Predicting eSports Wins Utilizing Real-Time Data

Daniel Craig

CUNY - DATA698

Dr. Arthur O'Connor

1 ABSTRACT

Esports is a growing industry with an abundant amount of viewers and commercial value. One of the forefront Esports games is Dota 2, with tournaments holding prize pools up to $40 million dollars. As a commercial commodity, predicting wins and showcasing likelihood of a match's outcome to the audience holds entertainment value and has been incorporated into features of the Dota 2 client. While past research focuses on using intensive Game State Integration apps and avoids using heroes as features themselves, this research utilizes historical professional match data available through API to explore the effects of using embedding vectors to represent heroes, since the model cannot evaluate a categorical value of a hero name like 'Vengeful Spirit' and thereby including hero composition as a feature where other studies have not, on forecasting gold values and the use of the same embeddings as features in classifier models to hopefully increase accuracy. Embedding vectors are a row of values a model creates and alters during training to represent a categorical variable. This research found that embeddings slightly decreased predictive ability compared to non-embedded models in scenarios where gold is forecasted only 1 minute ahead, but increased predictive ability in forecasting gold at 5 minutes ahead.

2 INTRODUCTION

Esports is a rapidly growing industry, currently evaluated at $4.3 billion dollars in 2024, with a 7% growth rate over the next 4 years and expected value of $5.8 billion dollars by 2028 [1]. In the realm of competitive gaming, Dota 2 is one of the most lucrative titles a professional eSports player can compete in. Viewership for Dota 2's global tournament, The International, garnered nearly 1.5 million viewers in 2023 [2]. The International holds 7 of the top 10 highest prize pools in eSports history with the highest record of $40 million dollars and Dota 2 as a game holds 8 of the top 10 spots [3]. The game holds roughly a 200,000 player count at any point in the day with peaks of up to 600,000 [4]. The game is in a 5v5 team format in a top-down per-

spective, such as real-time strategy games StarCraft or Command & Conquer, with players selecting from a pool of 100+ heroes with unique abilities with the goal of destroying the enemy team's "Ancient", taking players 35 to 45 minutes to complete the game. The "Ancient" is the furthest building from the enemy team's spawn location and serves as an equivalent to a King in chess. The players compete for two primary resources; gold and experience. These resources can be gained by being near or killing enemy players or units. Players are placed on a square map with three distinct areas called "lanes." There are two lanes that trace the outside of the square and one lane that spans the bottom-left to top right corner diagonal of the square. Non-controllable units called "creeps" run down the three lanes where they clash against enemy creeps. Players compete for killing more creeps than their enemies to gain gold and experience until they are strong enough to destroy the enemy's Ancient. Due to the limited areas a player can earn gold, teams give priority to specific players and are typically denoted by position. With 5 players, the positions range from 1 to 5, with 1 being the most dependent on gold and 5 being the least. There are several defensive buildings called towers that are placed in each lane that serve as deterrents to keep enemies from destroying a team's Ancient. Teams contend with each other using the unique abilities of their hero, typically with millions of dollars on the line.

As a heavily viewed eSport, statistics, predictions, and match favorability are presented during live feeds of professional matches with an entire production crew serving live overlays and stat blurbs during gameplay. One feature that was embraced and developed by the game's developer, Valve, is a live feed of win favorability, in the form of percentages, throughout the game. Yu et al. (2018) [5] found the feature to be 68% accurate at the half-way point in games. This favorability can drastically change throughout the course of the game swinging from 80% favorability to as low as 30% at times. While a formal explanation of Valve's methodology for

predicting a win during a live game has not been made public, there have been several forays into predicting wins.

This research seeks to add to this existing knowledge focused on enhancing viewer experience with meaningful win prediction probabilities by training a model to give win probabilities every minute using historical data accessed via API by exploring the impacts of using embedding vectors to represent heroes and their relationships with gold over time. Deviations from existing knowledge and this paper are the significantly reduced workload related to data collection, the use of embedding vectors to capture unique hero and gold relationships, and a utilization of neural net architecture to improve predictions. The model design is intended to be a Long Short-Term Memory (LSTM) model, which is a specialized recurrent neural network, forecasting over hero gold acquisition and team fight outcomes, with a Random Forest and XGBoost model layer to create win probabilities per minute mark. This change in approach from other papers, that use a Game State Integration application during each game, avoids intensive time needed to collect data for model training and a change in features guided by domain expertise to reduce feature management.

3 LITERATURE REVIEW

Song et al. 2015 [6] predicted wins based on only the heroes selected during the drafting stage of a match before any play across 3000 matches. A logistic regression approach was utilized with features chosen as heroes drafted and hero combinations for a testing accuracy of around 60%. Akhmedov and Phan [7] created a game – state integration app to pull data during a game live for roughly 100 games, performed a sensitivity analysis around the variable *player.gold* to determine highly correlated features to create a new dataset of variables for use in

a multi-step forward prediction and fed into Linear Regression, Neural Net, and Long Short-Term Memory (LTSM) models for impressive 82%, 88%, and 93% accuracies, respectively. While this method was effective, the paper did not explore the impact of this on predicting on which team won. Yang et al. 2016 [8] combines pre-game features in player ranking and player hero statistics with in-game variables as features in a multi-model architecture to create a final prediction. Yang et al. 2016 [8] used pre-game features in a logistic regression model which results in a 70% accuracy in prediction. Gold, death, and experience variables, gathered from live games, were used in an Attribute Sequence Model (ASM) to predict the transition probability of the variable *player.gold* to quantify likelihood of gold change. The outputs from the pre-game logistic regression model and the ASM transition probability metric are combined into a final logistic regression model to create a prediction in which team would win. A comparison of prediction accuracies between pre-game, real-time, and combined models are compared over game duration with the combined models sitting between 75-80% within the first 20 minutes of a game and 85-95% after the 25 minute mark. A common theme across papers using a time-series of *player.gold* and a lack of focus on involving which hero is related to player.gold is apparent.

As some studies progressed into deep learning utilizing recurrent or convolutional neural networks, others returned to classic machine learning algorithms such as V.J. Hodge et al. (2021) [9]. V.J. Hodge et al. (2021) [9] focused on a Weka Logistic Regression, Weka Random Forest, and Microsoft's LightGBM algorithms for prediction and parsing features using a replay parser. Specific features were not detailed outside of being labeled as 'in-game metrics'. These models were tested during a live event, ESL One Hamburg 2017, in coordination with the event handler to generate live stats and enhance commentary and analysis by the talent panel. Game stats were retrieved using another Game State Integration app. Accuracy for predictions in professional

games were placed at 75%. This is a stark difference from Yang et al.'s (2016) [8] approach of utilizing pre-game features in a layered architecture approach.

4 HYPOTHESIS

This paper seeks to answer whether representing heroes, their expected future value of gold, and the difference in gold between teams and positions to a model will increase accuracy in predicting which team will win in comparison to not including hero representations. Using a simple gold differential between teams, as other research has done, does not account for latent implications between gold amounts on specific heroes and its implication on the game. To represent heroes and their unique relationships with gold, embedding vectors will be used. To accomplish capturing a hero's relationship with gold and forecast values, an LSTM forecasted gold value and the LSTM's learned embeddings for each hero will be fed into a Random Forest Classifier and an XGBoost classifier to predict which team will win the game. This research will use each position's amount of gold (10 features, 5 for each team), an embedding for each hero (10 embeddings), the difference in gold between opposing team positions (5 features), and a total gold team differential value (1 feature) that measures gold difference between teams for a count of 26 features used in modeling to answer the research question. The intuition behind the research question lies in a deeper understanding of the relationship between hero choices and positions.

The position 1 naturally requires more gold to be effective, but once large amounts have been gained this role utilizes gold more effectively than any other to impact a game. In contrast, the position 5 does not require much gold at all to impact the game significantly, and typically does not drastically impact a game with more gold but position 5 heroes are typically stronger than any other hero early into the game. Out of the 124 heroes that can be chosen for a role,

there is intuition amongst the community that despite having set gold priorities within a team, certain heroes gain gold faster due to abilities and may require more or less gold to impact a game even within their allotted position. While two different heroes may be played as a position 1, one of them can be adept at gathering gold but has little impact with high amounts, while the other is poor at gathering gold and extremely impactful with high amounts. Even within the same position, this illustrates the relationships between a hero, its ability to impact a game, the amount of gold it has, and the time of the game. Due to this relationship between gold, time, and power for each hero, the amount of gold on which heroes at which times is important in determining a winner.

