CSGO Round Result Classification

Allen Cheng, Jayden Wang, Joestar Song, Ted Xu, and Weison Xu

High School Regression Model Project

Vipul Goyal

September 9, 2021

Background:

CSGO is a multiplayer first person shooter with many players around the world. Because of its popularity, there are many professional competitions about it. There are many ways to play in this game, among which the competitive mode used in the game is the most classic. In this mode, players are divided into two teams, terrorists (T) and counter terrorists (CT). The goal of terrorists in the game is to place a bomb and protect it until it explodes, or eliminate all counter terrorists. The goal of the counter terrorists is to eliminate all terrorists or dismantle bombs after they are placed. Once a team has achieved their goal, a small round is over. In competitive mode, the team that wins the first 16 rounds will win the whole game.

In the game, players need money to buy weapons and equipment. In the first round, players only have enough money to buy pistols. At the end of each round, players will be rewarded with a certain amount of money. When you win the round or kill the enemy, you will be rewarded with more money. Then you can buy more advanced weapons with more expensive money, so you have a greater chance of winning the next round. Therefore, the first round is very important. Winning the first round allows you to buy better weapons, while the opponent can only use pistols, so your team has a great chance to win several rounds in a row.

From the game rules, there are many factors that affect the outcome of the game. HP can determine the probability of a player killing another. Obviously, it only takes one shot for a person with full HP to kill a person with low HP, but it takes several shots to kill a person with full blood.

Of course, the different damage of different weapons determines the different survival probability of players when fighting. On the other hand, helmets and vests can reduce injuries, which is also a factor to be considered. Furthermore, the number of people alive, whether the bomb is placed and the current score are the overall situation we can consider.

In terms of weapons, AKM and AWP are the two weapons we mainly consider. These two weapons are the favorite weapons used by players. They have one thing in common, that is, their

damage is very high. The game data shows that AKM can kill the enemy with one shot when it hits the head with helmet, which is the only automatic rifle that can do this. Similarly, AWP is a long-range weapon that one shot on body can kill enemy who have vest.

Finally, whether the bomb is placed has a great impact on the outcome of the game. If the bomb has been placed, T is more likely to win. If the one who carrying the bomb is killed, the bomb will fall, and the CT is more likely to win.

Introduction:

After knowing some basic knowledge, like which guns have choose most frequently or how to put the bombs in a better place these stuffs. What we need to do next is to connect the game we play into the program we do, In other words, how can we convert the popular computer games into a computer knowledge based problems. Before we start dealing with a new problem, the most important point is that why we want to deal with this problem “the prediction of the CSGO” and what’s the inspiration of this thinking, we have think many of question. For example, “What types of machine learning models perform best on this data set?”, “How often does the team with most money win?”, or “What attributes should your team have to win? Health, armor or money?” Among these inspirations, we find we need to do.

Logistic Regression with Stochastic Gradient Descent:

Gradient Descent is one of the most used algorithms to find the local minimum of a cost function. By taking the derivative (or finding the gradient) of the function, it is obvious to find the trend of the function, which tells about the slope of the cost function at that specific point. The slope of the function demonstrates the rate at which the function is increasing or decreasing at the given

point. Gradient Descent is to find the gradient of a cost function to estimate the value of a lower point on the cost function compared to the given point.

Gradient Descent:

Suppose the hypothesis is h(x), where x can be a value or a matrix that takes in the training data. Theta matrix is the weight for each feature in the data.

Step 1: $h(x) = \theta^T x$

In the logistic model, h(x) is served as a input to logistic function. Therefore the percentage hypothesis will be z(x):

Step 2: $z(x) = \frac{1}{1+e^{-\theta^T x}}$

The cost function that is used in the logistic model is cross entropy cost function:

Step 3: $J(x) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log z(x^{(i)}) + (1-y^{(i)}) \log(1 - z(x^{(i)}))\right]$

The value m is the rows of data in x matrix.

By applying regular gradient descent, the update of each component of theta matrix will be:

Step 4: $\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (z(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Where alpha is the learning rate, theta j represents one of the component in theta matrix.

