Background

Counter-Strike: Global Offensive (CS:GO) stands as one of the most iconic and enduring multiplayer first-person shooter (FPS) games in the gaming landscape. The essence of CS:GO lies in its tactical team-based gameplay, where two teams, the Terrorists (T) and the Counter-Terrorists (CT), compete against each other in rounds typically lasting a few minutes. The main mode involves bomb defusal, where terrorists aim to plant and detonate a bomb while counter-terrorists try to prevent it or defuse it. Alternatively, victory can be achieved by eliminating the opposing team.

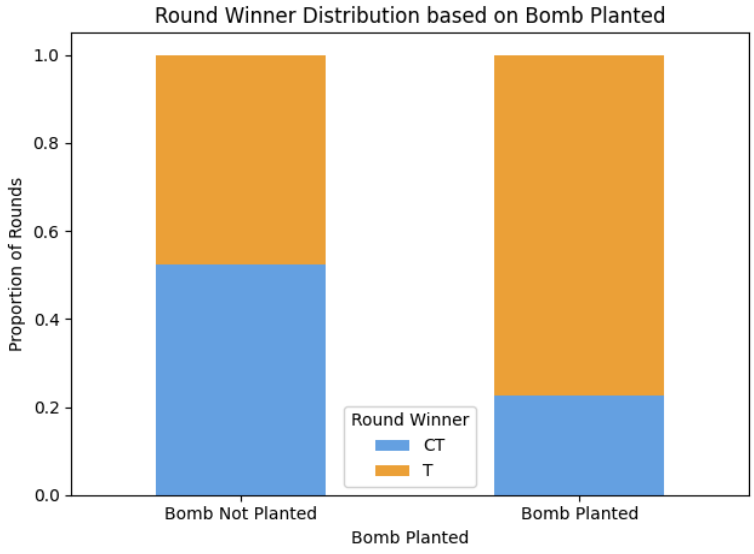
CS:GO has emerged as a major player in the esports industry, boasting significant financial stakes. Prize pools for CS:GO tournaments can reach staggering amounts, with some of the largest events offering millions of dollars in winnings. Professional players and teams compete for a share of these lucrative prizes, making CSGO one of the most financially rewarding titles in esports. We aim to use several machine learning models to find the best way to predict which team will win any given round, depending on any given snapshot of the round.

Data

The dataset was originally published by Skybox as part of their CS:GO AI Challenge, running from Spring to Fall 2020. The data set consists of ~700 games (with 122,411 observations) from high level tournament play in 2019 and 2020. Each datapoint is a "snapshot", which captures all of the data at a random point during any given game. Warmup rounds and restarts have been filtered. A round snapshot is recorded every 20 seconds until the round is decided. We seek to predict the winning team of the round given any one of these snapshots.

