Final report

Qihang Xu xu903@purdue.edu
Tianyi Yang yang1193@purdue.edu
Zhaoze Zhang zhan2507@purdue.edu
Patrick Ubelhor pubelhor@purdue.edu

DOTA2 is a popular 5v5 online game. In this this project, we would like to predict the outcome of a match based on the line ups chosen by each team.

In our dataset, each row represents 1 game.
In the first column whose value ranges over {-1,1}, a value of 1 indicates that radiant (one of the two sides) wins, -1 means the other size wins, (namely, dire). Column 2,3,4 are general info of a match. (Clusters, game mode, etc., which we would not pay too much attention to at this stage of the project because they don't really have any significant effect on the outcome of a match.) Starting from column 5 are -1 or 1s representing heroes chosen by each side (1 -> by radiant, -1 -> by dire, 0 = not chosen by either side in this game). Each game contains strictly 10 heroes, so there would be 5 positive 1s and 5 negative 1s in each row with no exception.
Since the original data set contains 90,000+ samples, we choose 2000 of them to simplify the testing process. More specifically, the first 2000 rows (samples) from original dataset dota2Train.scv would be chosen.

Simple example:

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	…
-1(dire wins)	Irrelevant info	Irrelevant info	Irrelevant info	0(not chosen)	-1(on dire)	0(not chosen)	…
1(radiant wins)	Irrelevant info	Irrelevant info	Irrelevant info	1(on radiant)	1(on radiant)	-1(on dire)	…
-1(dire wins)	Irrelevant info	Irrelevant info	Irrelevant info	-1(on dire)	0(not chosen)	0(not chosen)	
…	…	…	…	…	…	…	…

Where
col 5 maps to the hero "antimage"
col 6 maps to the hero "axe"
col 7 maps to the hero "bane"
…, etc.

URL where the above dataset available:

https://archive.ics.uci.edu/ml/machine-learning-databases/00367/

number id to hero name mapping could be found in this link(in case you are interested)

https://github.com/kronusme/dota2-api/blob/master/data/heroes.json

We've successfully imported the original .csv data set into the program.

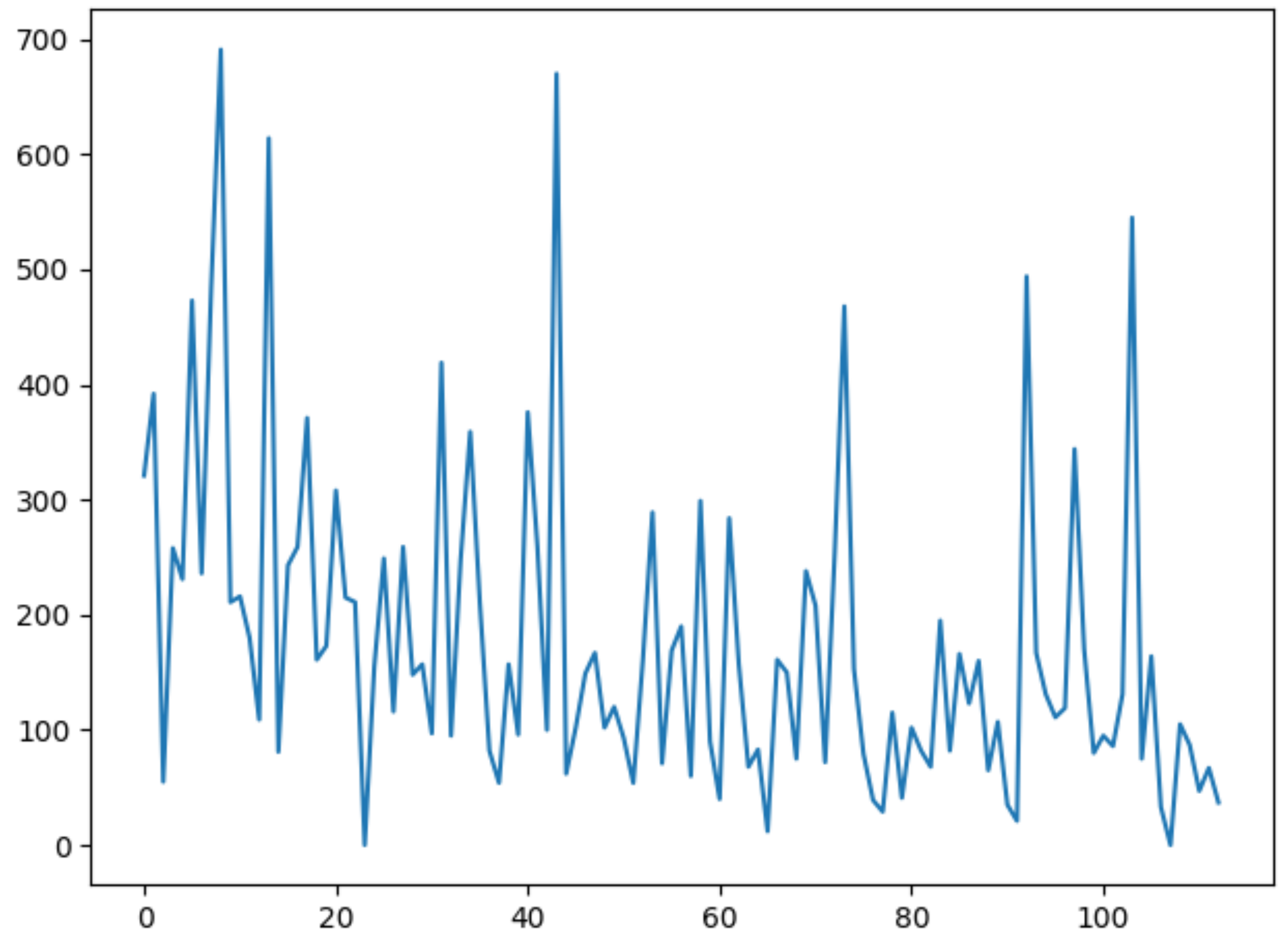
Samples are loaded into x which is of (2d) numpy array type.