Stochastic Gradient Descent:

In the Stochastic Gradient Descent, Step 1, Step 2, Step 3 are the same. But in Step 4, when using Stochastic Gradient Descent, the update of the theta will be:

Changed Step 4: $\theta_j := \theta_j - \alpha\,(z(x^{(n)}) - y^{(n)}) \cdot x_j^{(n)}$

$x^{(n)}$ represents a random row of data in x matrix. $y^{(n)}$ represents the y value that corresponds to the row of $x^{(n)}$ that is chosen. Instead of doing the summation of m rows of data, Stochastic Gradient Descent only needs to do one row of calculation, making the model converge faster.

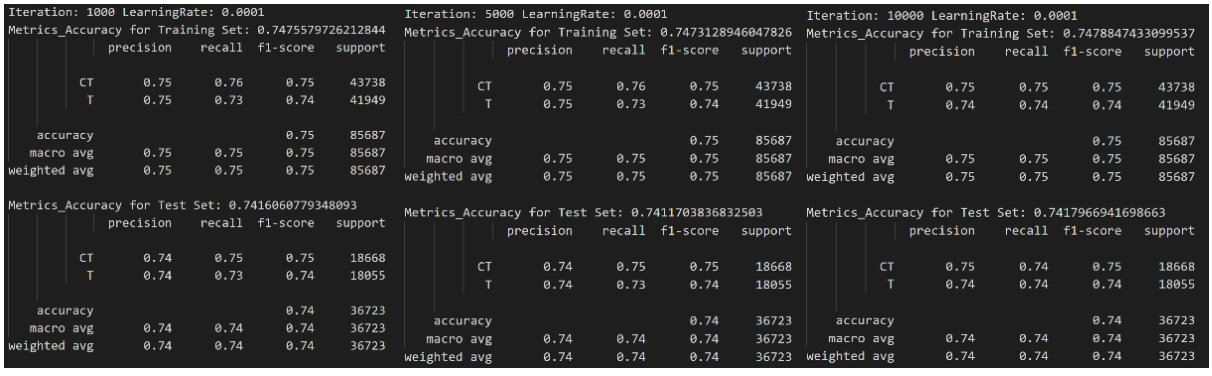
However, Stochastic Gradient Descent is also very sensitive to the learning rate alpha because the randomness of choosing a data point may cause the error to be magnified by the large learning rate.

Running Logistic Regression with Stochastic Gradient Descent:

Code Access: https://github.com/Weisonorz/CSGO-Round-Winner-Prediction

By running Logistic Regression, the accuracy of the model fluctuated around 0.747 to 0.748. When the learning rate is changed the accuracy and iteration increases, the accuracy changes very little.

When learning rate is set to 0.0001, from 1000 iterations to 10000 iterations, the accuracy decreases first then increases.



Iteration: 1000 LearningRate: 0.0001
Metrics_Accuracy for Training Set: 0.7475579726212844

	precision	recall	f1-score	support
CT	0.75	0.76	0.75	43738
T	0.75	0.73	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7416060779348093

	precision	recall	f1-score	support
CT	0.74	0.75	0.75	18668
T	0.74	0.73	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

Iteration: 5000 LearningRate: 0.0001
Metrics_Accuracy for Training Set: 0.7473128946047826

	precision	recall	f1-score	support
CT	0.75	0.76	0.75	43738
T	0.75	0.73	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7411703836832503

	precision	recall	f1-score	support
CT	0.74	0.75	0.75	18668
T	0.74	0.73	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

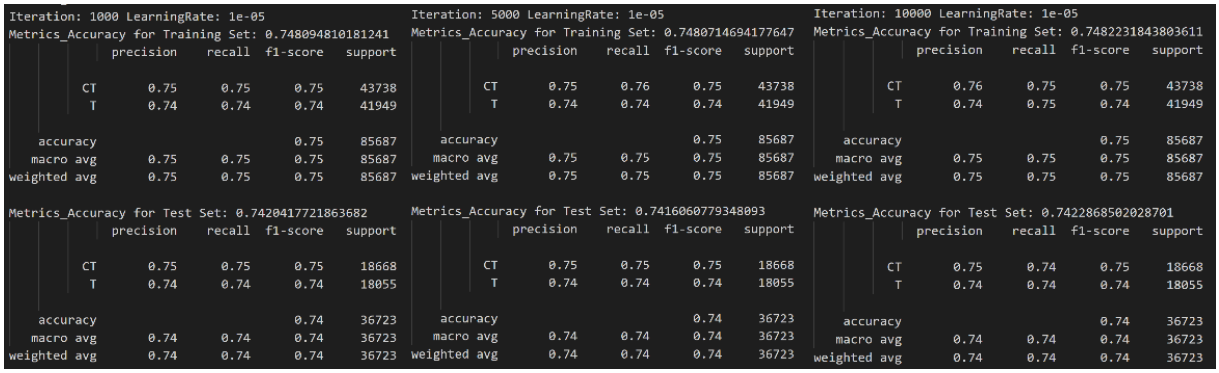
Iteration: 10000 LearningRate: 0.0001
Metrics_Accuracy for Training Set: 0.7478847433099537

