Dota2 Match Predictions Using Logistic Regression

Charles Zange crz34@drexel.edu

March 19, 2019

Abstract

This project created four Logistic Regression models to predict match outcomes from the video game Defense of the Ancients 2. Each model corresponds to a different moment in time during a match, drawing slices of information based on what data could be observed at that instant. The models are built from two distinct datasets that cover player skills using TrueSkill, player performance, character selection, and match scores. Results show that models with player skill features and in-game performance metrics perform much better than models based solely on character selection. Future extensions of this project can build on its findings to develop a prediction algorithm that updates in real time.

Contents

1 Background

Defense of the Ancients 2 (Dota2) is a video game with mass appeal and readily available data to support machine learning.

Dota2 is a published by Valve Corporation under lead designer 'IceFrog.'[1] The impact of IceFrog's Dota2 project on Valve, the video game market, and eSports worldwide cannot be understated. In prizes alone, Dota2 tournaments have given away earnings of more than $100 million as of 2017 (Stubbs). The top team of 2018's pinnacle tournament took home over $11 million (Int [2019b]). That same tournament, viewed onsite by 20,000 people, drew 15 million live-streamers online (Int [2019a]).

Dota2's match data is available through OpenDota.com, a volunteer project by Albert Cui, Howard Chung, and Nicholas Hanson-Holtry (Albert Cui [2019]). OpenAI, the nonprofit research and development company, cosponsors this effort.[2] OpenDota.com offers an API to pull data on players and matches, though there are restrictions on the number and size of API calls. For research, data from more than 50,000 matches have been hosted on Academic Torrents (Aca [2019]) and, more recently, Kaggle projects ('devinanzelmo' [2016a]). This project pulls data from a single Kaggle distribution comprised of two datasets, one of 50,000 matches and another of over 900,000 match outcomes, hosted under a CC BY-SA 4.0 license. A thorough description of this data is provided later in this paper.

1.1 Advantages of Dota2 as a test case

I chose Dota2 over other video games for five reasons:

1. 5v5 team match
    - There are two teams, Radiant and Dire. Dota2's 5v5 format demands teamwork and complicates the metric of individual player performance.
2. One map only
    - In first person shooter franchises like Halo or Counter Strike, players compete on a variety of maps, each with distinct advantages and disadvantages to a specific team. Dota2 uses only one map that has largely stayed the same since v7.20 (Map [2019]). This evens the playing field (literally) for analysis.
3. Minute-to-minute updates on individual player performance
    - The OpenDota API offers three statistics on each individual player's in-match performance every 60 seconds. These are helpful for comparing team performances over time.
4. Individual rankings of player skill
    - One of the datasets for this project provides TrueSkill ratings for each player's skill level. This metric, explained in greater detail in Appendix A of this paper, is helpful for identifying individual performance within a team-based match.
5. Large set of playable heroes
    - There are over 100 heroes (= characters) to play, each with their own unique abilities, adding variety to matches.

[1] 'IceFrog' is a pseudonym. Curiously, it is also the only name used by this designer that is publicly known. Even the publishing company, Valve, has declined to publicly reveal his identity. A speculative article from 2005 suggested IceFrog was actually 'Abdul Ismail,' but Valve pushed back, declaring the article fake. `https://dota2.gamepedia.com/IceFrog`

[2] OpenAI presented a full five-member AI team of players in the 2018 International tournament, calling the team "OpenAI Five." That project drew in part from OpenDota's API. `https://openai.com/five/`

1.2 Summary of the approach

This project will tackle the supervised, binary classification problem of predicting match outcomes using four separate Logistic Regressions. Each regression will be targeted to a specific moment in time, t, that corresponds to the slices of incomplete information available at different points during the match:

- $t = 0$, before the match begins and before players have chosen heroes
  - Goal: utilize player skill ratings to predict match outcomes.
- $t = 1$, once players have selected heroes for the match but before play begins
  - Goal: engineer a new feature set to score the game's heroes by their in-game functions and strengths, then use that new feature set to predict match outcomes based on the heroes selected by both teams.
- $t = 10$, ten minutes into the match
  - Goal: expand the dimensionality of the $t = 1$ regression with a summary of each team's performance 10 minutes into the match. This summary will be a sum for each team of observable player achievements like earning Gold and gaining XP.
- $t = n$, at the conclusion of the match
  - Goal: to find the maximum accuracy for outcome prediction based on the results of the match. Note that in Dota2, the final score does not necessarily reflect the winner.

The overall objective of this project is to create a foundation of insights that will inform a live prediction system, pulling real-time data from OpenDota (or a similar system) to predict outcomes based on in-game features.

2 Related Work

2.1 Principle sources

The approach to this project is motivated by a 2016 presentation by Professor Amit Bhattacharyya of the UC Berkeley's School of Information on predicting the outcome of National Football League games (Bhattacharyya [2016a]). In his lecture published December 8, 2016, Bhattacharyya laid out his initial setup, workflow, and resulting tables (Bhattacharyya [2016b]). He consulted a relatively sparse set of features: a) current and last year's win-loss record, week of the season, b) home game v. division game, and c) the spread of which team was favored and by how much. Bhattacharyya chose a Logistic Regression because of its strength in differentiating between binary options. The sigmoid form of the logistic curve provided both a prediction and a confidence level with each selection: the further the point on the sigmoid curve was from the center, the more confident the algorithm could be in its prediction.

Bhattacharyya used Python's scikit module to perform his Logistic Regression. He used the default setting for the regression's solver, 'liblinear'. This open source library supports large-scale linear classification in both Logistic Regression and Support Vector Machines (Rong-En Fan [2017]). I will use the same open source library through the same solver for each of the four Logistic Regressions in this project. For reference, the log likelihood function at the heart of Logistic Regression and the gradient ascent learning function are included below:

$$l(Y|X,\theta) = logP(Y|X,\theta) = \Pi_{t=1}^{N} Y_t \ln g(X_t,\theta) + (1 - Y_t) \ln 1 - g(X_t,\theta)$$

Update function:

$$\theta = \theta + \frac{\eta}{N} X^T (Y - g(X,\theta))$$

I based this project off Bhattacharyya's strategy because of its simplicity and effectiveness. His approach, however, was limited to outcome prediction at $t = 0$ and his limited feature space did not account for individual player actions that supported their teams. In this project, I will expand Bhattacharyya's methodology by testing at multiple moments of time t and expanding the feature space with player-specific data.

Further research and guidance for this approach came from Dota2 prediction projects on Kaggle. The first project by user 'devinanzelmo' served as the source of project data and summary on the structure of the tables ('devinanzelmo' [2016a]). 'devinanzelmo' further posted kernels from this repository on setting up his own prediction problem using different features ('devinanzelmo' [2016b]). I consulted his work to get started with understanding the data, but I ultimately made fundamentally different choices on selecting features and predicting outcomes because I did not support his approach. 'devinanzelmo's prediction kernel averages all player gameplay metrics by account rather than by match, and then applies those arithmetic averages to individual matches as features to indicate a player's performance level. I disagree with this choice. In Dota2, players who play 'Support' roles will collect less gold than those who play other roles like 'Carry' or 'Nuker.' Yet 'Support' players can still perform at a very high level and push their team to success. For this reason, I saw an opportunity to build a new algorithm using different features.