5 STATISTICAL METHODS & VARIABLES

An LSTM model was used to forecast hero gold values both one minute out and five minutes out. To forecast a minute or five minutes out is computationally simple, but to predict a win or loss on all feature values at every minute is computationally intensive. A Random Forest and XGBoost classifier model was used to predict a win or loss. To show feasibility of operating in a live scenario, yet accommodate the practical limitations of this research, the final features to predict a win or loss were kept to their states in the 15, 20, 25, 30, 35, 40, and 45 minutes. The LSTM model would forecast the anticipated gold of the hero for each player to each of these minutes and the subsequent Random Forest and XGBoost models would use those values as features. This method may be as a result of practical limitations, but also gives insight into the ability to predict with these models if accounting for the likelihood of a game to end at a certain point in time. With the current patch of the game, games tend to end between 30 and 40 minutes.

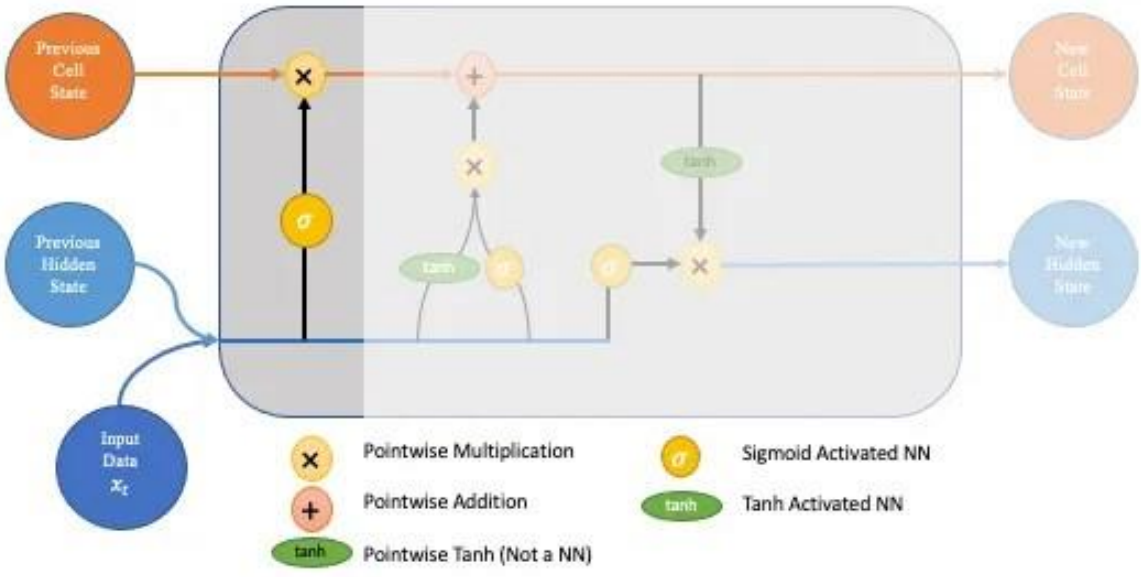
Feeding a single step forecasting model its own predictions can lead to exposure bias and introduce more error than training the model to forecast out five steps, thus the five steps model needed to be made. A Temporal Fusion Transformer (TFT) model was considered for the task, but due to computing limitations was determined not applicable. TFTs rely on having significantly more data than an LSTM and tend to be more variable in their predictions.

For forecasting with the LSTM, feature variables will be heroes in the form of their hero_id value and the time-series of each heroes' gold for a match. From the perspective of an LSTM model predicting on one time-series, hero_id will be a categorical variables with time-lagged gold values. In summary, this creates a multi-variate time-series prediction with a categorical variable and time-lagged features for each time-series. The number of time lags used are 1, 5, and 10 with performance dictating which will be used as a final model. Since the LSTM model cannot handle categorical variables as they are, hero_id will be represented as an embedding vector. Embedding sizes are 8 and 84. LSTM Models are tested without any embedding to compare and serve as evidence to conclude whether involving heroes as a feature is useful in predicting future gold values. Since there are 124 different heroes a significantly sized one-hot encoding object would be needed that would likely impact model performance, thus an embedding matrix is used. It is important to note that the embedding vector combined with the other heroes' time-series data is the technique used to ensure that heroes and their relationships with gold are represented. The LSTM model will use the embedding for each hero to distinguish the unique hero relationships with gold, capturing which heroes gain more or less gold, as well as capturing which heroes are given priority in gaining gold by the team. The LSTM will predict a gold amount for the given time-step of each of the 10 heroes in a match and pass these values, as well as the embedding vector, to the Random Forest and XGBoost models to predict a win or a loss.

An LSTM model is a type of recurrent neural network (RNN) that handles the vanishing gradient problem often seen in other RNN's by controlling the information in the cell states. The basic anatomy of an LSTM model contains two states and three gates. The first state, referred to as the cell state, carries information between iterations or time steps in the model. One can think of this as the long-term memory deemed important for the model. The second state, referred to as the hidden state, handles the most recently passed information to determine updates to the cell state by using the three gates. One can think of this as the short-term memory aspect of the model that gleans important information for longer term use.

Figure A: Forget Gate

(Dolphin, 2022) [10]



As the time-step value is passed to the hidden state, the first gate called the ‘forget gate’ evaluates the information and updates values in the cell state to remember or forget the past information by point-wise multiplication. This is effectively viewing the current information in the hidden state for current context and determining past information in the cell state to determine which of it is still relevant. The forget gate uses a sigmoid activation function that forces mathematical values between 0 and 1 to accomplish this “remembering” or “forgetting” to alter the values in the cell state.

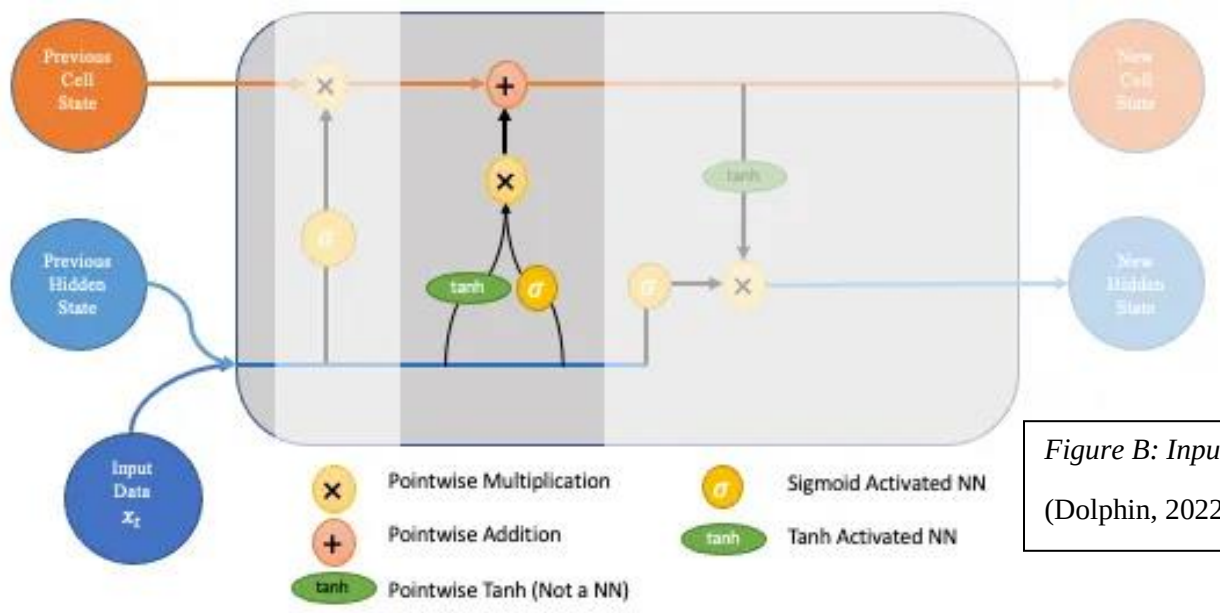


Figure B: Input Gate

(Dolphin, 2022) [10]

The second gate, called the ‘input gate’, contains two parts and is responsible for adding information to the cell state. The input gate first evaluates which information currently held in the hidden state should be used to update the cell state with a sigmoid function forcing values between 0 or 1. Secondly, it creates a vector of candidate values based on the current time-step in the hidden state. The vector of sigmoid values from the first part and the candidate values from the second are multiplied together to create the new values which are added to the cell state.