	precision	recall	f1-score	support
CT	0.75	0.75	0.75	43738
T	0.74	0.74	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7417966941698663

	precision	recall	f1-score	support
CT	0.75	0.74	0.75	18668
T	0.74	0.74	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

When learning rate is set to 0.00001, from 1000 iterations to 10000 iterations, the accuracy also decreases first then increases.



Iteration: 1000 LearningRate: 1e-05
Metrics_Accuracy for Training Set: 0.748094810181241

	precision	recall	f1-score	support
CT	0.75	0.75	0.75	43738
T	0.74	0.74	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7420417721863682

	precision	recall	f1-score	support
CT	0.75	0.75	0.75	18668
T	0.74	0.74	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

Iteration: 5000 LearningRate: 1e-05
Metrics_Accuracy for Training Set: 0.7480714694177647

	precision	recall	f1-score	support
CT	0.75	0.76	0.75	43738
T	0.74	0.74	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7416060779348093

	precision	recall	f1-score	support
CT	0.75	0.75	0.75	18668
T	0.74	0.74	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

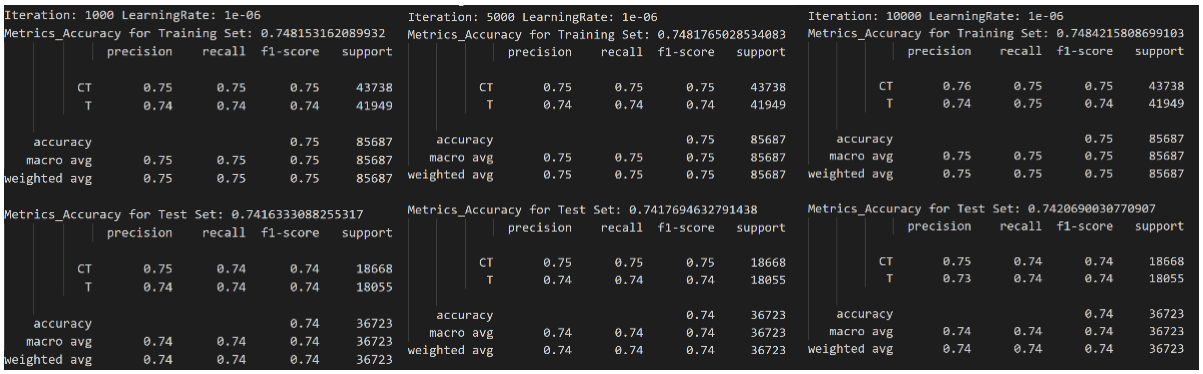
Iteration: 10000 LearningRate: 1e-05
Metrics_Accuracy for Training Set: 0.7482231843803611

	precision	recall	f1-score	support
CT	0.76	0.75	0.75	43738
T	0.74	0.75	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7422868502028701

	precision	recall	f1-score	support
CT	0.75	0.74	0.75	18668
T	0.74	0.74	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

When learning rate is set to 0.000001, from 1000 iterations to 10000 iterations, the accuracy increases gradually.



