CS 165A – Artificial Intelligence, Winter 2021

Machine Problem 1

(100 points)

Due: ***Thursday, Feb 18, 2021, 11:59pm***

Notes:

- Make sure to re-read the "Policy on Academic Integrity" on the course syllabus.
- Any updates or corrections will be posted on the Announcements page in web page of the course, so check there occasionally.
- You have to work individually for this assignment.
- Each student must turn in their report and code electronically.
- Responsible TA for Machine Problem 1: Kaiqi Zhang kzhang70@ucsb.edu
- **Visit Piazza for updates and possible changes**

---

1. Problem Definition

The current exponential increase in coronavirus disease 2019 (COVID-19) is reaching a calamitous scale in the United States, potentially overwhelming the health care system and causing substantial loss of life. During the entire course of the pandemic, one of the main problems that healthcare providers have faced is the shortage of medical resources and a proper plan to efficiently distribute them.
They have been in the dark failing to understand how much resource they could even in the very next week as the COVID-19 curve has swayed very unpredictably. In these tough times, being able to predict what kind of resource an individual might require at the time of being tested positive or even before that will be of great help to the authorities as they would be able to procure and arrange for the resources necessary to save the life of that patient.
The precondition of a patient influences ones fertility rate. In this homework, you will need to predict whether a patient unfortunately loss ones life from this disease using a Naive Bayes Classifier. A Naive Bayes Classifier computes the likelihoods of a feature that fall into different categories and most commonly chooses the category that is the most probable. Please refer to the lecture notes and textbook for these concepts and explain any design and model choice you make in a report.

1.1 Program Input

You will be given two separate datasets, a training dataset which will be contained in a file named "*covid_train.csv*", and a public testing dataset which will be contained in a file named "*covid_valid.csv*", which are randomly sampled from a public dataset. These files are in Comma-separated values (CSV) (https://en.wikipedia.org/wiki/Comma-separated_values) format, which can be opened with either Excel or text editor, and you can read it in the same way as text file. Each line in the file represents one row, and columns are separated by comma ",". Each of these files will have an arbitrary number of lines (rows) and each line represents one record. **The 5th column, "date_died", is the label.** A valid data indicates that the patient

unfortunately passed away on that day, and "9999-99-99" means the patient survived. **You need to predict whether the patient passed away from the disease (label 1) or not (label 0).** The 2nd and 3rd columns, the patients' date of entry to hospital and date of first symptom may have format either "date-month-year" or "data/month/year", and your program need to handle both cases. The column "age" represents the actual age of the patient. The rest columns contain binary labels "1" and "2", and other values (97-99) means this feature is unknown. You should choose which features to use and how.
Your program should take as input the two file path names with the training and public testing datasets through **command-line arguments** (https://en.wikipedia.org/wiki/Command-line_interface#Arguments).
Here is an example of how you should execute your code depending on which language you use (assuming that the dataset files are in the same directory with the executable):

- C/C++: `./NaiveBayesClassifier covid_train.csv covid_valid.csv`
- Python: `python NaiveBayesClassifier.py covid_train.csv covid_valid.csv`

1.2 Executable

The executable of your program must be named "NaiveBayesClassifier". Please also provide an executable wrapper script that just executes your code. Name this script "NaiveBayesClassifier.sh". If you're using C++, please include the commands to build the executable in this script (eg, `make` or `g++ NaiveBayesClassifier -o NaiveBayesClassifier`). For C/C++ you might have the following simple script:

```
#!/bin/bash
make
./NaiveBayesClassifier $@
```

for Python 3:

```
#!/bin/bash
python3 ./NaiveBayesClassifier.py $@
```

The above scripts will execute your code when you run the wrapper scripts like this:

```
./NaiveBayesClassifier.sh training.txt testing.txt
```

These scripts also allow command line arguments which will be directly passed to your code.

1.3 Program Flow

Your program should open and read every line (document and label) of the training dataset, and using the existing knowledge of the labels to train a Naive Bayes Classifier. You need to time how long this phase takes and print it at the end. Once you have built the classifier, your program should apply it on the training dataset and calculate its accuracy. Then, your program should also apply the classifier on the public testing dataset and calculate a separate accuracy. This last phase (applying the classifier on the two datasets) must also be timed.

1.4 Evaluation

The dataset is imbalanced. In the training set, about 94% of the records are negative (0), and in validation set and testing set, about 72% of the records are negative (0). Your program should at least beat the naive predictor that outputs all 0s. In other word, your program should get an accuracy at least 75% to get credit.

1.5 Program Output

The output of your program mush include and only include the prediction **label (0 or 1)**, one label per line. 0 means the patient survived, and 1 means the patient passed away. All output should be directed in the standard output, ***DO NOT*** write the results in a file.

During grading we will use the same training dataset but a different testing dataset (which is called the ***private*** testing dataset). So don't use the public testing dataset to train your classifier hoping to achieve greater accuracy.