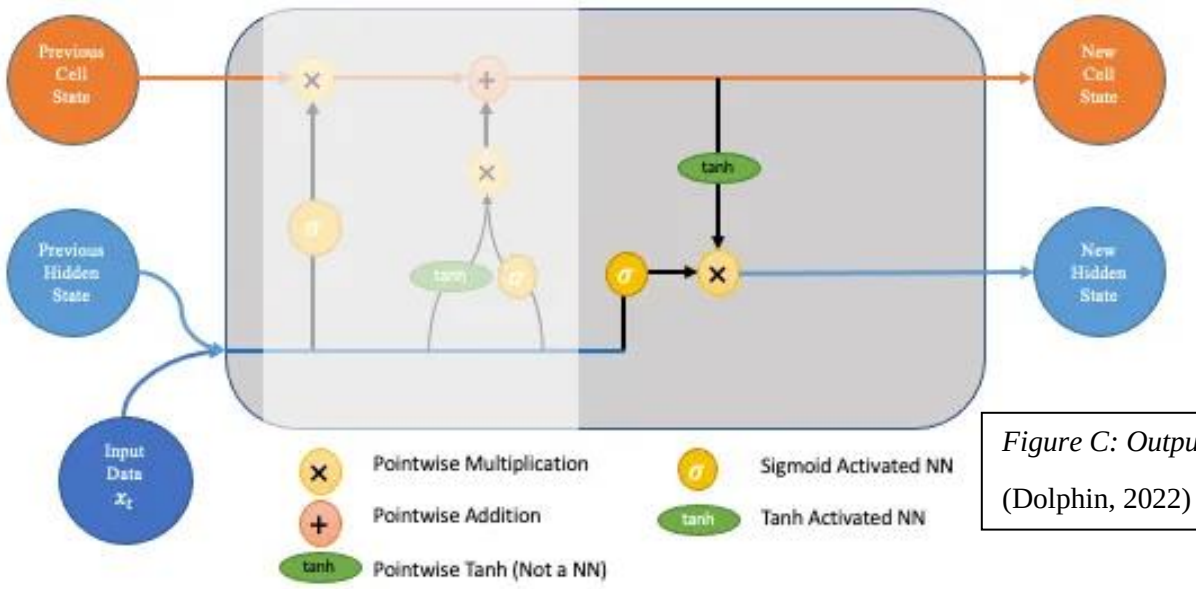


Figure C: Output Gate (Dolphin, 2022) [10]

The third gate, called the 'output gate', controls what the hidden state will look like for the next iteration of the model by using the cell state. This would be akin to dealing with a current situation using one's long-term memory. First, the output gate applies a tanh function, which forces values between -1 and 1, to the cell state. Secondly, the hidden state and current input data are concatenated and passed through a sigmoid activation function. The tanh values of the cell state and the sigmoid values of the hidden state are then point-wise multiplied together and passed on as the new hidden state to iterate over the next time-step as input data and repeat the process (Mohammadi et al., 2015).

A differentiator, of practical importance for companies, between this research and others is the lack of a Game State Integration used by [7], [8]. And [9] to collect all data available from a game for training purposes, and instead relying on the recorded minute mark values of variables in the historical database to re-create a "live-game" scenario for the model to train on. Papers focusing on live-game statistics with a type of game state integration tended toward smaller sample sizes due to the work involved with data collection. The advantage of using the recorded

minute mark values of variables that this paper suggests is access to hundreds of thousands of samples that can be used as training data. The disadvantage is the lack of numerous other variables that a Game State Integration app is able to gather. The data will be collected using the OpenDota API that gives access to Valve's database storing Dota 2 game metrics. The database is updated per game and thousands of games are played every day.

6 RESULTS

The dataset contained 10,000 matches of professional Dota 2 matches played from January 1st, 2024 to April 24th, 2024. The hypothesis of this paper is that using embedding vectors to represent heroes, when combined with past gold values will lead to a better prediction in future gold values for each hero and thus the outcome of the game due to the community's long-held belief that gold amounts on different heroes are worth more or less on certain heroes at different points in the game.

The LSTM models were trained with varying amounts of lookback windows, horizons, and embedding sizes. Pending the lookback parameter, a hero's time-series is segmented into rolling-windows and used as features in combination with the hero embedding to serve as input features. All models used 64 dimensions in their hidden node. The RMSE metric was used for interpretability as it keeps the errors in the same unit measured. For a frame of reference for how impactful an amount of gold can be in a game, 2000 gold is worth about a low-impact item, 4000 gold is worth about a medium-impact item, and 6000 gold items are high-impact and can be game-winning. Each of the test errors below are averaged across the number of time steps pre-

dicted on in a match. A correct interpretation of them is the average amount of gold the prediction was off by at a single minute. The following LSTM models were trained with their associated parameters:

Lookback	*Horizon*	*Embed Size*	*Train RMSE*	*Test RMSE*
10	1	84	31	42
10	5	84	63	113
5	1	84	30	38
5	5	84	47	91
1	1	84	31	62
10	1	8	31	35
10	5	8	44	138
5	1	8	34	105
5	5	8	39	138
1	1	8	40	73
10	1	N/A	29	28
10	5	N/A	203	249
5	1	N/A	30	30
5	5	N/A	287	355
1	1	N/A	35	35

The two best LSTM models are highlighted in green, one containing an embedding, and another without an embedding. These were used to predict a gold value on the time-series of the test data at the respective minute marks: [15, 20, 25, 30, 35, 40, 45] to be treated as the feature variable ‘last_gold’ for the Random Forest and XGBoost classifiers. Specifically, the 10 Lookback, 1 Horizon, no embedding model which scored a 29 and 28 in training/testing respectively, and the 10 Lookback, 1 Horizon, 8 embedding for its 31/35 scores. It seems that the only time

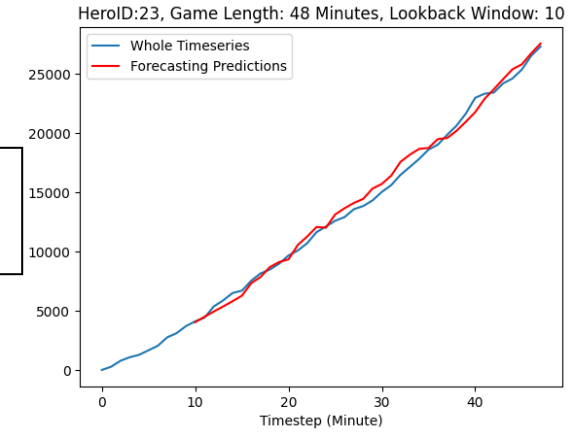
the embeddings outperformed models lacking the embedding is when the forecast had horizons at 5. While a per-minute model would most likely be used in a production settings for simplicity, its important to note that if a model wanted to predict when a game would end and evaluate a winner at that time, a forecast requiring a horizon would be aided by having a hero representation through the embedding. There also seems to be a trend that more lookback steps tend to decrease error within embed size groups.

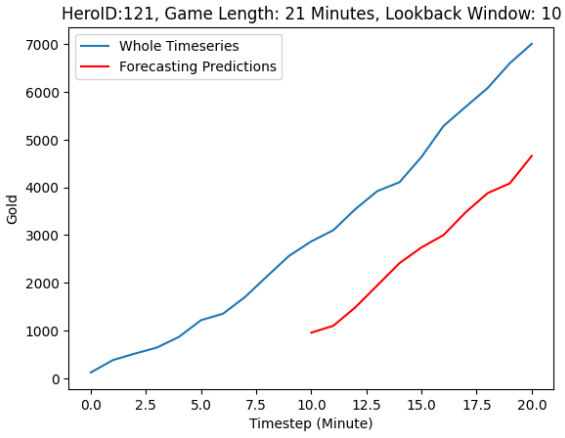
6.1 LSTM Performance Plots

For both models, the best and the worst predictions on a single match are show cased below.

