


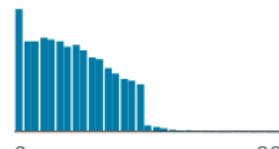
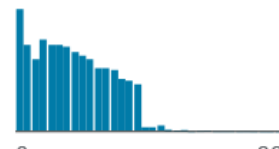
CSGO Round Winner Prediction Classification

Introduction:

CSGO: Counter Striker: Global Offensive is a popular online multiplayer first-person shooter game that players love all over the world of the unpredictability of the game is decided by many factors. The players are divided into two teams: Terrorist(T) and Counter-terrorist(CT). There are five players on each team and the objective is to defuse a bomb or rescue the hostage for the Counter-terrorists and plant a bomb for the terrorists. Each teams will be awarded currencies based on the performance of each round and they will be able to buy guns, flashbangs, or grenades to help with defense and offense in the game. There are different kinds of maps provided to the players such as Inferno, Mirage, Dust II, etc. The game consists of 30 rounds and the players swap sides after 15 rounds and the winners are decided by the best of 30 rounds. The objective of this project is to train the model based on Random Forrest, MLP Classifier, and Logistic Regression and find out the accuracy of the Models individually.

Dataset Description:

The dataset totally consists of 122411 Snapshots which is 122411 data entries and 97 attributes. All the entries are individual and different from each other. 96 of the attributes imply the data entries such as CT score(Counter Terrorists), T Score(terrorist score), time left, maps, etc. All of these columns are the determinants of the result which is the final column of the dataset. The final column has 2 string values. CT or T. CT implies the counter-terrorists and T implies the terrorists. There are no null values in the dataset and 94 of the attributes are floating numbers, 2 attributes are objects and 1 attribute is boolean. The conversion will be explained in the Pre-Processing techniques. An example of the dataset is given below:

# time_left	# ct_score	# t_score	▲ map
Time left in the round.	Current score for Counter-Terrorist team.	Current score for Terrorist team.	The current map.
			de_inferno 19%
0.01 175	0 32	0 33	de_dust2 18%
			Other (76455) 62%
175.0	0.0	0.0	de_dust2
156.03	0.0	0.0	de_dust2
96.03	0.0	0.0	de_dust2
76.03	0.0	0.0	de_dust2
174.97	1.0	0.0	de_dust2
114.97	1.0	0.0	de_dust2
94.97	1.0	0.0	de_dust2

Pre-Processing Technique:

We have to import the following:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

Then we will read the dataset of CSV extension using the pandas library. To check if all the data are precise and correctly imported we can see the desired number of data. I used `dataset.head(5)` which imported the first 5 data. To define the attributes and inputs, I used the shape method which provided the rows, and columns which indicate inputs and attributes.

To see if the dataset has any null value, `dataset.info()` is called and it shows all the information about the dataset. We can see that all the values are non-null, so we do not have to perform any null value elimination method. To make sure of no null values the following process is being used:

```
# checking for null values
dataset.isnull().sum()

time_left      0
ct_score       0
t_score        0
map            0
bomb_planted   0
..
ct_grenade_molotovgrenade  0
t_grenade_molotovgrenade  0
ct_grenade_decoygrenade    0
t_grenade_decoygrenade    0
round_winner              0
Length: 97, dtype: int64
```

But we can analyze that, numbers 3,4, and 96 columns are not floating or integers. We have to categorize the values using any one of the methods Labeling, One hot encoding, etc. I used the Labeling method.

```
from sklearn.preprocessing import LabelEncoder

# encoding non-numerical values to numerical values

categorical_features = ["map", "bomb_planted", "round_winner"]

for i in categorical_features:

    le = LabelEncoder()

    dataset[i] = le.fit_transform(dataset[i].values)
```

Using LabelEncoder(), all the columns are now transferred into integer types. So, all the attributes will return a number type which will provide us with more accurate precision on models.

```
dtypes: float64(94), int64(3)
memory usage: 90.6 MB
```

Now, we have to differentiate the attributes, and if we analyze we can see that the first 96 attributes are the features and the last attribute is the result based on the feature. So, I differentiated the features.

```
# differentiating X (set/columns of features) and y (result)

X = dataset.iloc[:, :-1]

y = dataset.iloc[:, -1] # list[-1]
```

Then, finally, we have to scale the values to predict the pattern and increase the accuracy.

```
from sklearn.preprocessing import MinMaxScaler

# scaling the features

scaler = MinMaxScaler()

scaler.fit(X)

X_final = scaler.transform(X)
```

We have to split the dataset into test and train. I have allocated 40% for the test size.

```
from sklearn.model_selection import train_test_split

# splitting the dataset

# train = 60%

# test = 40%

x_train, x_test, y_train, y_test = train_test_split(X_final, y, test_size=0.4, random_state=42)
```