Iteration: 1000 LearningRate: 1e-06
Metrics_Accuracy for Training Set: 0.748153162089932

	precision	recall	f1-score	support
CT	0.75	0.75	0.75	43738
T	0.74	0.74	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7416333088255317

	precision	recall	f1-score	support
CT	0.75	0.74	0.74	18668
T	0.74	0.74	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

Iteration: 5000 LearningRate: 1e-06
Metrics_Accuracy for Training Set: 0.7481765028534083

	precision	recall	f1-score	support
CT	0.75	0.75	0.75	43738
T	0.74	0.74	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7417694632791438

	precision	recall	f1-score	support
CT	0.75	0.75	0.75	18668
T	0.74	0.74	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

Iteration: 10000 LearningRate: 1e-06
Metrics_Accuracy for Training Set: 0.7484215808699103

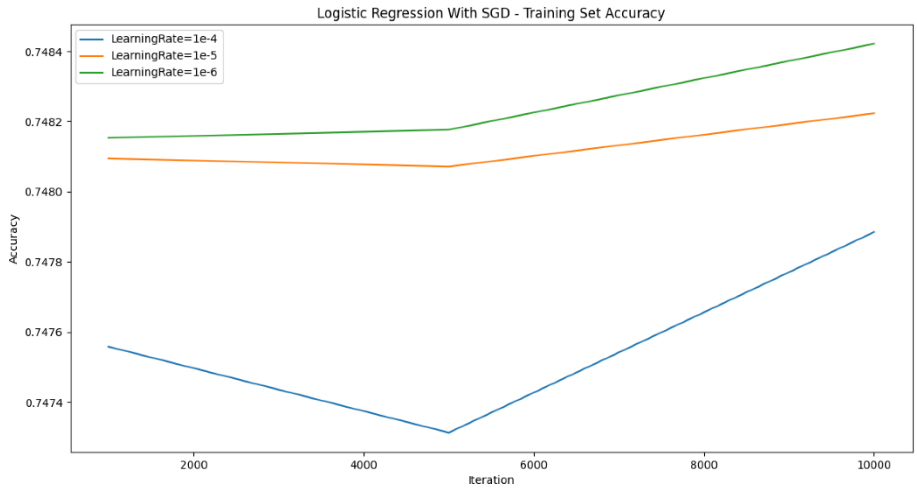
	precision	recall	f1-score	support
CT	0.76	0.75	0.75	43738
T	0.74	0.75	0.74	41949
accuracy			0.75	85687
macro avg	0.75	0.75	0.75	85687
weighted avg	0.75	0.75	0.75	85687

Metrics_Accuracy for Test Set: 0.7420690030770907

	precision	recall	f1-score	support
CT	0.75	0.74	0.74	18668
T	0.73	0.74	0.74	18055
accuracy			0.74	36723
macro avg	0.74	0.74	0.74	36723
weighted avg	0.74	0.74	0.74	36723

Training Set	Learning Rate = 1e-4	Learning Rate = 1e-5	Learning Rate = 1e-6
1000 Iteration	0.747557973	0.74809481	0.748153162
5000 Iteration	0.747312895	0.748071469	0.748176503
10000 Iteration	0.747884743	0.748223184	0.748421581
Tendency	Decrease Then Increase	Decrease Then Increase	Increase

Plotting all the training set accuracy:

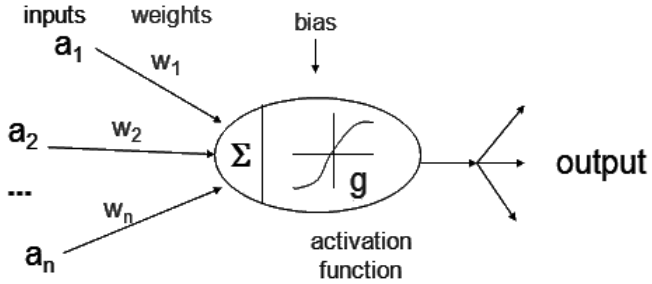


With line plot, it is clearer to see the tendency of the change of the training set accuracy. One mistake in the project is the lack of posing the change of loss function. Because the accuracy is better to reflect the final training result rather than the training process.

Neural Network with Logistic Activation Function

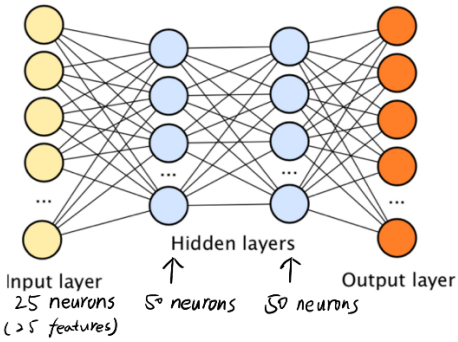
Neural Network is an algorithm to use a large collection of sub-units to calculate a training result, where sub-units are considered to be neurons of the whole Neural Network. Neurons take in several inputs and multiply each inputs by a randomized weights and then use the back propagation algorithm to update the matrix of weight in order to optimize the result.

