**Supervised Learning**

**CSGO Round winner**

**Problem Statement -**

The objective of this project is to build and compare multiple machine learning algorithms

for the classification of round winners in the game CS:GO.

**Dataset Description -**

CS:GO is a tactical shooter, where two teams (CT and Terrorist) play for a best of 30 rounds,

with each round being 1 minute and 55 seconds. There are 5 players on each team (10 in

total) and the first team to reach 16 rounds wins the game. At the start, one team plays as

CT and the other as Terrorist. After 15 rounds played, the teams swap side. There are 7

different maps a game can be played on. You win a round as a Terrorist by either planting

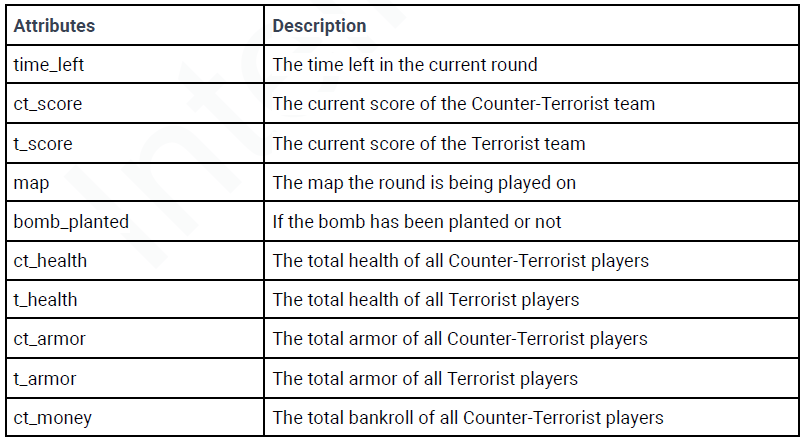
the bomb and making sure it explodes, or by eliminating the other team. You win a round as

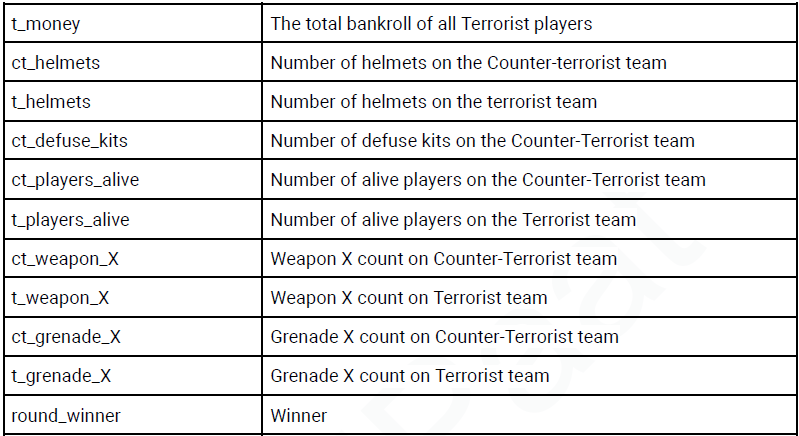
CT by either eliminating the other team, or by disarming the bomb, should it have been

planted.

CSGO round winner dataset contains 122410 records with 97 attributes. Some of the

important attributes description is given below –





**Data preprocessing:**

Loading the data:

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

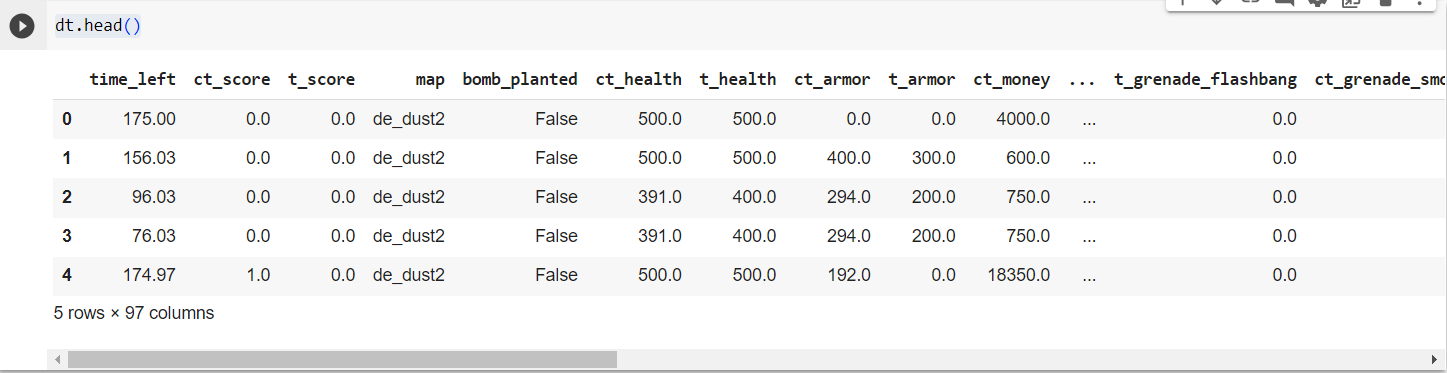
warnings.filterwarnings("ignore")

