BMI6319 Binary Classification to Predict Reoccurrence of Breast Cancer

Submitted in Partial Fulfillment

For

SBMI Ph.D. student Qualification Examination writing test

Fall 2019

Jingqi Wang

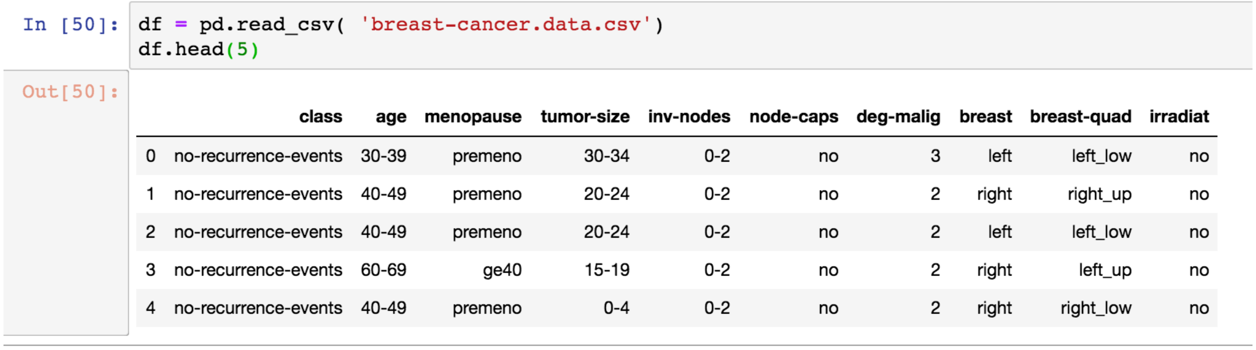
December 19, 2019

Introduction

As a Ph.D. student in SBMI, I was exposed to the predictive modeling task to detect the reoccurrence of breast cancer. After exploring the dataset, I decided to use the binary-classification algorithm for this task. In this project, I will first load the data set and prepare the dataset for the machine learning toolkit, which is 'Liblinear' in this case. To make sure all the works are reproducible, I also create a self-contained docker image to run the whole experiment.

Experiment design

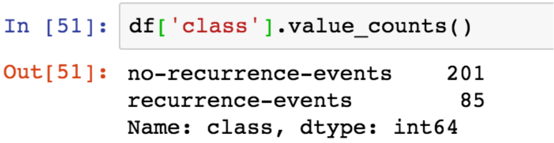
1. **Status of the Dataset:** The dataset is stored as a CSV file. There are 286 record in total, with 201 as negative cases and 85 as positive cases. The distribution quite balanced. As a result, a regular binary classification algorithm is a good start.



```
In [50]: df = pd.read_csv( 'breast-cancer.data.csv')
         df.head(5)
Out[50]:
```

	class	age	menopause	tumor-size	inv-nodes	node-caps	deg-malig	breast	breast-quad	irradiat
0	no-recurrence-events	30-39	premeno	30-34	0-2	no	3	left	left_low	no
1	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	right	right_up	no
2	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	left	left_low	no
3	no-recurrence-events	60-69	ge40	15-19	0-2	no	2	right	left_up	no
4	no-recurrence-events	40-49	premeno	0-4	0-2	no	2	right	right_low	no

Figure 1. Sample of the input dataset



```
In [51]: df['class'].value_counts()

Out[51]: no-recurrence-events    201
         recurrence-events        85
         Name: class, dtype: int64
```

Figure 2. Status of the data labels.

2. **Feature columns:** There are 9 columns which can be utilized as features. Most of them are category data, except for the ‘Degree of malignancy’. As a result, I’ll collect all the possible values of each feature and index them to build the machine learning model. For

example, the ‘no-recurrence-events’ will be encoded as ‘0’; and the ‘premeno’ will be encoded as feature index ‘4:1’.

```
no-recurrence-events,40-49,premeno,20-24,0-
2,no,2,left,left_low,no
```

will be converted into

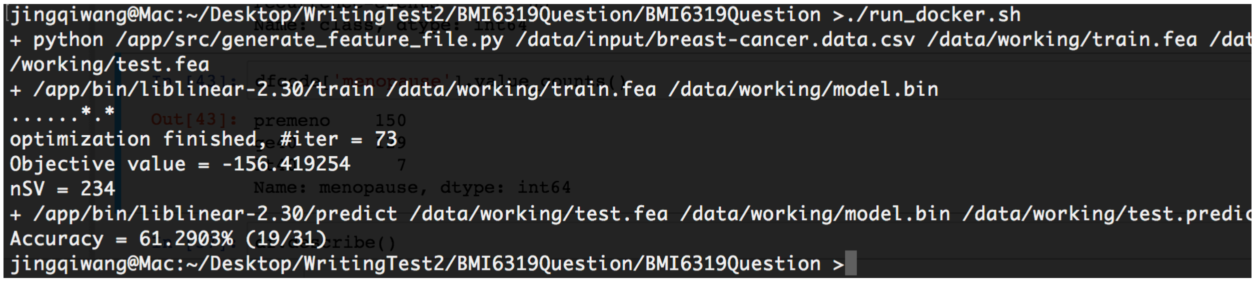
```
0 2:1 4:1 5:1 7:1 8:1 9:1 10:1 11:1 12:1
```