Each neurons are doing the operation like this:



Each neuron takes in the inputs a_1, a_2, a_3……a_n, then multiply inputs by weights and adding up with a bias term. Finally, generate the output with an activation function. (In this project, the activation model will be logisc function). By accumulating the number of neurons and the layer of neurons, the neural network will be larger and deeper.

In the later section, the neural network size will be represented with a tuple. For example, neural network size=(50,50) is shown in the graph to the right, where the hidden layers contains two layers, each hidden layers has 50 neurons.



Running the Neural Network:

This is the highest accuracy during several changes of parameter. Generally, the Neural Network has higher accuracy compared to Logistic Regression with Stochastic Gradient Descent. Neural Network has an accuracy higher than 0.79.

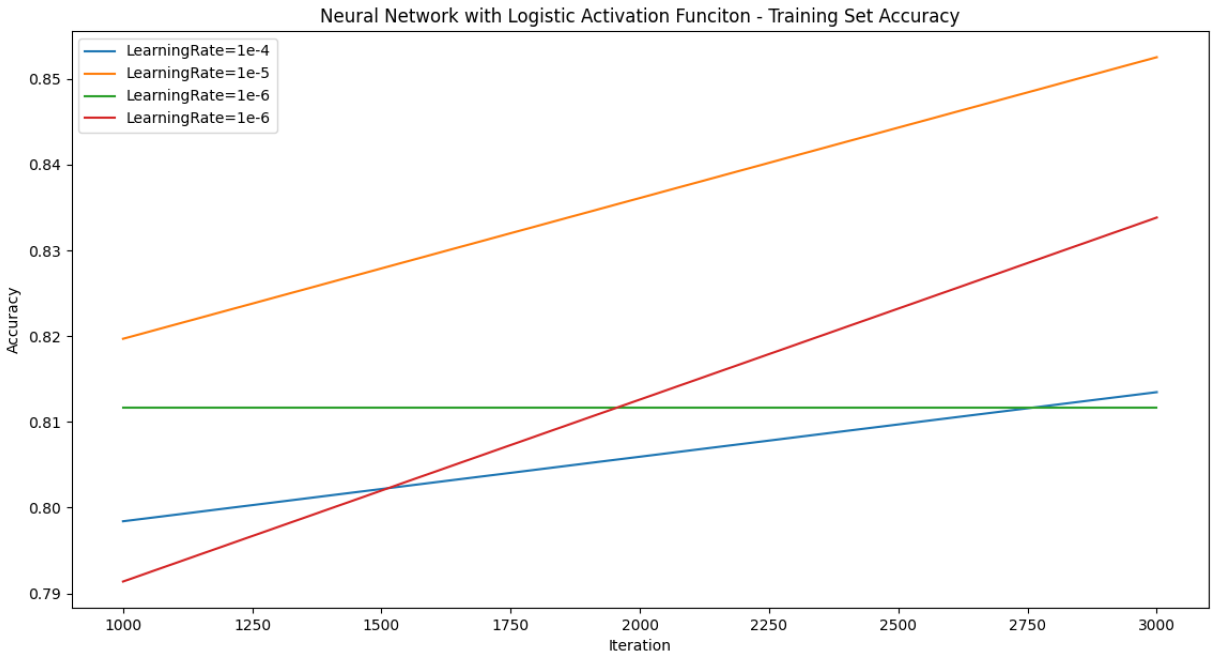
hidden layer size= (50,50,50), iteration=3000,

```
Metrics_Accuracy for Training Set: 0.8525213859745352
              precision    recall  f1-score   support

         0.0       0.87      0.84      0.85     43722
         1.0       0.84      0.87      0.85     41965

    accuracy                           0.85     85687
   macro avg       0.85      0.85      0.85     85687
weighted avg       0.85      0.85      0.85     85687

-------------------------------------------------
Metrics_Accuracy for Test Set: 0.76241047844675
              precision    recall  f1-score   support

         0.0       0.78      0.75      0.76     18684
         1.0       0.75      0.78      0.76     18039

    accuracy                           0.76     36723
   macro avg       0.76      0.76      0.76     36723
weighted avg       0.76      0.76      0.76     36723

The Program Ends
```