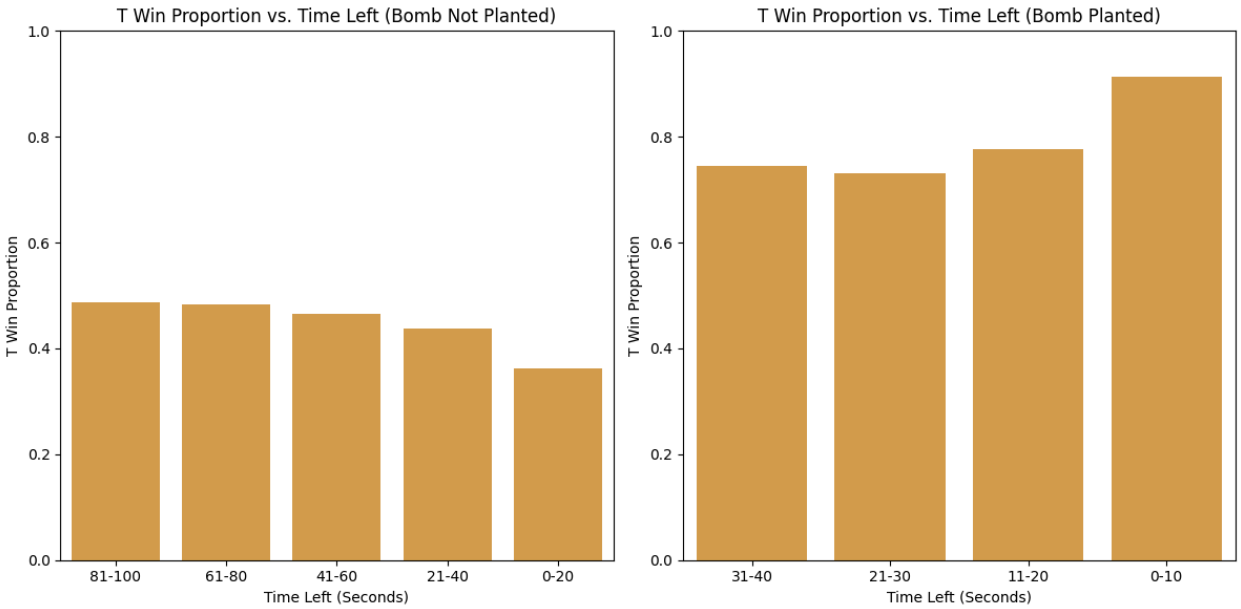
Exploratory Analysis

We will now begin to explore the data, focusing on the proportion of round wins across different predictor variables that are considered extremely important in the modern day competitive scene. The first variable we will examine is bomb_planted, and the correlating win proportion.

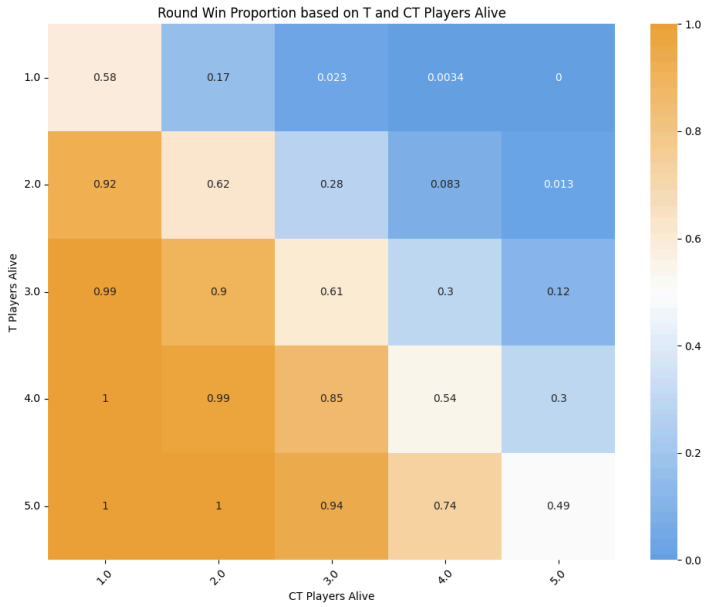


As we can see from the table, it is indeed true that the proportion of round wins for the T goes up a lot depending on if the bomb is planted or not, this follows intuitive beliefs as this is a winning condition for the T team. We will now examine the proportion of T side wins depending on how much

time is left in the round. It is important to remember that the round timer gets reset to 40 seconds when the bomb gets planted, so we will examine the win proportion across these two groups:



In rounds where the bomb is not planted, the win proportion for the terrorist team tends to decrease as the time left decreases. This decline in win proportion can be attributed to the increased pressure on the T side to execute their objectives swiftly, leading to riskier plays and higher likelihoods of being eliminated by the counter-terrorist team. Conversely, when the bomb is planted, the win proportion for T increases as the time decreases. This phenomenon occurs because the T side gains an advantage upon successfully planting the bomb, forcing the CT side to act quickly to defuse it. As time dwindles, the CTs face greater challenges in defusing the bomb, allowing the T side to capitalize on the time pressure and secure more round wins.