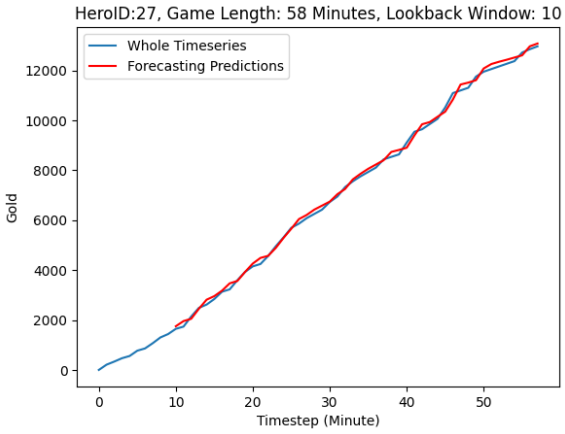
Figure D: LSTM Results

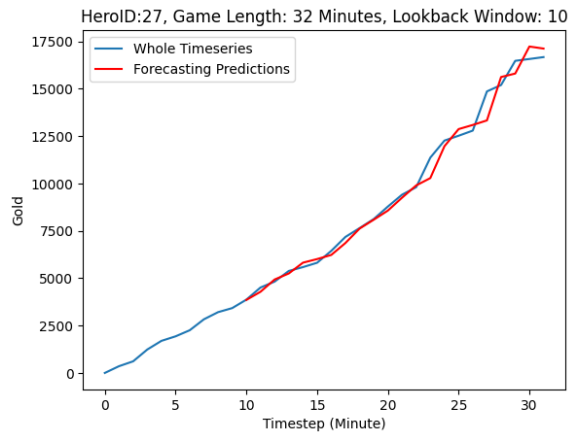
Lookback 10 Horizon 1 Embed 8



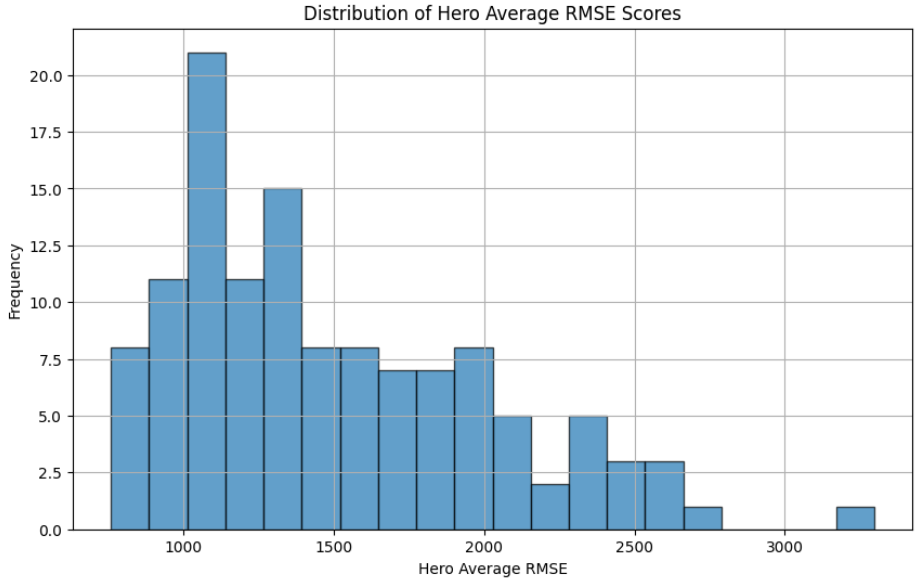


Lookback 10 Horizon 1 No Embed





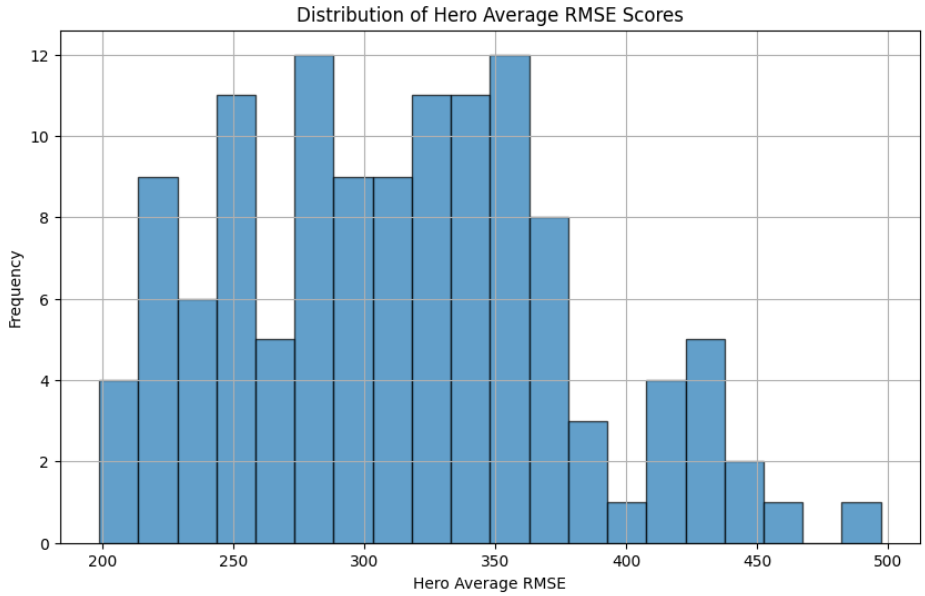
Its clear that while the embedded model can perform quite well, it also has higher variance in its poor predictions. While this showcases the best and worst, the scatter plots and histograms give an idea of the over model performances. Despite what seemed to be promising test scores for the embedded model, drilling into hero-specific differences reveals a lack of accuracy.



Lookback 10 Horizon 1 Embed 8

While most heroes hover around 1000 to 1500 gold off in a match, this pales in comparison to the predicted values without an embedding.

Figure E: Hero Distribution Errors



Lookback 10 Horizon 1 No Embed

Impressive, by comparison, the non-embedded model estimates a difference of no more than 500 gold off for any hero over the course of a match.