By changing the size of hidden layer and the iteration times, the model's accuracy varied:

Training Set	(50,50)	(50,50,50)	(50,50,50,50)	(50,50,50,50,50)
1000 Iteration	0.798417496	0.819692602	0.811605028	0.791391926
3000 Iteration	0.813472775	0.852521386	0.811605028	0.833825434
Tendency	Increase	Increase	Increase/Equal	Increase

Plotting the accuracy of training set:



According to diagram, the accuracy of neural network increases as the number of iterations increases. Learning rate is less sensitive to neural network than logistic regression with Stochastic Gradient Descent.

Methodology:

Code Address: https://github.com/AllenCheng5186/CS-GO-Round-Winner-Classification/blob/main/CSGO%20Round%20Winner%20Classification.ipynb

The program processed by using Python, and the majority of classifier where taken from the sklearn library. The training and testing set are divided 80%:20% randomly using 'train_test_split' function from sklearn, and the 'accuracy' that we will use as our score metric during this study is the percentile of the correct predictions of the classifier in the test set (a perfect classifier would always have accuracy=1).

Firstly, we need to know how many maps they played. There is too much data in the whole dataset, so we only use the whole dataset on map "de_infono" which has 23811 pieces of data. The original dataset has 97 columns that contain too much data of weapons and grenades, but most of them is not relevant to the results, except for AWP and AK47. So, we only remain the data of both weapons. We add a column for the round they`re in, ct_score + t_score and delete one of them, t_score. Secondly, we need make the dataset be visualized properly. Then, use a heatmap to see which data is relevant to our results. Replace "C" and "CT" with bool value.

```
de_inferno     23811
de_dust2       22144
de_nuke        19025
de_mirage      18576
de_overpass    14081
de_train       13491
de_vertigo     11137
de_cache         145
Name: map, dtype: int64
```

	round_winner	time_left	ct_score	round	bomb_planted	
628	0	174.92	0.0	0.0	False	
629	1	114.92	0.0	0.0	False	
630	1	94.92	0.0	0.0	False	
631	1	74.92	0.0	0.0	False	
632	1	174.95	1.0	1.0	False	

We use four approaches, KNN, RandomForestClassifier, SVM, neural network to make models and use "metrics.classification_report" to calculate the accuracy.

The Result of KNN

	precision	recall	f1-score	support
0	0.72	0.90	0.80	3845
1	0.83	0.59	0.69	3299
accuracy			0.75	7144
macro avg	0.78	0.74	0.74	7144
weighted avg	0.77	0.75	0.75	7144

The Result of RandForrestClassifier

	precision	recall	f1-score	support
0	0.89	0.90	0.90	2606
1	0.88	0.87	0.87	2157
accuracy			0.89	4763
macro avg	0.89	0.88	0.88	4763
weighted avg	0.89	0.89	0.89	4763