Figure 3. Example of the data convertion

3. **Split data into train and test for ML:** In this experiment, I split the dataset into two parts, 90% of the whole records collected as training dataset and the remaining 10% is used as testset. The model is trained on the training set and evaluated on the testset.

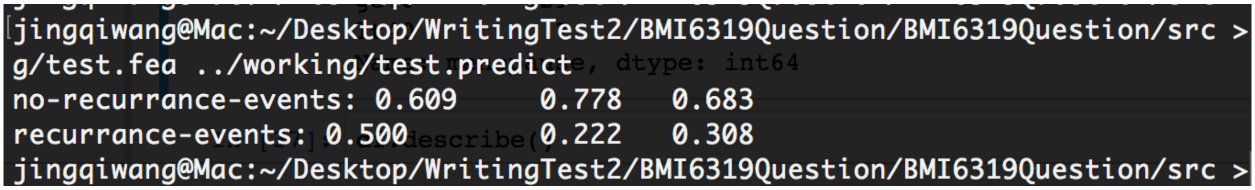
Preliminary Results

1. **Accuracy with default parameters:** The overall accuracy is 61.29%; Table 1 shows the precision, recall and f-sore of the individual label.



```
jingqiwang@Mac:~/Desktop/WritingTest2/BMI6319Question/BMI6319Question >./run_docker.sh
+ python /app/src/generate_feature_file.py /data/input/breast-cancer.data.csv /data/working/train.fea /dat
/working/test.fea
+ /app/bin/liblinear-2.30/train /data/working/train.fea /data/working/model.bin
......*.*
optimization finished, #iter = 73
Objective value = -156.419254
nSV = 234
+ /app/bin/liblinear-2.30/predict /data/working/test.fea /data/working/model.bin /data/working/test.predic
Accuracy = 61.2903% (19/31)
jingqiwang@Mac:~/Desktop/WritingTest2/BMI6319Question/BMI6319Question >
```

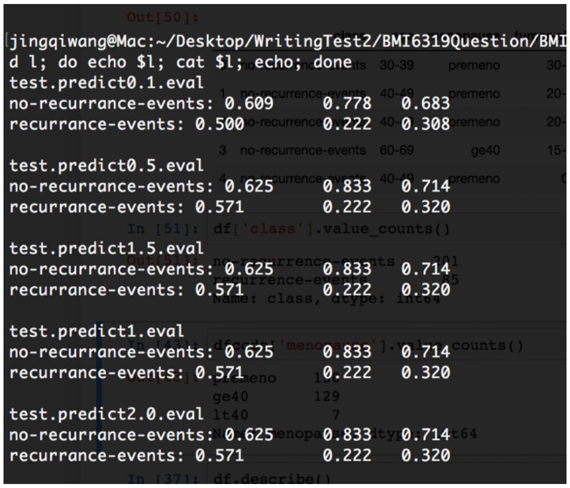
Figure 4. Execution result with default liblinear parameters



```
jingqiwang@Mac:~/Desktop/WritingTest2/BMI6319Question/BMI6319Question/src >
g/test.fea ../working/test.predict
no-recurrance-events: 0.609     0.778   0.683
recurrance-events: 0.500        0.222   0.308
jingqiwang@Mac:~/Desktop/WritingTest2/BMI6319Question/BMI6319Question/src >
```

Figure 5. Evaluation on individual labels

2. **Accuracy with hyper-parameter optimization:** I tried the cost (-c) in the classifier and get some improvents on (c>=0.5). As shown in Figure 6.



```
[jingqiwang@Mac:~/Desktop/WritingTest2/BMI6319Question/BMI
d l; do echo $l; cat $l; echo; done
test.predict0.1.eval
no-recurrance-events: 0.609     0.778   0.683
recurrance-events: 0.500        0.222   0.308

test.predict0.5.eval
no-recurrance-events: 0.625     0.833   0.714
recurrance-events: 0.571        0.222   0.320

test.predict1.5.eval
no-recurrance-events: 0.625     0.833   0.714
recurrance-events: 0.571        0.222   0.320

test.predict1.eval
no-recurrance-events: 0.625     0.833   0.714
recurrance-events: 0.571        0.222   0.320

test.predict2.0.eval
no-recurrance-events: 0.625     0.833   0.714
recurrance-events: 0.571        0.222   0.320
```

Figure 6. hyper-parameter optimization

Package structure and Usage

Fiture 7 shows the folder structure of the source code. ‘Liblinear’ package is located in the ‘bin’ folder and python codes are in the ‘src’ folder. All the tools and experiment settings are connected with the ‘run.sh’ script.