Labels, namely, the first column of the original dataset, are loaded into y and is transformed to a 1 d array.

The 2nd to 4th column of the dataset are information about the game modes, game type, cluster (in which geometric zone the game has been played). which we are not interested right now, and has been removed from the original dataset (together with the 1st column which indicates the outcome of a match).

As a result, in the processed dataset, each column id , starting from the 1st, are representative of a hero in DOTA2, and the entire dataset could then be used as a collection of samples we need to train our classifier.

Visualization of the processed dataset:



Since each feature of a sample is only of value -1 or 1, and there are 100 features, it doesn't make a lot of sense to graph a feature vs feature graph, our group decided to plot something similar to a frequency diagram where the value on the x axis indicates the hero id, and the value of the y axis indicates how many times that particular hero has been chosen (out of 2000 games). It can be seen that some heroes frequently appear in a game (700 out of 2000), while some are poorly chosen. (less than 100 out of 2000 games).

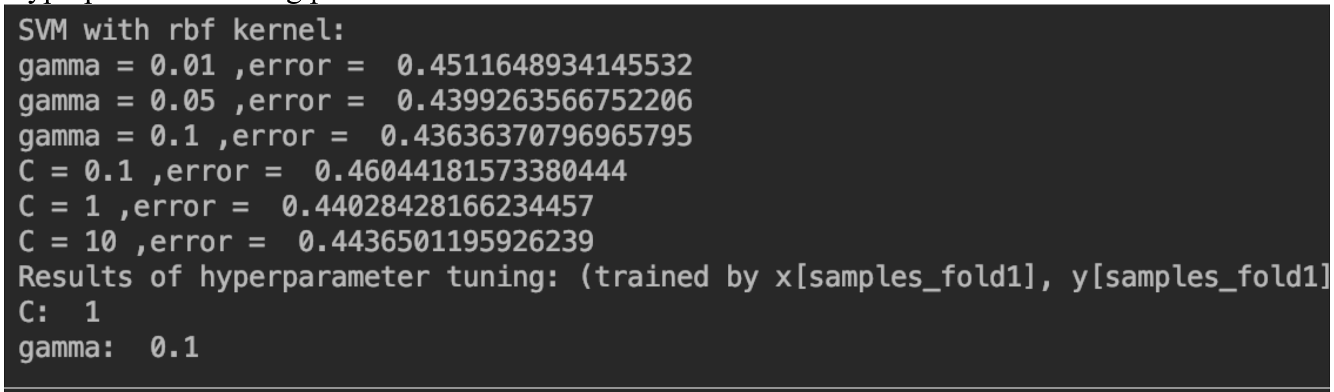
We've implemented a 2-fold cross validation onto the dataset, samples with labels 1 and -1 are evenly divided into each fold.

Bootstrapping have been done within each of the folders to do hyperparameter tuning for C and gamma for SVM with rbf kernel. (C only for SVM with linear kernel.)