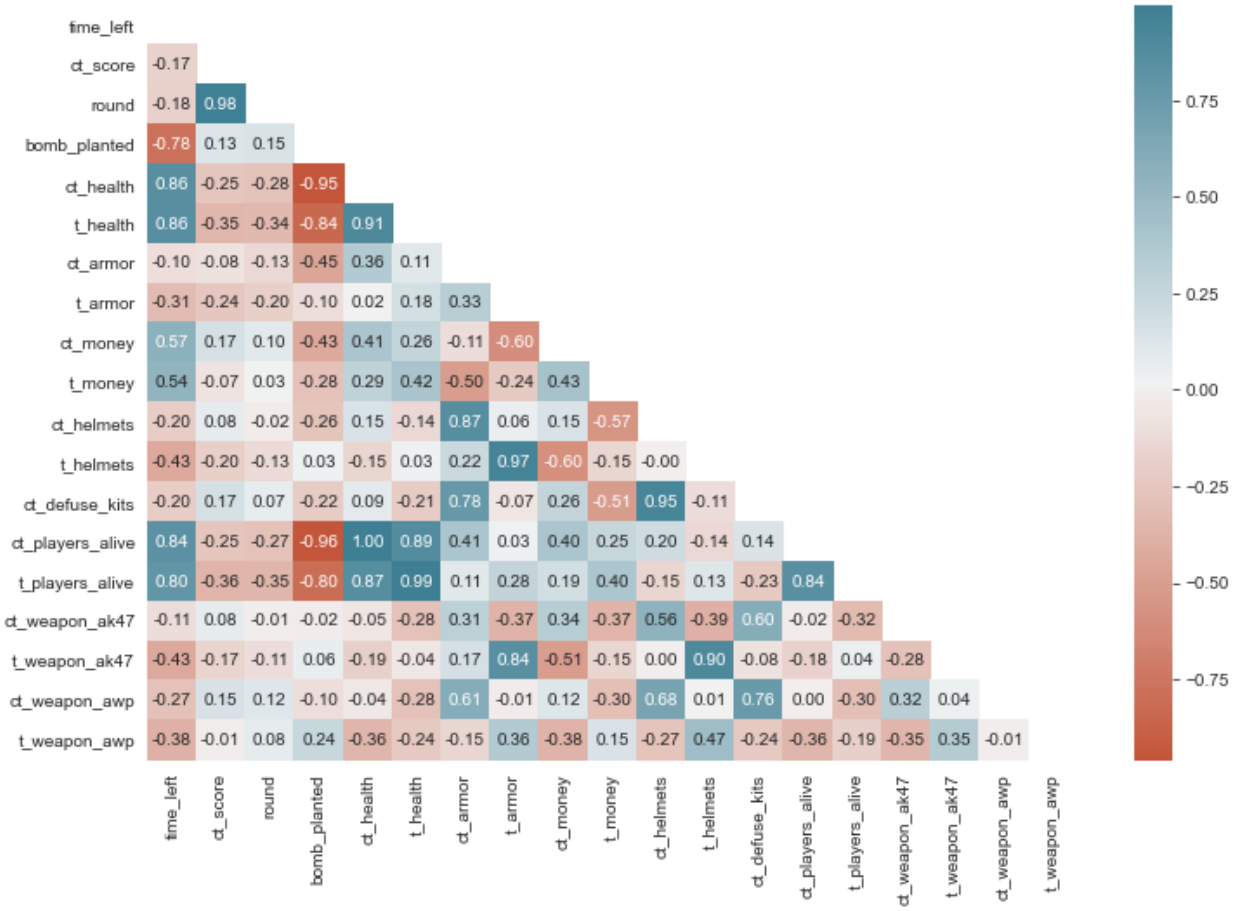
SVM's result

	precision	recall	f1-score	support
0	0.65	0.85	0.74	2573
1	0.72	0.46	0.56	2190
accuracy			0.67	4763
macro avg	0.69	0.65	0.65	4763
weighted avg	0.68	0.67	0.66	4763