The second project, also by 'devinanzelmo' on Kaggle, created the "Match Outcomes" dataset that is used in the first regression, $t = 0$, of this project ('devinanzelmo' [2016c]). Using a Python library, he approximated the Microsoft TrueSkill score of each player in a collection of over 900,000 matches. TrueSkill is built from the Elo system for rating chess players (further described in the footnote below) and is an improvement for skill predictions. Using this data, I introduce arithmetic means of those TrueSkill calculations and Logistic Regression to predict match outcomes in Regression $t = 0$.

The third and final supporting project was performed on Kaggle by user 'davidmercury' ('davidmercury' [2016]). His project attempted to predict match outcomes based on hero data by creating a feature space of Hero IDs. He divided his dataset into a Radiant and Dire team based on the individual identities of the heroes themselves using numbers to substitute for names for his specific analysis. By keeping Hero IDs and Hero Names split out, he focused on the individual heroes distinct from one another on both teams rather than the team's balance of hero functions to form a cohesive unit. I disagreed with this approach because I believe that it removes the heroes from the context of their functions during the match. Dota2 heroes have specialized roles ('Support,' 'Durable,' 'Jungler,' etc.) that contribute together to a team's makeup. I aim to compare the makeup of a team's balance and not specifically the identity of individual heroes. Later in this paper, I will describe the features I created based on Dota's 'Heropedia' data to highlight hero functions in testing (Cho [2019]).

2.2 Additional support and considerations

I consulted an article on using Logistic Regression to predict match outcomes in the video game Overwatch to see another real-world example like Bhattacharyya's approach (Yang [2018]). This article helped me an application of Logistic Regression in a team-based competition, though Overwatch and Dota2 have different mechanics.

An ID3 Decision Tree or a Random Forest would have provided strong alternatives to Logistic Regression. Dota2 match outcome prediction is a binary choice with a supervised outcome. Research and experience from class highlighted two potential drawbacks that pushed against decision trees.

1. Concerns about overfitting an inaccuracies
    - Further research online highlighted the risks of overfitting (Log [2016]) and potential issues with accuracy (Perlich [2017]). I noted from these sources that trees need pruning to be effective, whether through consistent testing with a validation set during construction or pruning in post. This would further delay the construction of an algorithm to make choices based on the data. Random Forest would address overfitting, but would also be more difficult to interpret once trained.
2. Using continuous values in trees is not impossible, but it takes effort and consideration

- Continuous values need special handling when working with trees. It would take more time to consider and apply this kind of feature engineering and categorization to the Dota datasets.

A Support Vector Machine (SVM) is another strong alternative to Logistic Regression. Ultimately, I chose to stick closely to Bhattacharyya's methodology on Logistic Regression using the 'liblinear' open source library. This ensured consistency between each of my testing frameworks and kept the focus on regressions with confidence levels. In future iterations of the project, it would be valuable to try additional solver libraries in both Logistic Regression and SVM to compare performance.

Additional support came from a project led by Aaron Schlegel to predict outcomes of sheltered animals at the Austin Animal Center (Schlegel [2018]). His project also used Python for preprocessing and model construction, aiming to determine the likelihood of adoption, transfers, animal returns, etc. based on a variety of features. While his team's outcome prediction model is not consistent with the Logistic Regression style from Bhattacharyya's NLF draft picks, the team's use of Python for exploratory data analysis and feature engineering was helpful when building my first dataframes.

2.3 Source of data

The data are broken into two sets ('devinanzelmo' [2016a]).

- "Match Outcomes" set. Over 900,000 matches with corresponding player account numbers and calculated TrueSkill scores, without supporting environment data (objectives, hero names, and other non-player information). The TrueSkill scores of players were calculated by 'devinanzelmo' in Python ('devinanzelmo' [2016c]) (see Appendix A).

- "Match Players" set. 50,000 matches with supporting player and environment data. This data was originally pulled in 2016 and is therefore missing some of the newest heroes (112 then, now 117), map tweaks, and MMR data. Nevertheless, the dataset is otherwise complete and usable for testing on features still available in the newest data from OpenDota's API.

Importantly, the two datasets cannot be cross-referenced. There is no overlapping intersection of matches because the datasets were pulled at different times for different purposes. This unfortunately meant that the TrueSkill scores of the "Match Outcomes" set could not be applied to regressions from the "Match Players"

Below are two tables showing the total available data for the project. Table 1 is for the "Match Outcomes" set. Table 2 is for the "Match Players set."

Figure 1: "Match Outcomes" dataset

Table	Description
MATCH_OUTCOMES	Two rows per match_id: one row for the Radiant team, one row for the Dire team. It expands information on match outcome, directly identifying which five players won (by account_id) and whether they were playing on Radiant. 1,828,588 rows (914,294 matches of two rows with one row per team)
PLAYER_RATINGS	Player's skill measured by TrueSkill score, with per-player average and standard deviation. 834,226 rows

Figure 2: "Match Players" dataset

Table	Description
MATCH	Identifies the match (match_id), the winner (Radiant v. Dire), and the game mode, plus details on objectives accomplished by the end of the match. 50,000 matches
ABILITY_IDS	List of all the abilities available by the more than 100 heroes in the full DotA2 roster, by number and name. 688 rows
ABILITY_UPGRADES	Tracks the ability upgrades made by each player in each match, plus the time during the match that the upgrade was earned. 8,939,599 rows
CHAT	Chat logs by match_id. 1,439,489 rows
CLUSTER_REGIONS	Location of servers ('clusters') that run DotA2 matches worldwide, plus server number. 53 rows
HERO_NAMES	Names of heroes by hero_id number. 112 rows
ITEM_IDS	Names of items by item_id number. 189 rows
OBJECTIVES	Objectives accomplished by players during the match, marked at the time of the accomplished objective. 1,173,396 rows
PATCH_DATES	Date that DotA2 was patched to a newer version. Patches usually include balance fixes, though they occasionally include new heroes, abilities, items, or map changes. 19 rows
PLAYER_TIME	Data on how player has used his or her time during the game. Every sixty seconds, three statistics are captured for each player: current XP, current Gold, and current 'Last Hits' (kills on non-player heroes that give XP and Gold). Individual players are recorded by player slots. 2,209,778 rows
PLAYERS	Summary of a Player's performance per match, counting all gold/XP/Last Hits/Hero Kills/etc. earned. 50,000 rows
PURCHASE_LOG	Log of the items purchased during a match. 18,193,745 rows
TEAMFIGHTS	'Teamfight' is, in general commentary and analysis, a subjective term usually defined as a conflict involving most if not all players in the same area of the map at the same time, with multiple (or potentially all) players casting abilities in quick succession with one another. 539,047 rows
TEAMFIGHTS_PLAYERS	Players involved in a 'Teamfight' in the 'Teamfights' table, with their recorded damage/deaths/earnings. 5,390,470 rows (10 row per teamfight, one row per player)

Subsequent triage eliminated the following tables from consideration.