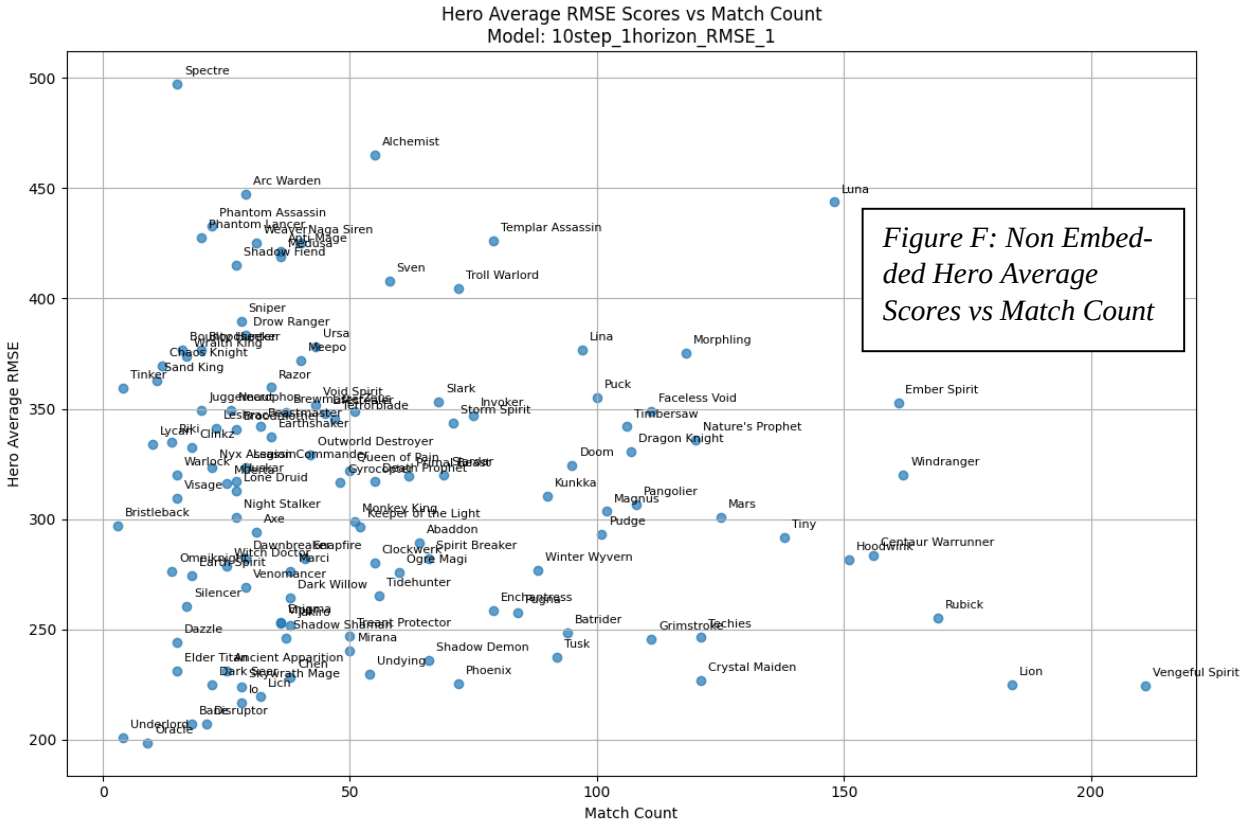


Figure F: Non Embedded Hero Average Scores vs Match Count

In attempts to characterize the heroes with high or low error scores, as well as see if any are prone to a lack of data by observing match count, the scatter plot for the non-embedded model shows that the worst performing heroes are typically considered “feast or famine” heroes that either do extremely well or extremely poorly. Most of these heroes are known for being extremely fast at gaining gold, and if not actively challenged by the enemy, will win the game. These types of heroes were hopeful targets for the embedding method to out-perform a non-embedded method. Note that the y-axis does not exceed 500, leaving a relatively unimpactful error in context of predicting based on gold values.

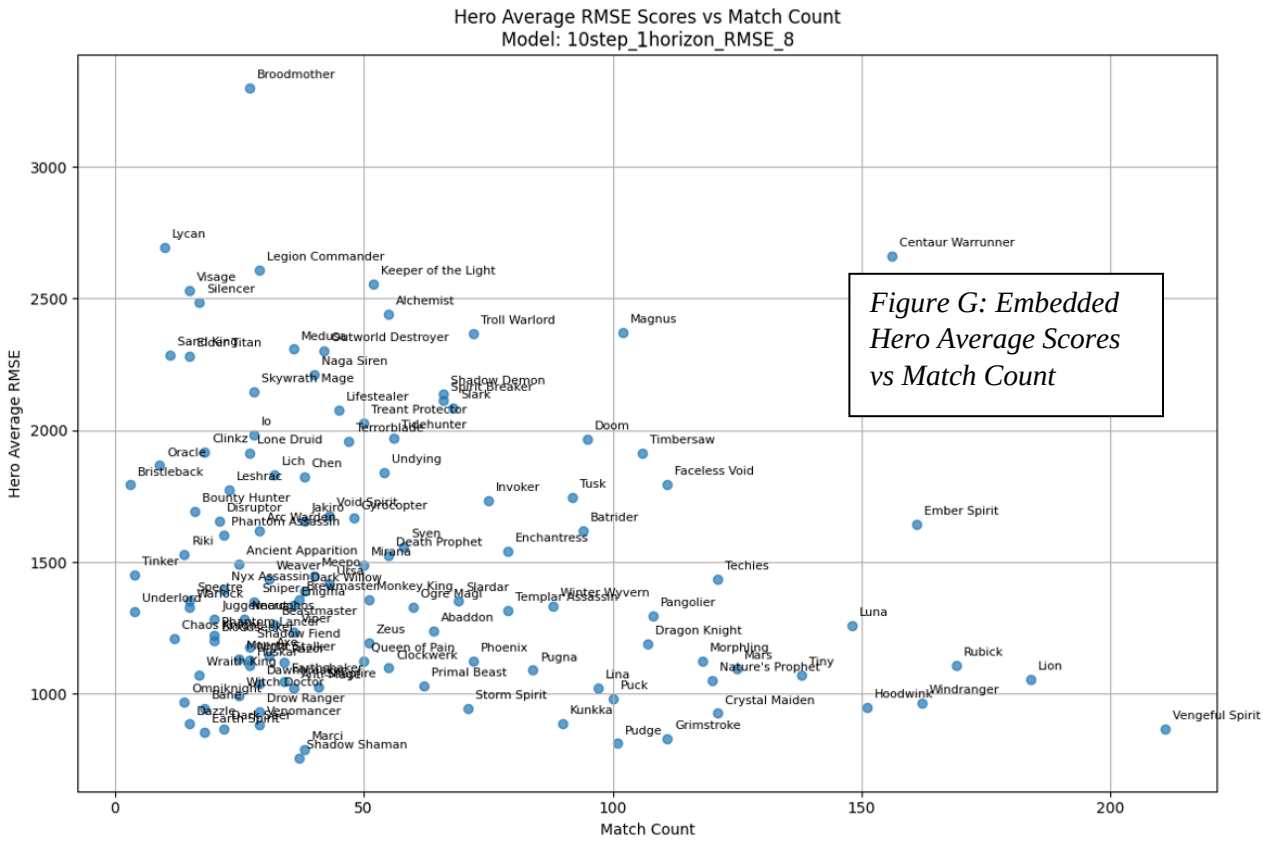


Figure G: Embedded Hero Average Scores vs Match Count

While most heroes are under a 2000 gold error, there are some similar heroes with the higher errors, such as Alchemist. In comparison by hero, the embedded model seems clearly outclassed by the non-embedded model. After predicting on the test data with both the 10 lookback, 1 horizon 8 embed and no embed models, the predicted values and classifiers were tested.

6.2 CLASSIFIERS: RANDOM FOREST & XGBOOST

The Random Forest and XGBoost models were trained on the following features:

Variable Name	*Description*	*Type*
team_diff	Difference in gold between teams. Positive values refers to a Radiant Team advantage. Negative refers to a Dire Team Advantage	Numerical, Continuous
pos[1-5]_diff	Difference in the 'last_gold' values between each position. i.e. 'pos1_diff', 'pos2_diff' Differences across positions (pos1_pos2_diff) were not evaluated. Positive values refers to a Radiant Team advantage. Negative refers to a Dire Team Advantage	Numerical, Continuous
rad_pos[1-5]_last_gold	Predicted/Given gold amount at the forecasted minute for the Radiant Team. Positive values refers to a Radiant Team advantage. Negative refers to a Dire Team Advantage	Numerical, Continuous
dire_pos[1-5]_last_gold	Predicted/Given gold amount at the forecasted minute for the Dire Team. Positive values refers to a Radiant Team advantage. Negative refers to a Dire Team Advantage	Numerical, Continuous
rad_pos[1-5]_embed_[1-84]	When an embedded model is used to forecast the 'last_gold' value, these features are added from the	Numerical, Continuous

	embedding vector weights trained in the LSTM model	
dire_pos[1-5]_embed_[1-84]	When an embedded model is used to forecast the 'last_gold' value, these features are added from the embedding vector weights trained in the LSTM model	Numerical, Continuous

Note that the feature size grows extremely large depending on the LSTM model used; requiring models resistant to high feature counts in the Random Forest and XGBoost models. The non-embedded models held 16 features. The models using an LSTM with 8 embedding features per hero held 96 features and models using an LSTM with 84 embeddings held the largest feature set at 856. Despite such a high feature count, the models performed well. For a baseline, the Random Forest and XGBoost models were tested on data using the actual value of 'last_gold' to compare the impacts of using the LSTM to predict the value. Performance metrics for the four resulting combinations between the 3 factors of model, hero_id embedding, and using predicted values over the real values are shown below:

6.2.1 Random Forest | No Embed | Real Last_Gold Value

Model	*Minute*	*Accuracy*	*Precision*	*Recall*	*F1 – Score*
RF	15	.74	.74	.74	.74
RF	20	.78	.78	.78	.78
RF	25	.78	.78	.78	.78
RF	30	.78	.78	.78	.78
RF	35	.76	.76	.76	.76
RF	40	.78	.79	.78	.78
RF	45	.67	.67	.67	.67

6.2.2 XGBoost | No Embed | Real Last_Gold Value

Model	*Minute*	*Accuracy*	*Precision*	*Recall*	*F1 – Score*
XG	15	.73	.73	.73	.73
XG	20	.76	.76	.76	.76
XG	25	.79	.79	.79	.79
XG	30	.75	.75	.75	.75
XG	35	.77	.77	.77	.77
XG	40	.74	.74	.74	.74
XG	45	.73	.74	.73	.74

Between the XGBoost and Random Forest models without embedding or predicted values, the Random Forest tended to out-perform at the 40 minute mark but lost drastically at the 45 minute mark.

