STAT 486 Report

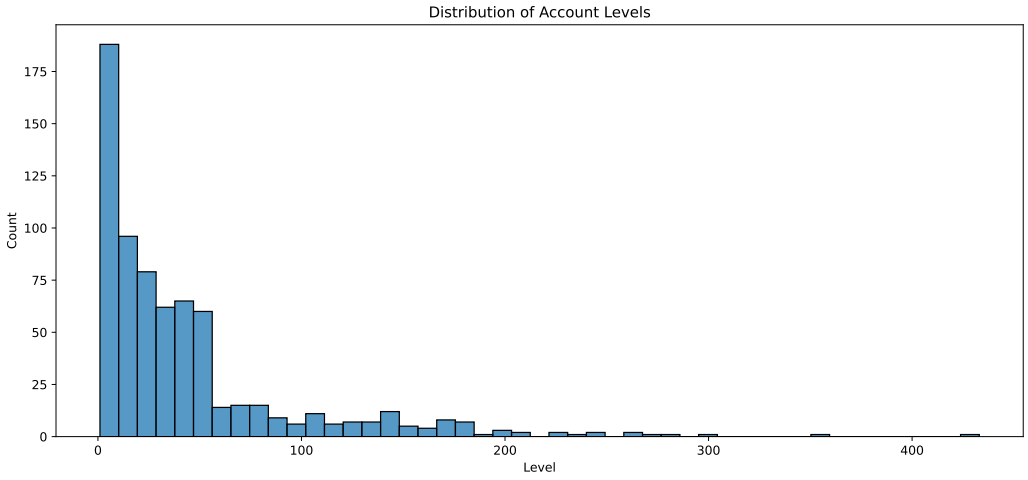
Bryce Marshall, Justin Ross, Tyler Smith, Cameron Slaugh

2024-04-16

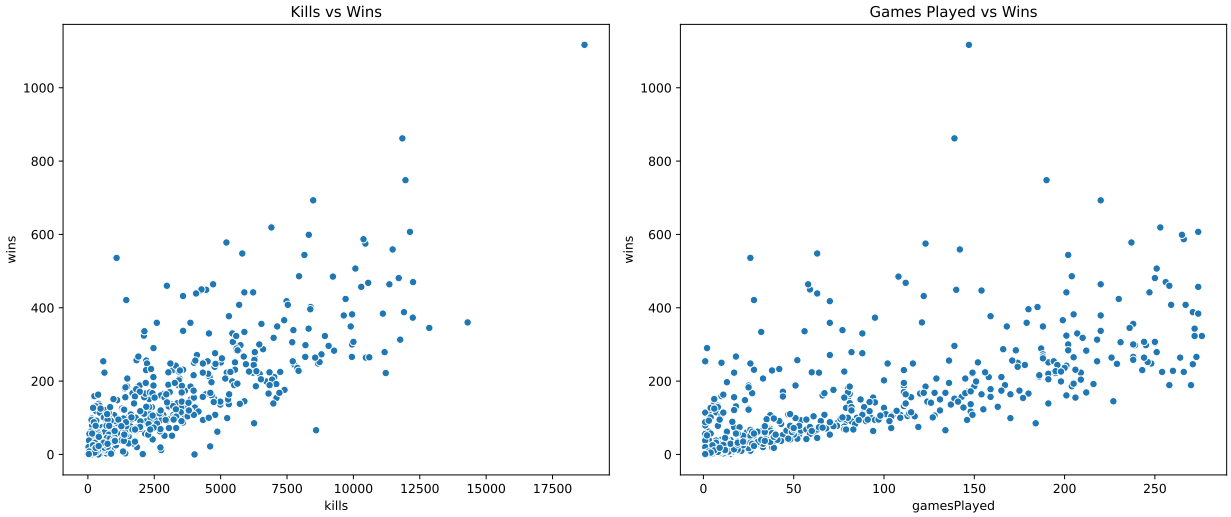
Introduction One of life's most desperately desired achievements is a win in Call of Duty. As such, our goal in this analysis is to determine the factors that contribute to success in online multiplayer matches. The data for our analysis was collected from Kaggle, and contains a number of variables relating to player performance.

Exploratory Data Analysis Our dataset originally included data on 1558 unique Call of Duty players. However, many of these players had not played any games and as such they were removed. We also removed significantly large outliers. As a result, our dataset now consists of 694 unique players.

