SIADS 696 Milestone II Project Report
Predicting Dota 2 Match Results from Game Metrics and Chat Logs

Abhay Baliga (abhayb), Hugh Ding (dinghugh), Seth Fleming (sethflem)

Introduction

Dota 2 is a video game belonging to the multiplayer online battle arena (MOBA) genre. A single match involves two teams of five players who spawn in their respective bases at opposing corners of a predefined square map. The primary objective of a match is to deplete the health of the main structure (*Ancient*) in the opposing team's base. Each player controls a unique hero for the match, and uses that hero's abilities and attacks to accomplish objectives that aid their team in destroying the enemy Ancient. These objectives include killing enemy players, non-playable opposing enemies (*creeps*), neutral enemies (*jungle monsters*), and destroying enemy buildings including *towers* (interchangeably *turrets*) and *barracks* (interchangeably *inhibitors*).

Player-level metrics, such as heroes and player attitude (learned from in-game chat logs), were used to determine which methods an individual player should focus on when working towards victory. These insights aim to better inform players' in-game and communicative decision-making. We enjoy playing similar types of games, so we were interested in discovering the best data-informed strategies to help improve our and others' gameplay.

The unsupervised portion of this project included conducting a PCA on the selected features to reduce dimensionality, after which we then clustered the data using KMeans to identify different groups of heroes. The supervised portion included training regression, decision tree, and neural network models to predict if a game was won or lost based on the players' statistics. Feature importance was then extracted from the regression and decision tree models to indicate which objectives/goals were most crucial in locking down a win. Functions for both portions were primarily from the scikit-learn library [1].

In the unsupervised portion, we found that despite the significance players place on official and unofficial role distinctions, the clustering results demonstrated a surprising lack of clarity. Roles and play styles were found to be more fluid than anticipated, with heroes adapting to different roles based on team composition. Last hits on creeps emerged as a primary factor in differentiating clusters, while expected distinctions in kills and deaths were not as pronounced.

In the supervised portion, we found that building-related objectives (towers killed, barracks destroyed, and tower damage dealt) were by far the most important features for predicting victory. Removing building-related features caused model accuracy to drop substantially; removing other groups of features did not have the same effect.

Related Work

In this section, we reviewed similar studies' findings that bear significance to our investigation into MOBA games and machine learning. By examining prior work, we aimed to identify gaps, assess methodologies, and draw insights that informed the rationale and contribution of our project.

During our Milestone I exploratory data analysis, we investigated which factors lead to victory in two popular MOBA games, League of Legends (LoL) and Dota 2 [2]. In this project, we instead only discussed Dota 2 due to its richer available dataset. In addition, instead of finding correlations between wins and broad match-level metrics, we used player-level metrics, such as heroes played and player attitudes (learned from in-game chat logs), to determine which goals an individual player should focus on when working towards victory.

A similar analysis on Dota 2 from kaggle focuses on the EDA and simple predictions of team 1's win/loss and as well as individual player win/loss [3]. Our analysis went deeper, adding several features engineered from existing ones that were excluded in their project's analysis, such as chat logs, as well as delving into feature importance. Another project using LoL's data highlighted different types of supervised models to predict if a team will win or lose [4]. In our project, we took the extra step to optimize the tuning of the hyper-parameters to ensure that more accurate predictions were made.

Data Sources

The Dota 2 dataset was retrieved from kaggle and comes in the form of multiple csv files [5]. Seven files were used in this project, and altogether, contained features related to player chat logs, hero names, match logistics & outcomes, objective statistics, player information, and team statistics. The data consisted of 50,000 ranked matches from 2019. The data cleaning process is discussed in the *feature engineering* section.

Feature Engineering

See Appendix A for a full list of features.

Chat sentiment analysis.

The goal of this analysis was to provide an additional layer of insight into each match through the lens of communication. Players are able to type out messages either to their team or to everyone in the match. However, the data does not specify under which channel players communicate. They can choose not to communicate, or to communicate virtually as much as they'd like (spamming is met with a warning to stop and typing is temporarily taken away for a brief time).

Using the chat logs from in-game, we sought to answer a series of questions that would allow us to get a better picture of how players were communicating, and derived a set of relevant features. How many players participated in the in-game chat each match and on which team? How positive or negative were their messages? How long or short were their messages?

The chat file's relevant features were match id, message, and player number. The TextBlob polarity [6] provides a high-level assessment of sentiment, revealing whether the language used is predominantly positive, negative, or neutral. NLTK Vader [7] sentiment analysis offers a nuanced perspective, capturing the intensity and context of sentiments, thereby providing a more detailed understanding of emotional dynamics within the team. Word count serves as a straightforward metric, indicating the volume of text produced by a player, which can be indicative of their level of engagement and collaboration. Meanwhile, the average word length unveils the style and complexity of communication, distinguishing between more thoughtful and articulate exchanges and those that are more casual or succinct. By aggregating these features on a per-player basis and calculating their averages, the project aimed to offer a comprehensive view of communication patterns, allowing for the identification of consistent sentiments, communication styles, and potential impacts on gameplay outcomes. This holistic analysis contributes to a deeper understanding of the role of communication in Dota 2 matches and its implications for team dynamics.

Supervised.

This was largely adapted from our milestone 1 work, except for 1) the addition of chat metrics and 2) an adjustment to how we standardized values.

Initially, the match-related information, team objectives data, and players' chat metrics were loaded from csv files into a pandas table. The match data was then cleaned, filtering for ranked 5v5 matches with durations, and new columns were created to quantify destroyed towers and barracks for each team. The team objectives data was processed to extract specific objectives and determine which team achieved them, then aggregated based on match and team. All datasets were then merged into a cohesive dataframe, and relevant metrics were selected.

Values for time-dependent metrics were normalized to be per minute, and clear outliers were removed for certain fields. Finally, the data was standardized by centering values around 0 and scaling them to between -1 and 1.

We originally thought to include our unsupervised clusters as features for supervised modeling, but because we ended up having to perform unsupervised clustering separately on winners and losers, it no longer provided useful features for predicting wins.

Unsupervised.

Using the same feature engineering process as the supervised analysis, the normalized and centered player-level data was used. Hero ID was also included and then translated into hero name. A dictionary of heroes to their corresponding in-game attributes was manually compiled and merged into the player data. The hero names and attributes provided human-readable labels for visualizations of the clusters.

From this table, a hero-specific table was created, grouped by the hero name. Principal Component Analysis (PCA) was used to create three components to project the data on. The number of components was decided through manual inspection of eigenvalues.

Principal Components provide a method to reduce the dimensionality of the data to better perform other clustering methods such as KMeans or simply visualizing the data. Other dimensionality reduction methods do not preserve the integrity of the data as well as PCA does.

Supervised Learning

At this stage, our data was formatted, cleaned, and scaled. Our next step was to understand the relationships between features to help us choose suitable models.

Model considerations.

Feature representations.