data=pd.read\_csv(r'/content/csgo\_round\_snapshots (1).csv')

**.head()**

Displaying first five rows.

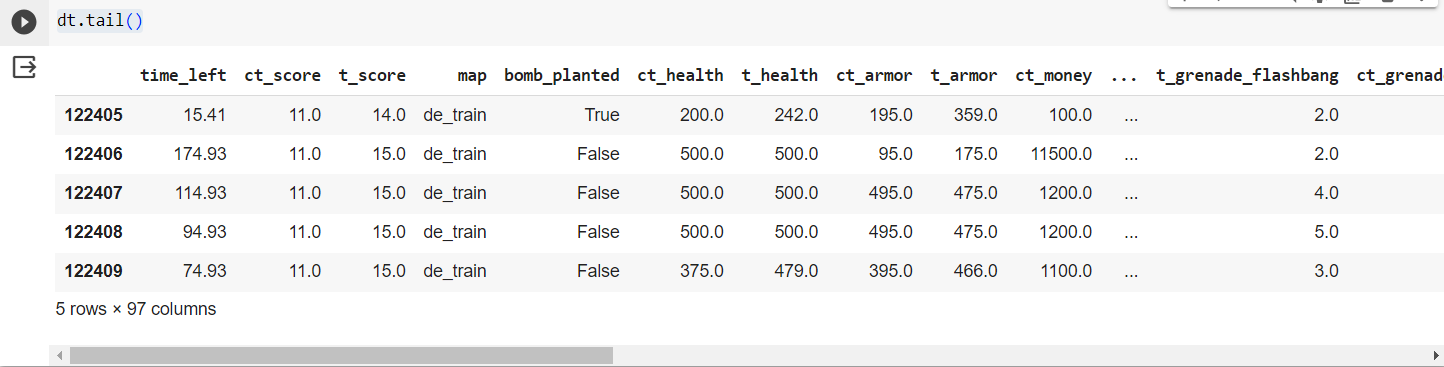
dt.head()



**.tail()**

Used to show rows by descending order

dt.tail()



**.shape**

Shows the row and column of the dataset.

dt.shape

(122410, 97)

