v. 1

CS 165B – Machine Learning, Fall 2023

Machine Problem #3
Due Tuesday, November 28 by 11:59 pm

TA: Esha Singh, Ruiquan Li
Email: esingh@ucsb.edu, rli667@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 are allowed and it is recommended to use the library **sklearn** extensively. You can use any default Python libraries and the imported packages at the beginning of ***mp3.py.*** However, there is no GPU in GradeScope, so make sure your code can run on 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, Keras, or Tensorflow in Gradescope, so please limit use to only sklearn.

In this project, you should implement **any classifier** for binary classification (not limited to linear classifiers). You are free to implement either any of the classifiers (you can directly use sklearn or implement the classifiers yourself) studied in class (e.g. linear logistic regression, SVM, naive Bayes) or more complex classifiers like a multi-layer perceptron (MLP) classifier, kernel SVM, Boosted Decision Trees, etc) The goal is 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.

Again you can make use of **any** of the classifiers provided in sklearn as long as it fits into the memory and run-time constraints.

A short example of how to implement any classifier from sklearn -

```
# import classifier
from sklearn.dummy import DummyClassifier
# create classifier object
clf = DummyClassifier(strategy="most_frequent")
# fit input data for training
clf.fit(X_train, y_train)
# make prediction
y_pred = clf.predict(X_test)
```

Like MP2, you may 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 applications to reduce the number of risky customers. Usually, users at risk should be 3%, so the imbalanced data is a big problem.

Change the **Python 3** program ***mp3.py*** that creates a classifier (for example: MLP classifier or SVM or any as per your choice) 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.
- *mp3.py*: the Python file you need to work on and submit.
- *MP3_Fall2023.pdf*: this instruction file

Code Instructions

In this project, you need to implement any (supported by sklearn) classifier for binary classification. As an example, you can refer to the multi-layer perceptron classifier. Specifically, there are several things to do:

1. Preprocess the features. There are different kinds of features: binary features, continuous features, and categorical features. You may conduct any form of feature selection, preprocessing, or feature expansion. Examples of data-preprocessing with sklearn can be found here: https://scikit-learn.org/stable/modules/preprocessing.html
2. Implement any classifier with sklearn.
3. Update the parameters of the classifier based on the training data.
4. Evaluate the performance of 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 the 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 features, you can normalize them, or divide them into several bins and convert them to categorical features.
- Examples of some typical values of hyperparameters that are present in various classifiers (*please refer to sklearn documentation for any classifiers that you use to get **exact** information on what hyperparameters to tune*):

    - 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 the **F1 score** for evaluation. The definition can be found in previous MP2 instructions or here. 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 mp3.py.

You can test your program with the ***train.csv*** and ***dev.csv*** provided in this starter package. You can directly run ***mp3.py*** for checking:

$ **python mp3.py**

This will output several metrics you get on the dev dataset. The final score will be graded on another holdout private test set — different from your dev set. (we will train using the same training set).

Submission Instructions

You should upload ***mp3.py*** to Gradescope for grading. Please put all the code into one file.

*[**Note:** Make sure that your data pre-processing code should either be implemented or called by run_train_test(training_data, testing_data)]*

Grading Instruction:

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

Manual Grader (50 points): If your code can run successfully (and it trains a machine learning classifier — simply "return 0" does not get you anything), you will get full points. Otherwise, your score will be based on completeness.

AutoGrader (50 points): Your score depends on the **F1 score** on the private test:
• 15% weightage: If your F1 score is equal to 0.40 or more. You get 15% for just matching 0.40
• 35% weightage: Additional prorated marks when your F1 score is above 0.40:
Score = 35% * (Your score - 0.40)/(0.50-0.40)

• If your F1 score is below 0.40:
Score = 0

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

- Top 10: +10 bonus points, top 10 submissions get 10 bonus points in this part.

Submissions with scores identical to the boundary will also be included. For example, F1 score of the 10th is 0.41, then all submissions with F1=0.41 will be included in the Top 10.

TA baseline F1 score: 0.50

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 not accepted.

Additional Leaderboard instructions:

1. Your leaderboard submissions will be evaluated on test data (dev.csv) shared with you. After the MP3 assignment deadline, we will re-evaluate your final submissions on private test data and use that score for the **final** leaderboard ranking.
2. Note: your leaderboard scores may change after MP3 is completed. We will only report your dev-set score on the leaderboard before the deadline.

   So if you train on the dev set you may ovefit and get a very high F1 score on the leaderboard before MP3 completes, but you will see a drastic decrease in your final leaderboard F1 score.