This is called overfitting and it would make your classifier very accurate on this specific dataset only, but not on unseen reviews. Also, there will be a time limit of 10 minutes and your program must finish execution within this time. In this testing dataset, the labels are hidden and a dummy label (N/A) will be placed instead. You should simply ignore these labels, and print your prediction result.

Note: You may not want to treat the label as part of the features.

1.5 Implementation Restrictions

You may use C/C++ or Python3 for this programming assignment. The grading environment is based on Ubuntu 18.04. If you're writing with Python3, you can use any built-in module, as well as numpy, pandas or scipy. Any machine learning libraries, eg. Scikit-learn, are not allowed. If you're writing with C/C++, you can use any built-in library.
Keep in mind while implementing the classifier that it must finish running within 15 minutes on gradescope. If it runs for more than 15 minutes, the grader will terminate your program and your accuracy score will be 0.

2. Report Preparation

As for the report, it should be between 1 and 3 pages in length (no more than 3 pages) with at least 11pt font size and should consist of at least the following sections:
• **Architecture**: Explanation of your code architecture (briefly describe the classes and basic functionality)
• **Preprocessing**: How you cleaned and how you represent a document (features)
• **Model Building**: How you train the Naive Bayes Classifier
• **Results**: Your results on the provided datasets (accuracy, running time). Also give a sample of the 10 most important features for each class (positive or negative)
• **Challenges**: The challenges you faced and how you solved them
• **Weaknesses**: Weaknesses in your method (you cannot say the method is without flaws) and suggest ways to overcome them.

There should be only one pdf report which includes all the above (no additional readme file). Please include your name, email and perm number in your report.

3. Submission Guidelines

- C/C++: Include your source code files (.cpp, .c, .hpp), your header files ".h" if you have any, and any other files that are needed to compile and run your code successfully. Do not submit the executable file. **Please include the script to compile your program in "`NaiveBayesClassifier.sh`".** Autograder will not compile your code automatically.
- Python: Include your source code files ".py", an script file named "`NaiveBayesClassifier.py`" and any other files that are needed to run your code successfully. Please indicate which version of python you use in the report.

Please ensure that your code is executable as follows:
```
% bash NaiveBayesClassifier.sh training.txt testing.txt
```

4. How to Submit

We are migrating to Gradescope!

Submitting through Gradescope is easy. just submit these files directly:

Makefile (if needed)
NaiveBayesClassifier.sh
source code
PDF report

Do not submit any data. Do not put them in a directory or zip file. If you did everything correctly, the autograder should run and give you your accuracy in a few minutes. There is no limit on the number of submissions, but only your last submission will count.

The Gradescope autograder is quite barebone by default, so make a post on Piazza if it does not have the package you need.

If, for some reason, you are not able to run on Gradescope, you can still submit it and I'll grade them manually. Please list in the report how to run your code and what's the environment needed in this case. However, programs that doesn't run on Gradescope will not be actively graded before the due date, and as such you will not be able to see your accuracy score in real-time.

5. Grading

Grade Breakdown:
- **20%** Complete Report
- **10%** Makefile and execution specifications if needed.
- **10%** Correct program specifications: reads file names from command line arguments, produces correct standard output results, no segfaults or exceptions
- **60%** Test accuracy and runtime of your algorithm

Plagiarism Warning: We are going to use software to check plagiarism. You are expected to finish the project by yourself without using any code from others.

5.1 Leaderboard and Bonuses

The top-3 scorer will get extra credit for this assignment. You can view the current leaderboard on Gradescope.

Bonus tier 1: The student whose classifier testing accuracy is ranked at the top will get 50% extra credit.
Bonus tier 2: The students whose classifier testing accuracy is ranked among the top-3 will get 25% extra credit.

If there's a tie in accuracy, the one with lower run time will take the lead. There is no limit on the number of submissions, but only the last submission will count.

5.2 Accuracy and Time

The main weight of the grade (60%) will be based on the testing accuracy (on ***private*** testing dataset) and running time of your model. You should make your model as accurate as possible, but keep in mind that the worst-case accuracy is 0.50, which can be achieved by choosing a random label for each document. Be aware that your code will be terminated if it is not finished within **15 minutes** and your accuracy score will be 0.

Your accuracy score will be linearly interpolated between the scores listed below:

accuracy on private test dataset	Percentage score for the accuracy component
0.750	0%

0.751	80%
0.800	80%
0.900	100%

The grading scale is designed so that any implementation of Naïve Bayes that performs better than random guess will get at least 80%. If you did better than random guess but could not reach accuracy of 0.75, we may bump you up if you can demonstrate in your report that you have made a reasonable amount of progress in your implementation of Naïve Bayes. A correct implementation of multinomial Naïve Bayes should get accuracy approximately 0.87. We leave an additional 30% headroom to encourage you to go beyond that.