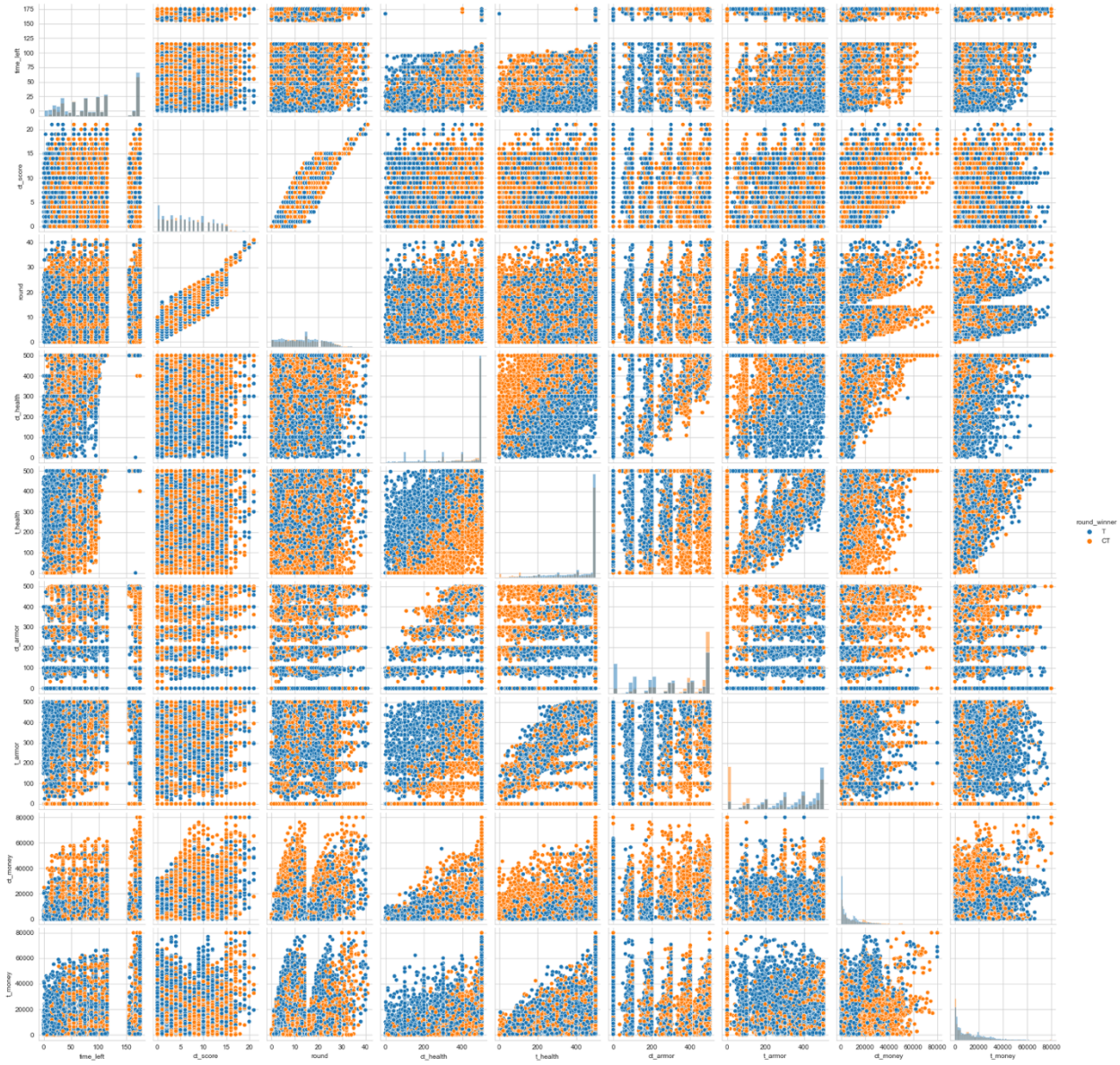
The Result of Neural Network with 2000 Epochs

```
Epoch [1000/2000], Loss: 0.0579
Epoch [2000/2000], Loss: 0.0226
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2012
1	0.99	0.99	0.99	1799
accuracy			0.99	3811
macro avg	0.99	0.99	0.99	3811
weighted avg	0.99	0.99	0.99	3811



Use A Heatmap to See Which Data which is Relevant to Our Results



The Dataset in All 9 Features

Contributions:

Ted Xu—Background Page.

Jayden Wang—Introduction, Data Analysis, Inspiration (proposed discussion problems), Time Controller, and Decision maker.

Weison Huang—Logistic Regression, Data Analysis Code, Data Analysis Graph, Regular Logistic Regression, Neural Network with Logistic Activation Function, and Training Data Comparison

Allen Cheng—Different Approach to Filter Data, SVM, KNN, Random Forest Classifier, CNN for Classification, and Presenting Part of Coding.

Joestar Song—Conclusion, APA Citation, and reorganizing the final report.

Works Cited

GD&SGD. Simple Book. (n.d.). https://www.jianshu.com/p/c497b41f8caa.

Jordan, J. (2018, January 26). *Neural networks: Representation*. Jeremy Jordan. https://www.jeremyjordan.me/intro-to-neural-networks/.

sklearn.linear_model.sgdclassifier. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html.

Tetris AI, Experiments 1 & 2: Single PARENT evolutionary algorithm. Hacking Dan | Tetris AI, Experiments 1 & 2: Single Parent Evolutionary Algorithm. (n.d.). https://blog.hackingdan.com/2017/11/tetris-ai-experiment-1-2-single-parent-evolutionary-algorithm.