As the difference in the number of players between the terrorist and counter-terrorist teams increases, the team with more players tends to have a higher proportion of round wins, reflecting the

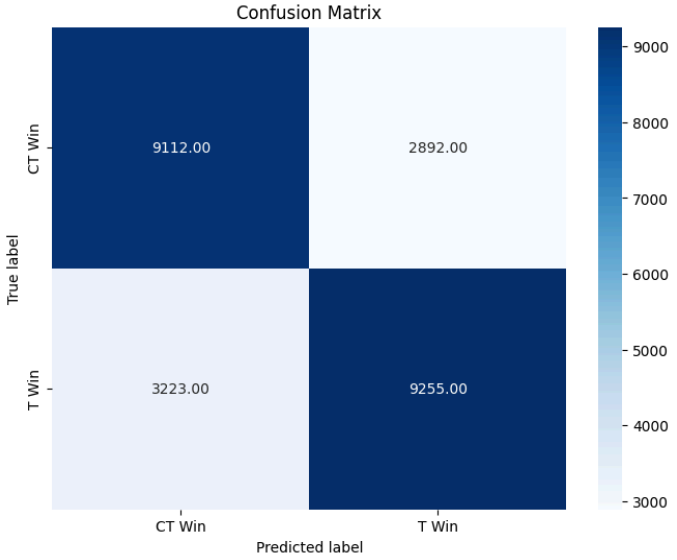
advantage of numerical superiority in engagements. Moreover, in rounds where the number of players on each team is evenly matched, T's exhibit a slight edge in round win proportion, corroborating a widely held belief within the professional scene regarding the strategic potency of T-side play.

Logistic Regression

Logistic regression is a suitable choice for this dataset due to its ability to model binary outcomes, which aligns well with predicting the winner of each round in a CS:GO game (T or CT). Logistic regression is a type of statistical model used for binary classification tasks, where the outcome variable is categorical and has only two possible outcomes. In CSGO, the round winner is either T or CT, making it a binary classification problem. Logistic regression works by fitting a logistic curve (sigmoid function) to the data, which allows it to estimate the probability that an instance belongs to a particular class. It models the relationship between the independent variables (features) and the probability of the binary outcome using the logistic function, which ensures that the predicted probabilities fall within the range of 0 and 1. This makes logistic regression particularly useful when you need probabilistic predictions and interpretability of model coefficients.

In the context of CSGO, logistic regression can utilize various features from the dataset, such as time left, player health, weapons, and economic status, to predict the probability of each team winning a round. It's interpretable, computationally efficient, and performs well with moderately sized datasets like those typically encountered in gaming analytics. Additionally, logistic regression allows for easy interpretation of coefficients, providing insights into the importance of different features in determining the outcome of rounds in the game.

In analyzing professional games data, logistic regression proves particularly valuable due to its probabilistic nature, offering insights into the nuanced dynamics of round outcomes. Professional players often leverage subtle advantages to secure victories, making precise probabilistic predictions crucial. Logistic regression's ability to estimate the probability of each team winning a round allows for a granular understanding of how various in-game factors influence outcomes.