- "CHAT" – Natural Language Processing is outside the scope of this project.
- "ABILITY IDS" and "ITEM IDS" – The names of items and abilities did not affect project testing.
- "CLUSTER REGIONS" – Geographic distance from the cluster could affect a player's connection to the online match, which makes it more difficult for players with slower connections to participate. While this does affect a game's outcome, the player data does not show a player's geographical location during the game. Therefore, server locations could not be compared.
- "TEAMFIGHTS" and "TEAMFIGHTS PLAYERS" – I was not confident that TEAMFIGHTS would be a firm, objective metric of performance given the subjective definition of the term in general commentary and analysis. These two tables were removed to help keep the data clean.

- "PATCH DATES" – After thorough consideration, this table was dropped from the final model. Software patches can affect match outcomes by introducing new heroes, changing hero abilities, or making minor changes to the play map. That said, my goal was to develop an outcome predictor with a longer view of DotA2, able to generalize results indifferent of patches now and into the future. For this reason, "PATCH DATES" was dropped.
- "PURCHASE LOG", "OBJECTIVES", and "ABILITY UPGRADES" – These metrics of player performance were dropped from the project due to their relative size (18,193,745 rows, 1,173,396 rows, and 8,939,599 rows respectively) and because of potential pitfalls in streamlining multi-player tables into team-specific results. For example, TrueSkillsigma for a team could be averaged, but how could a team's specific ability upgrades (often 20 or more) be combined into a feature while preserving their timing and order in a team summary? Though not considered for this project, these and other performance questions are worthy of future analysis.

3 Main Content

3.1 Recreating the project

This project used the following tools:

- Jupyter (IPython)
  - This project was coded in Python using the Jupyter Notebook environment (Jup [2019]). Jupyter is built from IPython (IPy [2019]). This environment allows for inline results of calculations and visualizations, tying together Python libraries like numpy and scikit with in-progress results from Pandas dataframes. The original code was all developed in Jupyter and later converted to standard Python 3. It is easiest to view the project and its results using Jupyter Lab.
- Python 3, plus Python mathematical and scientific libraries
  - Python 3: this language's versatility and superior UTF-8 handling placed it at the heart of all coding aspects of this project (Pyt [2018]).
  - NumPy: this library's array-style resources are in some ways comparable to Matlab's matrices. NumPy supported the mathematics of data normalization for this project (Num [2018]).
  - Scikit-learn: the linear model and metrics packages from scikit-learn combined to provide the Linear Regression algorithm used in this project (Sci [2019]).
- Pandas and Microsoft Excel
  - Pandas Data Analysis Library: Pandas through Python provided the bulk of the pre-processing resources for the project. Pandas is built from NumPy and many NumPy array operations are included natively within a Pandas dataframe (Pan [2018]).
  - Microsoft Excel: One a few occasions (listed in comments within the coding section), I used this application to provide a larger view of the total dataset and to perform a small number of bulk operations (Exc [2019]).

This project broke each of the four regressions into two separate coding blocks: pre-processing and regression. The pre-processing scripts each produce one intermediate .CSV file that is then used by the regression scripts for analysis. The pre-processing scripts build progressively upon each other, either through direct reference to the source files from Kaggle or pulling from the intermediate files of the previous step ($t = 2$ relies on $t = 1$, for example). Note that the intermediate file for Regression $t = 0$ required minor tweaks in Microsoft Excel to be correctly formatted for the first regression. Further details are provided in the README.txt file that accompanies this paper as well as in comments within the project code.

3.1.1 Feature Engineering

In addition to the features available in the "Match Outcomes" and "Match Players" datasets, new features were engineered for two of the regressions.

- $t = 0$ Regression: "Match Outcomes" using TrueSkill to predict outcome

The goal of this regression is to predict outcomes based on player skills. To do so, player skills somehow must be translated into a team's overall potential. Rather than leave player skills separated into distinct features, I chose to perform arithmetic mean without weighting to combine the pre-calculated TrueSkills of each player into a team average. My logic was to treat a team as if it were playing as a single cohesive unit with common goals, rather than a collection of individuals, and to grant equal weight to all five players regardless of their role or skill. Four new features, two per team, were engineered to fulfill this comparison:

1. *Mean(TrueSkillmuRadiant)* - Mean of each Radiant player's average TrueSkill.
2. *Mean(TrueSkillmuDire)* - Mean of each Dire player's average TrueSkill.
3. *Mean(TrueSkillsigmaRadiant)* - Mean of each Radiant player's standard deviation TrueSkill.
4. *Mean(TrueSkillsigmaDire)* - Mean of each Dire player's standard deviation TrueSkill.

- $t = 1$ Regression: "Match Players" using hero data

The goal of this regression is to predict outcomes based on the function of heroes selected by each team for the match, regardless of the skills of individual players. To restate from an earlier discussion of 'davidmercurey's analysis, I aimed to focus on each hero's top three functions over hero identity with this project to emphasize a team's hero function composition. To that end, I consulted Dota2's 'Heropedia' which lists each hero along with its specializations and assigned point values to the significance of each specialization (Cho [2019]). The results are included along with the project code (see README.txt for details). Points were applied using the following logic:

- Six skill points were assigned to each hero for up to three top specializations: 3 points for the first, 2 points for the second, and 1 point for the third. This was done to focus on a hero's key strengths. Most heroes have three or more specializations.
- Heroes with two specializations received 4 for the first and 2 for the second.
- Heroes with one specialization received 6 points for that specialization.
- Two sets of additional features were added: Melee vs. Ranged (1 point to one or the other) and Strength vs. Agility vs. Intelligence (1 point for one). These classes are fully independent – no hero received partial points or multiple classes from these sets.
- When combined to form a team, these point values were summed, not averaged, to account for potential extra points in heroes with single-characteristic emphasis.

Here is a sample of the resulting table:

I chose this approach based on research in hero skills using Dota2's Gamepedia, specifically the hero attributes table which presents a nuanced differentiation of a hero's strengths (Tab [2019]). I considered using the Gamepedia attributes table instead of the engineered features described above, but ultimately came to the conclusion that the attributes table insufficiently described a hero's function during the game. The statistics of a hero's 'Strength' and 'Agility' scores do not necessarily determine if a hero's function is 'Support' or 'Durable.' A hero with high strength could be either a 'Carry' (a principle attacker in the match) or a 'Support' (back line defender) depending on its abilities. With Regression $t = 1$, I wanted to emphasize the hero's primary functions over its core statistics.