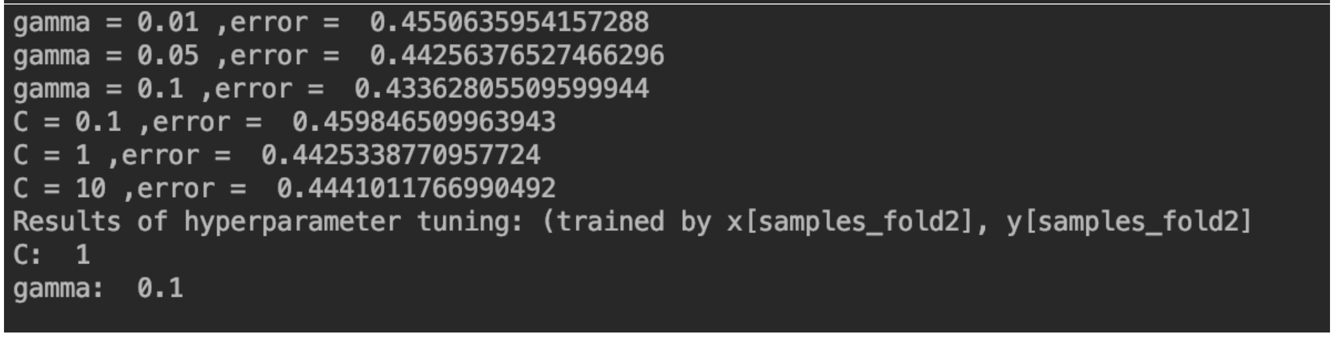
each time we've done hyperparameter tuning within a fold, we then perform cross-validation on the corresponding training and validation sets.

Experimental results:

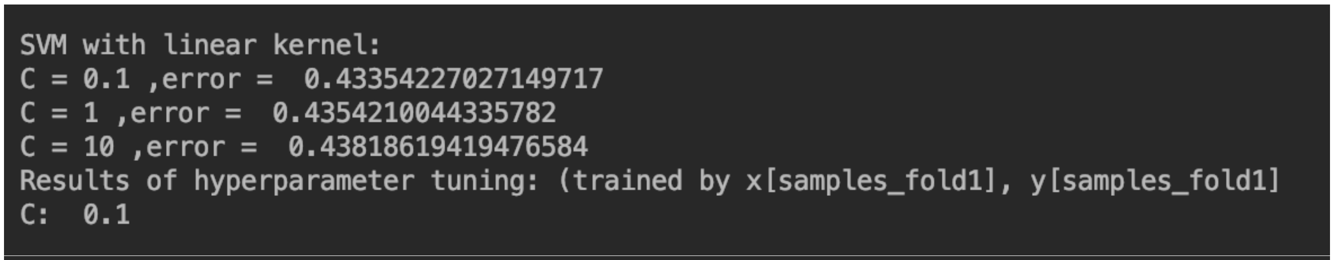
Hyperparameter tuning part:



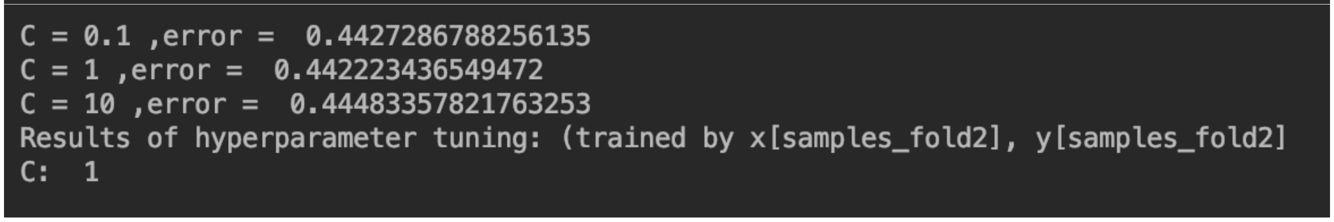
```
SVM with rbf kernel:
gamma = 0.01 ,error =  0.4511648934145532
gamma = 0.05 ,error =  0.4399263566752206
gamma = 0.1 ,error =  0.43636370796965795
C = 0.1 ,error =  0.46044181573380444
C = 1 ,error =  0.44028428166234457
C = 10 ,error =  0.4436501195926239
Results of hyperparameter tuning: (trained by x[samples_fold1], y[samples_fold1]
C:  1
gamma:  0.1
```



