# Capstone II Final Report

---

## Introduction

**Context:**

It is of value for any sports better, coach, or owner to understand what aspects of the game and player statistics contribute most to winning or losing a game. It is no different for professional esports, where player statistics are indicative of how strong a team is in any given match. It is for this primary reason that I became interested in setting out to find which statistics are most useful in predicting whether a match will result in a win or a loss for one of the most proficient esports of all time, Counter-Strike: Global Offensive (CSGO).

Since it's released in August of 2012, CSGO has grown into one of the most played first-person-shooters (FPS) and most watched esport in the world. Even ten years after it's initial launch, the game remains as one of the most competitive video games one can play and still retains close to one million concurrent players at any given time. Like many players around the world, as I grew up playing the game I became increasingly interested in what makes a player or team successful in a given match. My primary goal with this project was to determine what elements of gameplay contribute most to the result of a match and how they differ for individual player and aggregated team data.

**Problem Statement:**

How can we build a model that will output the result of a b01 match (win/loss) using individual player performance as well as aggregated team performance features?

**Success Criteria:**

Choosing the best model based on the highest F1-score while also considering runtime. The aim at the start of the project was to achieve 90% accuracy or better with our best model.

**Stakeholders:**

The primary stakeholders are CSGO team owners, coaches, and players. Secondary ones include match betters.

**Data Sources:**

Data source link: https://www.kaggle.com/datasets/mateusdmachado/csgo-professional-matches

All of the data was scraped from HLTV.org, the leading CSGO site in the world that contains all match data from the many years of competing at the professional level. Specifically, the data contains professional match information from November 2015 to March 2020.

---

## Data Wrangling

**Data Importing and Setup:**

The initial data on Kaggle was made up of 4 tables: *Results.csv, Picks.csv, Economy.csv,* and *Players.csv.* For our project we only needed *Results.csv* and *Players.csv.* These two files contain data on the match results and statistics for each player in each match.

The sole purpose that we needed the results file was to create our target variable of match outcome. To do this, we extracted the winning team from each map and deleted all duplicate match_ids since we only need the final match result rather than the result for each map played in the match. We then dropped all columns except match_id and the winning_team column that we just created. We then merged our new dataframe with the players dataframe that until now has been untouched. Finally, we used our winning_team column to create our target variable which was a column containing either win or loss for each player in any given match.

**Data Cleaning:**

Next we subsetted our data to only include best of 1 (b01) matches, followed by dropping any unnecessary columns. We renamed columns and explored the number of unique values as well as data types for each feature. We assured that no duplicate values were included in our data and looked into dealing with null values in our data. Some rows with null values were just dropped completely and others were handled by imputing values that made the most sense. For example, flash_assists was dealt with by imputing missing values using the median since we needed the column to contain integers.

We next dropped the null value rows as they tended to have missing values across multiple columns and they made up a very small subset of our data (around 2.15%). Doing this gave us a final dataframe with no null values. We then converted our data types to represent the correct ones for each feature, dropped a few more columns that aren't useful, and got our final dataframe for the individual player performance. We then performed the same steps of dropping null values and checking data types, but added the step of aggregating over our matches by team, taking the average of numerical columns and the maximum of categorical ones. This set us up nicely to transition into the EDA stage of our project.