Figure 3: Engineered hero features

	name	hero_id	localized_name	Carry	Disabler	Lane_Support	Initiator	Jungler	Support	Durable	Nuker	Pusher	Escape	Melee	Ranged	Strength	Agility	Intelligence
0	npc_dota_hero_antimage	1	Anti-Mage	3	0	0	0	0	0	0	1	0	2	1	0	0	1	0
1	npc_dota_hero_axe	2	Axe	0	1	0	3	0	0	2	0	0	0	1	0	1	0	0
2	npc_dota_hero_bane	3	Bane	0	2	0	0	0	3	0	1	0	0	0	1	0	0	1
3	npc_dota_hero_bloodseeker	4	Bloodseeker	3	2	0	0	1	0	0	0	0	0	1	0	0	1	0
4	npc_dota_hero_crystal_maiden	5	Crystal Maiden	0	2	0	0	0	3	0	1	0	0	0	1	0	0	1
5	npc_dota_hero_drow_ranger	6	Drow Ranger	3	2	0	0	0	0	0	0	1	0	0	1	0	1	0
6	npc_dota_hero_earthshaker	7	Earthshaker	0	1	0	2	0	3	0	0	0	0	1	0	1	0	0
7	npc_dota_hero_juggernaut	8	Juggernaut	3	0	0	0	0	0	0	0	2	1	1	0	0	1	0
8	npc_dota_hero_mirana	9	Mirana	3	0	0	0	0	2	0	0	0	1	0	1	0	1	0
9	npc_dota_hero_morphling	10	Morphling	3	0	0	0	0	0	1	0	0	2	0	1	0	1	0
10	npc_dota_hero_nevermore	11	Shadow Fiend	4	0	0	0	0	0	0	2	0	0	0	1	0	1	0
11	npc_dota_hero_phantom_lancer	12	Phantom Lancer	3	0	0	0	0	0	0	0	1	2	0	1	0	1	0
12	npc_dota_hero_puck	13	Puck	0	2	0	3	0	0	0	0	0	1	0	1	0	0	1
13	npc_dota_hero_pudge	14	Pudge	0	3	0	2	0	0	1	0	0	0	1	0	1	0	0
14	npc_dota_hero_razor	15	Razor	3	0	0	0	0	0	2	1	0	0	0	1	0	1	0
15	npc_dota_hero_sand_king	16	Sand King	0	2	0	3	0	1	0	0	0	0	1	0	1	0	0
16	npc_dota_hero_storm_spirit	17	Storm Spirit	3	0	0	0	0	0	0	1	0	2	0	1	0	0	1
17	npc_dota_hero_sven	18	Sven	3	2	0	1	0	0	0	0	0	0	1	0	1	0	0
18	npc_dota_hero_tiny	19	Tiny	3	0	0	0	0	0	0	2	1	0	1	0	1	0	0

Figure 4: Regression summary

Time	Features
D(t=0)	MATCH_OUTCOMES{match_id, account_id_0, account_id_1, account_id_2, account_id_3, account_id_4, win, rad} PLAYER_RATINGS{ account_id, total_wins, total_matches, trueskill_mu, trueskill_sigma}
D(t=1)	MATCH – project-generated engineered features on hero data from PLAYERS{hero_id, player_slot} and HERO_NAMES{hero_id}
D(t=10))	MATCH – add PLAYER_TIME{gold, xp, last_hits} to D(t=1)
D(t=n)	MATCH – add PLAYERS{gold_per_min, xp_per_min, kills, deaths, assists, denies, last_hits} to D(t=10)

3.2 Logistic Regression

Above (Figure 4) is a simplified list of the final features for each of the four regressions.

3.2.1 Models

Each regression was evaluated as a binary choice between 0 = *Direvictory* and 1 = *Radiantvictory*. Regressions were calculated using the 'liblinear' open source library (Rong-En Fan [2017]). The feature, coefficient, and intercept of each resulting equation are shown below.

Figure 5: "Match Outcomes" $t = 0$

REGRESSION T=0 Player skill only	**Dire mean TrueSkillmu**	**Dire mean TrueSkillsigma**	**Radiant mean Trueskillmu**	**Radiant mean Trueskillsigma**
Coefficient:	-1.06511126	-0.02104529	1.10191568	0.02481441
Intercept:	0.03281206			

Figure 6: "Match Players" $t = 1$

REGRESSION T=1 Hero data only	**Carry Dire**	**Carry Radiant**	**Disabler Dire**	**Disabler Radiant**
Coefficient:	0.02744527	-0.00789097	0.01722094	0.00962746
	Initiator Dire	**Initiator Radiant**	**Jungler Dire**	**Jungler Radiant**
Coefficient:	0.04155458	-0.07244957	-0.0144358	-0.04207553
	Support Dire	**Support Radiant**	**Durable Dire**	**Durable Radiant**
Coefficient:	-0.04769721	0.05063569	-0.08759323	0.08697741
	Nuker Dire	**Nuker Radiant**	**Pusher Dire**	**Pusher Radiant**
Coefficient:	-0.02239317	0.02306302	0.00456972	-0.01508217
	Escape Dire	**Escape Radiant**	**Melee Dire**	**Melee Radiant**
Coefficient:	0.06688836	-0.06160272	-0.03304307	0.0493681
	Ranged Dire	**Ranged Radiant**	**Strength Dire**	**Strength Radiant**
Coefficient:	0.03304307	-0.0493681	-0.00798974	-0.00853639
	Agility Dire	**Agility Radiant**	**Intelligence Dire**	**Intelligence Radiant**
Coefficient:	-0.03481506	0.02565702	0.04270936	-0.01643402
Intercept:	0.00109443			