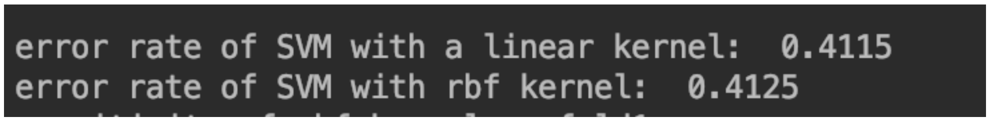
```
gamma = 0.01 ,error =  0.4550635954157288
gamma = 0.05 ,error =  0.44256376527466296
gamma = 0.1 ,error =  0.43362805509599944
C = 0.1 ,error =  0.459846509963943
C = 1 ,error =  0.4425338770957724
C = 10 ,error =  0.4441011766990492
Results of hyperparameter tuning: (trained by x[samples_fold2], y[samples_fold2]
C:  1
gamma:  0.1
```



```
SVM with linear kernel:
C = 0.1 ,error =  0.43354227027149717
C = 1 ,error =  0.4354210044335782
C = 10 ,error =  0.43818619419476584
Results of hyperparameter tuning: (trained by x[samples_fold1], y[samples_fold1]
C:  0.1
```



```
C = 0.1 ,error =  0.4427286788256135
C = 1 ,error =  0.442223436549472
C = 10 ,error =  0.44483357821763253
Results of hyperparameter tuning: (trained by x[samples_fold2], y[samples_fold2]
C:  1
```



```
error rate of SVM with a linear kernel:  0.4115
error rate of SVM with rbf kernel:  0.4125
```

Error rate after performing 2-fold cross-validation using SVM with rbf kernel:
0.4125

Error rate after performing 2-fold cross-validation using SVM with linear kernel:
0.4115

Originally we planned to show

1.Plots of number of samples versus accuracy.
2.Plots of k-fold cross - validation.
3. ROC curve.
Top 10 column # J $\in$ {1, 2,……, d} that has the most positive/negative impact on the outcome of a match.

As the additional experimental results.

However, we later decided just to show tables of different hyper parameter values and their corresponding sensitivity & specificity value mainly because of time issues and, for ROC curve particularly, the different sensitivity and specificity dots by changing hyperparameters cannot be reflected on the graph well.

SVM with rbf kernel fold1: (C = 1 fixed)

	sensitivity	specificity
Gamma = 0.01	0.821	0.3
Gamma = 0.05	0.697	0.454
Gamma = 0.1	**0.701**	**0.467**

SVM with rbf kernel fold2: (with C = 1 fixed)

	sensitivity	specificity
Gamma = 0.01	0.841	0.257
Gamma = 0.05	0.693	0.447
Gamma = 0.1	**0.693**	**0.451**

SVM with linear kernel fold 1:

	sensitivity	specificity
C = 0.1	0.651	0.513
C = 1	**0.627**	**0.546**
C = 10	0.616	0.548

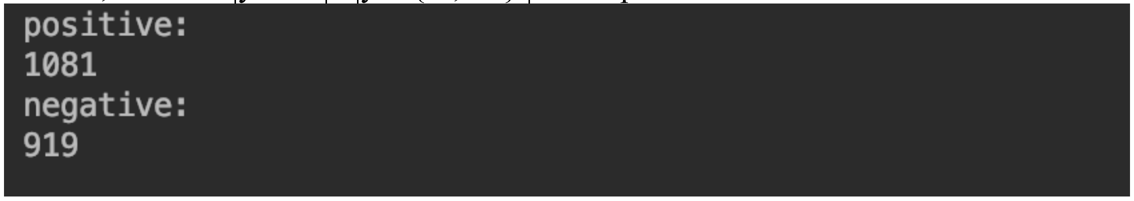
SVM with linear kernel fold 2:

	sensitivity	specificity
C = 0.1	0.685	0.486
C = 1	**0.661**	**0.505**
C = 10	0.654	0.503

Conclusion:

Not surprisingly, the value we get after hyperparameter tuning above turns out to be more or less the optimal choice of parameter value in the table.
Both the table and the error rate suggest that, the linear kernel SVM and the rbf kernel SVM share a similar performance on classifying the outcomes of a match.

Chance, which is |y = - 1| / |y = {-1, +1} | in the processed data set could be calculated as



```
positive:
1081
negative:
919
```

919/2000 = 0.4595, which is obviously larger than the error rate we get. That suggests that our classifiers do help in predicting the outcome of a match, although it’s nowhere close to ideal.

However, it is totally understandable to have an error rate of around 0.4 because drafting (choosing heroes) doesn’t have that much of an effect on the final outcome of a DOTA2 match, what definitely matters more is the player’s level, which is unfortunately not recorded in this dataset.