Document version history

Version	Key changes	Changed on	Changed by
2024/09/12/01	First draft	12-SEP-2024	MDSR
2024/09/12/02	The code should be in a Jupyter notebook file	13-SEP-2024	MDSR

CSE472 (Machine Learning Sessional)

Assignment# 2: Logistic Regression with Bagging and Stacking

Introduction

In ensemble learning, we combine decisions from multiple learners to solve a classification problem. In this assignment, you will implement a Logistic Regression (LR) classifier with bagging and stacking.

Programming Language/Platform

- Python 3 [Hard requirement]

Dataset preprocessing

You need to demonstrate the performance and efficiency of your implementation for the following three different datasets.

1. https://www.kaggle.com/blastchar/telco-customer-churn
2. https://archive.ics.uci.edu/ml/datasets/adult
3. https://www.kaggle.com/mlg-ulb/creditcardfraud

They differ in size, number and types of attributes, data quality (missing attribute values), data descriptions (whether train and test data are separate, attribute description format, etc.), etc. Your core implementation for LR, bagging and stacking must work for all three datasets without any modification. You should add a separate dataset-specific preprocessing script/module/function to feed your learning engine a standardized data file in matrix format. On the evaluation day, you will be given another new dataset for which you must create a preprocessor. Any lack of understanding about your own code will severely hinder your chances of success. Here are some suggestions for you,

1. Design and develop your own code. You can take help from tons of materials available on the web, but do it yourself. This is the only way to ensure you know every subtle issue that needs to be tweaked during customization.
2. Do not assume anything about your dataset. Keep an open mind. Deal with their subtleties in preprocessing.
3. Use Python library functions for common preprocessing tasks such as normalization, binarization, discretization, imputation, encoding categorical features, scaling etc. Visithttp://scikit-learn.org/stable/modules/preprocessing.html for more information.
4. Go through the dataset description given in the link carefully. Misunderstanding will lead to incorrect preprocessing.
5. For the third dataset, don't worry if your implementation takes long time. You can use a smaller subset (randomly selected 20000 negative samples + all positive samples) of that

dataset for demonstration purpose. Do not exclude any positive sample, as they are scarce.

6. Split your preprocessed datasets into 80% training and 20% testing data when the dataset is not split already. From the training set, separate out 20% data for validation. You can use the Scikit-learn built-in function for the train-test split. For splitting guidelines, see https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

Use bagging (sampling with replacement) to generate 9 training sets, and then train LR models on each of them. These will be the base learners for the stacking ensemble. You should use every tool in your arsenal to achieve good and generalizable results. Experiment with the different regularizers discussed in the class.

Then train another LR model as the meta classifier to finalize the stacking ensemble. You should also develop a simple majority voting based ensemble and make a comparative analysis.

Write clean and modularize your code.

Performance Evaluation

1. Always use a constant seed for any random number generation so that your experiments are reproducible.
2. Draw violin plots for each performance metric for the 9 bagging LR learners. (See https://medium.com/analysts-corner/how-to-create-violine-chart-in-python-3517e5d4a652)
3. Make a comparative analysis table as follows.

Performance on Test set

	Accuracy	Sensitivity	Specificity	Precision	F_1-score	AUROC	AUPR
LR*							
Voting ensemble							
Stacking ensemble							

** For LR, report average ± stdev for the 9 bagging LR learners*

Submission

1. Upload the codes in Moodle within **20-SEP-2024 (Friday) 10:00 PM**. (Strict deadline)
2. You need to submit a report file in PDF format containing the following items (No hard copy is required.):
    a. Clear instructions on how to run your script to train your model(s) and test them. (For example, which part needs to be commented out when training each dataset,

how to run evaluation etc.) We would like to run the script in our computers before the sessional class.

b. The table(s) and plot(s) mentioned in the performance evaluation section with your experimental results.

c. Any observations.

3. Write code in a Jupyter Notebook file (*.ipynb). Please provide description of the various code segments. Rename it with your student id. For example, if your student id is1905123, then your code file name should be "1905123.ipynb" and the report name should be "1905123.pdf".
4. Finally make a main folder, put the code and report in it, and rename the main folder as your student id. Then zip it and upload it.

Evaluation

1. You have to reproduce your experiments during in-lab evaluation. Keep everything ready to minimize delay.
2. You are likely to give online tasks during evaluation which will require you to modify your code.
3. You will be tested on your understanding through viva-voce.
4. If evaluators like performance, efficiency or modularity of a particular code, they can give bonus marks. This will be completely at the discretion of evaluators.

Warning

1. Don't copy! We regularly use copy checkers.
2. First time wrongdoers (either copier or the provider) will receive **negative** marking because of dishonesty.
3. Repeated occurrence will lead to severe departmental action and jeopardize your academic career. We expect fairness and honesty from you. Don't disappoint us!