- All predictive features are represented as continuous numeric values before they are passed into the model, which is intuitive and facilitates regression modeling.
    - As part of the feature engineering process, variables that change over the course of the game have been normalized by duration. All variables are scaled to between -1 and 1 and centered around 0 prior to modeling to discourage any one variable from skewing the results. There are no missing values.
- The output (winning) is a binary value of either 0 or 1.

Correlations.

We first augmented a correlation heatmap we had constructed for milestone 1 (figure 1). We found the following:

- Almost all of the features from the original dataset - that is, those representing in-game metrics and objectives - have a positive correlation to winning the game, as well as to each other.
- Death count instead has a clearly negative correlation with other in-game metrics and objectives.
- The two chat sentiment (polarity) features have a high correlation with each other, but very weak correlation with other features.
- Chattiness features (word count, word length) also have a high correlation with each other but weak correlation with other features, including chat sentiment.

This means our model would have to handle highly correlated features effectively.

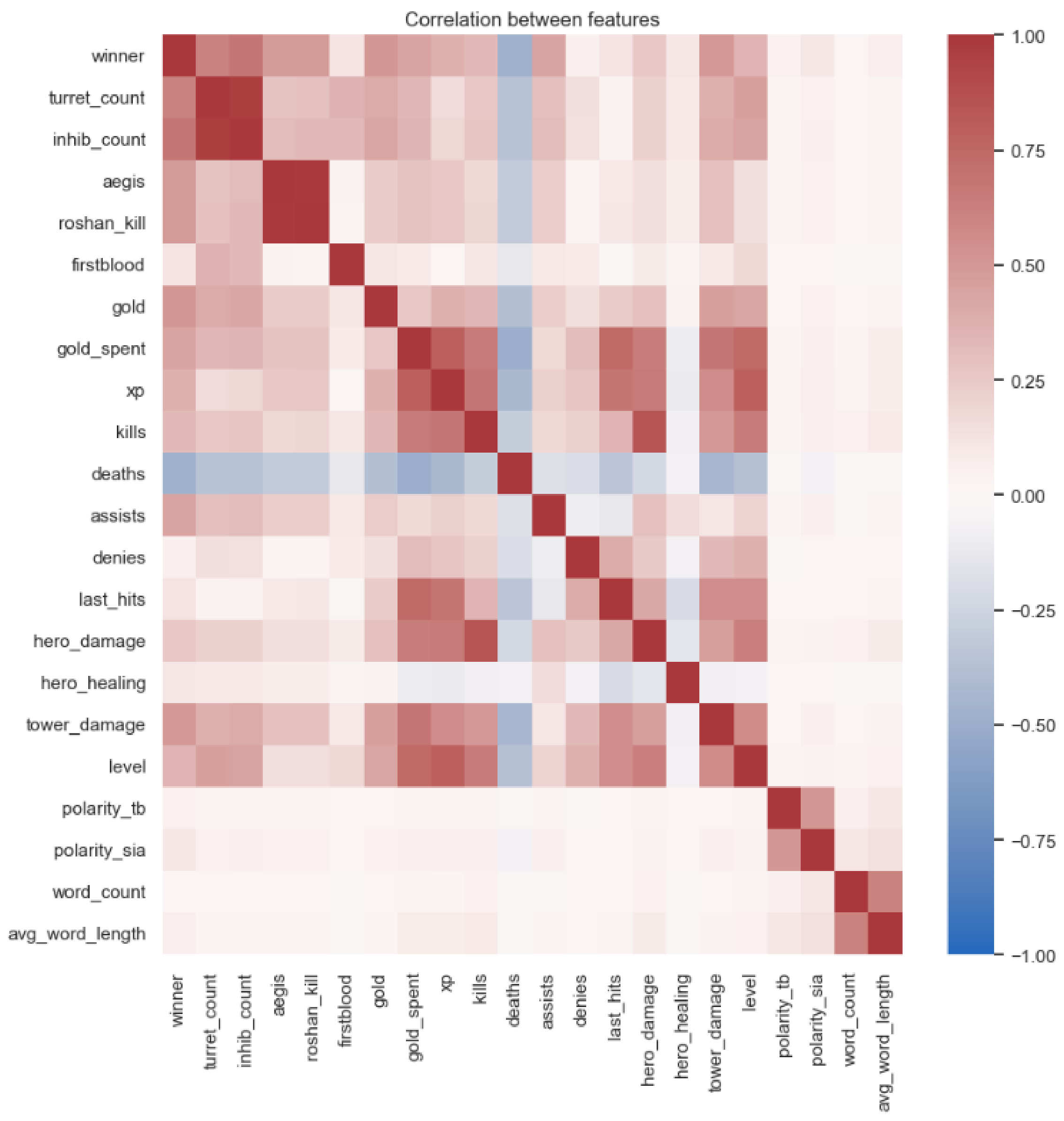


Figure 1: many variables are significantly correlated with each other.

Relationship to winning.

Since we were ultimately using our features to predict chances of winning, it was worth analyzing each predictor's relationship to winning. For example, linear regression would not accurately capture a quadratic relationship.

Intuitively (based on domain knowledge), we would expect the following:

- All of the original metrics should have a roughly linear relationship with winning, since they are objectives we aim to obtain during a match. When deciding whether to pursue an objective based on its cost/benefit analysis, the only cost is its opportunity cost.
    - For example, even though the correlation between winning and last hits is relatively weak, we expect that, all else held equal, getting more last hits would improve your win chances, since you are rewarded with gold; there are no scenarios where (again, all else held equal) you would rather

not get last hits.

- Developer statements would lead us to believe that chat polarity has a (positive) linear relationship with winning, but they are incentivized to claim this to discourage toxicity.

We then manually inspected differences in distributions using boxplots of each feature between winning and losing players to decide whether these assumptions were reasonable. (See *Data exploration -> Correlation to winning* in the *supervised.ipynb* appendix C notebook for corresponding visualizations.)

As expected, for the vast majority of the objectives-related features, the overlap in value between winners and losers was little to none, indicating a likely linear relationship. High chat polarity (positive sentiment) did indeed seem to increase linearly with win chances, although chattiness (e.g. measured by word count) did not.

Choice of models.

We now had two considerations:

1. Some features are highly correlated, and we don't have an easy way to decide which correlated features to exclude. This means we can rule out models like K-NN and Naive Bayes that assume otherwise.
2. Most of the features have a linear relationship to winning. This means we can assume that using highly complex models for non-linear relationships should not be necessary.

A third important consideration is that we wanted our models to be relatively *interpretable*. Our end goal wasn't simply to create a model that can retroactively predict winners of past games based on in-game metrics; it was to inform players about which metrics they should target in-game. Our set of features cannot realistically cover every in-game scenario, and players would need to make decisions on the fly (rather than pausing the game to run the model each time) based on their understanding of our analysis. This ruled out some of the more powerful techniques, like neural networks and ensemble learning.