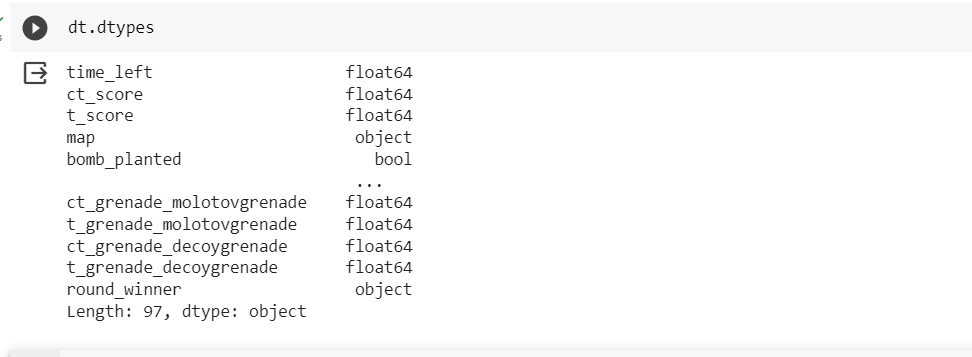
**.columns**

Shows each number of columns in the dataset



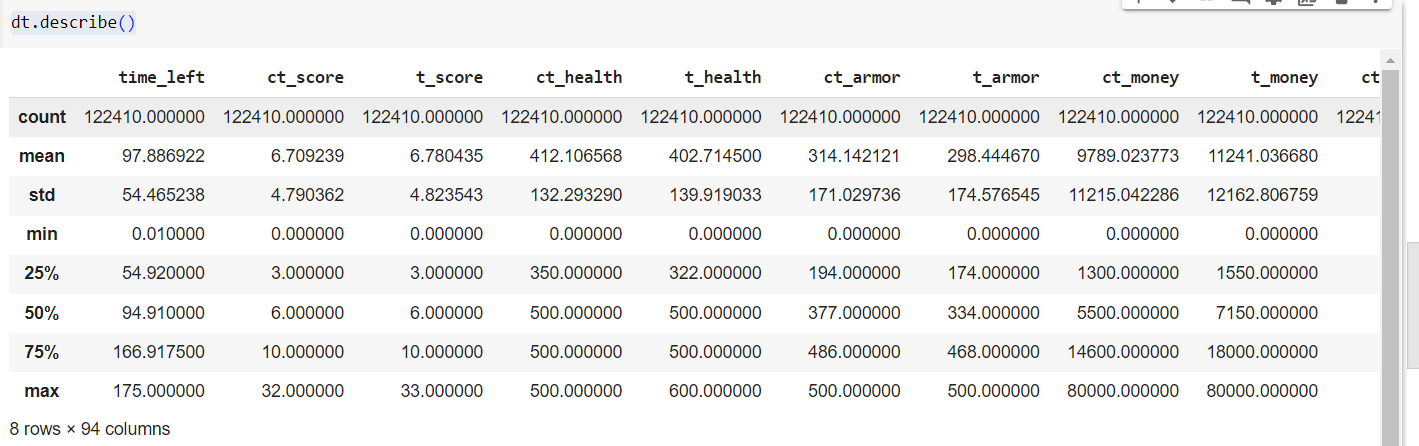
**.dtypes**

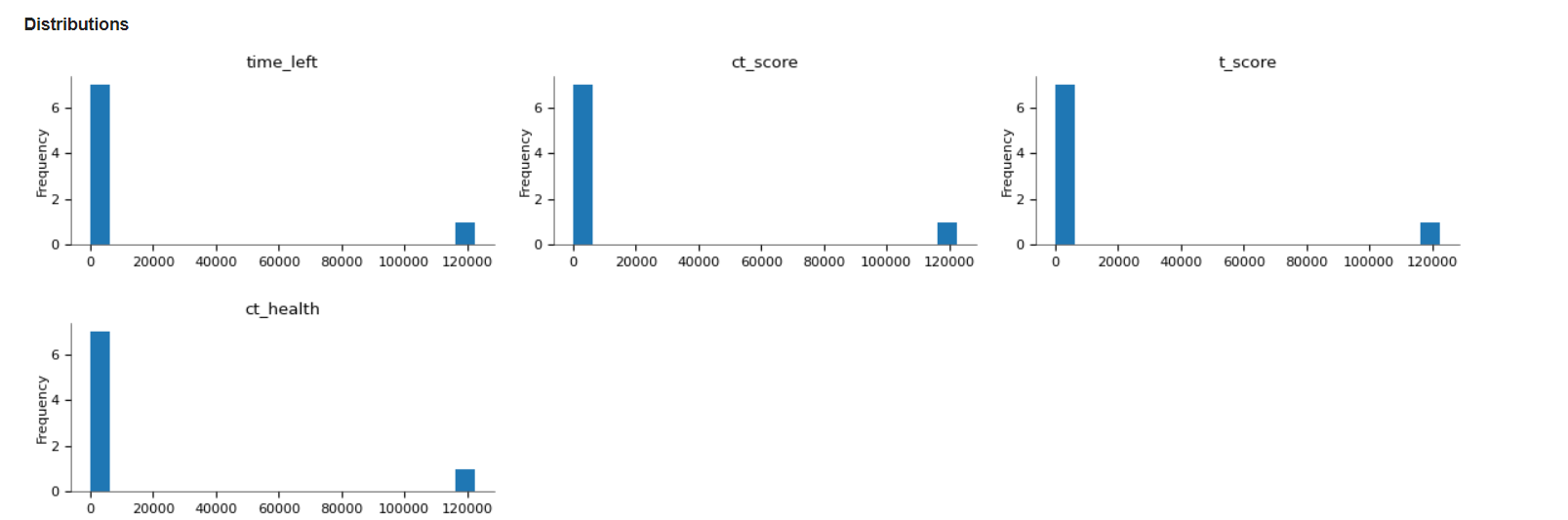
Data type of each column is displayed by this attribute