---

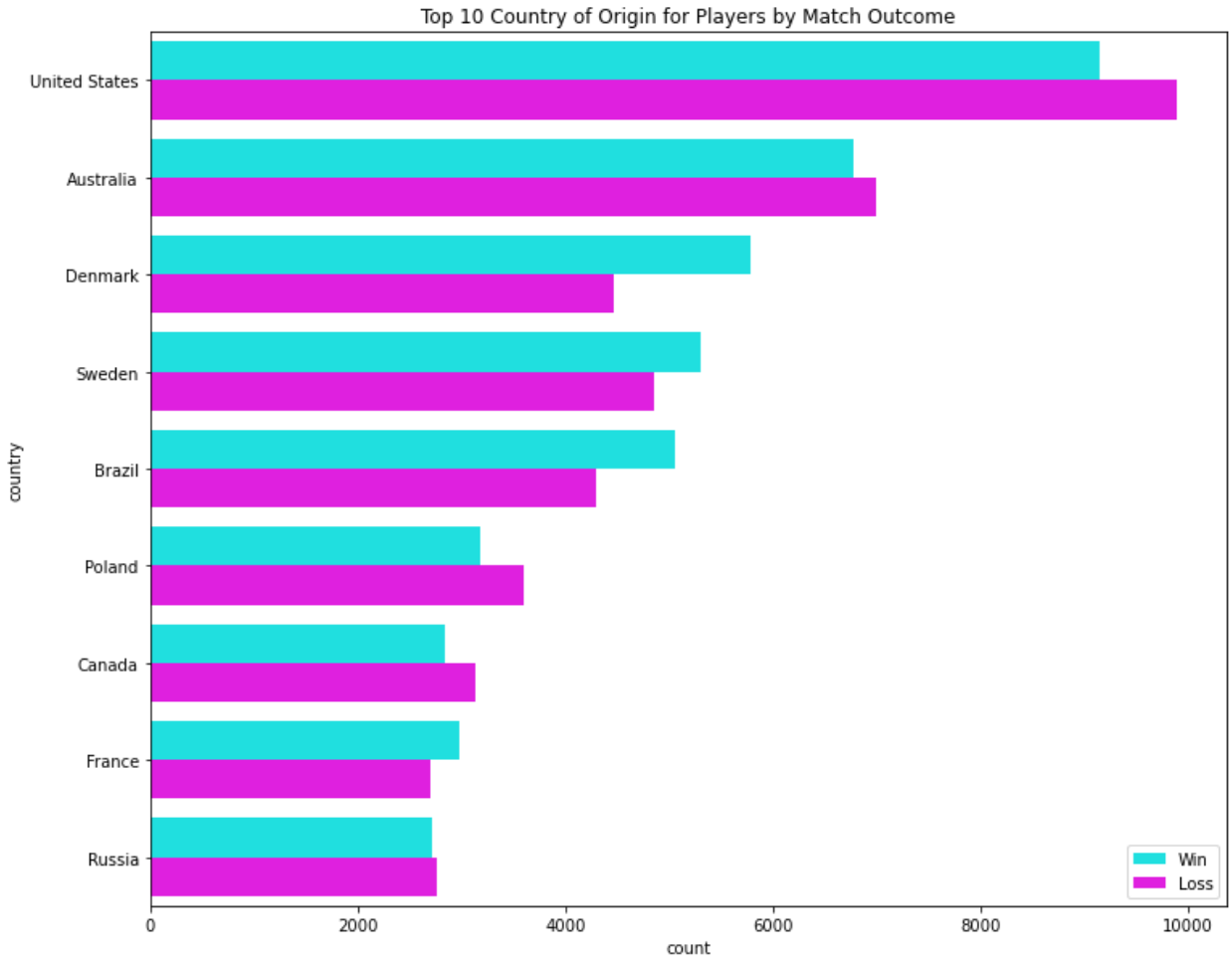
## Exploratory Data Analysis

**Exploring Our Data:**

To explore our data, the first thing that needed to be done was splitting our data once again by win and loss. To recap, this meant we now have 4 dataframe of interest: a win and loss dataframe for individual player performance, and a win and loss dataframe for aggregated team performance.

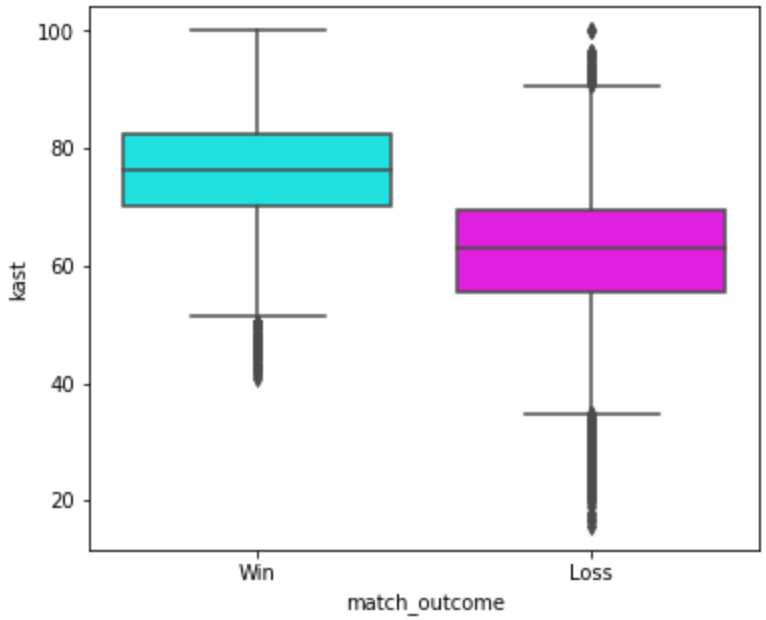
**Individual Player Performance:**

We first looked at the distributions of all of our numerical feature by plotting histograms. We then took a look at the countplot of top ten countries the players came from by match outcome, as seen in the plot below:

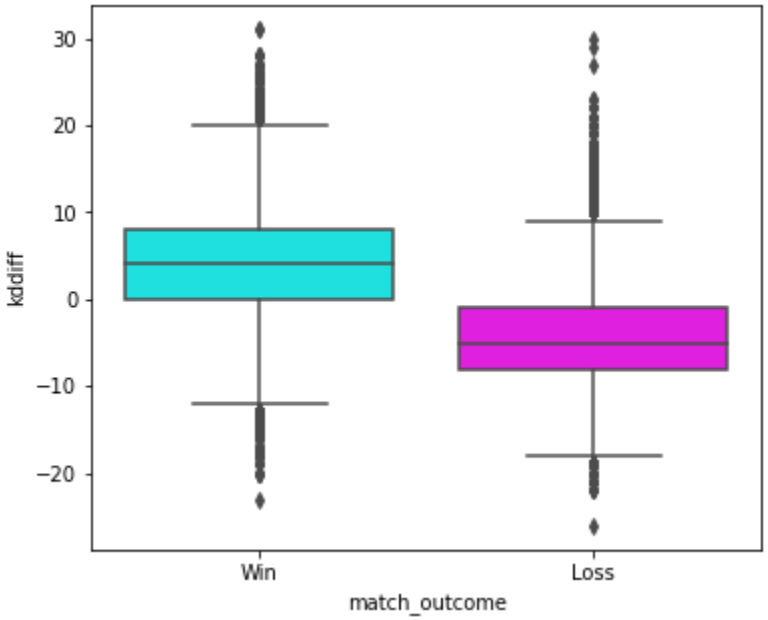



It's worth noting that player country origin is not included in our aggregated team data later as there exists many teams with players from varying countries. We see that the majority of players are from the United States, and should keep in mind that there are a good amount of countries represented with few players being born there (not displayed in the plot). If these countries are among the highest for feature importance in our best model later on, we should be a bit skeptical as there aren't many cases that could be analyzed since there may just be 1 player from a country that has played many matches.

