BIM 155
Xianglong Wang

Project 2
Colon Cancer Screening

2025-04-27
xlowang@ucdavis.edu

1. Introduction

Welcome to Project 2, colon cancer screening. You may find approaching this project to be simpler than Project 1; indeed, features have been generated for you. However, you will find out that the number of features gets overwhelming: there are 20073 features available for you to work with. It is up to you on how you manage these features and control potential overfitting with just 148 data points.

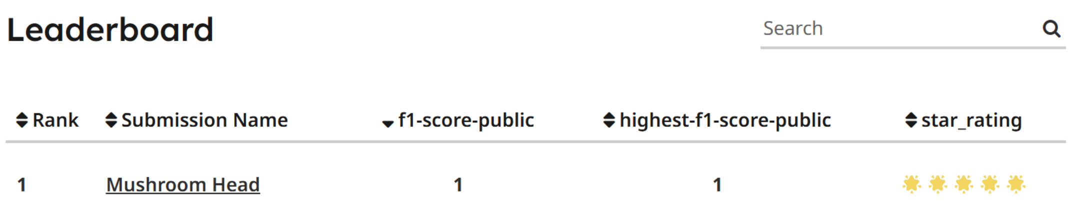
The data you have are from 148 total subjects. The subjects include 50 healthy subjects and 98 subjects with Stage II colon cancer. Stage II colon cancer typically have limited local spread and recover relatively well; however, colorectal cancers have high recurrence rates, and for stage II colon cancer, about 20% of the patients' cancer come back after surgery removal. Among the 20%, many subjects experience metastatic spread of the cancer cells and end up losing their lives.

Samples were obtained from the 148 subjects during colonoscopy (healthy) or surgery removal of the tumors (cancer patients), yielding 246 samples. A mucosa (normal) sample were obtained for each healthy subject, and a pair of normal/tumor samples were obtained for each cancer patient. DNA sequences of each sample were then obtained and the prevalence of 20070 genes were provided for you, along with the side (left/right) of the sample and the subject's age and sex. Note that the genetic information may not contain information whether the gene has mutated. 148 of these samples (30 healthy and 59 normal/tumor samples) are available for you in the training set, and the rest of the subjects are contained in the test set.

Your goal is to try using the available genetic information to predict whether the sample is normal (class 0) or contains cancer tissue. If the sample has cancer, determine whether it is unlikely to recur (class 1) or will recur within the next three years (class 2). To simplify the project, I did not provide you with information about mRNA sequencing and copies (which can reflect genetic mutation, for example, since mutations affect mRNA length and stability). Therefore, do not be too surprised if you cannot get your performance to be extremely high (> 0.90 ICC).

Here is some important information for Project 2.

- The deadline for submission is 5/14, 11:59pm. The late deadline is 100 hours after the regular deadline, 5/19, 3:59am.
- You have three chances to submit your predicted results within any continuous 24-hour period on GradeScope to `project-debug`. If you submit more than three times over a day, then the GradeScope will reuse your most recent submission and tell you that you have submitted too many times.
- Remember, `project-debug` is not graded. `project-debug` will tell you your performance on a public set as shown in Figure 1, where the score of your **current submission** and your **previous submission** will be given to you, along with the star rating that we promised in lecture.
- You will have three total submissions available for `project-final` on GradeScope. You should wisely choose your model to submit based on your perceived confidence and of course, the performance on `project-debug`.
- The highest score and the star rating of your `project-final` should update with additional submissions (and takes the highest). However, if you submit too many times, GradeScope will lock you out. Therefore, I recommend you submit `project-final` after you finalized your `project-debug` submissions.


Leaderboard

Rank	Submission Name	f1-score-public	highest-f1-score-public	star_rating
1	Mushroom Head	1	1	★★★★★

Figure 1: A sample leaderboard of project 1-debug.

- Due to GradeScope's limitations, there is no way for me to recall a submission.
- The autograder should work even if your `Id` field is out of order.
- We are expecting ties for top place of this project. Therefore, this project requires you to implement a `train(df)` and `predict(df, model)` function to allow your model to be executed over the cloud with the efficiency of your code measured. Code that runs faster will be ranked ahead of the code that runs slower to break ties.
- The fastest model within the top 5 submissions will get a 1-point bonus added to their performance grade.
- The autograder's Python environment has the following packages: `numpy`, `scipy`, `matplotlib`, `scikit-learn`, `statsmodels`, `pandas`. Just in case you would like to try complex methods, we also enabled `xgboost` (no GPU). If you really need to use extra packages, send Dr. Wang an email.
- Although it may be possible for you to find the data source (and acquire the data), the setup of my autograder disallows the use of any pre-trained models. Therefore, all features need to be computed in the `train()` and `predict()` functions for the autograder to mark your submission complete.
- Please submit a writeup about your solution of the project according to our syllabus guidelines to `project-writeup`. We are not expecting more than 1 page of writeup for this project.