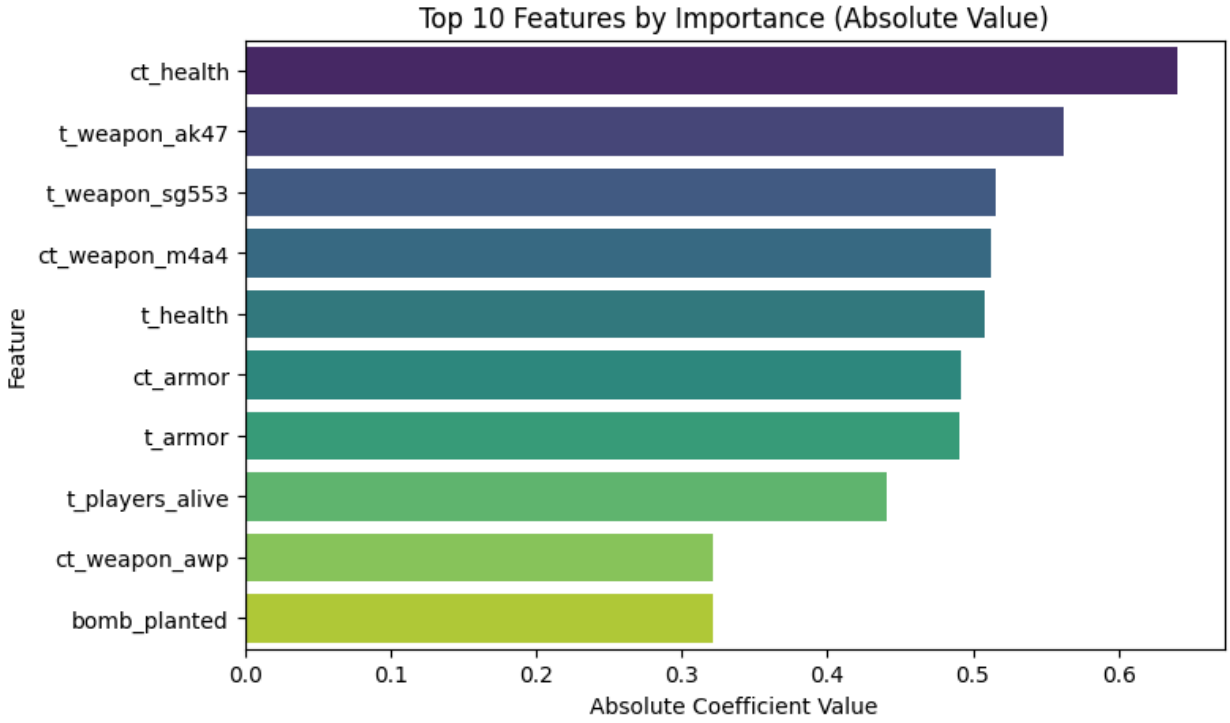
As we can see from the confusion matrix, the logistic regression model performed moderately well on the testing data obtaining an accuracy of about 75%. This is around the accuracy we would expect for a dataset such as this one. Inaccuracy in predicting round outcomes in Counter-Strike: Global

Offensive may stem from anomalies within individual rounds, such as unexpected plays or strategic missteps leading to unpredictable results. These anomalies can introduce uncertainty into the dataset, challenging the predictive power of models like logistic regression by deviating from the expected patterns of gameplay.

Logistic Regression With Feature Selection

Feature selection using logistic regression involves analyzing the coefficients of each feature to identify their impact on the prediction of round outcomes in CS:GO. Features with larger coefficients generally have a greater influence on the model's predictions and can be prioritized, while features with smaller coefficients may be considered for removal to simplify the model and improve its interpretability and performance.

To select the 10 most important features from the logistic regression model, we examined the absolute values of the coefficients and chose the top 10 features with the highest coefficients. Then, we built a new logistic regression model using only these 10 features to create a more focused and interpretable model while potentially improving its predictive performance.