Figure 7: "Match Players" $t = 10$

REGRESSION T=10 T=1 + 10 minutes	**Carry Dire**	**Carry Radiant**	**Disabler Dire**	**Disabler Radiant**
Coefficient:	0.03695539	-0.0252565	0.01695094	0.00675322
	Initiator Dire	**Initiator Radiant**	**Jungler Dire**	**Jungler Radiant**
Coefficient:	0.04231519	-0.07125435	0.00534528	-0.06991897
	Support Dire	**Support Radiant**	**Durable Dire**	**Durable Radiant**
Coefficient:	-0.05128501	0.06372561	-0.08794901	0.09221852
	Nuker Dire	**Nuker Radiant**	**Pusher Dire**	**Pusher Radiant**
Coefficient:	-0.04303888	0.0385553	0.00772819	-0.01311013
	Escape Dire	**Escape Radiant**	**Melee Dire**	**Melee Radiant**
Coefficient:	0.0714312	-0.06359616	-0.05640949	0.06979992
	Ranged Dire	**Ranged Radiant**	**Strength Dire**	**Strength Radiant**
Coefficient:	0.05640949	-0.06979992	-0.01068629	-0.00822525
	Agility Dire	**Agility Radiant**	**Intelligence Dire**	**Intelligence Radiant**
Coefficient:	-0.05347797	0.043515	0.063963	-0.03437898
	gold_t_0 Radiant	**lh_t_0 Radiant**	**xp_t_0 Radiant**	**gold_t_1 Radiant**
Coefficient:	0.23130091	-0.06624305	0.20921477	0.18358423
	lh_t_1 Radiant	**xp_t_1 Radiant**	**gold_t_2 Radiant**	**lh_t_2 Radiant**
Coefficient:	0.02711764	0.14639824	0.20558886	-0.02139668
	xp_t_2 Radiant	**gold_t_3 Radiant**	**lh_t_3 Radiant**	**xp_t_3 Radiant**
Coefficient:	0.17687963	0.22928574	-0.04727303	0.19740332
	gold_t_4 Radiant	**lh_t_4 Radiant**	**xp_t_4 Radiant**	**gold_t_128 Dire**
Coefficient:	0.2189073	-0.02935809	0.17327173	-0.23586138
	lh_t_128 Dire	**xp_t_128 Dire**	**gold_t_129 Dire**	**lh_t_129 Dire**
Coefficient:	0.07382622	-0.17889191	-0.18734588	0.00838758
	xp_t_129 Dire	**gold_t_130 Dire**	**lh_t_130 Dire**	**xp_t_130 Dire**
Coefficient:	-0.14855514	-0.16143397	0.01607373	-0.20672613
	gold_t_131 Dire	**lh_t_131 Dire**	**xp_t_131 Dire**	**gold_t_132 Dire**
Coefficient:	-0.19966553	0.04131507	-0.16691512	-0.18986756
	lh_t_132 Dire	**xp_t_132 Dire**		
Coefficient:	0.04476207	-0.19115533		
Intercept:	0.01284749			

Figure 8: "Match Players" $t = n$

REGRESSION T=N T=10 + end results	**Carry Dire**	**Carry Radiant**	**Disabler Dire**	**Disabler Radiant**
Coefficient:	-6.79E-02	-1.56E-02	6.24E-02	2.29E-02
	Initiator Dire	**Initiator Radiant**	**Jungler Dire**	**Jungler Radiant**
Coefficient:	-3.32E-02	5.70E-02	-3.92E-02	2.76E-03
	Support Dire	**Support Radiant**	**Durable Dire**	**Durable Radiant**
Coefficient:	7.61E-02	-1.24E-01	-2.94E-02	4.94E-02
	Nuker Dire	**Nuker Radiant**	**Pusher Dire**	**Pusher Radiant**
Coefficient:	-9.62E-02	6.35E-02	-1.23E-01	1.84E-01
	Escape Dire	**Escape Radiant**	**Melee Dire**	**Melee Radiant**
Coefficient:	2.06E-01	-1.22E-01	3.72E-02	2.03E-03
	Ranged Dire	**Ranged Radiant**	**Strength Dire**	**Strength Radiant**
Coefficient:	-3.72E-02	-2.03E-03	1.83E-02	-5.57E-02
	Agility Dire	**Agility Radiant**	**Intelligence Dire**	**Intelligence Radiant**
Coefficient:	3.43E-03	-1.47E-02	-2.23E-02	7.24E-02
	gold_per_min Dire	**gold_per_min Radiant**	**xp_per_min Dire**	**xp_per_min Radiant**
Coefficient:	-6.83E+00	6.25E+00	-6.17E-01	7.19E-01
	kills Dire	**kills Radiant**	**deaths Dire**	**deaths Radiant**
Coefficient:	1.82E-01	-7.89E-02	3.71E-03	-3.69E-02
	assists Dire	**assists Radiant**	**denies Dire**	**denies Radiant**
Coefficient:	1.36E-01	-1.31E-01	-7.58E-02	1.96E-01
	last_hits Dire	**last_hits Radiant**	**level Dire**	**level Radiant**
Coefficient:	1.60E+00	-1.42E+00	-1.24E+00	8.81E-01
Intercept:	0.90665727			

4 Evaluation

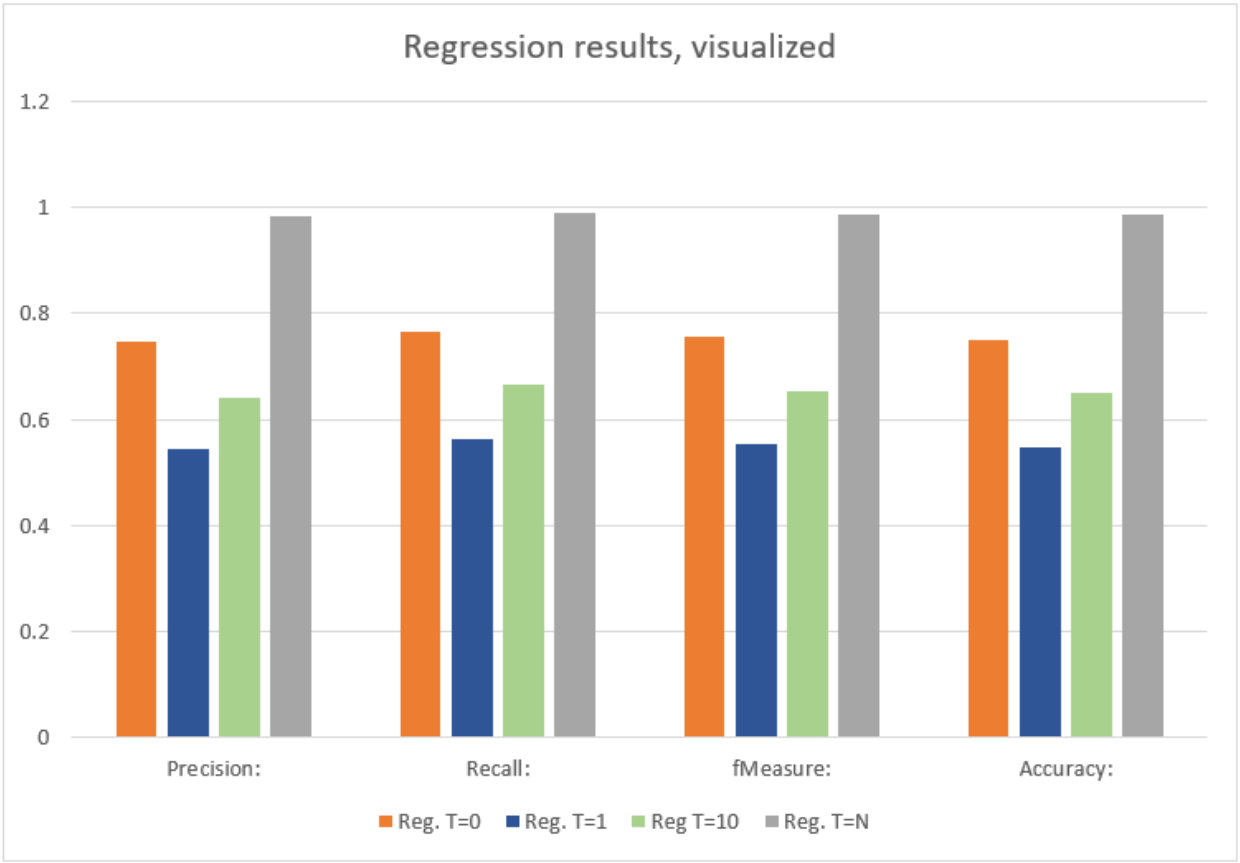
4.1 Statistics

Here are the results from Regression $t = 0$ focused on the "Match Outcomes" dataset and Regressions $t = 1$, $t = 10$, and $t = n$ from the "Match Players" dataset.