6.2.3 Random Forest | Embedded | Real Last_Gold Value

Model	*Minute*	*Accuracy*	*Precision*	*Recall*	*F1 – Score*
RF	15	.70	.70	.70	.70
RF	20	.76	.76	.76	.76
RF	25	.77	.77	.77	.77
RF	30	.79	.79	.79	.79
RF	35	.79	.79	.79	.79
RF	40	.77	.80	.78	.77
RF	45	.69	.70	.69	.69

6.2.4 XGBoost | Embedded | Real Last_Gold Value

Model	*Minute*	*Accuracy*	*Precision*	*Recall*	*F1 - Score*
XG	15	.73	.73	.73	.73
XG	20	.75	.75	.75	.75
XG	25	.79	.79	.79	.79
XG	30	.77	.77	.77	.77
XG	35	.77	.77	.77	.77
XG	40	.75	.76	.75	.75
XG	45	.71	.73	.71	.71

The embedding vectors used to represent heroes and their relationship with gold showed a marginal increase in performance in the Random Forest model over the non-embedded values as well as the embedded XGBoost model. Overall, introducing the embedding vector representation of the heroes does not seem to make a significant difference in classifying wins when forecasting gold values 1-minute out scoring between the non-embedded models for actual gold values.

6.2.5 Random Forest | Embedded | Predicted Last_gold Value

'last_gold values' were predicted by the 10 Lookback 1 Horizon 8 Embed model

Model	*Minute*	*Accuracy*	*Precision*	*Recall*	*F1 - Score*
RF	15	.71	.71	.71	.71
RF	20	.74	.74	.74	.74
RF	25	.78	.78	.78	.78
RF	30	.80	.80	.80	.80
RF	35	.80	.80	.80	.80
RF	40	.78	.78	.78	.78
RF	45	.64	.64	.64	.64

6.2.6 XGBoost | Embedded | Predicted Last_gold Value

'last_gold' values were predicted by the 10 Lookback 1 Horizon 8 Embed model

Model	*Minute*	*Accuracy*	*Precision*	*Recall*	*F1 - Score*
XG	15	.71	.71	.71	.71
XG	20	.75	.75	.75	.75
XG	25	.78	.78	.78	.78
XG	30	.77	.77	.77	.77
XG	35	.77	.77	.77	.77
XG	40	.75	.76	.75	.75
XG	45	.71	.73	.71	.71

The same weakness at 45 minutes appears for the Random Forest model, but it does begin to perform quite well at the 30 and 35 minute marks. The XGBoost model remained relatively unchanged from introducing predicted values.

6.2.7 Random Forest | No Embed | Predicted Last_gold Value

'last_gold' values were predicted by the 10 Lookback 1 Horizon No Embed model

Model	*Minute*	*Accuracy*	*Precision*	*Recall*	*F1 - Score*
RF	15	.74	.74	.74	.74
RF	20	.77	.77	.77	.77
RF	25	.77	.77	.77	.77
RF	30	.78	.78	.78	.78
RF	35	.78	.78	.78	.78
RF	40	.79	.80	.79	.79
RF	45	.69	.70	.69	.69

6.2.8 XGBoost | No Embed | Predicted Last_gold Value

'last_gold' values were predicted by the 10 Lookback 1 Horizon No Embed model

Model	*Minute*	*Accuracy*	*Precision*	*Recall*	*F1 - Score*
XG	15	.78	.78	.78	.78
XG	20	.87	.87	.87	.87
XG	25	.80	.81	.80	.80
XG	30	.80	.80	.80	.80
XG	35	.82	.84	.82	.82
XG	40	.89	.90	.89	.89
XG	45	.71	.71	.71	.71

While the results of classifying a win were only slightly favoring an un-embedded model in the prior iterations, the combination of using an XGBoosted Classifier without embedding vectors to represent heroes on predictions of gold values by an LSTM model using a 10 lookback 1 Horizon, also lacking a hero embed, held quite robust results ranging from 80% - 89% accuracy at predicting games at different minute marks. While the expectation was that predicted values would only increase error, a stark improvement in performance is noted here.

