CS 373 (Spring 2020): Assignment 5 - Prediction

Due 11:59pm EDT, 2 April, 2020. Updated 3/24/20 to clarify provided data.

For this assignment, you will build and evaluate a classifier using Python. You can choose what approach to use; you will be graded based on 1) your description of how you did this (how it address the issues in an appropriate fashion), and 2) how good your classifier does at solving the problem given.

The task is a simple form of what is known as *transfer learning*. Normally, we think of machine learning as having training and test data drawn I.I.D. from the same distribution. With transfer learning, the training and test sets are drawn from *different* distributions. There are many real-world examples where this is useful. For example, suppose we want to hold a clinical trial of a potentially dangerous cancer treatment. Because of the risk, we only give it to people who have used up all other treatment options, but are still in reasonably good shape. The test is successful for some patients, and we have developed a machine learning model that can identify which of the patients it works for. We want to apply this model to determine who should use the drug - but the people we are applying to include patients who have not used up all other treatments, or who are not in good shape. Can we expect the model to work well?

A simpler analogy, that directly models what you will do, is: Assign a limited set of resources to those who will use them most effectively. (An example might be college admissions, the number of applicants who COULD succeed at Purdue may exceed the number who can be admitted.) You have training data that is divided into two classes (those who use the resources effectively, and those who don't.) and you classify which is which. While this seems a simple binary classification problem, there is a catch. In the training data, 50% of the individuals use the resources effectively, and 50% don't. However, we do not have enough resources to give them to half the individuals, we only have enough for 10%. Given data *from the same distribution as the training data*, your classifier should only return the "+" class approximately 10% of the time at test, even though half the samples would be labelled "+". In other words, your classifier should detect data points that are **strongly positive**.

In this assignment, the difference from traditional machine learning is simply that the distribution of the class varies between the training data you are given (50% +) and the unseen test data (10% +). The distribution of other features is the same. You want to develop a model that can transfer from the data you will use for training, to the unseen data (that we will use to test your model.) In other words, it *should* output + roughly 10% of the time, when given data drawn from the same distribution as the training data.

Note that you have to make this decision independently for each new item, you can't

"remember" how many + you have output and output only the required number. We may give you a test problem where all the test data are drawn from the − class, if you give 10% + in this case, you'll lose points. It is only when the test data are from the same distribution as the training data (50% + and 50% − in the training data) that you should produce roughly 10% + output.

A simple way of doing this would be to train a classifier, and if the output is a +, then roll a multi-sided dice. If it comes up 1, output +, otherwise output − . (Quick math exercise - how many sides should this dice have?) You should be able to do better, choosing the *most* likely to be +. Another would be to sample your training data, so that you have only 10% from the + class, and use this to train your classifier. Unfortunately, this is also unlikely to select those who would make the best use of the resources, and you'll find you have too little training data for this to work well.

You should be able to work out good solutions based on k-nearest neighbor, decision trees, or Naïve Bayes. Choose one of the three models for this assignment. The method you use is up to you. You should include in your write-up both a mathematical analysis of why your solution should return roughly 10% of + class, as well a discussion of why it should give better results than the "random sampling" approaches above.

While we do not proscribe the use of classifiers in Python libraries, you will probably find it easier to program this "from scratch". (Unless you want to use the "choose randomly" approach above, which will not earn full credit.)

HINT: Think carefully about how your chosen algorithm makes the final decision (prediction). Can you tune this to achieve the 10% + objective?

Dataset Description

All required data files can be found in

`https://www.cs.purdue.edu/~clifton/cs37300/asn5/`

They are also available to you in Vocareum, although you may find you want to edit or otherwise create additional files, and not just use the ones provided.

You will use the student performance dataset `data.csv`, which has 500 student records with 31 attributes. The last attribute G is a binary label indicating a student's performance ('+' or '-') and more information about the attributes can be found in `README.txt`. You are also provided with a test dataset `data_test.csv` consisting of 100 student records.

Note that the test dataset we gave you is a subset of the training dataset we gave you. To do actual tests of accuracy, you will need to split a test dataset out of the training data you are given. When we test your program, we will use different training and test datasets. You should not expect the test data items to exactly match anything in the training data. You can count on the training data having 50% + and 50% −, but the test data could be from a different distribution (e.g., a test set that is all − in the original distribution, where you would be expected to produce almost all −; or all +, where you would be expected to output around 20% +.)

Input/Output Specifications

Your program should take as input a training dataset and a test dataset, i.e., two command line arguments for the paths of training set and test set respectively. It will be called as follows:

```
predict.py <path of training set> <path of test set>
```

The format of the files is given in `sample_train.csv` and `sample_test.csv`. Note that `sample_test.csv` isn't the same format as data.csv or data_test.csv - the "G" column (class) is missing. This corresponds to a real-world use, where you don't know the prediction. `data_test.csv` corresponds to what you see when developing an algorithm - a test set where you try to predict, and the compare with the known true answer (although remember, data_test is based on the 50% + and 50% −, not what you should really be producing.)

Your program should develop a model based on the training set given as an input argument, and produce + or − for each of the test items and print it to standard output, one for each line and in the order given in the test dataset file (second argument). The outputs should look like this:

```
+
+
-
-
+
-
...
```

Note that your code will be evaluated on Vocareum, an online platform which runs an Ubuntu 16.04 image and has enough libraries pre-installed for this assignment. You should have received an email with instructions on accessing Vocareum if you haven't used it before. Direct link to this assignment is `https://labs.vocareum.com/main/main.php?m=editor&nav=1&asnid=130070&stepid=130071`, and is also available through Blackboard (under course content.) Once you login, you can upload your code (i.e., `predict.py`) along with sample datasets (i.e., `sample_train.csv` and `sample_test.csv`) to Vocareum, and use the interactive console to test the code. (The sample datasets should be preloaded, but for some reason aren't showing up yet. This should be resolved soon.) Please ensure that your code is compatible with the platform. Especially, you are only allowed to use libraries that are pre-installed on Vocareum. So you might want to check the Vocareum environment before starting programming (or, if you wish, do everything within that environment.)

What to Turn In

You should turn in, through Gradescope, a (typeset) write-up containing the following sections:

1. **Method Overview**: What basic classification approach you are using, and how you have modified it to support transfer learning. One paragraph.

2. **Constraint Satisfaction**: Show why your approach is expected to return 10% positive. This should be in the form of a mathematical proof sketch.

3. **Efficacy**: Discuss, or better still give a proof sketch, of why your method should return the *best* 10% of individuals.

4. **Empirical Evaluation**: You should test your approach and evaluate how well it works. Describe briefly the tests you have run and what they are evaluating; for each, give your test results as a table or graph (or single number, whichever is most appropriate.)

You will also need to submit your code through Vocareum (make sure you test that it runs properly under Vocareum, if you have developed elsewhere.)