Figure 9: Final statistics

Statistic	Reg. T=0	Reg. T=1	Reg. T=10	Reg. T=N
Precision:	0.746510005	0.54430085	0.639306089	0.983816509
Recall:	0.763746727	0.564545345	0.666950079	0.989577234
fMeasure:	0.755030004	0.554238293	0.652835572	0.986688463
Accuracy:	0.750150006	0.548483939	0.649354548	0.986310417

Figure 10: Final charts



4.2 Feature evaluation

In addition to the statistics identified above, I evaluated each regression to find its most discriminating features. Conceptually, the most discriminating features could be found directly using Information Gain on all of the real-valued features - either by breaking up the range of possible values into enumerated regions or using a Gaussian distribution - and comparing entropies. Alternatively, tools within Python's scikit library can perform this statistical analysis on a broader range of data. For this project, I selected the f_classif solver from scikit's feature_selection library to find the best features for predicting outcomes.

f_classif relies on the F-test from the Analysis of Variance (ANOVA) collection of models. In broad terms, the application of this test within the Dota2 data would be a comparison $F = \frac{variationbetweenmeans}{variationofclasseswithinsamples}$ for all features within the regression. In scikit's own analysis, this result is shown as an accuracy score, where accuracy is defined as the number of samples correctly predicted by this feature (Acc [2019]). The idea of this analysis is to find out which feature's variance corresponds to the highest variation of classifications, helping us identify which feature is the best predictor of outcomes (Min [2016]).

I varied the number of $kBest$ features to select for each regression, given that the regressions each had a different number of features. The results are included below along with sklearn's accuracy score. Normalizing this prediction to a true F score would require dividing every score by the total number of outcomes. I opted not to do this because the accuracy scores for many of these features are very small compared to the number of outcomes, avoiding potentially cumbersome decimal formats or scientific notation.

- Regression $t = 0$
  1. Radi_meanTrueSkillmu, the mean of Radiant's player TrueSkills. $Score = 9692.863$
  2. Dire_meanTrueSkillmu, the mean of Dire's player TrueSkills. $Score = 9102.588$
- Regression $t = 1$
  1. Dire 'Durable' class character. $Score = 186.703$
  2. Radiant 'Durable' class character. $Score = 181.240$
  3. Radiant 'Strength' class character. $Score = 92.534$
  4. Radiant 'Ranged' class character. $Score = 85.167$
- Regression $t = 10$
  1. Dire player slot 0 gold earned. $Score = 402.376$
  2. Radiant player slot 0 gold earned. $Score = 399.762$
  3. Radiant player slot 3 gold earned. $Score = 379.591$
  4. Radiant player slot 4 gold earned. $Score = 364.627$
  5. Dire player slot 2 gold earned. $Score = 343.025$
  6. Dire player slot 4 gold earned. $Score = 336.570$
  7. Dire player slot 1 gold earned. $Score = 324.038$
  8. Radiant player slot 1 gold earned. $Score = 319.609$
  9. Dire player slot 3 gold earned. $Score = 312.094$
  10. Radiant player slot 2 gold earned. $Score = 308.634$
  11. Radiant player slot 3 XP earned. $Score = 197.606$
  12. Dire player slot 0 XP earned. $Score = 196.171$
- Regression $t = n$
  1. Dire gold-per-minute earned. $Score = 102456.701$

2. Radiant gold-per-minute earned. $Score = 93649.567$
3. Dire XP-per-minute earned. $Score = 39237.223$
4. Radiant XP-per-minute earned. $Score = 30180.899$
5. Dire player kills. $Score = 22873.753$
6. Radiant player deaths. $Score = 22090.025$

5 Observations and Conclusions

It is tempting to draw comparisons between Regression $t = 0$ and the other three regressions. Unfortunately, since Regression $t = 0$ was evaluated on a separate "Match Outcomes" dataset compared to the other three on "Match Players," it would be unwise to place too much emphasis on direct comparisons. That said, there are conclusions to draw from Regression $t = 0$ independently.

The Precision, Recall, fMeasure, and Accuracy of $t = 0$ were, subjectively, impressive. The Logistic Regression based solely off a team's TrueSkill mean of means and mean of standard deviations accurately predicted match outcomes about 75% of the time. This is a considerable improvement over random selection's 50% accuracy. Independent of other conditions, I believe it is fair to conclude that (A) a player's skill is a principle feature that must be considered to successfully predict match outcome, and (B) the TrueSkill metric introduced an important and useful tool for making said prediction. Validation of Microsoft's TrueSkill metric is beyond the scope of this paper, but it has proven to be effective even independent of other features.

For Regressions $t = 1$, $t = 10$, and $t = n$, we can see an improvement with accuracy in each step of the game's playtime. Regression $t = 10$, trained after 10-minutes of gameplay, outperformed the $t = 1$ 54.8% accuracy at 64.9%. It is logical to expect that the outcome of a match is generally more predictable as a match progresses. That said, it would be interesting to see the shape of the curve that is formed by increasing accuracy over $\{t = 2, t = 3, \ldots t = n - 1\}$. As the Logistic Regression predicts with greater and greater certainty, its prediction certainty increases along its sigmoid function. I would be interested in seeing the shape and coefficients of a similar function generated by looking at increase and accuracy over total match time.

Regression $t = n$ performed with the highest accuracy (98.6%), though this result seems trivial as the regression can look at the entire set of final results, including the match score and the gold amounts of each hero. That said, the order of $kBest$ features is significant. Based on the ANOVA F-test, the most significant features in predict match outcome at $t = n$ are gold-per-minute earned by both teams, then XP-per-minute earned by both teams, and then the kill score (with Dire player kills and Radiant player deaths being virtually identical, minus rare cases of heroes being killed by Non Player Characters). I noted in introducing this topic that the final score of hero kills does not always reflect the winner of the match. That appears to be correct. A team's gold earnings and XP earnings are better predictors of the outcome of a match than the score, based on this analysis.