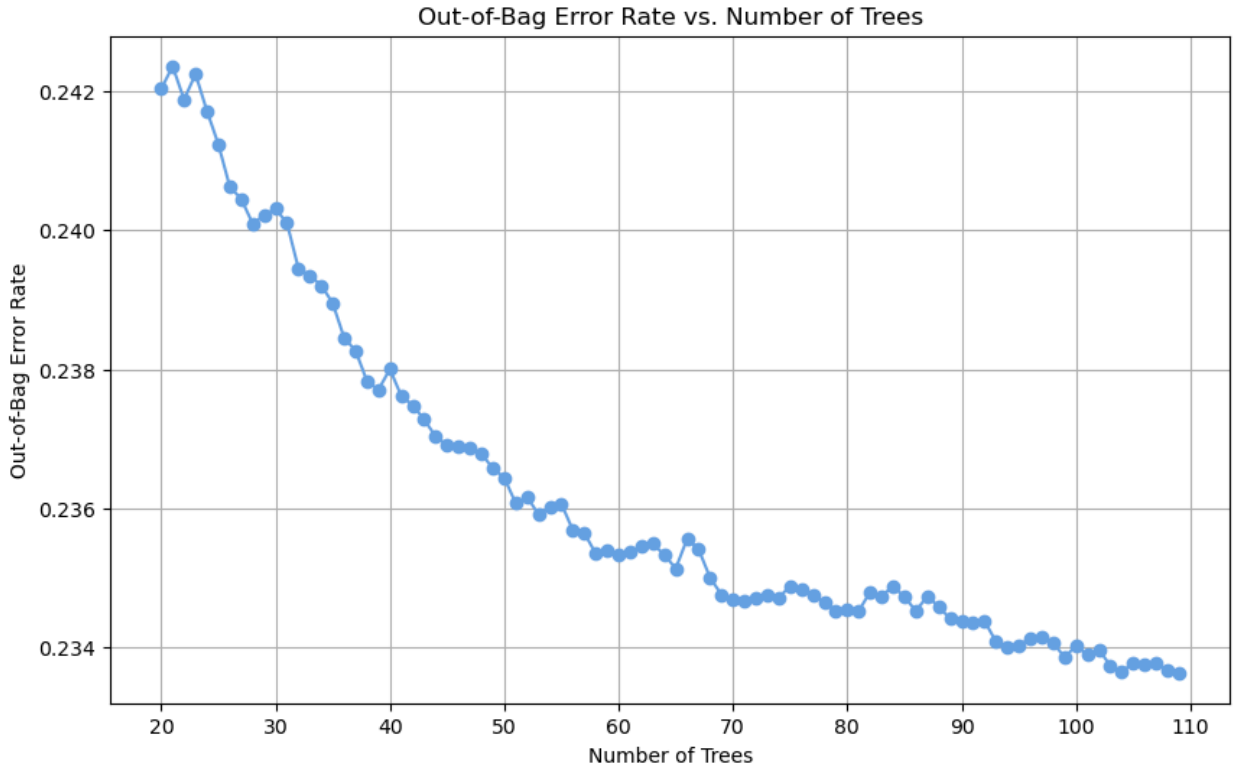


The new model had an accuracy of 74% which was only 1% lower than the original model with a lot less features. Although this did improve the performance of the model, we now know that for future studies we can capture most of the variance using many less features.

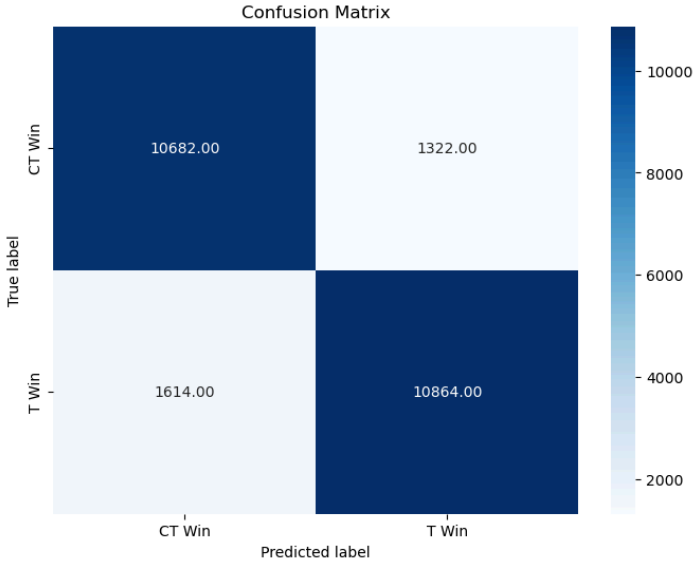
Random Forest

After considering which machine learning algorithm to select next, we opted to employ a Random Forest Classifier for our dataset. Random forests are particularly suitable for feature selection tasks and exhibit excellent scalability. Moreover, leveraging the Out-of-bag (OOB) error rate inherent in random forests allowed us to conserve both time and computational resources. Overall, selecting a