With these considerations in mind, the following models were both implemented and analyzed:
- Lasso regression (specifically logistic lasso regression, since winning is binary)
- Decision tree (this isn't explicitly for linear relations, but can be used for them)

For benchmarking purposes, we also trained a neural network (MLP classifier) as an example of a powerful model that can better capture complex relationships. However, we did not analyze it further due to its lack of interpretability.

Modeling process.

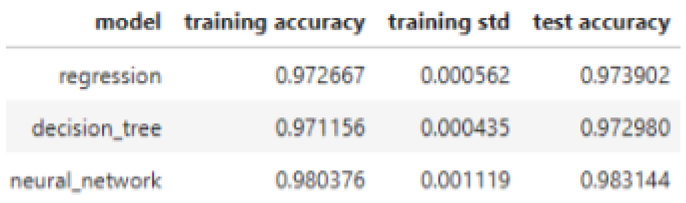
After setting a random state of 42 for reproducibility, and then splitting the data into training and testing, we conducted a 5-fold grid search for parameter tuning. We tried to iterate over all parameters that were important for model performance (based on lectures and scikit function documentation). For numeric parameters, we chose values on a roughly exponential scale to capture a wide range of possible behaviours.

Regression parameters:
```
  "C": [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100]
  "tol": [0.00001, 0.0001, 0.001, 0.01]
```
Tree parameters:
```
  "max_depth": [None, 1, 2, 5, 10, 50]
  "min_samples_leaf": [1, 2, 5, 10, 50]
  "max_leaf_nodes": [None, 2, 10, 50]
  "max_features": ["sqrt", "log2", None]
```
Neural network parameters:
```
  "hidden_layer_sizes": [(50, 50), (100,)]
  "alpha": [0.001, 0.01]
  "activation": ["identity", "logistic", "tanh", "relu"]
```

Results.

See Appendix B for a full list of regression coefficients and decision tree feature importances using the best performing parameters.

Model evaluation results.



model	training accuracy	training std	test accuracy
regression	0.972667	0.000562	0.973902
decision_tree	0.971156	0.000435	0.972980
neural_network	0.980376	0.001119	0.983144

Figure 2: all three models are fairly equal.

Both the regression and decision tree performed nearly equally well, with around 97% accuracy.

The neural network performed slightly better, with around 98% accuracy. This possibly indicates that some variable relationships are not entirely linear. Another possibility is that a complex model like a neural network is better able to capture more complex relationships between features, since we saw that many features are closely correlated. However, the performance increase is small, which seems to indicate that our assumption about linearity was fairly well-founded.

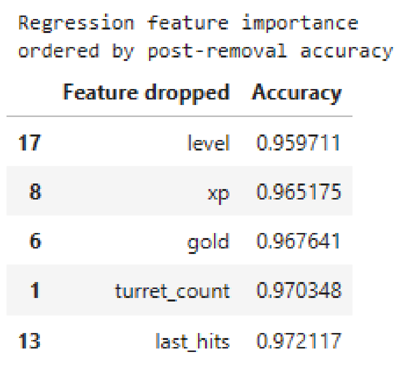
Evaluation metric.

We explicitly chose to evaluate our models on accuracy because we have perfectly balanced data and care equally about false positives and false negatives. With this in mind:

- Precision wouldn't be ideal because it fails to consider false negatives.
- Recall wouldn't be ideal because it fails to consider false positives.
- Other metrics like F1 score and AUC could be justifiable, but there are no substantial benefits to using these more complex and less intuitive metrics over accuracy.

Feature importance and ablation.

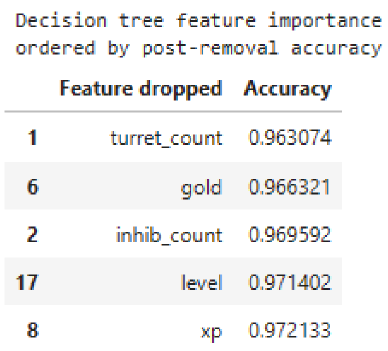
Using the best-performing parameters from our grid search, we performed feature importance testing on the regression and decision tree models by looping through each feature and re-calculating model accuracy with that feature removed.



```
Regression feature importance
ordered by post-removal accuracy
```

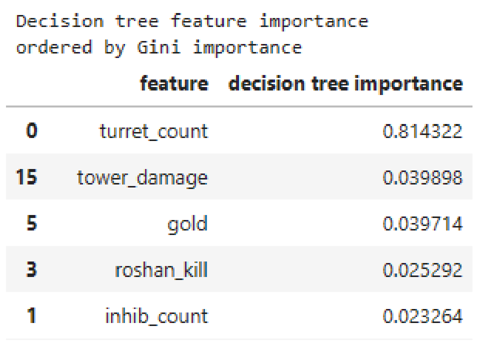
	Feature dropped	Accuracy
17	level	0.959711
8	xp	0.965175
6	gold	0.967641
1	turret_count	0.970348
13	last_hits	0.972117

Figure 3: based on the top 5 features for the regression model, none of the features are individually important.



```
Decision tree feature importance
ordered by post-removal accuracy
```

	Feature dropped	Accuracy
1	turret_count	0.963074
6	gold	0.966321
2	inhib_count	0.969592
17	level	0.971402
8	xp	0.972133



```
Decision tree feature importance
ordered by Gini importance
```

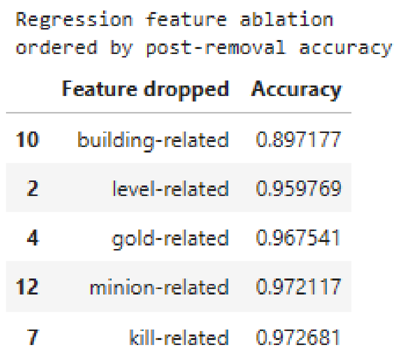
	feature	decision tree importance
0	turret_count	0.814322
15	tower_damage	0.039898
5	gold	0.039714
3	roshan_kill	0.025292
1	inhib_count	0.023264

Figure 4: turret count is much more important than other features based on Gini importance, but not feature removal.

For both models, removing any one feature did not substantially affect the model's original accuracy of ~97%. This confirms that our features are highly correlated, since the information lost from the removal of any one feature could easily be made up for with a combination of other features.

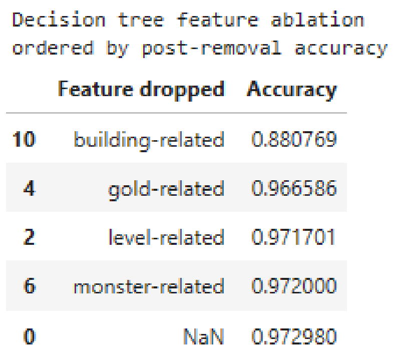
For the decision tree, this notably contrasted with the feature_importances_ attribute (described in figure 4, as well as Appendix B), which considered turret count to be an order of magnitude more important than every other feature. This perhaps indicates that while turret count was very important for that particular split, other features can also be split upon to produce comparable results.

