaluation and discussion
        - Report training time, testing time, memory consumption, etc
        - Analyze the experimental results, e.g., ablation studies of your techniques
        - Draw insightful conclusions
    - Summary of discoveries and future work.
    - **A contribution table**
        - In your report, include a contribution table indicating your percentage of contributions, in total 100% (grades will be adjusted accordingly).
        - Any teams without a contribution table will not be graded.
    - References

9. Presentation Guidelines

- 5-10 slide explaining your core ideas in the report
- **Face2Face presentation**
- Each team will have about 7 minutes for a presentation [depending on the number of teams].
    - Do not exceed the time requirement.

10. Submission Guidelines:

- Create folder with name *TeamName*.
    - Put all your code together in a folder with name *code*.
    - Rename your slides with name *slides* (with the extension name, such as *pptx* and *pdf*).
    - Rename your report with name *report* (with the extension name, such as *pdf*).
    - Put *code*, *slides* and *report* in the folder *TeamName*.
    - You should replace *TeamName* to your actual team's name.
- Compress this folder and your slides together as one zip file.
    - **Your submitted predictions in Kaggle must be able to be generated by using your codes. Otherwise, the marks of corresponding part will be 0.**
        - There are many ways to keep your code reproducible, i.e., we can get the same results in every running. For example, this webpage includes some methods to keep it reproducible in Python and PyTorch (a machine learning library). Another webpage introduces such methods for Keras (another machine learning library). For other libraries you plan to use in your code, please refer to its corresponding libraries to learn how to keep it reproducible.
- Follow the below example to name your zip file by replacing "*TeamName*" to your actual team's name:
    - *TeamName.zip*
- Your submission should be submitted by your **TEAM COORDINATOR** and the

deadline is **11:59 PM, April 7th, 2025**.

11. Grading Rubrics

- Code (6%):
    a. Code quality (3%)
    b. Code performance (3%)
        i. Evaluate the Macro-F1 of your prediction results on the testing data.
- Report (5%)
- Presentation (4%)

Outcome Presentations	%	**A+/A/A-**	**B+/B/B-**	**C+/C/C-**	**D+/D**	**F**
Code quality	3%	Programs are well-organized, makes good use of whitespace and comments. Variables have helpful names.	Programs are well-organized, easy to read and understand.	Programs can be read and is in a logical order.	Programs are runnable but barely readable.	Absent
Report Content	5%	Excellent, comprehensive and in-depth analysis with concrete facts/evidence	Clear analysis with good analysis supported by plenty of facts/evidence	Basic analysis with some level of facts/evidence	Barely relevant analysis with minimal facts/evidence	Absent
Presentation quality	4%	Very clear and logical	Good, easy to follow	Understandable, structured	Barely understandable	Absent