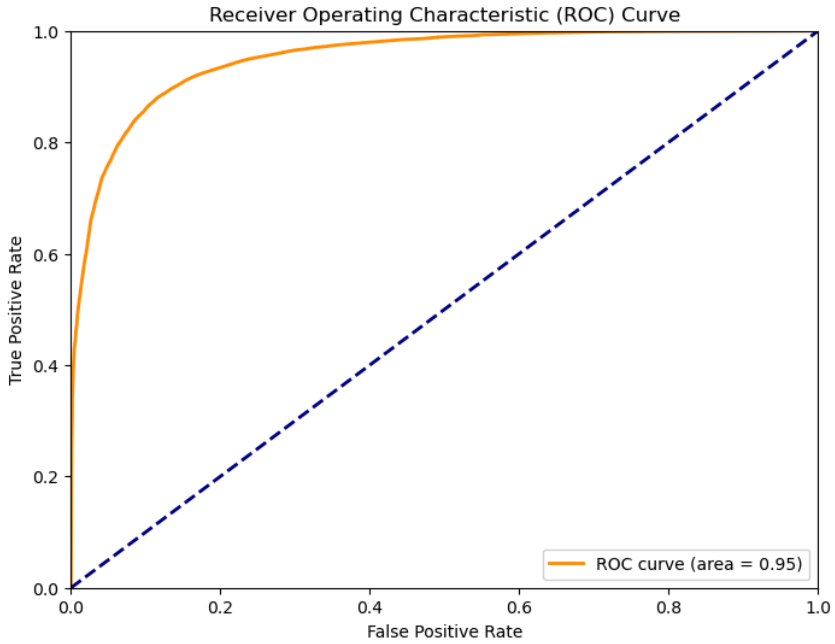
random forest was an ideal option for us as we had an immense amount of data (~120k observations) and a significant amount of features (~100 features).



Our model was trained on 100 trees, utilizing the Gini Index for generating results. We used the OOB error results from the figure above to select our number of trees. Furthermore, we chose the Gini Index as the criterion for splitting nodes in our Random Forest Classifier due to its effectiveness in creating decision trees that partition the data based on the impurity of classes.



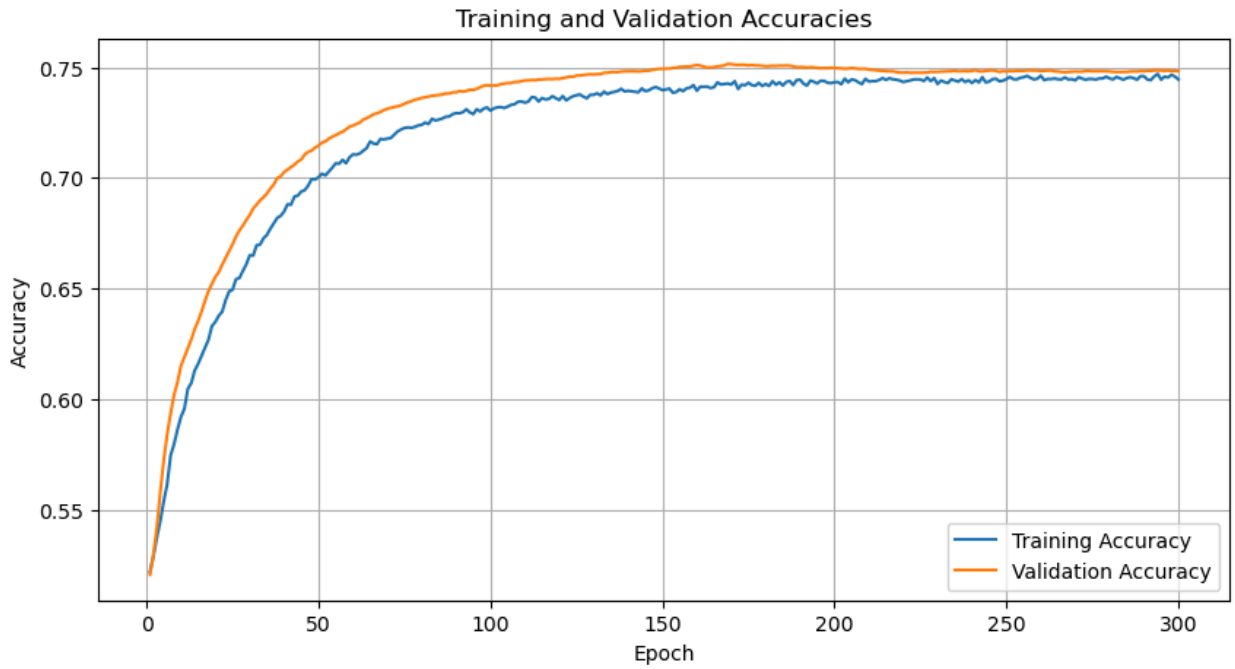
Upon evaluation, we achieved an accuracy of 88% on our test dataset (confusion matrix figure is shown above), accompanied by an impressive AUC-ROC score of 0.95 (as shown in the figure below). Of course, we use the AUC-ROC as our preferred metric as it serves as a robust evaluation metric for binary classification models, offering a comprehensive assessment of model performance across various threshold values. Its ability to remain unaffected by threshold selection and its sensitivity to class imbalance render it the preferred option for assessing the discriminative prowess and generalization abilities in binary classification tasks.