Following this, we performed feature ablation by grouping together features that, through domain knowledge, we determined to be similar. For example, one group consisted of features directly representing xp gained (levels, xp), while another group consisted of features directly representing contribution to kill count (firstblood, kills, assists).



```
Regression feature ablation
ordered by post-removal accuracy
```

	Feature dropped	Accuracy
10	building-related	0.897177
2	level-related	0.959769
4	gold-related	0.967541
12	minion-related	0.972117
7	kill-related	0.972681



```
Decision tree feature ablation
ordered by post-removal accuracy
```

	Feature dropped	Accuracy
10	building-related	0.880769
4	gold-related	0.966586
2	level-related	0.971701
6	monster-related	0.972000
0	NaN	0.972980

Figure 5: building-related features are the most important for model accuracy.

Feature ablation made it clear that building-related metrics (towers killed, tower damage, and inhibitors killed) were by far the most important, with their removal decreasing the accuracy of both models to below 90%. Other groups of metrics continued to have minimal impact on model accuracy.

The key takeaway from this is that **building-related metrics like towers killed are by far the most important features for predicting wins**.

(See *Model analysis -> feature importance* and *Model analysis -> feature ablation* in the *supervised.ipynb* appendix C notebook for the full tables.)

Parameter tuning sensitivity.

We performed parameter tuning for both the regression and decision tree models. For each parameter that our grid search used, we plotted model performance across all searched values for that parameter; other parameters were held constant to their best-performing values.

We found that the models generally weren't that sensitive to individual parameters. For example, figure 6 describes the decision tree parameter tuning for maximum depth. While maximum depth increases roughly exponentially on the x-axis, mean accuracy only slightly increases on the y-axis.

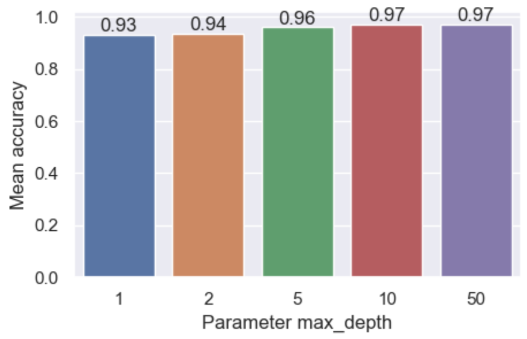


Figure 6: max_depth parameter tuning has little effect on decision tree model accuracy.

This makes us confident that our grid search was able to obtain model parameters that were quite close to optimal model performance.

See *Data analysis -> Parameter tuning sensitivity* in the *supervised.ipynb* appendix C notebook for visualizations on the sensitivity of each parameter.

Modeling decision trade-offs.

Given the needs of our analysis, our choice of models and evaluation metric (accuracy) are fairly clear-cut.

The main decisions we made that involved substantial trade-offs were in our parameter tuning choices. For numeric parameters, we manually selected values on a roughly exponential scale. We intentionally kept our parameter tuning quite sparse for two reasons:

1. To improve the running time of the models.
2. To limit overfitting. Ultimately, our dataset is somewhat small and only spans a time period of two weeks - it wouldn't be worth carefully tuning our models on dataset patterns that may not be generalizable.

The downside to conducting our grid search on a less granular level would be that the models wouldn't be as effective at predicting wins for our dataset specifically. However, the models ended up being reasonably accurate anyways (over 97%), and didn't seem to be too sensitive to parameter tuning, meaning the downside was fairly minimal in our case.

Checking mispredictions.

False positives and false negatives were fairly balanced for both models, as we would expect from using accuracy as the evaluation metric:

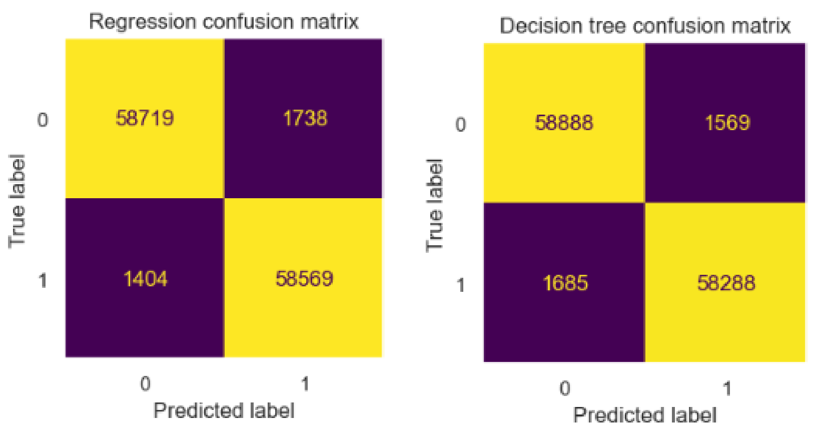


Figure 7: the types of errors are fairly well-balanced in both models.

In addition, the summary statistics of our incorrect predictions indicate that the mean values of the features are still very close to 0 (which is what we had centered them on).

Finally, we manually inspected three instances of incorrect predictions for both the regression and decision tree models (using the pre-normalized/centered/scaled values):

match_id	team	player_slot	hero_id	duration	winner	turret_count	inhib_count
21051	blue	0.0	106.0	3719.0	False	3.0	4.0
21051	red	130.0	42.0	3719.0	True	0.0	0.0

Figure 8: red team wins despite not destroying any turrets.

In match ID 21051, we see that the blue team destroyed three turrets and four inhibitors to the red team's 0. It is impossible to win a typical game without destroying at least three turrets and one inhibitor. This means that the blue team intentionally surrendered a game they had an advantage in, perhaps due to sportsmanship.

match_id	team	player_slot	winner	gold_spent	kills	deaths	hero_damage
22058	red	128.0	False	17160.0	3.0	3.0	7299.0
22058	red	129.0	False	22425.0	19.0	7.0	25714.0
22058	red	130.0	False	17465.0	12.0	5.0	16510.0
22058	red	131.0	False	13525.0	4.0	6.0	9536.0
22058	red	132.0	False	9510.0	6.0	11.0	10096.0

Figure 9: the strongest player is unable to carry their team.

In match ID 22058, player_slot 129 on the red team was predicted to win but instead lost. The prediction was reasonable given that they dominated the stats in terms of gold spent, kills, and hero damage. However, the rest of their team wasn't nearly as capable, and player_slot 0 was able to carry the blue team.

match_id	team	player_slot	winner	gold_spent	kills	deaths	hero_damage
39359	blue	0.0	True	1525.0	0.0	1.0	726.0
39359	blue	1.0	True	26550.0	9.0	2.0	16099.0
39359	blue	2.0	True	15405.0	7.0	2.0	11504.0
39359	blue	3.0	True	24750.0	11.0	7.0	14201.0
39359	blue	4.0	True	11895.0	4.0	10.0	4977.0

Figure 10: a player doesn't contribute to the game and is carried to victory.

In match ID 39359, player_slot 0 on the blue team is predicted to lose; given that they spend almost no gold and do very little damage, they may have disconnected from the game. However, the rest of their team seemingly manages to win a 4v5.