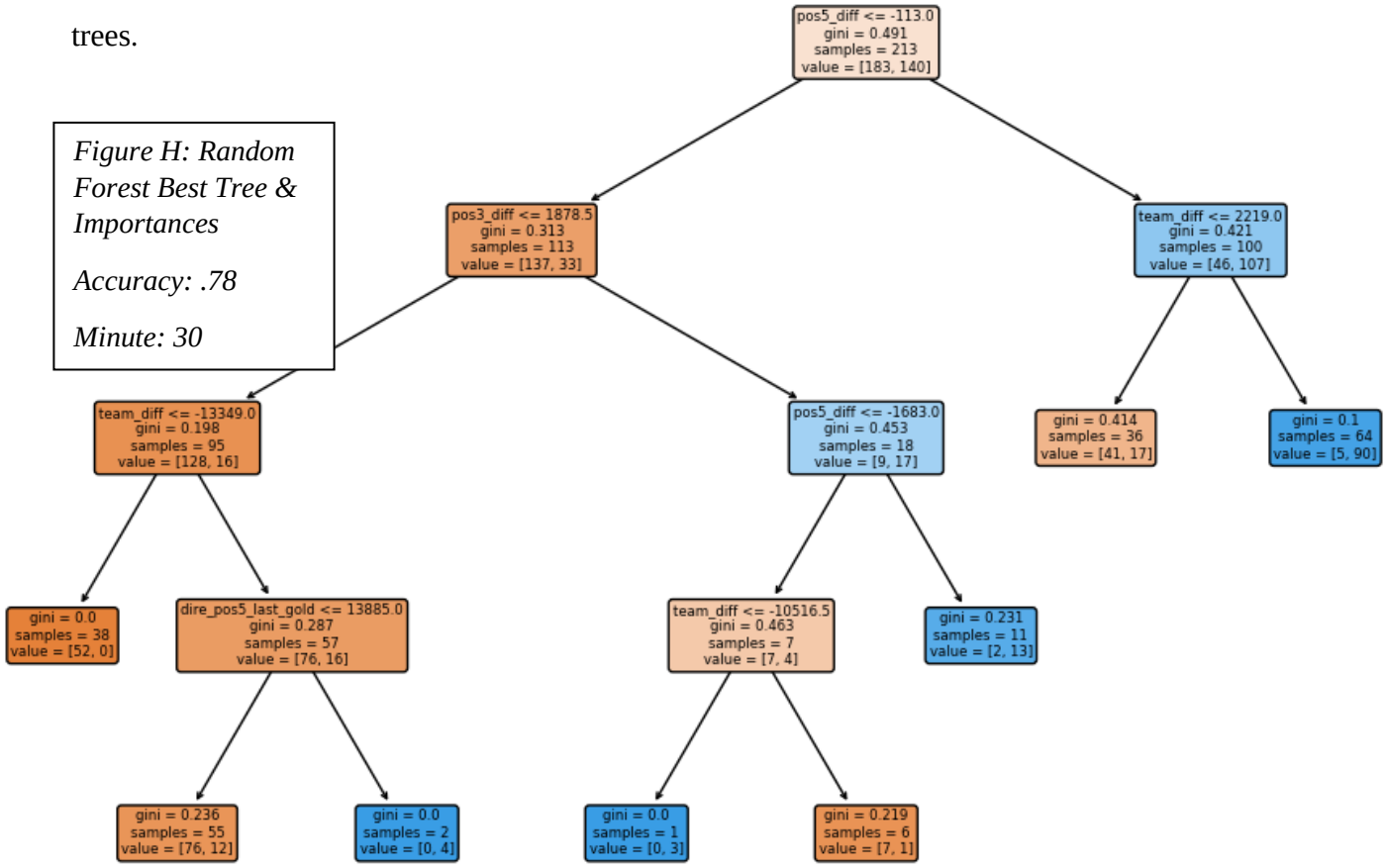
6.3 CLASSIFIER PLOTS

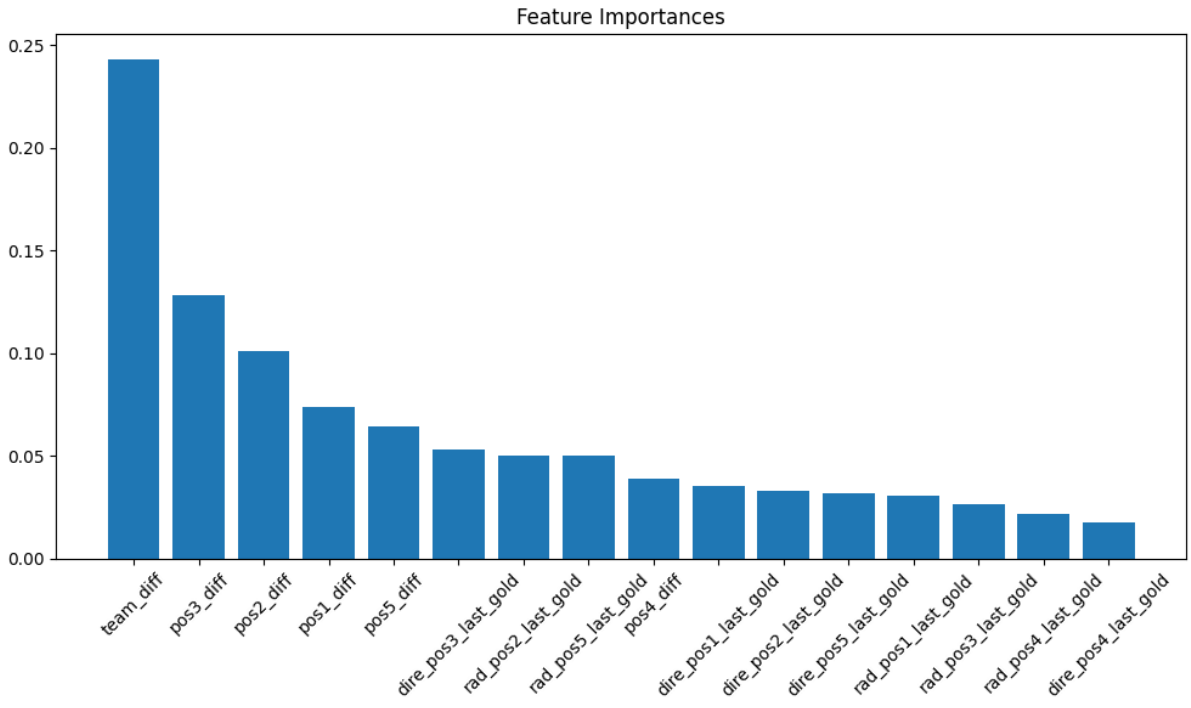
While a decision tree and feature importances cannot be shown for each minute predicted on, the following are examples of the highest performing models seen in 5.2.7 and 5.2.8. Following the arrows left path is indicative of a "true" evaluation of the rule in the node. Recall that these trees are taken as the best estimators and are still averaged across several hundred other trees.

Figure H: Random Forest Best Tree & Importances

Accuracy: .78

Minute: 30





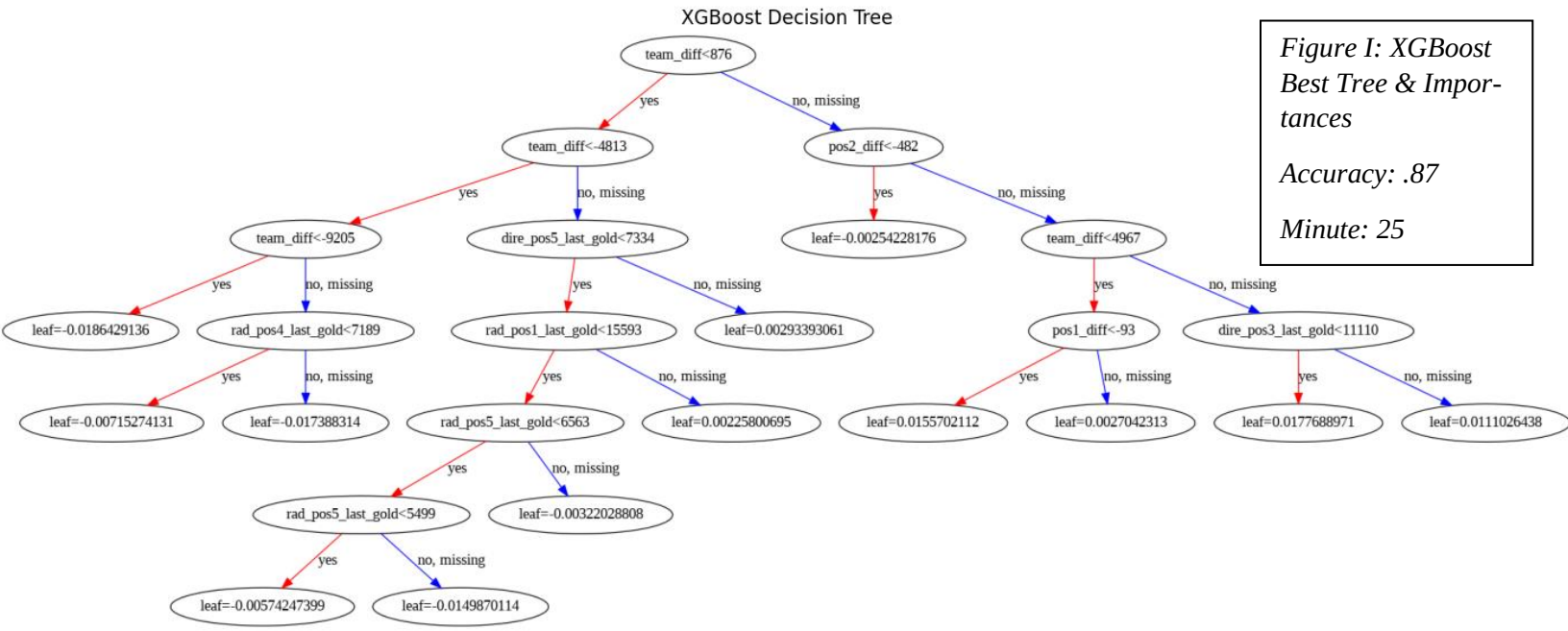
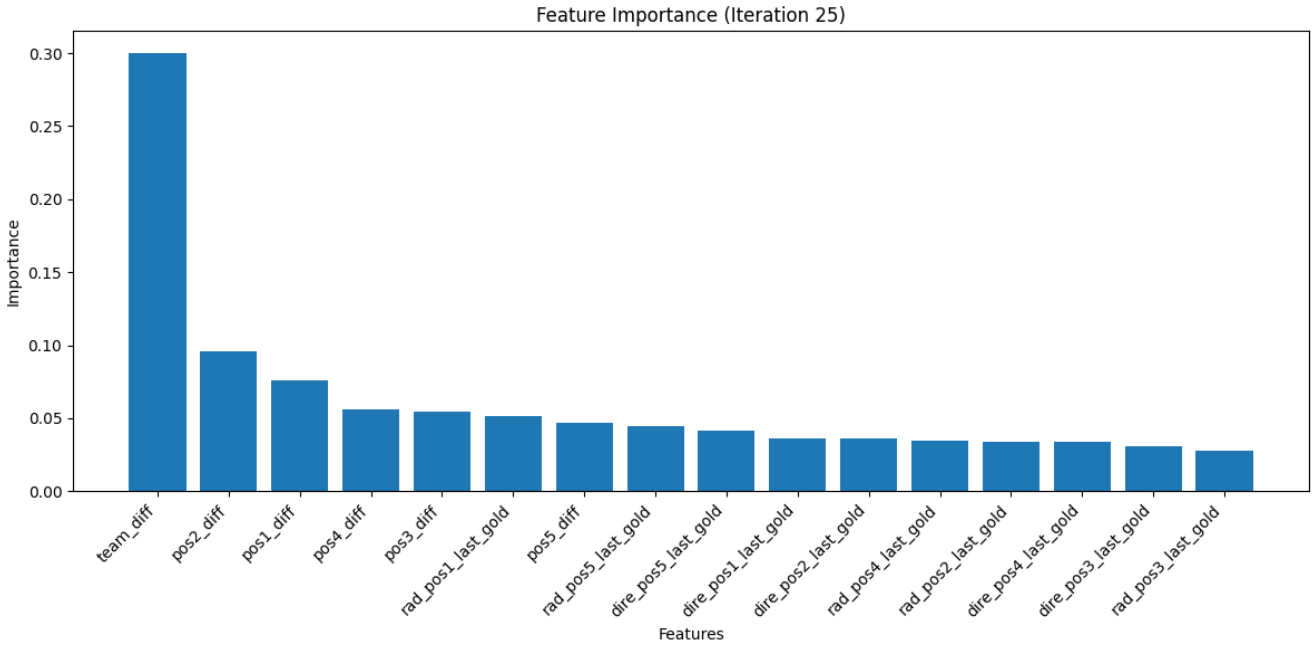


