CS 165B – Machine Learning, Spring 2023

Machine Problem #4
Due Wednesday, June 14 by 11:00pm

TA: Danqing Wang
Email: danqingwang@ucsb.edu

Note: This assignment should be completed individually. You are not supposed to copy code directly from anywhere else, and we will review the similarity of your submissions. You should write the code from scratch, and we recommend you using library **sklearn**. You can use any default Python libraries and the imported packages at the beginning of ***mp4.py.*** However, there is no GPU in GradeScope, so make sure your code can run in a **CPU machine with 6GB RAM**. The running time limitation in GradeScope is **30 minutes.** Note that we do not have deep learning frameworks such as pytorch or tensorflow installed in the docker.

In this project, you should **start from scratch** to implement **a multi-layer perceptron (MLP) classifier** to predict the decision of credit card applications. Remember to keep the function name and argument of ***run_train_test(training_data, testing_data)*** unchanged.

Like MP3, we use **Pandas** library to deal with the data.

Problem Overview:

In this project, we focus on a real-world classification problem: whether the bank should approve the application of the credit card based on client information.

Credit score cards are a common risk control method in the financial industry. It uses personal information and data submitted by credit card applicants to predict the probability of future defaults and credit card borrowings. The bank can decide whether to issue a credit card to the applicant. Credit scores can objectively quantify the magnitude of risk. Credit score cards are based on historical data. Given the client's historical behaviors, the bank can identify whether this client is a risky customer that tends to pay late billing. Combining this information with the corresponding application, the bank can make a better decision on credit card application to reduce the number of risky customers. Usually, users in risk should be 3%, so the imbalance data is a big problem.

Change the **Python 3** program ***mp4.py*** that creates a MLP classifier for the binary classification problem. The training and development data sets are available in the starter package. The detailed data information can be found in ***data.md***.

- *data/**: include ***train.csv*** for training and ***dev.csv*** for development. We will have an extra private test set for the final grading.
- *data*.md: describe the data information.
- *mp4.py*: the python file you need to work on and submit.
- *MP4.pdf*: this instruction file

Code Instructions

In this project, you need to implement a MLP classifier for binary classification. You can refer to multi-layer perceptron classifier. Specifically, there are several things to do:

1. Preprocess the features. There are different kinds of features: binary feature, continuous features, categorical features.
2. Implement a MLP classifier with sklearn.
3. Update the parameters of the classifier based on the training data.
4. Evaluate the performance on the development set.
5. Try different hyperparameters / preprocessing methods to get a better performance.

run_train_test(training_data, testing_data) function is called to train a classifier based on the training data and return the prediction on the testing data.

Hint:

- You **do not need** to use all features. There could be some redundant features.
- You **must** make sure that the training and test data are processed in the same way.
- For continuous feature, you can normalize them, or divide them into several bins and convert them to categorical features.
- Some typical values of hyperparameters:
  - Hidden size: 8, 16, 32, 64, 128, …
  - Hidden layer: 1, 2, 3, 4, …
  - Maximum epoch: 100, 200, 300, 400, ….
  - Learning rate: 0.01, 0.001, 0.0001, 0.03, 0.003, …
  - batch size: 8, 16, 32, 64, 128, …
- For the imbalanced dataset, you can refer to https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html

Evaluation Instructions

In this project, we will use **F1 score** for evaluation. The definition can be found in Lecture 4 P8. Specifically, we calculate the F1 score based on the positive class, which is the risky customers with target=1. We implement the evaluation metric in mp4.py.

You can test your program with the ***train.csv*** and ***dev.csv*** provided in this starter package. You can directly run ***mp4.py*** for checking:

$ **python mp4.py**

This will output several metrics you get on the dev dataset. The final score will be graded on another private test set (use the same training set).

Submission Instructions

You should upload ***mp4.py*** to Gradescope for grading. Please put all code into one file.

Grading Instruction:

The score on Gradescope is computed based on the **F1 score** using the following rule:

Manual Grader (60 points): If your code can run successfully, you will get full points. Otherwise, your score will be based on the completeness.

AutoGrader (30 points): Your score depends on the **F1 score** on the private test:

• If your F1 is above 0.2: $Score = min(100,\ 80 + 100 * (F1 - 0.2)) * 30\%$
• If your F1 is below 0.2: $Score = max(0, 80 - 100 * (0.2 - F1)) * 30\%$

Leaderboard (10 points + bonus points): This score is based on your ranking on the leaderboard **(before the normal deadline)**:

- Top 60: +5 points, totally get 5 points in this part with no bonus.
- Top 40: +5 points, totally get 10 points in this part with no bonus.
- Top 20: +5 bonus points, totally get 15 points in this part.
- Top 10: +5 bonus points, totally get 20 points in this part.
- Top 3: +5 bonus points, totally get 25 points in this part.
- Top 1: +25 bonus points, totally get 50 points in this part.

Submissions with scores identical to the boundary will also be included. For example, F1 score of the 40-th is 0.41, then all submissions with F1=0.41 will be included in Top-40.

Note that if you submit after the normal deadline, your ranking on the leaderboard will be based on your last submission before the normal deadline.

Late submissions are accepted with 20% reduced credit each day. For example, you will get 80% of full credit if one day late, 60% if two days late and so on so forth.