Continuing on the topic of significant features, $t = 0$ showed that a team's average TrueSkill mean was the best predictor of match outcomes. The only alternative in this small feature space would be TrueSkill standard deviation. It is logical that the player's average skill level would weigh higher than the variance of their play - players with higher averages are considered by TrueSkill to be more skilled, despite potential overlaps between the variances of closely ranked players.

The best prediction features of Regression $t = 0$ skewed heavily in favor of gold. Though two XP features rose into the top ranks, 10 of the 12 best predictors were all player gold earnings. This suggests that gold is the most significant in-game factor among those considered in the regression. Dota2 is sometimes considered a 'farming' game during which players have to find ways to earn XP and gold in otherwise unoccupied parts of the map in an effort to accelerate toward important item purchases. This insight could support players forming strategies in real time, re-emphasizing 'farming' time.

I was disappointed with the results of Regression $t = 1$ which looked purely at hero data. $t = 1$ had numerous engineered features designed to highlight a hero's strengths and abilities. This was in contrast

to another study published on Kaggle with the same dataset, in which the author focused exclusively on Hero ID and Hero Names. In the end, our algorithms performed similarly. By comparison, 'davidmercury's work on the same dataset with three algorithms produced the following accuracy: Support Vector Machine 48.5%, Random Forest 56.8%, and k-Nearest Neighbors 53.0%. My Regression $t = 1$ had an accuracy of 54.8%, only 4.8% more efficient than random selection and 2.0% less accurate that 'davidmercury's Random Forest model. This suggests that a new understanding of hero selection is needed to incorporate hero data into future models. There may be better methods for comparing the hero on opposing teams, perhaps by looking at individual matchups of specific hero roles to, for example, compare one teams attacking strength against another defending strength. Alternatively, it may also be true that hero selection is not as strong a predictor of match outcomes as generally believed by players and commentators.

Interestingly, the two best features for predicting outcomes from Regression $t = 1$ were both 'Durable' classes for Dire and Radiant, though the accuracy scores were small. This was surprising given the conventional wisdom of the game that the 'Carry'-heavy characters, charged with being on the front lines and literally 'Carrying' their team to victory, are not themselves the best outcome predictors. Further testing is needed to analyze the ideal balance of 'Durable' and other classes ('Ranged' and 'Strength' also appeared in the top 4).

6 Future extensions of this work

There are five primary opportunities to expand this project into future work.

First, TrueSkill: design a project that can calculate and subsequently evaluate the TrueSkill of individual players, comparing this result against algorithms that do not use TrueSkill on the same dataset. Regrettably, the current project was unable to answer those questions without a matching dataset for testing. Is TrueSkill more accurate than Valve's own Matchmaking Rating (introduced after project data was pulled)? Is TrueSkill more accurate than other predictions that do not rely on any player skill data? Do minutes of live gameplay need to be added to a dataset before another algorithm can outperform TrueSkill, and if so how many minutes?

Second: test other types of algorithms on the dataset to compare accuracy with Logistic Regression. Random Forests were proposed earlier. While I chose a different approach, I have seen other video game outcome projects that rely on Random Forests. Support Vector Machines could apply here, as could a Hidden Markov Model. It would be interesting to compare the performance of various algorithms on the dataset.

Third: increase the feature space of testing in the $t = 2$ through $t = n - 1$ period by adding in objectives, ability upgrades, and item purchases. As indicated in the discussion about data pre-testing, these three tables were not included in this project due in part to file size challenges. That said, the progression of a team through various objectives is an indicator of a team's overall performance in match, and this data could be helpful in predicting the outcome of an in-progress match.

Fourth: deeper understanding as teamwork as a distinct feature. There is hidden information in this dataset: how many 5v5 teams were truly planned squads, and how many were composed of random partners? Matchmaking in Dota2 is fluid. Sometimes all 10 members are chosen randomly from available players. Other times 2 or more players will already be linked online via headsets, collaborating in real time on common objectives. What features could be added to reflect this teamwork dynamic, and how could we evaluate the performance of teams vs. random players? In a future test, it would be interesting to untangle the team dynamic more fully.

Finally, I would be interested in exploring OpenDota's API further in search of live statistics from matches as they occur. With access to that data, it would be possible to develop software that can be run in iterations (perhaps every 60 seconds) from new in-game data to predict outcomes.

7 Appendix A: TrueSkill

TrueSkill is not native to Valve or Dota2. It is an algorithm designed by Microsoft that was later added to Python as a separate library. Valve introduced its own matchmaking metric, called Matchmaking Rank or MMR, which has subsequently been made available through the OpenDota API. Unfortunately, the MMR metric was not available at the time the dataset for this project was pulled from Kaggle, and therefore cannot be used in this project. For more information on MMR: `https://dota2.gamepedia.com/Matchmaking`.

TrueSkill is an extension of the Elo system used for rating chess players. TrueSkill expands on the Elo system in a few ways, most notably through the employment of Bayesian probability on continuous value data using a Gaussian probability density function. Unlike the Elo system, the TrueSkill ranking algorithm can be applied to individual players in team-based competition. The algorithm produces the average skill of a gamer (μ) and a degree of uncertainty about the player's skill (σ). In short, a player's skill uncertainty (σ) is relatively high when they have just started playing a game. Over time, the uncertainty tightens as their record of wins and losses provides more data for TrueSkill to assess their ability. TrueSkill scores can then be used by Xbox's matchmaking engine to produce what it calls "interesting" matches – that is, matches in which players have similar average (μ) TrueSkills, within some tolerance for uncertainty (σ).

Microsoft claims that TrueSkill can generate an accurate skill level for a player in as few as 3 matches of 16 player free-for-all competition. Without teams to improve or reduce a player's win percentages, more skilled players will rise quickly to the top of the TrueSkill charts. In a team-based game the prediction will take longer to converge. Games with 2 teams with 8 players per side can take as many as 91 matches before the uncertainty (σ) shrinks and the algorithm becomes more confident in its predictions.

Winning players with a higher skill average (μ) that far exceeds any overlap of uncertainty with their opponent ($(\mu_{winner} - \mu loser) > \epsilon$ fall into a category of expected wins, changing little about the uncertainty (σ) of their TrueSkills. But unexpected results where a weaker TrueSkill player defeats a stronger opponent earns that weaker player a significant gain in TrueSkill points proportional to the difference in the pre-match skills $\mu_{loser} - \mu winner$.

Ultimately the goal of TrueSkill within Microsoft's Xbox platform is matchmaking. The algorithm is used to maximize the change of a draw, as draws are considered 'interesting' matches in which both teams have an equal chance of winning. For two players A and B with TrueSkills (μ_a, σ_a) and (μ_b, σ_b), the chance of a draw is calculated as:

$$e^{\frac{-(\mu_a - \mu_b)^2}{2c^2}} * \sqrt{d}$$

where $d = \frac{2\beta^2}{c^2}$ and $c^2 = 2\beta^2 + \sigma^2_{winner} + \sigma^2_{loser}$ and β^2 is the variance of the performance around the skill of each player.