It feels reasonable that our models mispredicted these rows - for us, they would have passed the eyeball test as well (given the same amount of information, we would have predicted the same incorrect result that the model did). Since Dota is a widely accessible video game, players will occasionally act inconsistently and irrationally, which is not something we could predict from in-game statistics alone.

One thing we do notice is that some of the mispredictions are due to players performing very differently from the rest of the team. We could include more team-wide metrics to improve the accuracy of the model prediction. However, this is somewhat counter to the purpose of the analysis, which is to inform players what metrics to focus on, since a player can't control the actions of their teammates.

This gives us a good idea that it'd likely be more helpful to focus on our data quality and richness rather than our models to improve our analysis.

Unsupervised Learning

The goal of the unsupervised learning portion was to build clusters to attempt to identify unique playing styles or roles within the game. With these clusters identified, we could investigate whether there were more effective playing styles over others.

Methods description.

To find labeled clusters, the results of PCA clustering were fed into K-Means clustering methods. PCA was used to reduce dimensionality, and K-Means would then find clusters in the lower dimensional space. PCA retained the "shape" of the data, and K-Means was used because the clusters found were not in a complex shape while still providing insight into different playing styles.

As clustering can be a subjective endeavor, manual inspections of visualizations were the primary method of evaluating hyperparameters.

The alpha parameter and number of components were determined through visual exploration. A relatively low alpha of 0.1 was chosen. The results were not particularly sensitive to the alpha, but a lower alpha resulted in slightly more variance in the visualization, which was desired for ease of evaluation.

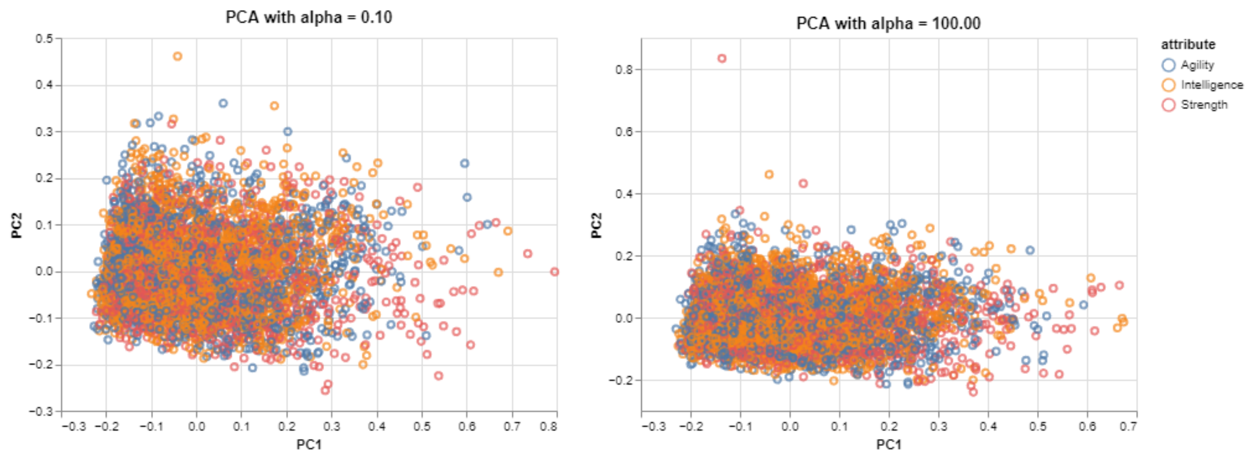


Figure 11: Tuning the alpha parameter for PCA fails to create clearly defined clusters.

With both hero level and player level data, the best kernel function found was the cosine. The cosine function was able to find two groupings of players at the hero level, labeled by the in-game attribute. However, one of the primary differences in the two clusters was due to hero winners and losers, which it was able to determine even after dropping the "winner" feature from the analysis.

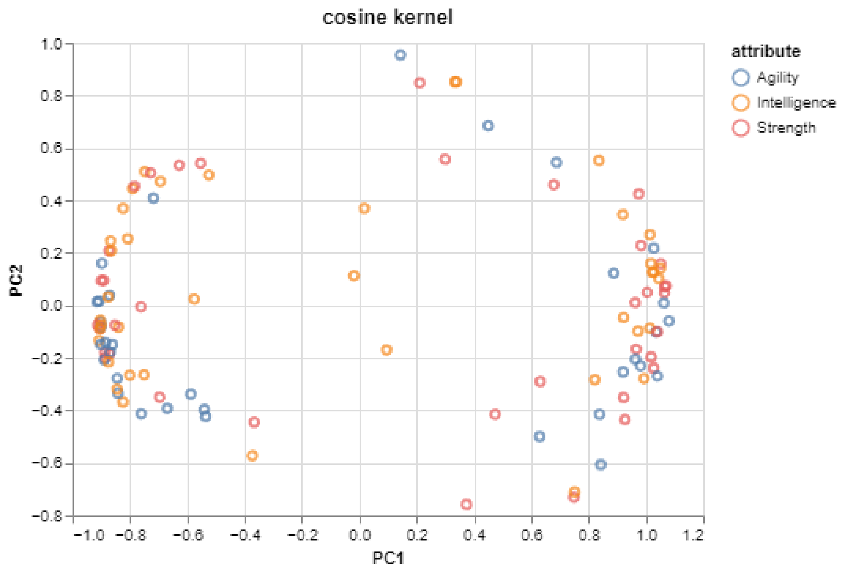


Figure 12: When tuning across different kernels, the cosine kernel seems to produce the clearest clusters by manual inspection.

When the two obvious clusters were investigated through K-Means, the last_hit and tower_damage features were found to be the differences in the groups given to the PCA, with 10% and 5% deviations from the mean, respectively. However, the winner feature also differed by 3%. When the K-Means clustering of 2 was applied to the player level data, the difference in win percentage surpassed 15%. Adding more clusters simply found players with different win percentages, but did not appear to describe different playstyles. The differences in other features were minimal and centered around the mean in both analyses. Instead of finding playing styles, the clustering was finding skilled versus unskilled players, which meant that during our analysis going forward, the data was separated into winning groups and losing groups.

The number of components to use was determined by eigenvalues. If too many components were used, the eigenvectors became too similar to the original metrics. The eigenvalues started dropping significantly after the first component, from 344.65 in the first, to 111 in the second, to 56.54 in the third, and then to 28.28 in the fourth, so a decision was made to cut off at three components.

After the PCA model was used to transform the data, K-Means clustering was applied, with 7 clusters arrived at through calculating the silhouette score, as well as manual inspection.

Evaluation.

The number of clusters was difficult to decide upon. Even with dimensionality reduction and after reducing to player-level features, clustering resulted in very noisy data. Evaluation was done via Silhouette Score on the clusters found. Silhouette score was graphed along increasing clusters, but there was no clear indication that a specific number of clusters was the most appropriate.

