CS480 – Project Phase 2
Assigned on: Monday, 11/2/2015
Due: Sunday, 11/15/2015, 11:59pm
Please submit your solutions through black board assignment page.

Description (the same from phase 1)

You are introduced an electronic product that has a $*value* and a $*price*. Each product can be in an *Excellent* condition, which works flawlessly, or in a *Trash* condition, where, hmm, it is trash. If you buy a product and if it is *Excellent*, your wealth increases by $*value*-$*price*. If, however, the product turns out to be *Trash*, your wealth decreases by $*price.* You do not know whether a product is *Excellent* or *Trash* until you buy it; however, there will be clues for you to make an informed decision. Your objective is to increase your wealth as much as possible.

The project will consist of multiple phases. The number of phases is tentatively set to 4. Depending on time and interest, we might need to eliminate one of the existing phases or add a new phase.

According to the syllabus (http://www.cs.iit.edu/~mbilgic/classes/fall15/cs480/CS480-FALL15-SYLLABUS.pdf), the project is worth 20% of the grade.

Important: the project is different from homeworks and programming exercises. The homeworks and programming are based on what is already covered in the lectures. The project, however, is not based on past materials in class; instead, it requires you to do research and innovative thinking. Therefore, you need to start each phase of the project as soon as it is assigned.

Phase 2

In this phase, unlike phase 1, you are not provided the probability of being Excellent. Rather, your agent needs to learn the features of Excellent and Trash products by going through an existing list of products whose conditions and features are known. Once it learns about Excellent and Trash products, given a product's features, it should be able to predict the probability of it being Excellent.

You are provided two python files and several product files (csv files):

- agents.py
  - This file has the base agent Agent and a baseline agent.
- simulate_agents_phase2.py
  - This file simulates the baseline agent and your agent.

The product files are pairs of files named as dataseti_train.csv and dataseti_test.csv where i ranges between 1 and 4. These files are used to create X_train, y_train, X_test, and y_test (X and y are described below.) Your agent is provided the X_train, y_train during training and it is tested using X_test. During testing, y_test is not revealed to the agent.

Your task is to create an agent called Agent_hawk_username where hawk_username is your Hawk username. For example, for me, it would be Agent_mbilgic.py file. You need to create a class called Agent_hawk_username that inherits the base Agent and overrides two methods:

1. train(self, X, y)
   Given:
   X: A matrix (2D numpy array) where rows correspond to products and columns correspond to feature values of those products. In this phase, each feature is a binary feature.
   y: A 1D numpy array where each entry corresponds to whether a product is Excellent or Trash. The i^{th} entry in y corresponds to the i^{th} row in X.
   Task:
   Train your agent to learn features of Excellent and Trash products.
2. predict_prob_of_excellent(self, x)
   Given:
   x: A 1D numpy array that corresponds to a single product.
   Task:
   Predict the probability of x being Excellent.

Rules:

1. You can use only numpy, scipy, and standard Python libraries. You cannot use any other external library.
2. Your idea and implementation has to be your own. You are not allowed to discuss it with your friends, except for help with Python. Your idea and implementation will be compared with your peers' solutions. Similarities will be investigated for possible cheating.

3. Researching and using existing machine learning techniques, including the ones in the textbook, is allowed (bot not required). However, if you decide to use an existing idea, then you have to explicitly state which machine learning technique you are using and provide a citation for it in your report. Regardless, the implementation has to be your own. For example, if you like, you can implement naïve Bayes or logistic regression as described here http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf.
4. Your agent is not allowed to read the train and test files directly. Your agent is not allowed to access y_test.
5. If you like, your agent can override the default constructor as long as we can still call the constructor using your agent's name and everything works as expected.
6. If you need, you can create additional classes and methods in your Agent_hawk_username.py file.

Submit a zip file that contains two (and only two) files:

1. agent_hawk_username.py file.
2. A report in .pdf format. The report should have:
    a. A detailed description of the train and predict_prob_of_excellent methods.
    b. The simulation results comparing your agent to the baseline agents. (This is the output from the simulate_agents_phas2.py file after you add your agent to the list.)

The simulation code evaluates the agents in three categories: agent wealth (the higher the better), log-loss (the smaller the better), and 0/1 error (the smaller the better). Your agent should beat the other agents in wealth. Your score will depend on your creativity, correctness of your implementation, how well your agent performs in wealth, documentation of your code, and quality of your report.

Please remember to put your files (agent_hawk_username.py and report.pdf) in a folder, zip it, and submit it.