Models Trained:

i) Random Forest: Random Forest is a well-known machine learning algorithm from the supervised learning approach. It may be applied to both classification and regression issues in machine learning. It is built on the notion of ensemble learning, which is a method that involves integrating several classifiers to solve a complicated issue and enhance the model's performance. Random Forest is a classifier that comprises a number of decision trees on various subsets of the provided dataset and takes the average to enhance the predicted accuracy of that dataset, as the name implies. Instead than depending on a single decision tree, the random forest collects the forecasts from each tree and predicts the final output based on the majority vote of predictions.

	precision	recall	f1-score	support
0	0.86	0.89	0.87	12004
1	0.89	0.86	0.87	12478
accuracy			0.87	24482
macro avg	0.87	0.87	0.87	24482
weighted avg	0.87	0.87	0.87	24482

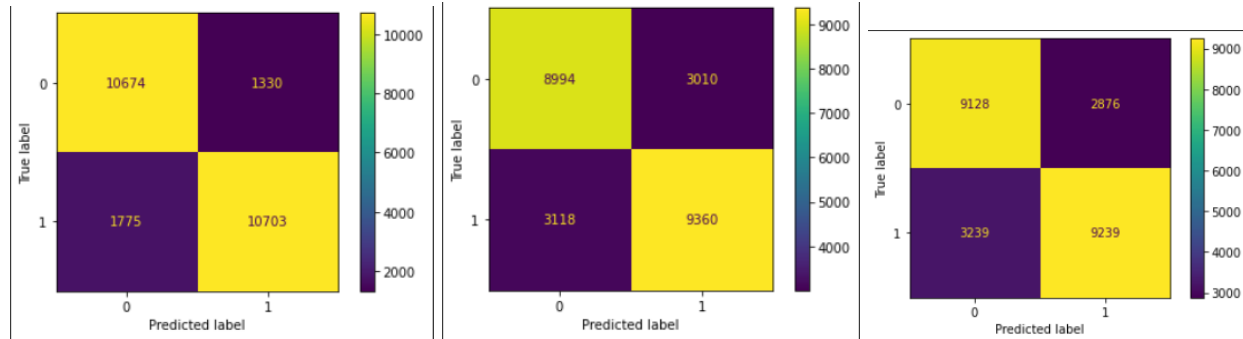
ii) MLP Classifier: Multi-layer Perceptron classifier, or MLPClassifier, is connected to a neural network by the name itself. MLPClassifier uses an underlying Neural Network to carry out the classification task, unlike other classification methods like Support Vectors or Naive Bayes Classifier. The implementation of MLPClassifier requires no more work than the implementation of Support Vectors, Naive Bayes, or any other classifier from Scikit-Learn, which is one similarity with those techniques' other classification algorithms.

	precision	recall	f1-score	support
0	0.74	0.75	0.75	12004
1	0.76	0.75	0.75	12478
accuracy			0.75	24482
macro avg	0.75	0.75	0.75	24482
weighted avg	0.75	0.75	0.75	24482

iii) Logistic Regression: One of the most often used Machine Learning algorithms, within the category of Supervised Learning, is logistic regression. Using a predetermined set of independent factors, it is used to predict the categorical dependent variable. In a categorical dependent variable, the output is predicted via logistic regression. As a result, the result must be a discrete or categorical value. Rather than providing the actual value of 0 and 1, it provides the likely value between 0 and 1. The accuracy of Logistic Regression is given below:

	precision	recall	f1-score	support
0	0.74	0.76	0.75	12004
1	0.76	0.74	0.75	12478
accuracy			0.75	24482
macro avg	0.75	0.75	0.75	24482
weighted avg	0.75	0.75	0.75	24482

Confusion matrix:



Random Forest

MLP Classifier

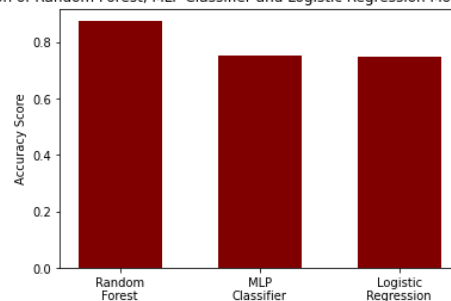
Logistic Regression

The confusion matrix consists of 20% of the data we have split for the test. From the above confusion matrix, we can analyze that for the Random Forest, we get 10674 accurate data for CT win and 10703 accurate data for T win. (Here, 0 is implying CT, and 1 is implying T as we have labeled). On the other hand, we get 1330 wrong data for CT win and 1775 wrong data for T win. For MLP Classifier, 8994 CT data are right and 9360 data of T is right. Finally, for Logical Regression, 9128 CT data is right and 9239 T data is right. If we calculate all the confusion matrices, we would see the accuracy scores are correct. For example, Random forest:

CT accuracy score: $10674 / (10674 + 1775) = 0.857 \sim 0.86$

T accuracy score: $10703 / (10703 + 1330) = 0.889 \sim 0.89$

Comparison of Random Forest, MLP Classifier and Logistic Regression Model accuracy scores



We can see that in terms of accuracy of the given dataset, Random forest is the best model.

Reference:

i) Dataset link: <https://www.kaggle.com/datasets/christianlillelund/csgo-round-winner-classification>