.describe()

It shows the count, mean, median and etc of the dataset





**Checking missing values and null:**



**Implementing machine learning algorithms:**

Importing the algorithms from Scikit Learn:

from sklearn.linear\_model import LinearRegression,LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import confusion\_matrix,accuracy\_score,r2\_score

Splitting the data and the output column:

x=dt.iloc[:,:-1]

y=dt["round\_winner"]

Train and Test split:

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_state=9999)

**Linear regression:**

* Using the linear regression from the scikit learn linear\_model library

le=LabelEncoder()

for i in dt:

    dt[i]=le.fit\_transform(dt[i])

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=999)

lr=LinearRegression()

lr=lr.fit(x\_train,y\_train)

y\_pred=lr.predict(x\_test)

* Training the model using the linear regression method.
* Using Label Encoder to encoding the categorical data with the fit and transform function
* Implementing linear regression training from scikitlearn to train and predict the label for the data.

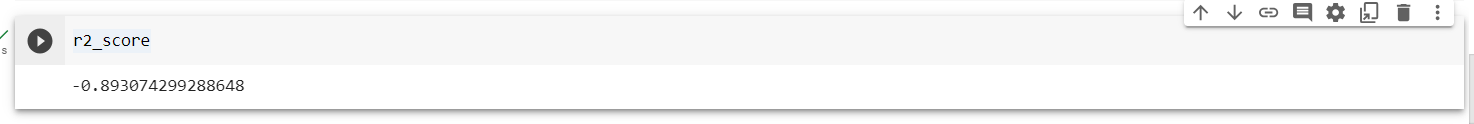
**R2 score:**

r2\_score=r2\_score(y\_pred,y\_test)

* Import r2\_score form the sklearn metrics package.
* R2 score is calculated for regression models to measure the proportion of the variance for a dependent variable that's explained by an independent variable in a regression model.

R2\_score for the linear regression model trained:

r2\_score



**Logistic regression:**

* Using the logistic regression from the scikit learn linear\_model library
* Training the data with logistic regression model

lg=LogisticRegression()

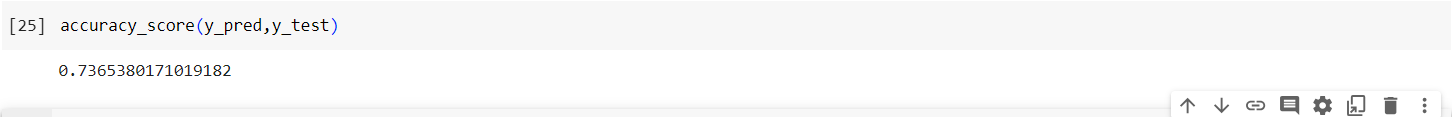
lg=lg.fit(x\_train,y\_train)

y\_pred=lg.predict(x\_test)

**Accuracy Score:**

* Calculating the accuracy score for the model trained.
* Accuracy score is imported from the sckitlearn metrics library.
* Accuracy score shows the proportion of the correct prediction over overall predictions

accuracy\_score(y\_pred,y\_test)

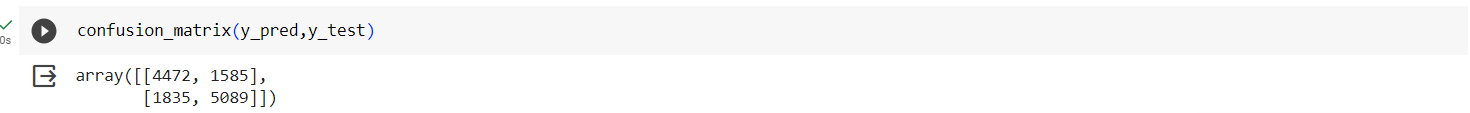


* The accuracy is 73% for the model trained using Logistic regression.

