HMC CS 158, Spring 2018
Problem Set 1 Project: Titanic Survival

Goals:

- To tackle the first steps in any ML pipeline: visualizing data and evaluating baseline classifiers.
- To start familiarizing yourself with the Python libraries `numpy` and `matplotlib`.

For this assignment, you can work individually though you are encouraged to work with a partner. We will continue analyzing this data set next week, so if you work with a partner, you must work with the same partner next week. You should sign up for partners on Canvas (People → PS1 Groups). If you are looking for a partner, try Piazza. If, after trying Piazza, you are having trouble finding a partner, e-mail Jessica.

Submission

You should submit any answers to the exercises in a single file `writeup.pdf`. This writeup should include your name and the assignment number at the top of the first page, and it should clearly label all problems. Additionally, cite any collaborators and sources of help you received (excluding course staff), and if you are using slip days, please also indicate this at the top of your document.

Your code should be commented appropriately. The most important things:

- Your name and the assignment number should be at the top of each file.
- Each class and method should have an appropriate doctsring.
- If anything is complicated, it should include some comments.

There are many possible ways to approach the programming portion of this assignment, which makes code style and comments very important so that staff can understand what you did. For this reason, you will lose points for poorly commented or poorly organized code.

When you are ready to submit, make sure that your code compiles and remove any debugging statements. Then rename the top-level directory as `<username1>_<username2>_ps1`, with the usernames in alphabetical order (e.g. `hadas_yjw_ps1`). This directory should include the electronic version of your writeup and main code, any files necessary to run your code (including any code and data provided by staff), and follow the same structure as the assignment directory provided by staff. So, for this assignment, your directory should have the following structure:

- `<username1>_<username2>_ps1/`
  - `data/`
    - `titanic_train.csv`
    - `titanic_test.csv`
  - `source/`
    - `titanic.py` (with your modifications)
    - `util.py`
  - `writeup.pdf`
  - `titanic.pdf` (pdf printout of `titanic.py`)

Package this directory as a single `<username1>_<username2>_ps1.zip` file, and submit the archive. Additionally, to aid the staff in grading, submit your `pdf`'s as separate files.

Parts of this assignment are adapted from course material by David Kauchak (Middlebury).

Introduction[1]

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this problem, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

Starter Files

---

code and data

- code : `titanic.py`
- data : `titanic_train.csv`, `titanic_test.csv`

documentation

- Metrics:
  http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

---

Download the code and data sets from the course website. For more information on the data set, see the Kaggle description: https://www.kaggle.com/c/titanic/data. (The provided data sets are modified versions of the data available from Kaggle.[2])

Note that any portions of the code that you must modify have been indicated with `TODO`. Do not change any code outside of these blocks.

[1] This assignment is adapted from the Kaggle Titanic competition, available at https://www.kaggle.com/c/titanic. Some parts of the problem are copied verbatim from Kaggle.

[2] Passengers with missing values for any feature have been removed. Also, the categorical feature `Sex` has been mapped to `{'female': 0, 'male': 1}` and `Embarked` to `{'C': 0, 'Q': 1, 'S': 2}`. If you are interested more in this process of *data munging*, Kaggle has an excellent tutorial available at https://www.kaggle.com/c/titanic/details/getting-started-with-python-ii.

1 Visualization [5 pts]

One of the first things to do before trying any formal machine learning technique is to dive into the data. This can include looking for funny values in the data, looking for outliers, looking at the range of feature values, what features seem important, etc.

(a) **(4 pts)** Run the code to make histograms for each feature, separating the examples by class (e.g. survival). This should produce seven plots, one for each feature, and each plot should have two overlapping histograms, with the color of the histogram indicating the class. For each feature, what trends do you observe in the data?

(b) **(1 pts)** We are often also interested in understanding the interactions between features. Of course, such a task is complicated as (1) visualizing interactions between m out of n features requires $\binom{n}{m}$ plots, and (2) as we live in a 3D-world, we have trouble interpreting plots in high dimensions. Therefore, we will focus here on pairwise interactions.

Make a scatter plot[3] of `Fare` as a function of `Age`, where each point represents an example (e.g. passenger) and the color of the point indicates the class (e.g. survival). Include this plot in your writeup, along with a 1-2 sentence description of any trends you observe.

2 Evaluation [2+2 pts]

Before trying out any classifier, it is often useful to establish a *baseline*. For this exercise, you will evaluate some simple (and stupid) classifiers on the data. We will model our classifers after those in `scikit-learn`[4].

(c) **(2+2 pts)** We have implemented one baseline classifier, `MajorityVoteClassifier`, that always predicts the majority class from the training set. Read through the `MajorityVoteClassifier` and its usage and make sure you understand how it works.

Your goal is to implement and evaluate another baseline classifier, `RandomClassifier`, that predicts a target class according to the distribution of classes in the training data set. For example, if 60% of the examples in the training set have `Survived = 0` and 40% have `Survived = 1`, then, when applied to a test set, `RandomClassifier` should randomly predict 60% of the examples as `Survived = 0` and 40% as `Survived = 1`.

Implement the missing portions of `RandomClassifier` according to the provided specifications. Then train your `RandomClassifier` on the entire training data set, and evaluate its training error. If you implemented everything correctly, you should have an error of 0.485.

Extra Credit (+2): For a binary classification task, prove that the expected accuracy of `MajorityVoteClassifier` is at least as high as that of `RandomClassifier`.

[3] You may find the pyplot API at http://matplotlib.org/api/pyplot_api.html useful.

[4] Almost all of the model techniques in `scikit-learn` share a few common named functions, once they are initialized. These are `some-model-name.fit(...)`, `some-model-name.predict(...)`, and `some-model-name.score(...)`. You can always find out more about them in the documentation for each model.