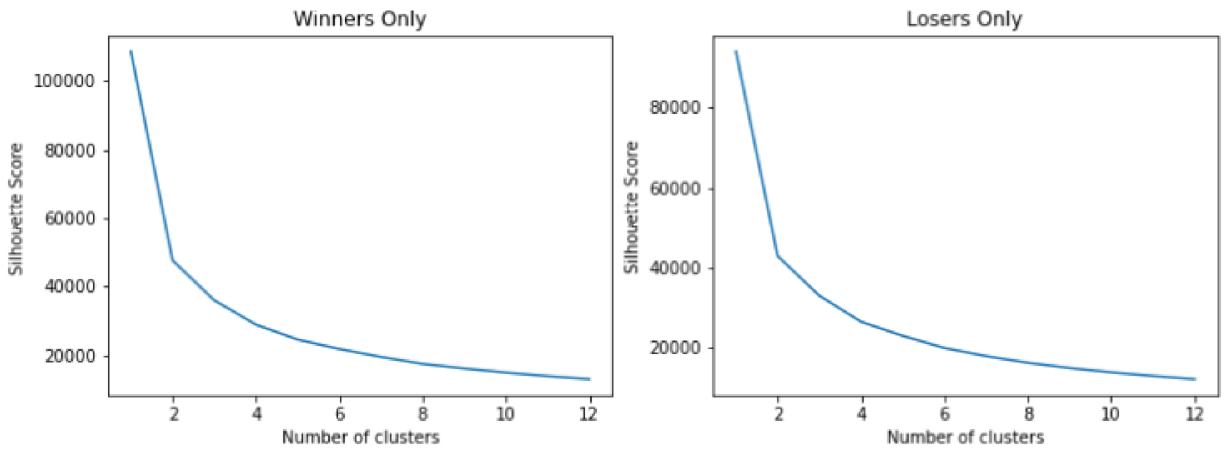


Figure 14: Silhouette score fails to define clear clusters.

Any number of clusters between 2 and 9 may have been appropriate, but 7 was chosen. Due to the clusters being so difficult to interpret, hierarchical clustering was also attempted to determine whether a different number of clusters could be appropriate.

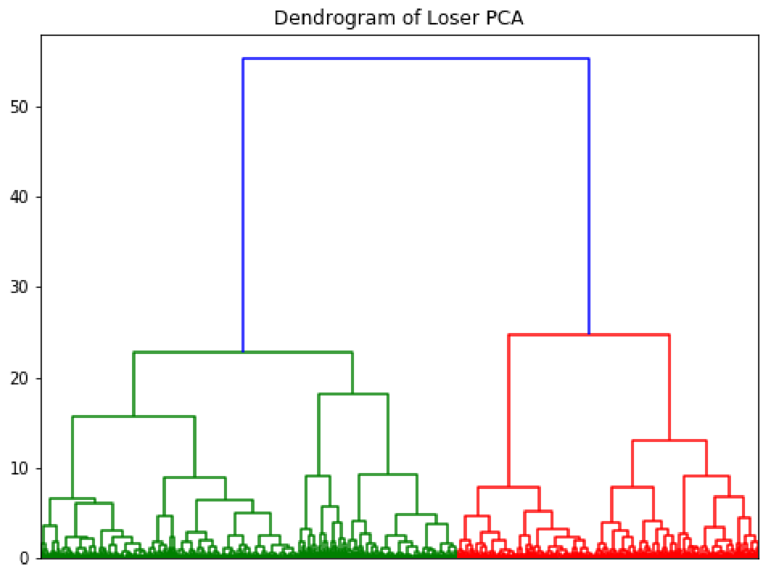


Figure 15: hierarchical clustering does not provide a clear number of clusters.

The dendrogram also resulted in clusters that were not straightforwardly interpretable. Using seven clusters created groupings where the silhouette score dropped, but the closeness score on the dendrogram was not too close.

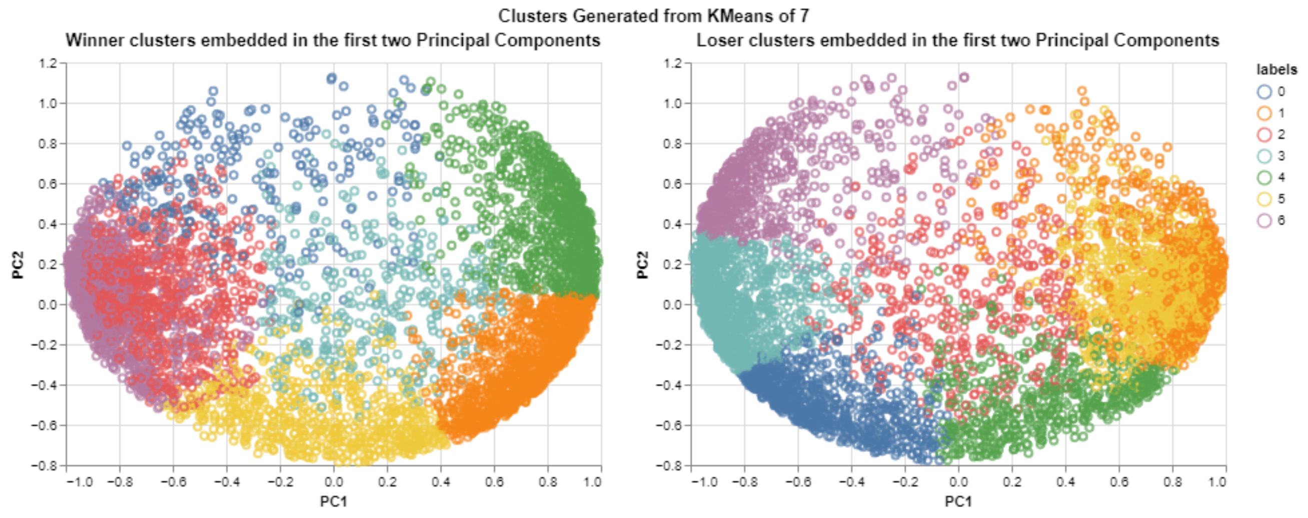


Figure 13: clusters generated by PC1 and 2.

The clusters were somewhat more of a spectrum rather than straight-forward differences, but 7 clusters still resulted in finding differences in playstyles. From previous tuning of the alpha hyperparameter in figure 12, we knew the results were not very sensitive to that. We also knew from our methods evaluation that if team level features such as number of towers left were kept in, the groupings found were winners and losers, so the analysis was sensitive to team-level features, but not as much to the player-level features, such as damage done to towers. Even reducing features to simply Kills, Deaths and Assists did not result in clear clusters. This can be found in the *Player PCA* section of the *dota_clusters.ipynb* notebook in Appendix C.