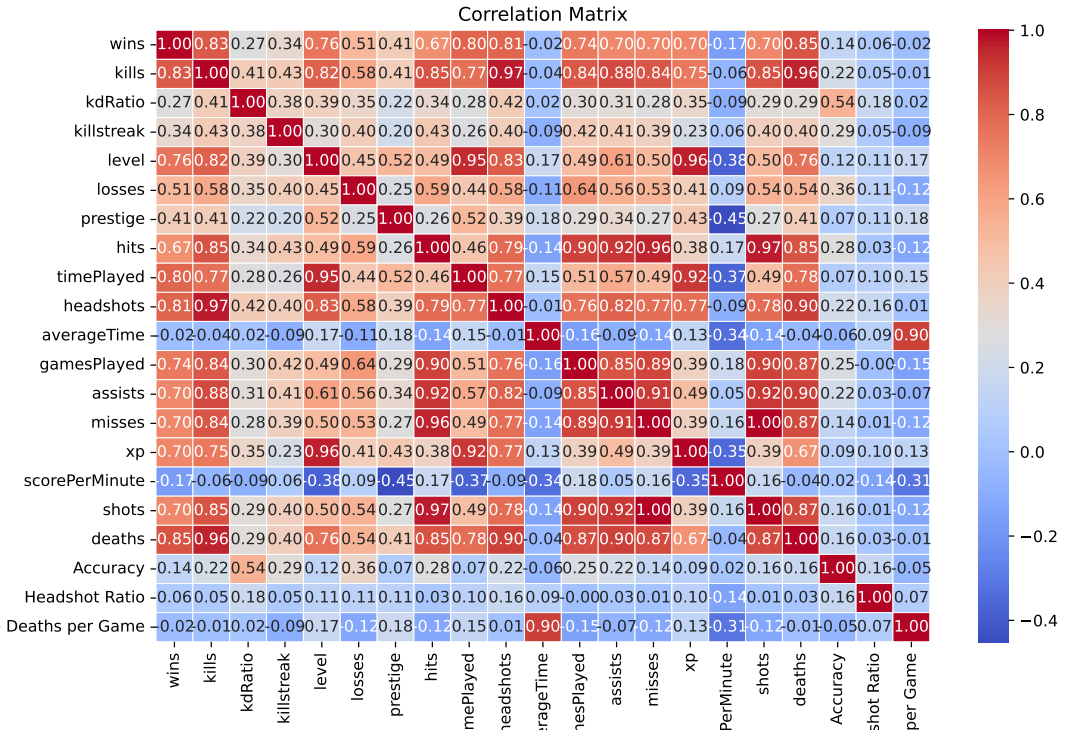
For the players in our analysis, the average number of online matches is 68, and the average number of total online wins is 116. The mean kill-death ratio is 0.838, suggesting that on average, players are killed more often than they kill their opponents. As the plot below demonstrates, most players have relatively young accounts with levels less than 50.



To better understand the relationships between the explanatory variables and the number of wins, we plotted some of the variables we thought would be most related to wins.



These plots demonstrate the high correlations between these variables and a player's number of wins. Kills has a correlation of 0.833 and games played has a correlation of 0.742. However, an issue we face is that there is correlation between some of our explanatory variables.



Due to the multicollinearity in the data, we have to be decisive in the model fitting process. We will have to be careful with linear models, as multicollinearity can cause a model to produce inaccurate model coefficients. To combat this, we will try a number of linear and non-linear models, as well as regularization and feature selection to create our best possible model.

Methods Feature engineering was key in making our model better at predicting game outcomes. First, we cleaned the data by removing any unusual values in the 'gamesPlayed' column using a method called Interquartile Range. This helped get rid of any misleading data.

Next, we created two new features to give our model more information to work with. The 'Accuracy' feature shows the ratio of hits to shots, and we made sure to handle cases where there were no shots to avoid errors. The 'Headshot Ratio' shows the ratio of headshots to kills, giving us more insight into players' aiming skills.

Before feeding the data into the model, we filled in any missing values using the median value of each feature. We also added some polynomial features to capture any non-linear relationships in the data. Finally, we standardized all the features to put them on the same scale, making it easier for the model to understand and learn from them.