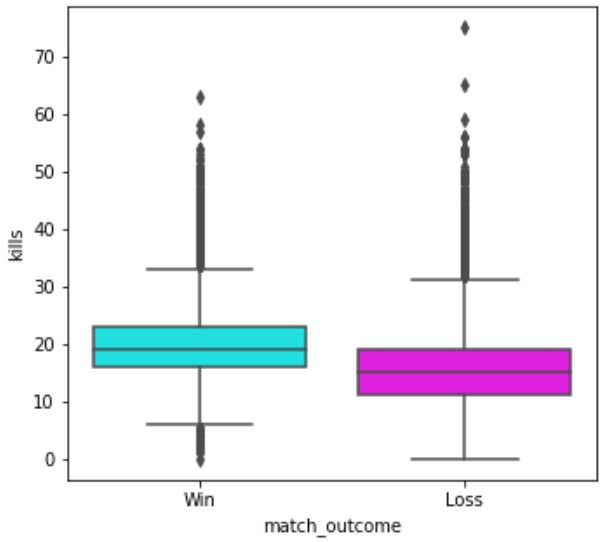
We then looked at the correlation heatmap for the win and loss data separately for the individual player data. Many of the correlations were expected to be high since many of the feature are calculated using other features (i.e. adr is an average of adr_ct and adr_t). Since we are mainly interested in how wins differ from losses, we plotted boxplot comparisons for each numerical feature. One of the plots that stood out the most was the kast comparison which can be seen below:

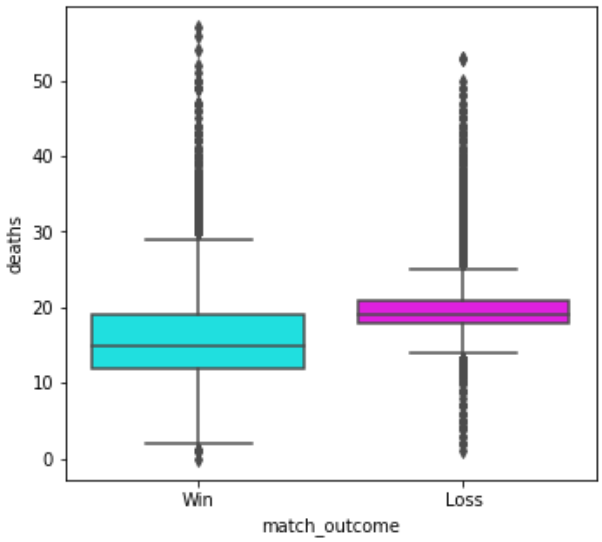



The plot shows that just about 75% of players that won their match had a kast (percentage of rounds the player killed, assisted, survived, or was traded) greater than 75% of players that lost their match. This tells us that kast should be quite useful in predicting a win or loss for any given player. We see a similar result in the comparison for kddiff (total kills minus total deaths) below:




What's interesting about this is that if we look at the boxplot comparison for kills and deaths, this distinction isn't nearly as clear, albeit still exists:



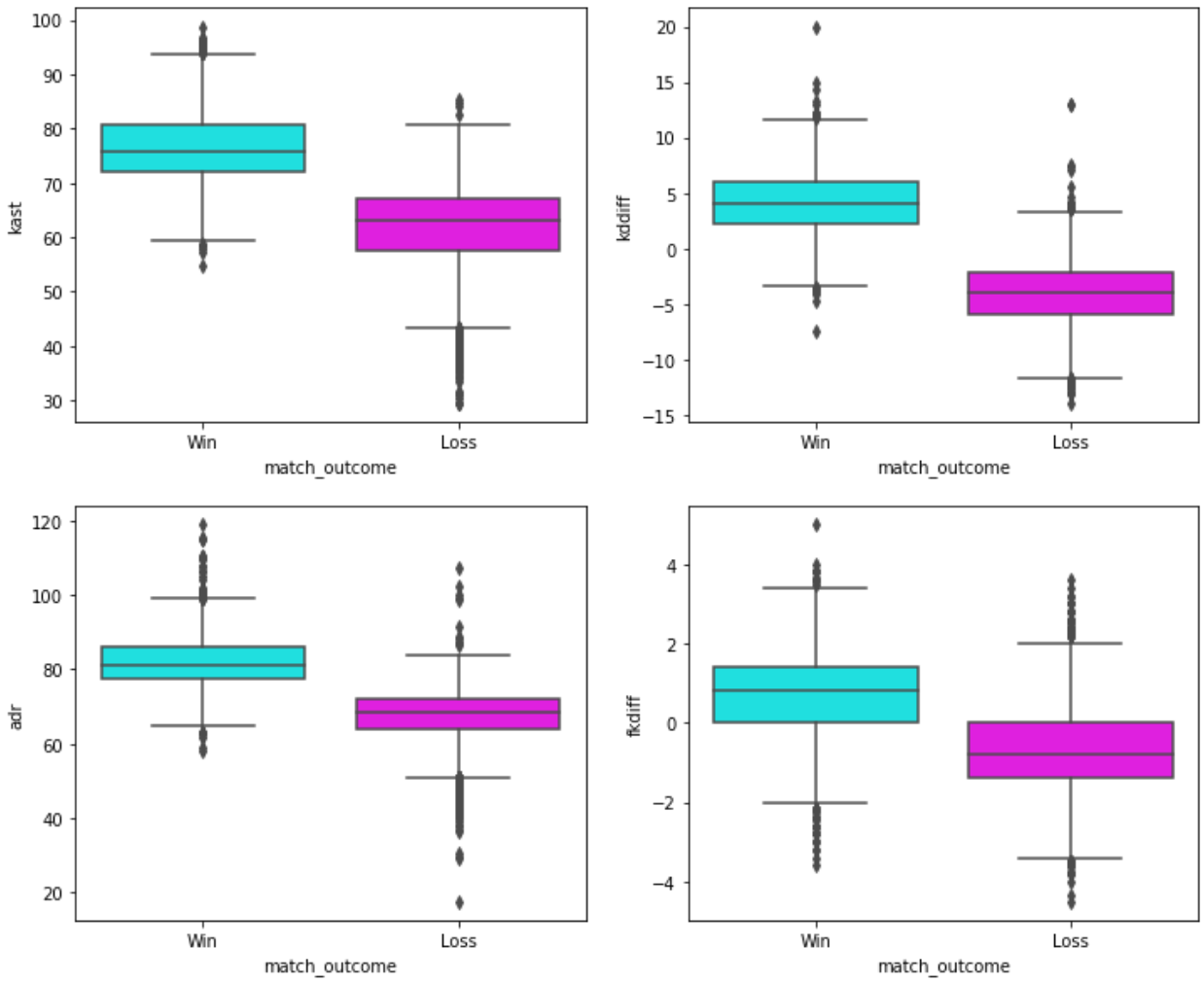




This suggests that while kills and deaths alone might be useful in building our model, the relation between them should be even more useful. It's also worth noting that players who lost matches vary much less in amount of deaths than those who won. The outliers for all of our features seemed reasonable enough to leave them in.

**Aggregated Team Performance:**

We again started out by looking at the distributions of all of our numerical features by plotting histograms. There were no meaningful categorical features to look at as once the aggregation was performed none were useful for our project moving forward. Like with the individual player data, we looked at the correlation heatmap for the win and loss data separately.

Plotting the boxplot comparisons, we saw similar results to the individual data but with even more distinguishable features by match outcome with even wider gaps between the outcomes. Plotted below are some of the features that have noticeable differences:




We should also mention that for kast, kddiff, and adr the related features for ct and t (i.e. for kast, kast_ct and kast_t) had very similar distributions. It is clear that for each of these features there exists a significant difference between win and loss data. This tells us that we should expect our aggregated team performance model to likely outperform the individual player performance model.

---

## Preprocessing

**Individual Player Performance:**

The first thing that needed to be done in this stage was splitting our data into training and test sets (80/20 split). In order to scale our data, we then needed to split the data for both the training and test sets further into numerical and categorical dataframes. The only categorical variable of interest in our case was the country variable. Once split, we fit the standard scaler to our numeric features in our training set, and then used the fit to transform (scale) both our training numeric data and our testing numeric data. To deal with the categorical variable country, we created dummy variables so that our future models could use the indicators in making predictions. Since we have 82 countries included in our data, this step created 82 new indicator columns for our training data. We applies the same method to our test data, and

found that 5 countries were not represented in the test data. To fix this, we simply added the 5 missing countries as new columns filled with zeros to represent no players being from them. Finally, we merged the numerical and categorical dataframes together. This means in the end we had one final training dataframe with both numerical and categorical features represented as well as a final test dataframe with the same features.

**Aggregated Team Performance:**

Everything that we did for the individual player performance data above was repeated for the aggregated team performance data except for the categorical steps. Since we didn't have any categorical features for our aggregated team data, all we needed to do was scale our data using a standard scaler (fit with our training data and transformed on both training and test sets). This set us up nicely to begin exploring different models that could help us solve our initial problem and achieve our goals.

---

## Modeling

**Individual Player Performance:**

The models we tried:

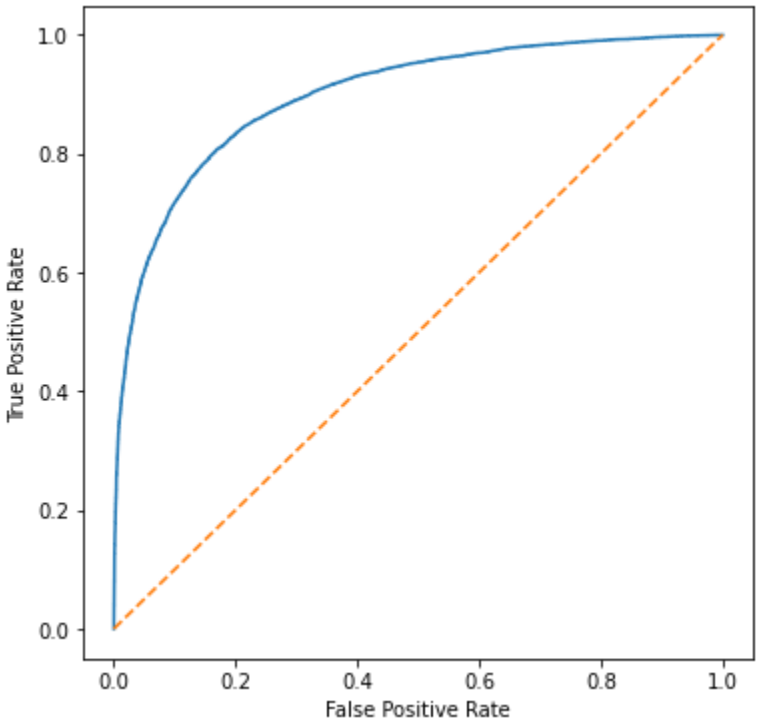
- *Logistic Regression*:
  - Hyperparameter Tuning Method: *RandomizedSearchCV*
  - Best Hyperparemeters:
    - *C*: 0.1
    - *penalty*: l2
    - *solver*: liblinear
  - Best Score on Training Set (Comparing Models):
    - *F1-score*: 0.817
  - Best Score on Test Set (Comparing Binary Classes):
    - *F1-score*: 0.82