Figure I: XGBoost Best Tree & Importances

Accuracy: .87

Minute: 25



Deeper inspection of the decision trees at each minute mark reinforces the idea that different positions' gold values have differing levels of importance depending on the time of the game. Both trees value *team_diff* highly, with subsequent hero position differences in gold afterwards. Checking trees across minute marks shows that varying positional differences prove more important at different points in the game such as *pos3_diff, pos2_diff* being the most important in

the model designed for minute 30, while the *pos1_diff* variable is more important in the model for minute 25.

7 RESULTS

While the hypothesis of the paper was that superior predictions for use in live-settings would be achieved by including embedding vectors as representations of heroes to accompany their gold metrics, the results in this paper disagree. Adding the embeddings using an embedding layer during the training process for the LSTM either slightly decreased performance or achieved parity. An argument could be made that a lack of data or the techniques used in the research were unfit to accurately capture the impact of a hero chosen and forecasting their gold, but any relationships between hero and gold gain seem to be captured in the lookback steps of an LSTM. It is interesting to note that embeddings did seem to outperform non-embedded forecasts for horizons of 5, meaning there could be some use for this technique in comparing whether longer horizon scenarios instead of just 1 minute would improve win predictions.

This research does show the practicality and use of the openly available historic data from Dota 2 to be used in predicting live games. Should the database be updated, and accessible, during a professional match, simple gold values and hero id's can be pulled to generate predictions on the minute mark to update win likelihood. This is a significantly less intensive method compared to other research using Game State Integration apps for metrics that resulted in close accuracies.

To continue research, comparing Multi-Layer Perceptron neural nets and Covolutional Neural Nets to Random Forest and XGBoost models in determining a winning game may

show them to be more capable models. This research does show improved performance in using forecasted gold values as features rather than using at-time values for classifying a win. This can be see when comparing the tables 6.2.2 – 6.2.4 and 6.2.8 in the Results section Further research could explore using different lengths of forecast horizons, such as 5,8, and 10, to generate the forecasted gold feature for the classifying models. In this scenario, using hero embeddings did have a significant improvement in RMSE as compared to models that did not use a hero embedding. Data concerning spells cast and outcomes of team fights are available at the same source as the data in this research and could be used as more features. Using information concerning which heroes died, the game time, gold value change between teams, and location of the fight could lead toward predicting what objectives are taken by the winner of the team fight and further account for which team would win. Pairing this with a model that can predict gold values may boost accuracy. An increase in data, to include highly skilled players in the public matchmaking realm, may also increase predictive ability as this research only focused on professional matches played under Patch 7.35, roughly 10,000 games, compared to the hundred thousands of games available.

8 Model Parameters & Code Appendix

All code was executed using PyTorch to build and predict with the LSTM, scitkit learn's random forest package, and the XGBoost for the XGBoost Classifiers.

Google Colab Notebooks are available here: Google Drive Colab Notebooks

Copies are also available on Github: d-ev-craig Github

XGBoost/Random Forest Model Parameters

The Random Forest models were set to a limit of 100 estimators and a cost complexity parameter of .01.

A grid search across the following parameters was used to identify the strongest XGBoost Model:

```
param_grid = {
    'n_estimators': [200, 300],
    'max_depth':  [10],
    'learning_rate': [0.01 ,0.1],
    'subsample': [0.8 ,1.0],
    'colsample_bytree': [0.8 , 1.0],
    'gamma': [0, 0.1, 0.3],
    'min_child_weight': [1, 5, 10]
}
```

The resulting strongest model parameters at each minute mark is below:

Minute	*Col sample*	*Gamma*	*Learning rate*	*Max depth*	*Min child weight*	*Estimators*	*Sample*

15	0.8	0	0.1	10	1	300	0.8
20	1.0	0	.01	10	5	300	1.0
25	1.0	0.1	.01	10	5	200	1.0
30	0.8	0	.01	10	5	200	0.8
35	0.8	0	.01	10	5	200	0.8
40	1.0	0	.01	10	10	300	0.8
45	1.0	0	.01	10	5	200	0.8

9 CITATIONS & REFERENCES

[1] *DOTA 2 TI viewers 2019 | Statista*. (2019). Statista; Statista. https://www.statista.com/statistics/518201/dota2-championship-viewers/

[2] *DOTA 2 TI viewers 2019 | Statista*. (2019). Statista; Statista. https://www.statista.com/statistics/518201/dota2-championship-viewers/

[3] *Top eSports tournaments by prize pool 2022 | Statista.* (2022). https://www.statista.com/statistics/517940/leading-esports-tournamets-worldwide-by-prize-pool/#:~:text=The%20eSport%20tournament%20that%20has

[4]Djundik, P. (2013, July 9). *Dota 2 Steam Charts* [Review of *Dota 2 Steam Charts*]. Steam Charts - SteamDB. https://steamdb.info/app/570/charts/#1y

[5]Yu, L., Zhang, D., Chen, X., and Xie, X., "*MOBA-Slice: A Time Slice Based Evaluation Framework of Relative Advantage between Teams in MOBA Games*", arXiv e-prints, 2018. doi:10.48550/arXiv.1807.08360.

[6] Song, K., Zhang, T., & Ma, C. (2015). *Predicting the winning side of DotA2.* https://cs229.stanford.edu/proj2015/249_report.pdf

[7] Akhmedov, K. and Phan, A. H., "*Machine learning models for DOTA 2 outcomes prediction*", arXiv e-prints, 2021. doi:10.48550/arXiv.2106.01782.

[8] Yang, Y., Qin, T., and Lei, Y.-H., "*Real-time eSports Match Result Prediction*", arXiv e-prints, 2016. doi:10.48550/arXiv.1701.03162.

[9] V. J. Hodge, S. Devlin, N. Sephton, F. Block, P. I. Cowling and A. Drachen, "*Win Prediction in Multiplayer Esports: Live Professional Match Prediction*," in IEEE Transactions on Games, vol. 13, no. 4, pp. 368-379, Dec. 2021, doi: 10.1109/TG.2019.2948469.

[10] Dolphin, R. (2022, February 28). LSTM Networks | A detailed explanation | towards Data science. *Medium*. https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

[11] Mohammadi, M., Mundra, R., & Socher, R. (2015). CS 224D: Deep Learning for NLP. In *Lecture Notes: Part IV*. https://cs224d.stanford.edu/lecture_notes/LectureNotes4.pdf