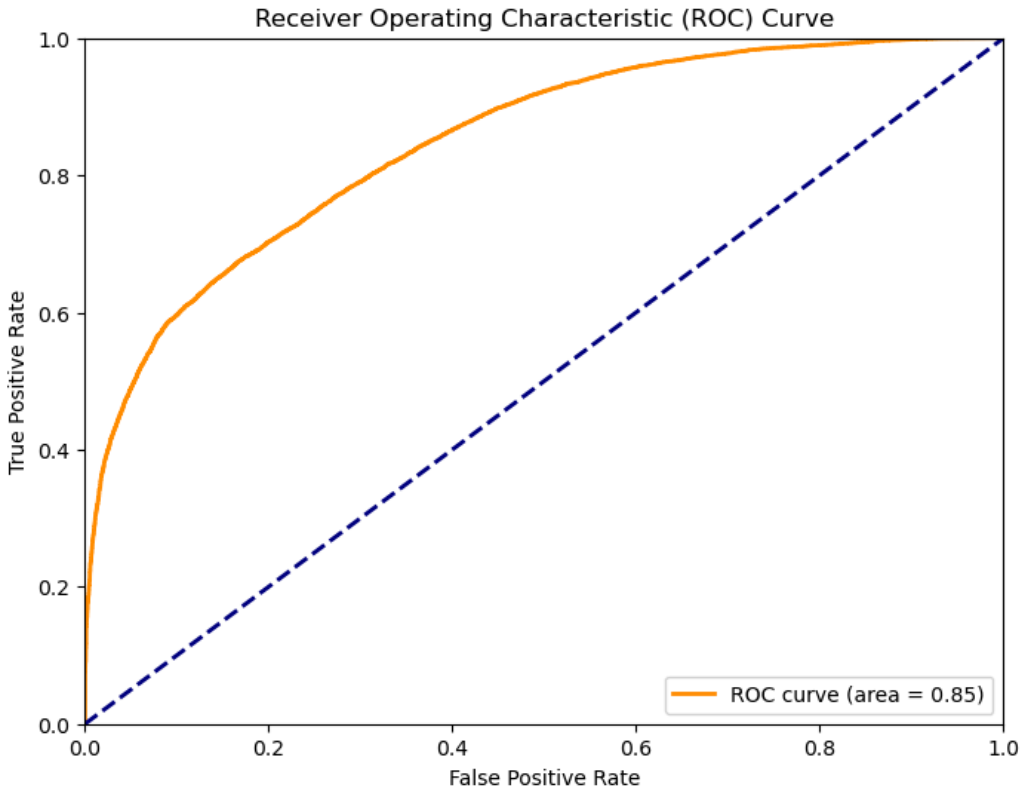


Neural Network

Due to how large of a dataset we had, we were curious to see if fitting a neural network could lead to highly accurate classification results. We decided to employ a multilayer perceptron neural network for this binary classification task. The architecture of our MLP consists of an input layer, two hidden linear layers, and an output layer. Each hidden linear layer (with 32 and 16 weights) is followed by a Tanh activation function and a dropout layer (dropout set to 20%) for regularization to prevent overfitting. The choice of Tanh activation function allows the model to capture complex non-linear relationships within the data, while dropout layers introduce randomness during training, encouraging robustness and generalization. Overall, this architecture aims to strike a balance between complexity and simplicity, leveraging non-linear transformations and regularization techniques to enhance the model's capacity to learn and generalize from the data.