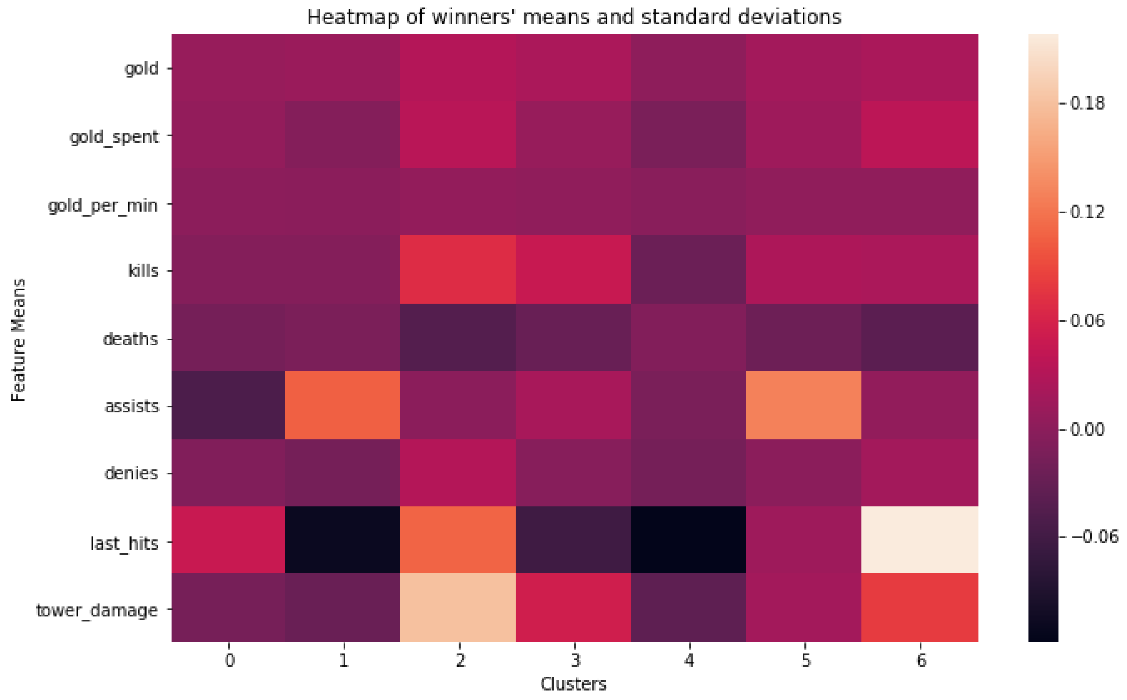


Figure 16: K-means produces the most noticeable clusters along last hits and tower damage.

The clusters mainly differed on last hits on the "creeps", which are non-player creatures that mindlessly attack towers but are generally fairly weak; players will kill them as a method to gain gold and defend their towers. We had expected greater differences in kills and deaths. Tower damage also was different in the clusters, and that can be attributed to roles within the game, but there is not enough difference to tell the roles from that alone. A similar heatmap of losers can be found in the *Heatmaps* section of the *dota_clusters.ipynb* notebook in Appendix C.

Discussion

Supervised results.

Our intuition stated that winning had a generally linear relationship with the most important features for predicting wins. This was mostly confirmed by the strong performance of the regression model, which had comparable accuracy to the decision tree (a simple non-linear model), and only slightly lower accuracy than the neural network (a complex non-linear model). Chat polarity and other features outside of objectives provided little insight into win chances, as we would expect.

What surprised us was just how much both the regression and decision tree emphasized towers destroyed compared to every other metric. When discussing whether a team is ahead, the typically discussed metrics are kills and gold; while these did have a positive correlation with winning, they were dwarfed in terms of feature importance by tower-related objectives.

The main challenge for the supervised analysis arose from a lack of computational power. Our data had 481,720 rows and 21 predictive features, which was large enough to make running an exhaustive grid search for our models on our personal machines difficult. We instead ran our script through the more powerful Great Lakes computing environment. We also reduced the granularity of our grid search; as described earlier, this ended up not being too detrimental, as our models had low parameter sensitivity and were fairly accurate anyways.

Unsupervised Results.

Roles are given both official and unofficial names. In practice, players will place great importance on them, often picking and banning heroes by available roles, so it was very surprising that these clusters were not obvious. If those were not how players were clustered, then we could have expected clusters around players at different skill levels. Instead, the clusters merged into one another in a painfully unclear manner. While playing differences could certainly be found, roles and playstyles were not set in stone. A hero will fulfill different roles depending on their team before the match starts, but it seems that even within a game, the role may be more fluid than we had envisioned. This may be because there are at least 9 official roles and only 5 players on the team. The roles overlap in some manner, and even our unsupervised learning methods have problems distangling them.

Some challenges came from the size of the data. Especially with unsupervised methods, the data has to be loaded into accessible memory, and our local machines were unable to run either PCA or KMeans on the full dataset. While writing and testing the code, it was run on a small sample of about 10000 players. We took advantage of the Great Lakes computing cluster to run our code on the full data. The full player data did not yield significantly different results than the sample.

The primary challenge with the unsupervised methods used was the difficulty in interpretation of clusters. The clusters were rarely clean, and when investigated, the differences were not obvious, as seen in earlier figures.

Next steps.

Supervised learning.

The most important addition for this project to produce actionable information would be to try to derive *causal analysis* from it. Our end goal is to inform a player which objective they should focus on to maximize their chances of winning. So far, we have only demonstrated that turret count is the strongest predictor of winning; we want to know the degree to which a player *choosing* to destroy a turret would increase their chances of winning.

We had planned to make causal inferences as part of this project through instrumental variables, but unfortunately couldn't find a suitable IV. We had intended to use hero win rates, which are typically balanced around 50%, but found that our dataset became somewhat imbalanced at the hero level (for example, the top 5 best heroes at destroying turrets all had win rates well over 50%) - this was likely due to the small time frame of around two weeks from which our data was gathered.

Some other potential areas of improvement to 1) make attempts at causal inference more credible and 2) allow our models to be more broadly insightful include:

- Increasing our sample size, especially with more up-to-date data. Our current data source was from several seasons ago, which would introduce concerns of covariate shift if applied to today's metagame.
- Adding new features, such as regional data (for investigating geographic trends) or player skill level.
- Adjusting all features to be player-level; right now, a few of our features are instead team-level (like turret count) due to dataset limitations, which means not every feature is treated equally.
- Engineering more features that could further improve our accuracy. While the features we tried to engineer for this project related to chat ended up not having a meaningful effect on our models, there are other avenues, such as contribution to team composition, that we haven't yet explored.
- Exploring player and match statistics at a more granular level, such as analyzing different points in time during a match, or seeing which objectives tend to follow others (e.g. turret damage after hero kills), rather than just looking at post-game statistics.

Unsupervised learning.

Like the supervised learning portion, adding or engineering new features, as well as working with more granular data, would help to improve our analysis. This would benefit us primarily in investigating how known hero roles play differently, such as how their statistics change over the course of the game. Since these roles are meaningful according to both official sources and domain knowledge, we expect there to be approaches from which we could find meaningful differences that weren't available to us given our dataset.

Sample size did not appear to be a major bottleneck for our unsupervised analysis.

Ethical Considerations

In the supervised portion of the project, the outcome of the game was predicted from simple game statistics and chat logs. The chat logs contain only messages and usernames which are not traceable to the actual player. Therefore, the outcome and the security of the data itself is not a concern.