The results of the match are then updated for each player based on the likelihood of the given result (difference in skill levels, accounting for each player's σ) and the result that actually occurred. The less the likelihood of the result, the more the point gain/loss affecting each player's TrueSkill.

As indicated in the body of this article, there are TrueSkill libraries in Python already (see `https://pypi.org/project/trueskill/`) and a TrueSkill Dota2 project on Kaggle ('devinanzelmo' [2016c]), however calculating TrueSkill in general can be challenging without an enormous dataset. Microsoft estimates that a 2 team/4 players per team competition would need 46 matches of data for TrueSkill to converge, and a 2 team/8 player match would need 91. Dota2 fits in the middle of that range with 2 team/5 player competition. With the large number of players worldwide that have data pulled from OpenDota's API (and the complicating factor of players hiding their Account IDs, creating 'fake' player accounts like $AccountID = 0$), a substantial dataset is required to generate 50 or more matches for even a modest number of players. That said, TrueSkill does theoretically move toward convergence even after a small number of samples, and is therefore useful within the context of this project.

For more information on TrueSkill, consult the following references: (Ralf Herbrich [2006],Tom Minka [2005])

8 Bibliography

References

Logistic Regression versus Decision Trees, 2016. URL https://blog.bigml.com/2016/09/28/logistic-regression-versus-decision-trees/. BigML. [Last modified 28 September 2016].

Understanding Analysis of Variance (ANOVA) and the F-test, 2016. URL http://blog.minitab.com/blog/adventures-in-statistics-2/understanding-analysis-of-variance-anova-and-the-f-test. The Minitab Blog. [Last accessed 17 March 2019].

NumPy, 2018. URL https://www.numpy.org/. Supported by NumFocus. [Last accessed 17 March 2019].

Pandas: Python Data Analysis Library, 2018. URL https://pandas.pydata.org/. Supported by NumFocus. [Last commit v0.23.4 3 August 2018].

Python 3.7.2, 2018. URL https://www.python.org/downloads/. Python Software Foundation. [Last commit v3.7.2 24 December 2018].

Academic Torrents, 2019. URL http://academictorrents.com/. [Last accessed 17 March 2019].

Accuracy Score, 2019. URL https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score. Scikit Learn. [Last commit v0.23.4 3 August 2018].

Choose a Hero, 2019. URL http://academictorrents.com/. Dota2. [Last accessed 17 March 2019].

Microsoft Excel, 2019. URL https://products.office.com/en-us/excel. Microsoft. [Last accessed 17 March 2019].

IP[y]: IPython Interactive Computing, 2019. URL https://ipython.org/. IPython. [Last commit v7.3 18 February 2019].

The International 2018, 2019a. URL https://en.wikipedia.org/wiki/The_International_2018. Wikipedia. [Last accessed 17 March 2019].

The International (Dota 2), 2019b. URL https://en.wikipedia.org/wiki/The_International_(Dota_2). Wikipedia. [Last accessed 17 March 2019.

Jupyter, 2019. URL https://jupyter.org/. Project Jupyter. [Last updated 7 March 2019].

Map, 2019. URL https://dota2.gamepedia.com/Map. Gamepedia. [Last accessed 13 March 2019].

scikit-learn: Machine Learning in Python, 2019. URL https://scikit-learn.org/stable/index.html. Scikit Learn. [Last commit v0.20.9 March 2019].

Table of hero attributes, 2019. URL https://dota2.gamepedia.com/Table_of_hero_attributes. Gamepedia. [Last accessed 13 March 2019].

Nicholas Hanson-Holtry Albert Cui, Howard Chung. *OpenDota FAQ*, 2019. URL https://blog.opendota.com/2014/08/01/faq/. Sponsored by DotaCoach.org, VPGame.com, OpenAI, and Rivalry. [Last accessed 13 March 2019].

Amit Bhattacharyya. *Using Machine Learning for Predicting NFL Games*, 2016a. URL https://datadialogs.ischool.berkeley.edu/2016/schedule/using-machine-learning-predicting-nfl-games. University of California, Berkeley. [Last modified 7 November 2016].

Amit Bhattacharyya. *Using Machine Learning for Predicting NFL Games — Data Dialogs 2016*, 2016b. URL 2016,https://www.youtube.com/watch?v=8emUyzczThY. University of California, Berkeley. [Last modified 8 December 2016].

'davidmercury'. *hero picking and match result*, 2016. URL https://www.kaggle.com/davidmercury/hero-picking-and-match-result. GitHub. [Last commit v18, 2016].

'devinanzelmo'. *Dota 2 Matches*, 2016a. URL https://www.kaggle.com/devinanzelmo/dota-2-matches. GitHub. [Last commit v3 2016].

'devinanzelmo'. *Setting up a prediction problem in dota 2*, 2016b. URL https://www.kaggle.com/devinanzelmo/setting-up-a-prediction-problem-dota-2. GitHub. [Last commit v2 2016].

'devinanzelmo'. *Dota 2 skill rating with TrueSkill*, 2016c. URL https://www.kaggle.com/devinanzelmo/dota-2-skill-rating-with-trueskill. GitHub. [Last commit v5 2016].

Claudia Perlich. *Answer: What Are The Advantages Of Logistic Regression Over Decision Trees?*, 2017. URL https://www.forbes.com/sites/quora/2017/06/19/what-are-the-advantages-of-logistic-regression-over-decision-trees/#2fd145282c35. Forbes. [Last modified 19 June 2017].

Thore Graepel Ralf Herbrich, Tom Minka. *TrueSkill: A Bayesian Skill Rating System*, 2006. URL March2019,https://www.microsoft.com/en-us/research/wp-content/uploads/2007/01/NIPS2006_0688.pdf. Microsoft. [Last accessed 13 March 2019].

Cho-Jui Hsieh Xiang-Rui Wang Chih-Jen Lin Rong-En Fan, Kai-Wei Chang. *LIBLINEAR: A Library for Large Linear Classification*, 2017. URL https://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf. National Taiwan University. [Last modified 8 December 2017].

Aaron Schlegel. *Extraction and Feature Engineering of Animal Austin Center's Shelter Outcomes Dataset using Requests and Pandas*, 2018. URL https://aaronschlegel.me/extraction-feature-engineering-aac-data-requests-pandas.html. [Last modified 5 February 2018].

Mike Stubbs. *Dota 2s $100 million milestone, visualised.* URL https://www.redbull.com/se-en/dota-2-100-million-milestone-visualised. RedBull. [Last accessed 17 March 2019].

Yordan Zaykov Tom Minka. *TrueSkill Ranking System*, 2005. URL https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/. Microsoft. [Published 18 November 2005].

Bowen Yang. *Predicting e-sports winners with Machine Learning*, 2018. URL https://blog.insightdatascience.com/hero2vec-d42d6838c941. Insight Data Science. [Last modified 2 March 2018].