‘Build.sh’ and ‘run_docker.sh’ are the entries for users who want to replicate the experiment.

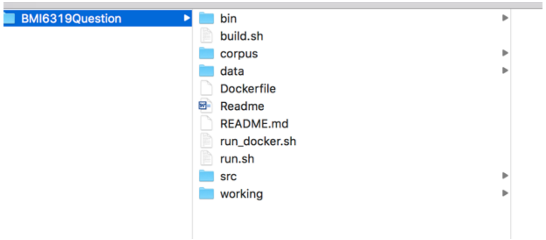


Figure 7. Source code folder structure

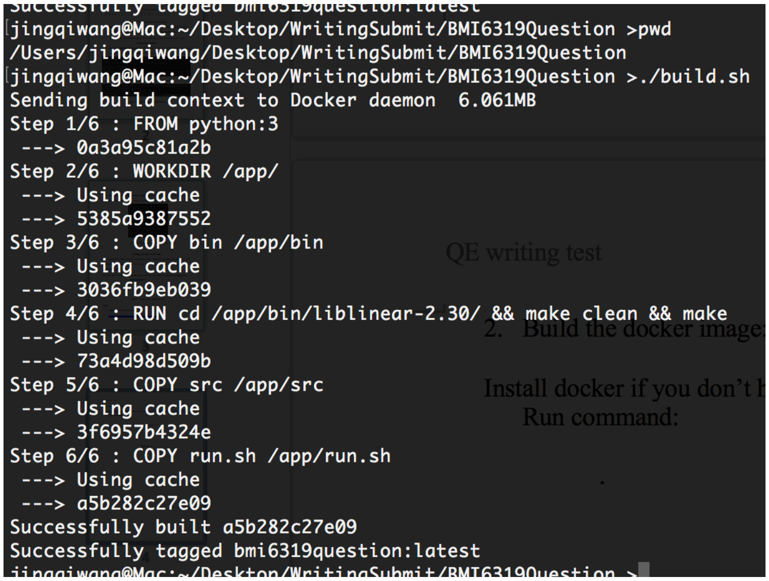
Usage:

1. **Check out the source code:**

   https://github.com/jingqimelax/BMI6319Question.git

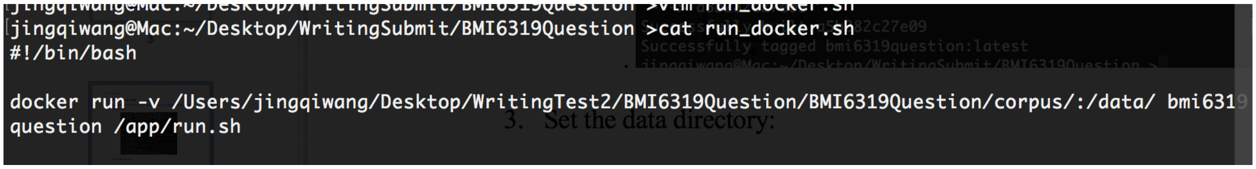
2. **Build the docker image:**

Install docker if you don’t have it yet: https://docs.docker.com/docker-for-mac/install/
Run command:



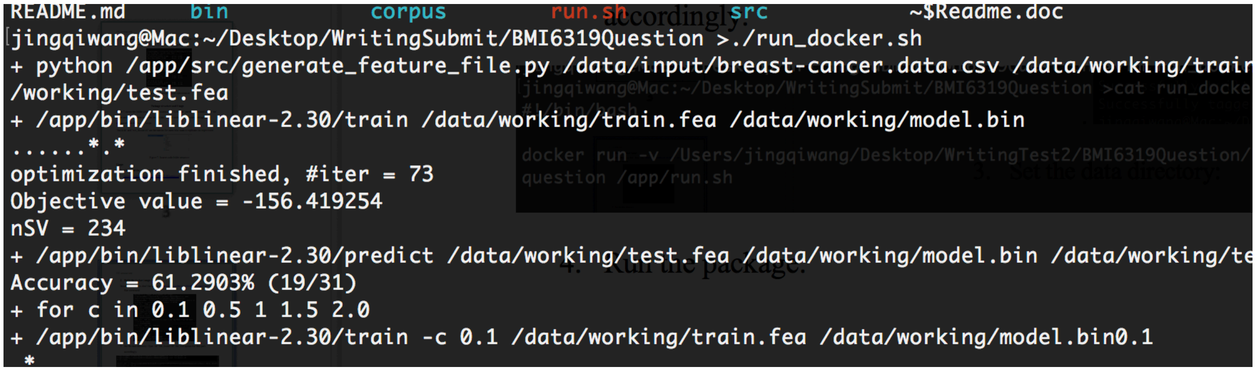
```
Successfully tagged bmi6319question:latest
[jingqiwang@Mac:~/Desktop/WritingSubmit/BMI6319Question >pwd
/Users/jingqiwang/Desktop/WritingSubmit/BMI6319Question
[jingqiwang@Mac:~/Desktop/WritingSubmit/BMI6319Question >./build.sh
Sending build context to Docker daemon  6.061MB
Step 1/6 : FROM python:3
 ---> 0a3a95c81a2b
Step 2/6 : WORKDIR /app/
 ---> Using cache
 ---> 5385a9387552
Step 3/6 : COPY bin /app/bin
 ---> Using cache
 ---> 3036fb9eb039
Step 4/6 : RUN cd /app/bin/liblinear-2.30/ && make clean && make
 ---> Using cache
 ---> 73a4d98d509b
Step 5/6 : COPY src /app/src
 ---> Using cache
 ---> 3f6957b4324e
Step 6/6 : COPY run.sh /app/run.sh
 ---> Using cache
 ---> a5b282c27e09
Successfully built a5b282c27e09
Successfully tagged bmi6319question:latest
jingqiwang@Mac:~/Desktop/WritingSubmit/BMI6319Question >
```

3. Set the data directory: Open the ‘run_docker.sh’ file, and update the data path accordingly.



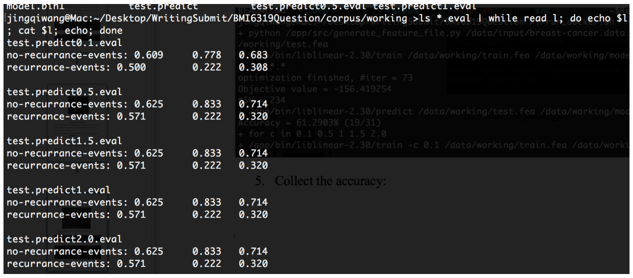
```
jingqiwang@Mac:~/Desktop/WritingSubmit/BMI6319Question >vim run_docker.sh
[jingqiwang@Mac:~/Desktop/WritingSubmit/BMI6319Question >cat run_docker.sh
#!/bin/bash

docker run -v /Users/jingqiwang/Desktop/WritingTest2/BMI6319Question/BMI6319Question/corpus/:/data/ bmi6319
question /app/run.sh
```

4. Run the package:



```
README.md      bin           corpus         run.sh         src            ~$Readme.doc
[jingqiwang@Mac:~/Desktop/WritingSubmit/BMI6319Question >./run_docker.sh
+ python /app/src/generate_feature_file.py /data/input/breast-cancer.data.csv /data/working/trai
/working/test.fea
+ /app/bin/liblinear-2.30/train /data/working/train.fea /data/working/model.bin
......*.*
optimization finished, #iter = 73
Objective value = -156.419254
nSV = 234
+ /app/bin/liblinear-2.30/predict /data/working/test.fea /data/working/model.bin /data/working/te
Accuracy = 61.2903% (19/31)
+ for c in 0.1 0.5 1 1.5 2.0
+ /app/bin/liblinear-2.30/train -c 0.1 /data/working/train.fea /data/working/model.bin0.1
 *
```

5. Collect the accuracy:



```
model.bin1          test.predict        test.predict0.5.eval test.predict1.eval
jingqiwang@Mac:~/Desktop/WritingSubmit/BMI6319Question/corpus/working >ls *.eval | while read l; do echo $l
; cat $l; echo; done
test.predict0.1.eval
no-recurrance-events: 0.609     0.778   0.683
recurrance-events: 0.500        0.222   0.308

test.predict0.5.eval
no-recurrance-events: 0.625     0.833   0.714
recurrance-events: 0.571        0.222   0.320

test.predict1.5.eval
no-recurrance-events: 0.625     0.833   0.714
recurrance-events: 0.571        0.222   0.320

test.predict1.eval
no-recurrance-events: 0.625     0.833   0.714
recurrance-events: 0.571        0.222   0.320

test.predict2.0.eval
no-recurrance-events: 0.625     0.833   0.714
recurrance-events: 0.571        0.222   0.320
```

Conclusion

In this experiment, I built a machine learning pipeline which include data loading, wrangling, and machine learning training and optimization. All the contents are organized as a docker image for users who want to replicate the experiment. Due to the limited development time, detailed analysis and fine tuning of the ML model are not achieved yet, such testing with additional features, n-fold-cross validation. All the source codes are available at github: https://github.com/jingqimelax/BMI6319Question.git