2. Setup

When you decompress the `BIM-155-project-02.zip` file and navigate to the `proj2/` folder, you will see the following files:

- `train.csv` has 20075 columns. The first few lines of the file looks like the following:

```
Id,TNFAIP8L1,...,sex,age,site,Expected
Train0,4.1453747566136,...,Male,74,Right,0
Train1,4.56335466730922,...,Female,84,Right,1
```

  The `Id` column is simply the subject id, which can be safely ignored for training. The Next 20070 columns are the results of genetic sequencing for each gene, with the name of the gene appearing in the header. The number of copies is represented by a real number. The last four columns contains the subject's biological sex (`Male`/`Female`), age as an integer, the extraction site (`Left`/`Right` side of the body), and the label of the sample (0: normal; 1: cancer; 2: cancer that will recur).

- `test_data.csv` contains 20074 columns. This file is structured like `train.csv` except that it does not have the label of the samples, which means that this file does not have an `Expected` column.
- `sample-submission.csv` is a sample prediction file that you can edit (but does not need to submit). Currently, the file looks like the following:

```
Id,Predicted
Test0,0
```

However, your Python code should properly modify/generate this file to reflect the actual predictions.

- `icc.py`. This file computes the intraclass coefficient used to measure the performance of your algorithm. Please do not modify this file.
- `proj2.py`. An almost-blank Python file implementing the skeleton of required submission files.

Among the test set, approximately 30% will be used for the public set in the `project-debug` and the rest of them will be used in `project-final`.

A valid submission to `project2-debug` or `project2-final` will consist of:

- `proj2.py`. Your display grade of the assignment is from the coding style checker using `pycodestyle`. Therefore, don't forget to pass your code to `autopep8`.
- Additional Python source files, if necessary.

For the `proj2.py` to pass the autograder, your `proj2.py` needs to contain at least these two functions:

```
model = train(df)
```

where a `model` object is returned from calling the `train()` function on a `df` variable, which is a `pandas.DataFrame`. Note that your `model` object can be a list if you decide to use a scaler or multiple models.

```
predict(df, model)
```

should take a `pandas.DataFrame` (likely containing the test set) and the `model` object you returned from `train()`, and write the predictions of the `df` into `predictions.csv`.

3. Hints

- The `train()` and `predict()` functions should be bare-minimum implementations of your routine for the best runtime. You could write other debug functions for you to fine tune your model within the same file, for example, when you perform cross-validation. However, for the best run time, the `train()` and `predict` functions should not contain any cross-validation, printing, or other unnecessary code to maximize the speed of your code.
- We have introduced regularization as a potential way to control overfitting of your machine learning algorithm. Alternative ways include feature selection-based methods (for example, selecting the features with the most importance) and principal component analysis (PCA), as well as other algorithm-specific methods of controlling overfitting.

4. Evaluation

The evaluation metric for this project is the intraclass coefficient. What we are using is ICC(3,1), which is a method for measuring the absolute agreement of two observers on the same complete data set. The reason why I chose ICC instead of F_1 score is that F_1 score simply will mark your observation wrong if you predicted a "2" (cancer to recur) as a "1" (cancer); however, a "2" predicted as a "0" is arguably a worse mistake than a “2” predicted as a “1”. Measuring the performance using the intraclass coefficient has the potential of mitigating the limitation of the F_1-score. The calculations of the ICC is very similar to the calculation of R^2 for regression models. To use the icc.py that I provided, please make sure to put the icc.py and the proj2.py together in your working directory, and call the function similar to the following

```
icc_score = icc(np.concatenate(
    (y_true.reshape(-1, 1), y_pred.reshape(-1, 1)), axis=1),
    icc_type='ICC(3,1)')
```

assuming that your y_true and y_pred are 1D np.ndarrays. You should get a numerical score between 0 and 1 to inform your own performance.