- *Random Forest*:
  - Hyperparameter Tuning Method: *RandomizedSearchCV*
  - Best Hyperparemeters:
    - *n_estimators*: 500
    - *max_features*: auto
    - *max_depth*: 20
  - Best Score on Training Set (Comparing Models):
    - *F1-score*: 0.812
  - Best Score on Test Set (Comparing Binary Classes):
    - *F1-score*: 0.82

- *Gradient Boosting*:
  - Hyperparameter Tuning Method: *RandomizedSearchCV*
  - Best Hyperparemeters:
    - *n_estimators*: 500
    - *max_features*: log2
    - *max_depth*: 10
    - *learning_rate*: 0.01

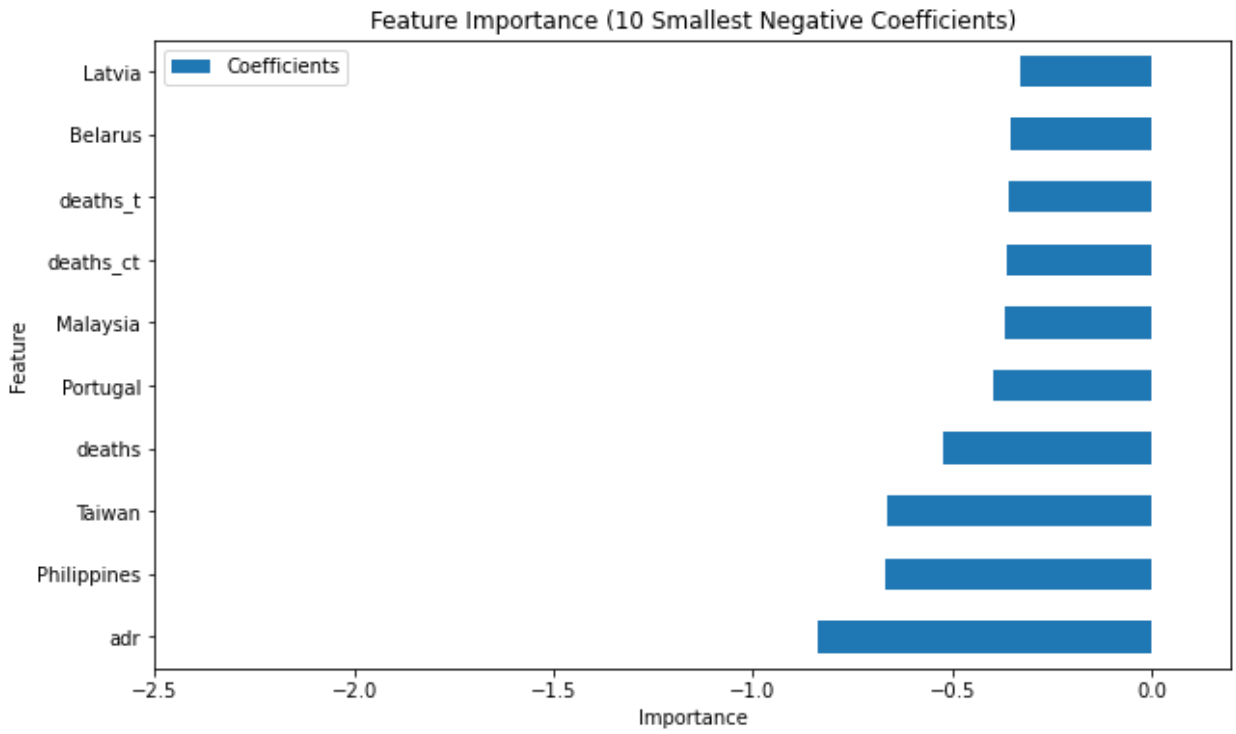
- Best Score on Training Set (Comparing Models):
  - *F1-score*: 0.813
- Best Score on Test Set (Comparing Binary Classes):
  - *F1-score*: 0.82

All three of these models performed about the same, so when determining the best model we selected the logistic regression one due to it being the simplest model, as well as having the highest score by a small margin. The ROC-curve for our best model can be seen below:



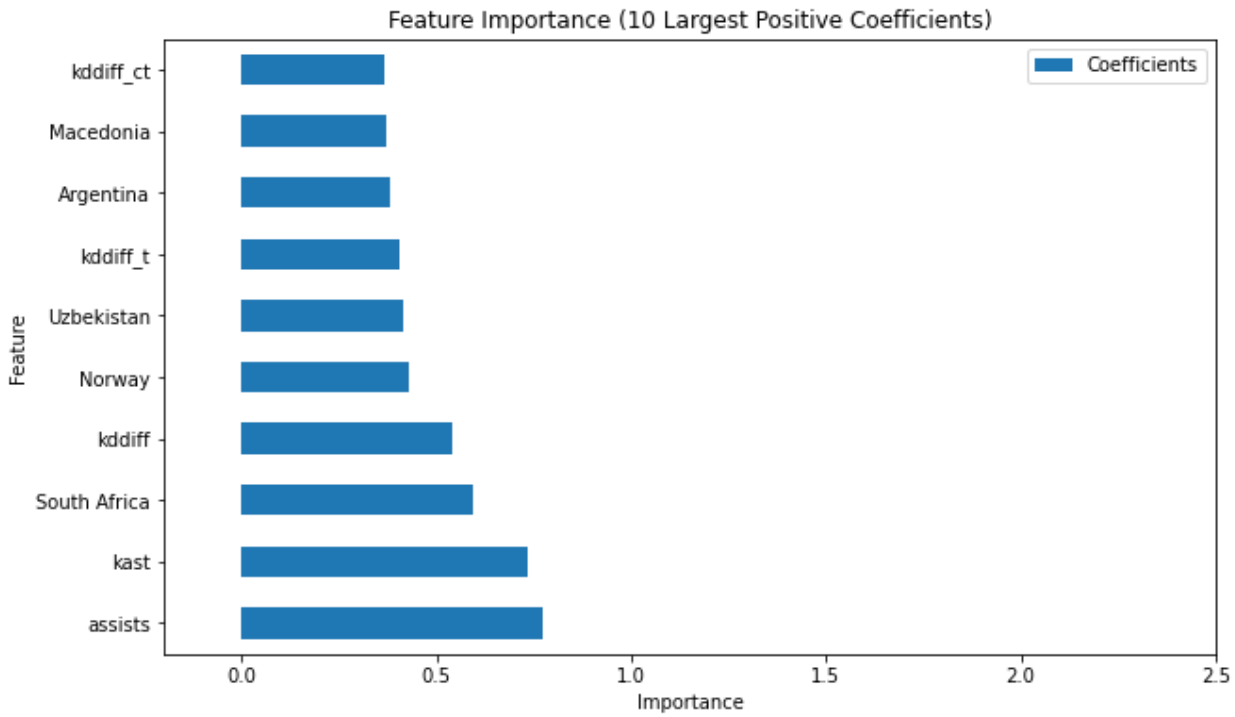

The ROC AUC score was 0.898. This is a pretty good score and tells us that our model definitely has some predictive power.

Before we look into the modeling for our aggregated team performance data, we should look at the coefficients of our logistic regression model to determine which features are contributing the most to our model predictions. Plotted below are the ten smallest coefficients:

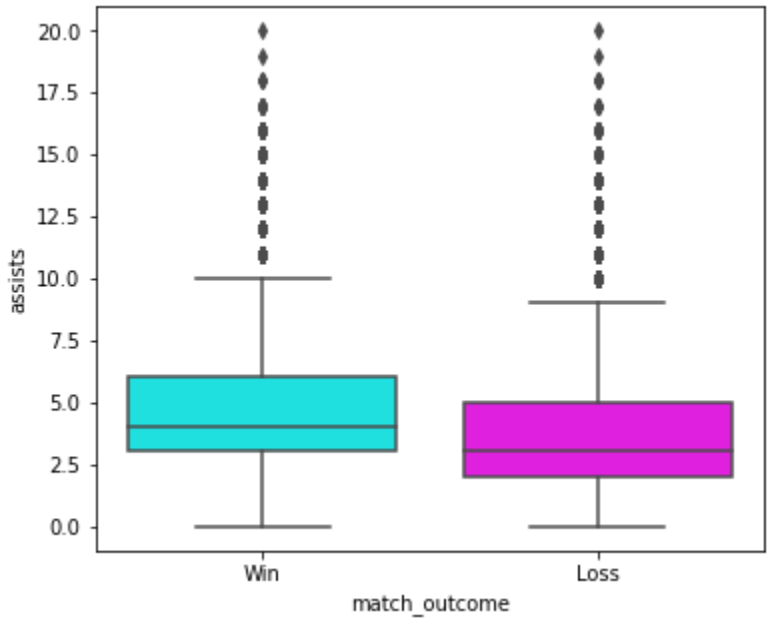



We can see that the six country features included in the plot are ones that are not represented very well in our sample. What's most interesting about this plot is that adr is the most important feature for predicting a match to be a loss. This is a bit of a head-scratcher at first, as we would expect a higher adr to contribute more to winning a match then losing. Seeing deaths, deaths_ct, and deaths_t is not surprising since we would expect that with more deaths comes a higher chance of losing the match. One possible reason that adr contributes so much to losses might be due to individual player performance not being a good indicator of the result of a match. One player having a high adr doesn't mean that the rest of the team is performing well, and maybe one person carrying all of the weight of their team is actually a bad thing. I suspect that there is more value in the whole team contributing somewhat equally, since afterall it is a team game for a reason.

Let's now take a look at the ten largest coefficients:




We see here that like before, the countries listed as most important for our model are those that were not represented well in our sample. The top numerical features contributing to the Win class are assists and kast. This means that as the amount of assists increase, and as the percentage of rounds where a player killed, assisted, survived, or was traded out increase, so does their chance to win the match. This makes sense and checks out for kast, with our boxplots from earlier that showed kast as being much higher for players that won their match. The assists being so important for our model is not that surprising logically, but the difference in distributions for the win and loss data was not as large, as we can see below:



The kddiff, kddiff_t, and kddiff_ct features follow close behind in importance level, which should also not be that surprising with the boxplots examined in the EDA stage.