Model	Description	Hyperparameters
Gradient Boosting	Builds a strong predictive model by combining multiple weak predictive models, typically decision trees	Number of estimators, Learning Rate, Max Depth, Subsample
K-Nearest Neighbors	New predictions are the averaged target values of the k nearest neighbors in the training data	Number of neighbors, Weights
Lasso/Ridge	Linear models that penalize the size of the model coefficients (this helps combat multicollinearity)	α
Linear Regression	Fits a linear equation to observed data	None
Neural Network	Maps input data to output values by passing data through multiple layers of interconnected nodes	Hidden Layers, Activation, Solver, Alpha, Learning Rate
Random Forest	Builds multiple decision trees and averages the predictions to get a more accurate and stable prediction	Number of Estimators, Max Depth, Minimum Samples Leaf
SVM	Creates a margin around predicted values, trying to find an optimal balance between fitting the data and avoiding overfitting	C, Kernel, Gamma

Model Selection To our surprise, tree based models did not consistently outperform linear models. We used a random forest to predict wins, but the best test RMSE that we obtained was higher than the RMSE that we obtained with linear models like linear regression and penalized linear regression.

Ensemble methods also did not show significant promise over single models. The test RMSE from the random forests and gradient boosting models performed about the same as single models like k-nearest neighbors and performed worse than linear regression and penalized linear regression.

The biggest challenge that we all faced with more complex models like random forests and gradient boosting was overfitting. We consistently obtained significantly lower training RMSE's than testing RMSE's which was a surprise. Even with extensive hyperparameter tuning on the number of trees, depth of trees, and number of features considered at each split, the testing RMSE never got super close to being less than or equal to the training RMSE. Hyperparameter tuning on the learning rate and number of estimators in the gradient boosting model yielded similar results to the random forest.

Gradient boosting and random forests are both robust machine learning methods and we are sure that with an even more thorough hyper parameter tuning process, we could have found a model that we were satisfied with that doesn't overfit the data. However, we determined that the computation would be too intensive for the marginal gain in accuracy. Even though k-nearest neighbors and penalized linear regression are simple methods, we are satisfied with the RMSE that it yielded and the short amount of time it took to run.

A Multiple Linear Regression had our second best test RMSE underfit some degree. Using a standard Linear model, we feature engineered our variables and standardized the data to receive our results which is only slightly worse than our Neural Network.

Best Model A Neural Network, which had our best test RMSE, still overfit by some degree. Using a MLP network we tuned the hyperparameters by using a grid search for hidden layers, activation, solver, alpha, and learning rate. Overall the model took longer to run than some of the others but it was our best model with a test RMSE of 49.91. The final hyperparameters used were:

Hyperparameter	Value
Activation	tanh
Alpha	0.01
Hidden Layer Sizes	100
Learning Rate	adaptive
Solver	sgd

Our model's most important features were Time Played, Level, Score per Minute, and Accuracy.

A K-means clustering analysis offered interesting insights into player performance differences. It split players into four groups based off their stats: new players, elite players, experienced and competent players, and intermediate active players.

- New Players (Cluster 0)
  - These players that have low wins (30.25) and kills (478.57). These players also have a low kill-death ratio (0.71) and engage minimally with the game, as indicated by lower levels (11.73) and moderate prestige (14.5).
- Elite Players (Cluster 1)
  - Players who are highly skilled veterans with high wins (395.79) and an extensive number of kills (9112.71). High levels (173.21) and maximum prestige (100.24) indicate significant time investment and expertise.
- Experienced and Competent Players (Cluster 2)
  - Players with moderate-to-high wins (211.98) and kills (4828.83), displaying a good kill-death ratio (0.96). With substantial prestige (62.08), these players are very good but might not play as intensively as the elite players.
- Intermediate Active Players (Cluster 3)
  - Characteristics: Players who engage deeply during play sessions with moderate wins (103.96) and kills (1654.91), high prestige (99.40), and significant game time per session (47.34 minutes).

Conclusion and Next Steps: The neural network stood out as the best model with a test RMSE of 49.91, highlighting its strong performance in predicting game outcomes. This means it's reliable for our goal of using player stats to forecast wins. The most important features were Time Played, Level, Score per Minute,

and Accuracy. This suggests that the best way to contribute to a win is to play more games, be accurate in your shots, and aggressive in your gameplay. These attributes will most heavily influence your success in online matches.

Looking forward, there's room for improvement. We can fine-tune the neural network's settings to potentially boost its accuracy further. Exploring other techniques like ensemble methods or hybrid models could also help enhance our predictions. It might be beneficial to refine our feature engineering to better capture the unique aspects of the game. Additionally, integrating real-time data and applying methods like reinforcement learning could make our model more adaptable to changing gameplay dynamics. To keep our model effective, regular updates with new data will be essential to ensure its predictions remain accurate and relevant over time.