However, in the context of treating players fairly, ethical concerns could emerge in the unsupervised portion of the project from the interpretation of clusters and the potential for stigmatization based on hero choices or playstyles. To mitigate these issues, it is essential to approach clustering results with caution, avoiding deterministic statements about player skills and preferences. Emphasizing the neutral nature of clusters as statistical patterns rather than judgments on player capabilities can help prevent reinforcing stereotypes. By presenting clustering results in a non-judgmental manner and respecting player diversity, ethical considerations can be prioritized throughout the unsupervised learning phase of the project.

Statement of Work

Abhay - Unsupervised Lead + report writing
Hugh - Supervised Lead + report writing
Seth - Chat Sentiment Analysis Lead + report writing

References

[1] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.

[2] Baliga, A., Ding, H., Fleming, S. (2023). Differences in MOBA Conditions.

[3] Winarso, Wega. (2023). Dota 2 Matches. Retrieved September 26, 2023 from https://www.kaggle.com/code/wegawinarso/chat-and-heroes-eda-win-predictions

[4] Acar, Ahmet. (2020). (LoL) League of Legends Ranked Games. Retrieved September 26, 2023 from https://www.kaggle.com/code/acarahmet/league-of-legends

[5] Anzelmo, Devin. (2019). Explore Player Behavior and Predict Match Outcomes. Retrieved September 26, 2023 from https://www.kaggle.com/datasets/devinanzelmo/dota-2-matches

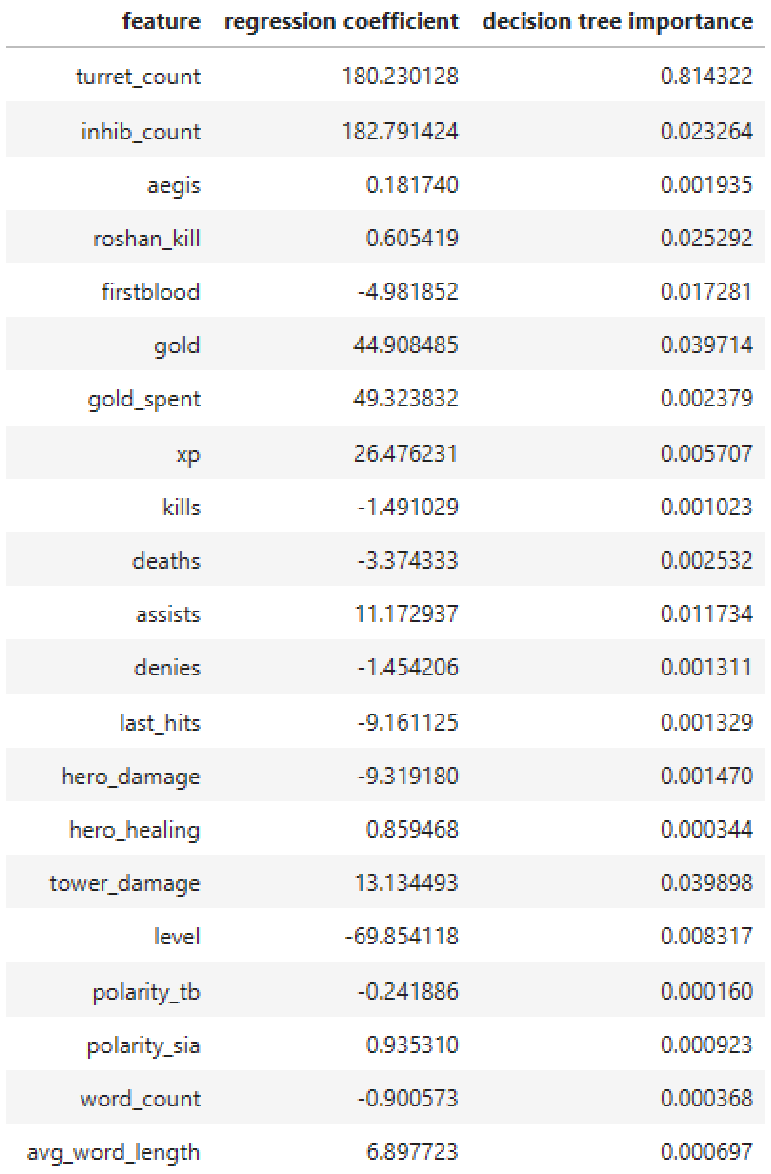
[6] Loria, Steven et. al. (2020). Sentiment Analysis. Retrieved September 26, 2023 from https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis

[7] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

Appendix A - Feature List

Feature	Description	Data Type
match_id	Unique match id number	int
team	Blue or red team	string
player_slot	Player number within game	int
hero_id	Unique hero number	int
duration	Game duration in seconds	int
winner	True or false	string
turret_count	Turrets destroyed	float
inhib_count	Inhibitors destroyed	float
aegis	Aegis obtained	float
roshan_kill	Roshan killed	float
firstblood	Player with the first kill	float
gold	Gold Earned	float
gold_spent	Gold used to purchase items	float
xp	Experience earned	float
kills	Kills	float
deaths	Deaths	float
assists	Assists	float
denies	Denies	float
last_hits	Creeps killed	float
hero_damage	Damage to enemy heroes	float
hero_healing	Healing to allies	float
tower_damage	Damage to enemy towers	float
level	Level at end of game	float
polarity_tb	Average chat polarity	float
polarity_sia	Average chat intensity	float
word_count	Average chat word count	float
averge_word_length	Average chat word length	float

Appendix B - Supervised regression coefficients vs decision tree feature importances



feature	regression coefficient	decision tree importance
turret_count	180.230128	0.814322
inhib_count	182.791424	0.023264
aegis	0.181740	0.001935
roshan_kill	0.605419	0.025292
firstblood	-4.981852	0.017281
gold	44.908485	0.039714
gold_spent	49.323832	0.002379
xp	26.476231	0.005707
kills	-1.491029	0.001023
deaths	-3.374333	0.002532
assists	11.172937	0.011734
denies	-1.454206	0.001311
last_hits	-9.161125	0.001329
hero_damage	-9.319180	0.001470
hero_healing	0.859468	0.000344
tower_damage	13.134493	0.039898
level	-69.854118	0.008317
polarity_tb	-0.241886	0.000160
polarity_sia	0.935310	0.000923
word_count	-0.900573	0.000368
avg_word_length	6.897723	0.000697

The regression uses a mixture of features, while the decision tree largely splits on turret count. Note that since our process has some randomness involved (through train/test splits), and features are highly correlated (so it may be possible to represent some features as combinations of other features), regression coefficient size isn't too meaningful. However, the degree to which the decision tree relies on turret count for feature splitting is too high to ignore, so we should pay attention to it when conducting further feature importance analysis.

Appendix C - Notebooks

Note that to run the notebooks and reproduce the results, you should download the kaggle dataset separately and save the files to a subfolder called “dota”.

Chat Sentiment Analysis

Supervised Learning

Unsupervised Learning