**Aggregated Team Performance:**

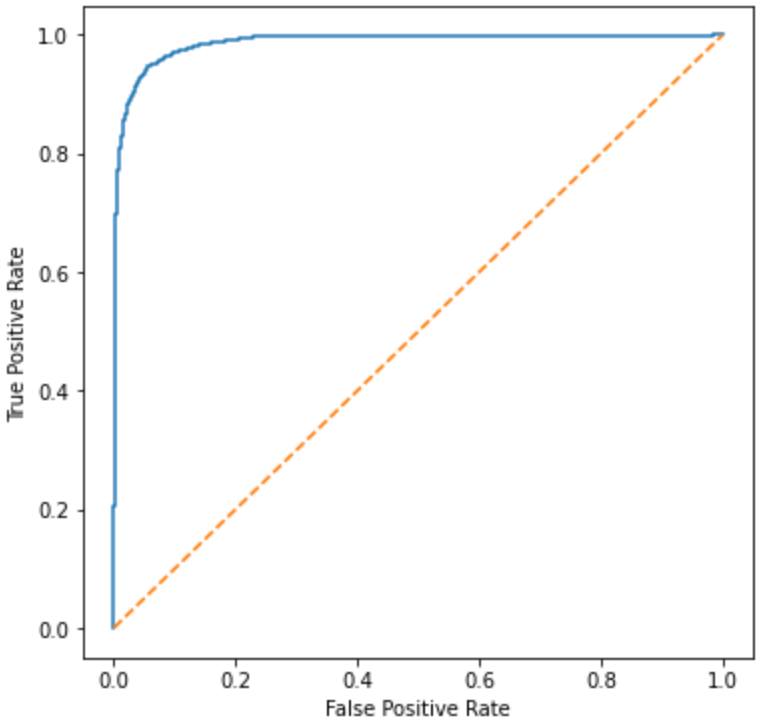
The models we tried:

- *Logistic Regression*:
  - Hyperparameter Tuning Method: *RandomizedSearchCV*
  - Best Hyperparemeters:
    - *C*: 10.0
    - *penalty*: l1
    - *solver*: liblinear
  - Best Score on Training Set (Comparing Models):
    - *F1-score*: 0.945
  - Best Score on Test Set (Comparing Binary Classes):
    - *F1-score*: 0.95

- *Random Forest*:
  - Hyperparameter Tuning Method: *RandomizedSearchCV*
  - Best Hyperparemeters:
    - *n_estimators*: 500
    - *max_features*: auto
    - *max_depth*: 10
  - Best Score on Training Set (Comparing Models):
    - *F1-score*: 0.943
  - Best Score on Test Set (Comparing Binary Classes):
    - *F1-score*: 0.94

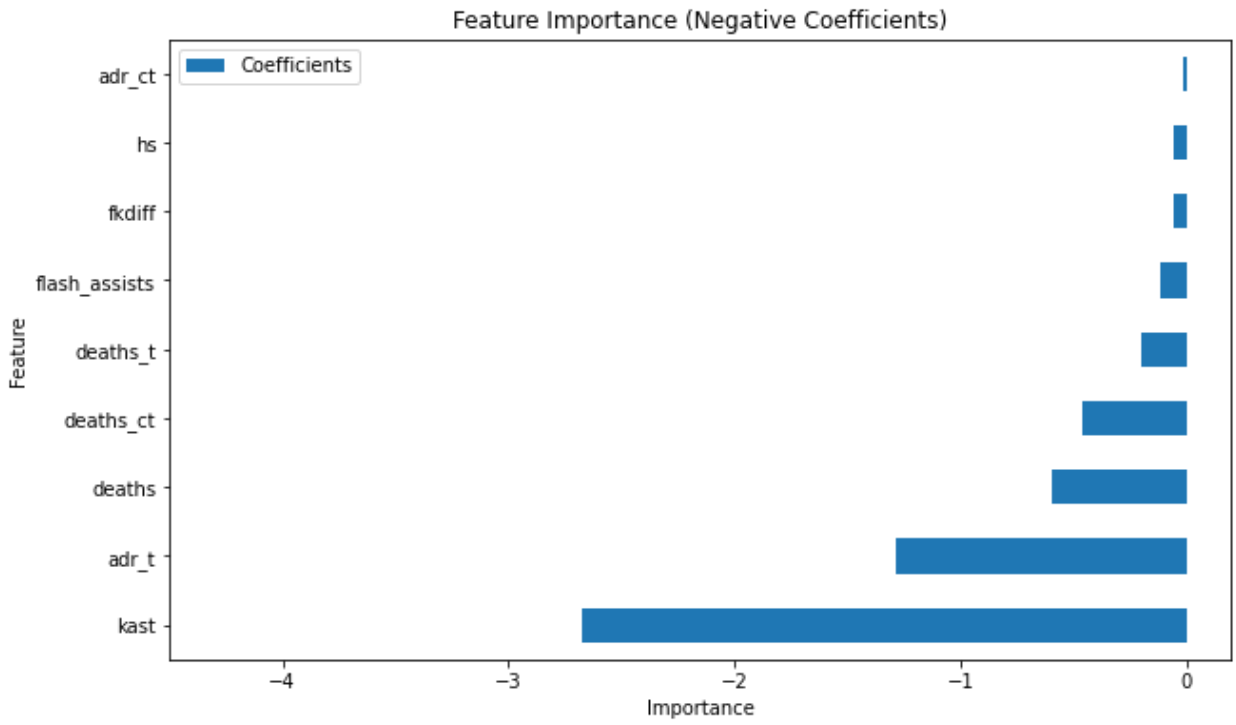
- *Support-Vector Machines (SVM)*:
  - Hyperparameter Tuning Method: *RandomizedSearchCV*
  - Best Hyperparemeters:
    - *gamma*: 0.1
    - *C*: 1.0
  - Best Score on Training Set (Comparing Models):
    - *F1-score*: 0.945
  - Best Score on Test Set (Comparing Binary Classes):
    - *F1-score*: 0.94

Once again, all three models performed about the same, so we took the simplest as our best model: logistic regression. The ROC-curve for our best model can be seen below:



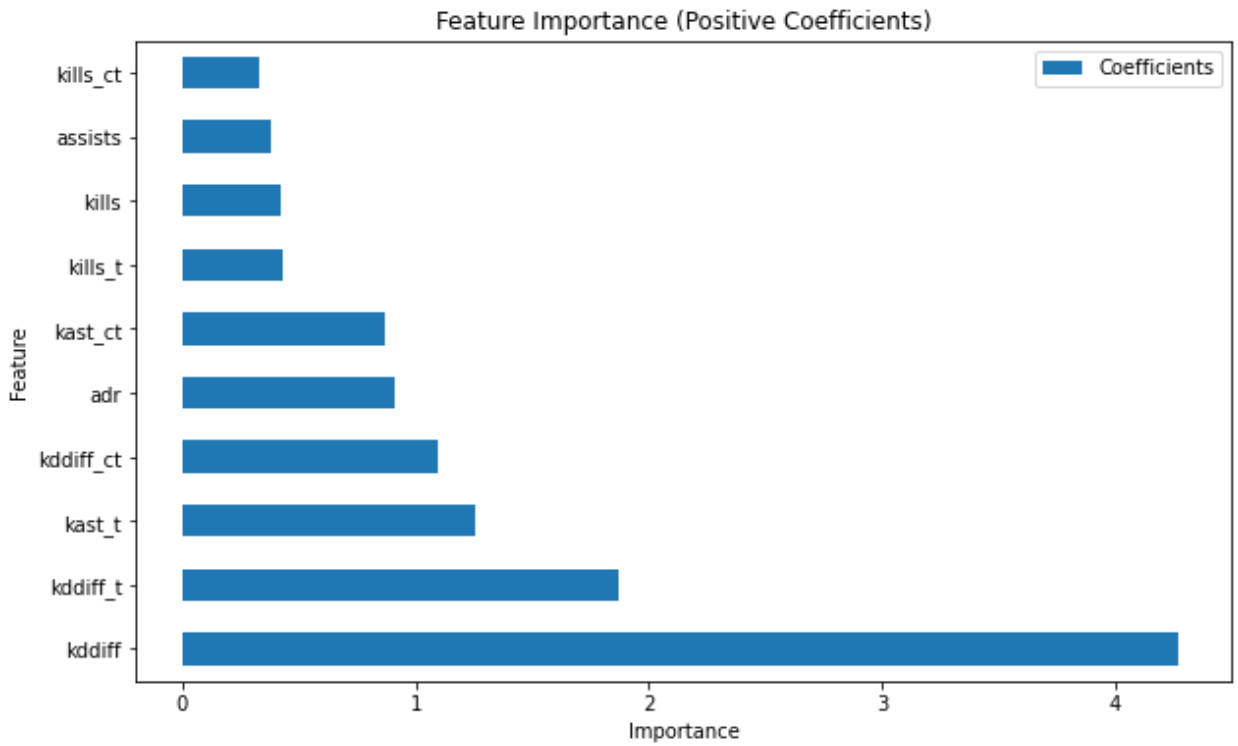

The ROC AUC score was 0.988. This is an extremely good score and tells us that our model definitely has some predictive power.

We should once again look at the coefficients of our logistic regression model to determine which features are contributing the most to our model predictions. It's also worth reminding ourselves that since we dropped observations with null values, some of the aggregations were not performed with five players on a team. These situations shouldn't make a huge impact since the dropped rows made up a very small subset of our data (about 2.15%). Plotted below are all of the negative coefficients:




To our surprise, kast is just as important for making predictions with our aggregated team data, just for the opposite class! The higher the team kast, the more likely they lost the match. This tells us that while kast is useful in predicting wins for individual player performance, the same is not true when all players on a team have their kast averages together. It's tought to know exactly why this may be true, especially when kast_ct and kast_t help predict the opposite class, win. Like before, we would expect all three death features to help predict the loss class, and the only other significant feature to mention here is adr_t, which suggests that having a lower team adr on the T side increases a team's chance of winning. I would think that this is due to the many rounds that are won on the T side by blowing up the bomb rather than dealing tons of damage.

Plotted below are all of the positive coefficients:




The plot shows kddiff being far and away the most useful feature in predicting the positive class, win. It's more than double as important as the next most important feature. Not only that, but three of the top four most important features are a version of kddiff. This is significant, and it is not that surprising since the EDA we did showed us that it would most likely be useful in predicting wins. Recall that kddiff is the kill-death ratio, and that deaths were always useful in predicting losses, so we can see why this feature would be important. Another interesting result from this plot is that while kast was found to be important in predicting losses, kast_t and kast_ct are useful in predicting wins. Also, adr is listed as important here, but adr_t and adr_ct contribute to predicting the opposite class. Assists are also not nearly as important in making predictions as they were for the individual player data.

The most important features for our aggregated team data model are rated as much more important than that of the individual player model. This matches our results that told us our aggregated team model performs better in making predictions overall. It also suggests that team performance is more indicative of which team wins any given match.

---

## Future Scope

- There are many models that I did not get the chance to try, and it is possible that there exists a better model that will produce more useful results.
- I only looked at b01 (best of one) matches for this project, and there still is a ton of data on b03s and even some b02s and b05s. These could be interesting to explore for many different reasons.

- After seeing how a feature could contribute to predicting one class, but it's related features pertaining to which side player(s) are on could contribute to the opposite class, I think that it would be interesting to explore how the sides compare overall across matches. There should be a way to split the original data so that the sides are comparable.

- One of the data spreadsheets that I did not use pertaining to map vetoes (the map selection process in matches) could be very interesting to explore and find out how map selection impacts team performance.

- For comparison purposes, it would be interesting to delve into a game like Valorant, which runs very similarly to CSGO but is newer, and see if similar results are gathered pertaining to common variables between the two games.