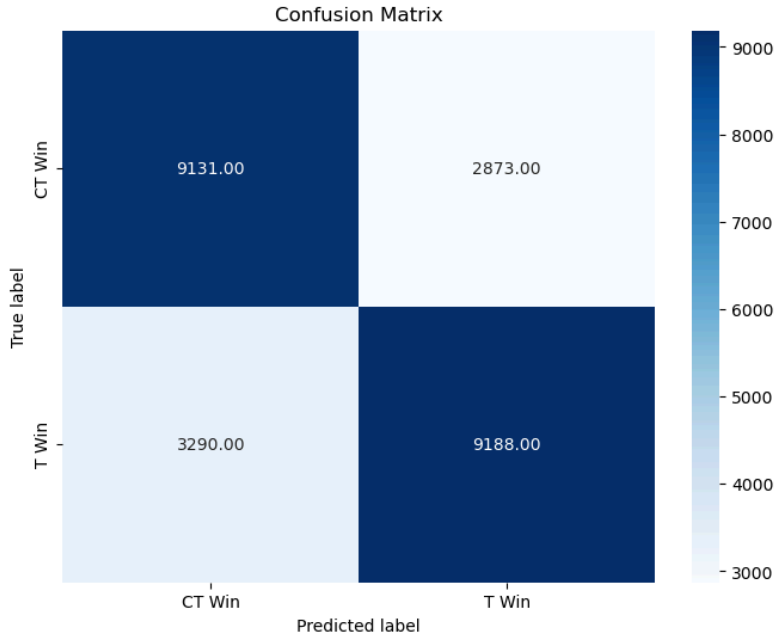
BCEloss was chosen for its compatibility with binary classification tasks, penalizing models based on the difference between predicted probabilities and actual labels. Adam was selected due to its adaptive learning rate mechanism and momentum-based updates, enabling efficient convergence during training.





The observed results from the first figure above indicate that the trained neural network model reached a plateau in performance after 300 epochs of training. The convergence of both training and validation accuracies to 75% suggests that the model might have learned to predict with a certain level of accuracy but struggled to improve further. We also notice that our validation accuracy is always higher than our training accuracy which is unusual for a neural network. However, we believe the underlying

cause of this anomaly stems from the fact that we have multiple dropout layers which allows our model to be more robust when shown new data. The AUC score of 0.85 indicates that the model's ability to discriminate between positive and negative instances is reasonably good.



However, the consistency of our test accuracy being at around 75% as well implies that the model's performance on unseen data aligns with its performance on training and validation sets, likely influenced by the use of the Adam optimizer and Binary Cross-Entropy loss function for stability and convergence. While the model achieved satisfactory accuracy and discrimination, further analysis is warranted to comprehend the reasons behind its plateau at 75%, besides architectural considerations. To potentially enhance the neural network's performance, future studies may benefit from adjusting the architecture, experimenting with hidden layers, neurons, and activation functions, alongside fine-tuning hyperparameters such as learning rate, dropout rate, and batch size for improved convergence and generalization.

Conclusion

In conclusion, our exploration of machine learning models for predicting round outcomes in CS:GO has revealed valuable insights and performance evaluations. Logistic regression initially provided interpretable results, albeit with modest accuracy, while feature selection refined the model's focus without sacrificing predictive power. Transitioning to ensemble methods, our Random Forest Classifier demonstrated superior performance, achieving an accuracy of 88% and a high AUC-ROC score of 95%, showcasing robustness and scalability. Although neural networks introduced complexity, a multilayer perceptron exhibited decent performance, plateauing at 75% accuracy, suggesting potential for further refinement. Overall, our analysis underscores the interplay between model complexity, interpretability, and performance, offering avenues for strategic insights in the evolving esports landscape of CS:GO. Further research and refinement hold promise for enhancing predictive capabilities and understanding gameplay dynamics.