CS 165A – Artificial Intelligence, Winter 2022

Machine Problem 1

(100 points)

Due Thursday, Feb 17 2022, 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: Nagarjun (nagarjun@ucsb.edu) Office Hours: Tue 3-4pm
- Visit Piazza for updates and possible changes
- *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.

---

1. Problem Definition

It is year 2100. The Martians and Earthers are celebrating the beginning of 22nd century with a huge Interplanetary Olympics Competition. You (yes, **you**) are now the part of the coaching team for Earthers Representation at the competition. Being an AI expert, you are asked to create a program which will judge the performance grade of athletes. So that the coaches can optimize the coaching routines for all athletes. Good luck and make Mother Earth proud!

You will be given a dataset which contains information about athletes. Each information instance includes attributes (features) like age, height, blood pressure, sit-up counts etc. The task is to predict whether the athlete has high grade of performance or not. You are asked to write a Python program which automatically classifies athletes' performance grade as either label 1 (High) or label 0 (Not High) by implementing a Naive Bayes Classifier. You also need to explain any design choice, the reasoning behind the approach to increase the accuracy of model 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 "training.txt" and a public testing dataset which will be contained in a file named "testing.txt". Each of these files will have arbitrary number of rows and 12 columns. Columns are separated by comma (',') with first 11 columns as features and 12th column as the label.

The name of the 11 columns in order are as follows: age, gender, height_cm, weight_kg, body fat_%, diastolic, systolic, grip_force, sit_and_bend_forward_cm, sit_up_count, broad_jump_cm. The target column is named class.

Your program should take as input the two file path names with the training and public testing datasets through command line arguments.

Here is an example of how you should execute your code.

Python3

```
python3 NaiveBayesClassifier.py training.txt testing.txt
```

Java **(openjdk should be compatible with version 11.0.13 on autograder)**

```
javac NaiveBayesClassifier.java
java NaiveBayesClassifier training.txt testing.txt
```

For C++ **(g++ should be compatible with version 7.5.0 on autograder)**

```
g++ -o naivebayesclassifier.c
./a.out training.txt testing.txt
```

For C **(gcc should be compatible with version 7.5.0 on autograder)**

```
gcc -o naivebayesclassifier.c
./a.out training.txt testing.txt
```

[Please note that the dataset files should be in the same directory as the python program]

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".

For Python 3:

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

Java:

```
javac NaiveBayesClassifier.java
java NaiveBayesClassifier $@
```

Cpp:

```
g++ -o naivebayesclassifier.cpp
./a.out $@
```

C:

```
gcc -o naivebayesclassifier.c
./a.out $@
```

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 (features and its 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 balanced. In the training, public and private testing sets, they all have 50% of instances with label 0 (Not High). Your program should beat the naive predictor that outputs all 0s and get an accuracy at least 55% to get credit.

1.5 Program Output

The output of your program must include and only include the prediction **label (0 or 1)**, one label per line. 0 means the performance grade of athlete is not high, and 1 means performance grade of athlete is high.

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 do not 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 one. Also, there will be a time limit of 10 minutes and your program must finish execution within this time.

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

1.5 Implementation Restrictions

If you are using Python3 for this programming assignment. You can use any built-in module, as well as numpy, pandas or scipy. Any machine learning libraries, eg. Scikit-learn, are not allowed.

Keep in mind while implementing the classifier that it must finish running within 10 minutes on gradescope. If it runs for more than 10 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 atleast 11pt font size and should consist of at least the following sections:

- **Architecture:** Explanation of your code architecture. (Describe basic functionality and classes)
- **Preprocessing:** How you represent an instance
- **Model Building:** How you train the classifier
- **Results:** Your results on the provided datasets (accuracy, running time, other metrics)

- **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

Include your source code files ".py, .java, .c, .cpp".
Include "`NaiveBayesClassifier.sh`" which contains info of compiling and running the program.

Please ensure that your code is executable as follows:

```
% bash NaiveBayesClassifier.sh training.txt testing.txt
```

4. How to Submit

We will be using Gradescope to submit the assignment. Submitting through Gradescope is easy. Just submit these files directly:

- 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%** Execution specifications
- **10%** Correct program specifications: reads file names from command line arguments, produces correct standard output results, no segmentation faults or exceptions
- **60%** Test accuracy and runtime of your algorithm

5.1 Leaderboard and Bonuses

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

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.

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 50%, which can be achieved by choosing 0 for each instance. Be aware that your code will be terminated if it is not finished within **10 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
<0.55	**0%**
0.55	**60%**
0.75	**100%**
1.0	**120%**

Table 1: Score calculation for different accuracy range

The grading scale is designed so that any implementation of Naive Bayes performs less than 0.55 will get **0** score for accuracy component. If you get better than or equal to 0.55, you will get at least 60%. For accuracy between 0.55 and 0.75, scores will be linearly interpolated between 60% and 100%. For accuracy between 0.75 and 1.0, scores will be linearly interpolated between 100% and 120%. A correct implementation of Naïve Bayes should get about 0.75+ accuracy. We leave an additional 25% headroom to encourage you to go beyond that.