**Confusion matrix:**

* Calculating the Confusion matrix for the model trained.
* Confusion matrix is imported from the sckitlearn metrics library.
* Confusion matrix is used evaluate the performance of the model. The matrix compares the actual target values with predicted values by the machine learning model.

confusion\_matrix(y\_pred,y\_test)

****

* From the confusion matrix about 4472 are predicted correctly by the model trained.

**Decision tree:**

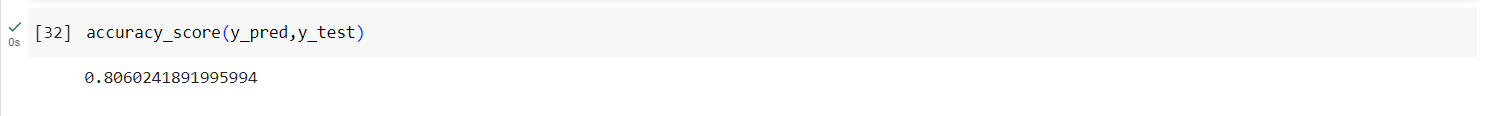
* Using the Decision tree classifier from the scikit learn tree library
* Training the data with Decision tree classifier

dtr=DecisionTreeClassifier()

dtr=dtr.fit(x\_train,y\_train)

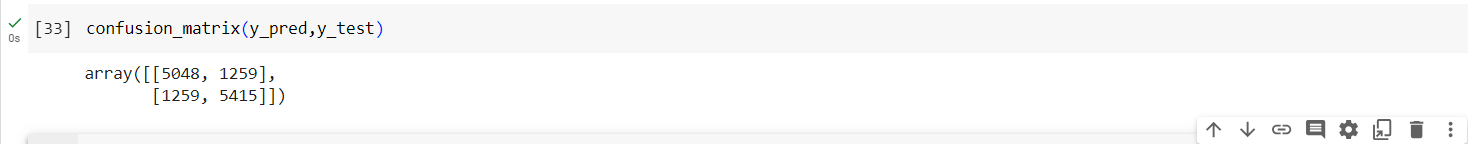
y\_pred=dtr.predict(x\_test)

**Accuracy Score:**



* The accuracy is 80.6%, the accuracy of the decision tree classifier model is higher when compared to the logistic regression model.

**Confusion matrix:**

****

Random forest:

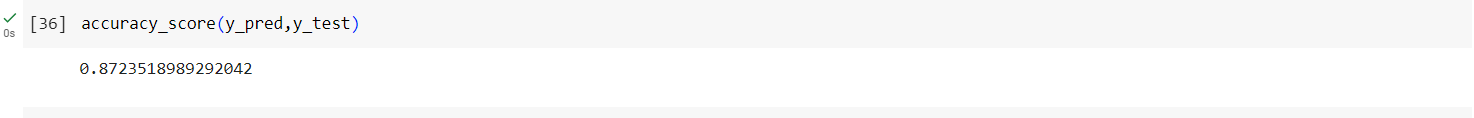
* Using the Decision tree classifier from the scikit learn tree library
* Training the data with Decision tree classifier

Rf=RandomForestClassifier()

Rf=Rf.fit(x\_train,y\_train)

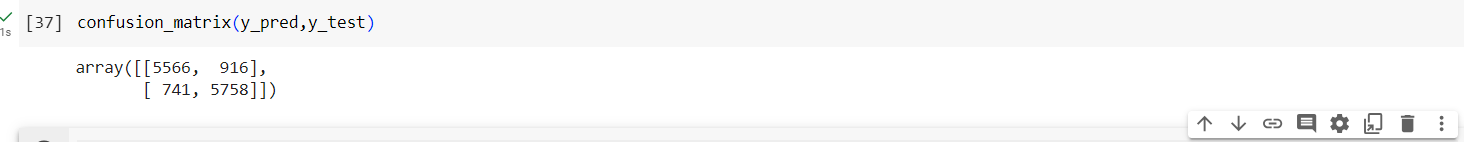
y\_pred=Rf.predict(x\_test)

**Accuracy Score:**



* The accuracy is 87.2%, the accuracy of the random forest classifier model is higher when compared to other models

**Confusion matrix:**



**Conclusion:**

From training the dataset of CGSO game using various models of machine learning it is observed that the Random Forest model works best on the data set prediction nearly **87.2%** of the data correctly.