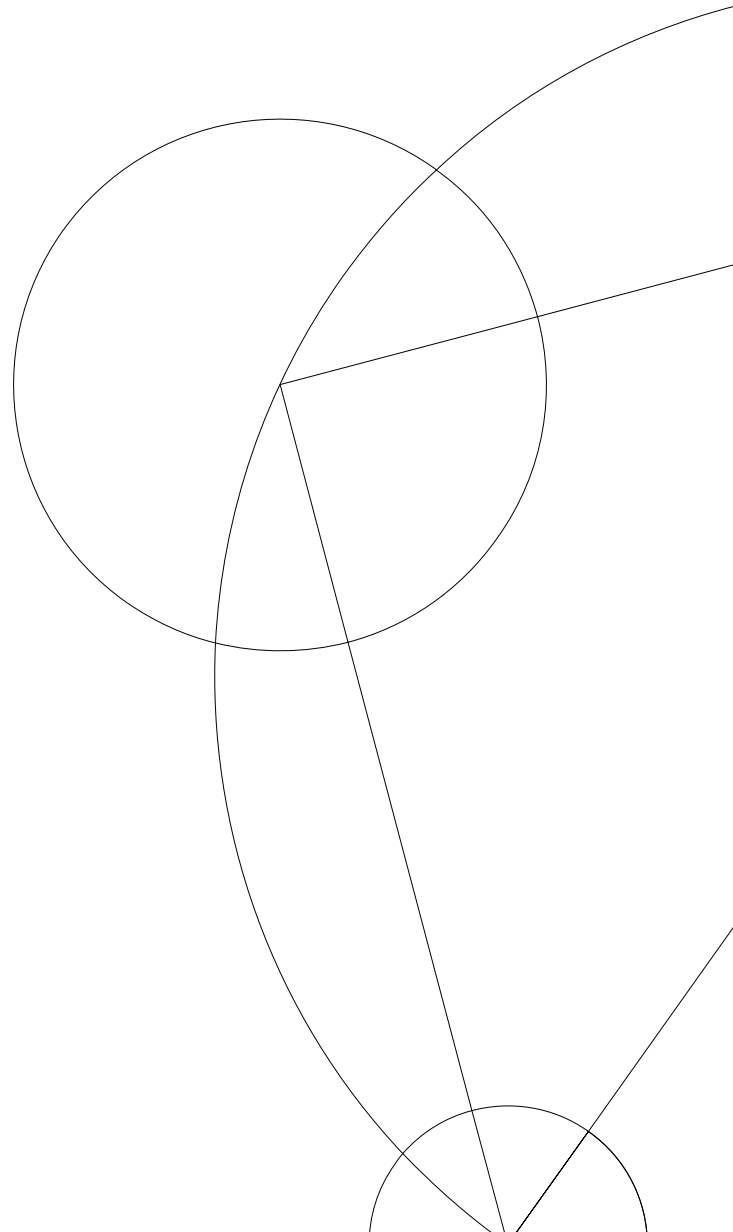# Optimizing Win Probability in Team Sports
Using Data Analysis and Machine Learning

Iversen, Bjarke Kingo
bjarkekingo50@gmail.com

Truong, Susanne
suzze-t@hotmail.com

December 10, 2018

Supervisor: Stefan Horst Sommer

1 Abstract

In this thesis we present three approaches for optimizing win probability in team sports. These approaches revolve around creating dynamic applicable algorithms for computing optimal team compositions, predicting win probability, and assessing player performance. Also, we present and discuss the underlying theory in order to make the algorithms easy to interpret and implement. We see that we can apply hierarchical clustering techniques and then utilize the *gap statistic* in order to cluster player profiles into roles, which can be used for several team sports. When doing live predictions on win probability, we construct several algorithms for predicting, which can likely be used for other team sports as well. We show that *logistic regression* and a modified *k-nearest neighbors* algorithm can be used for computing win probabilities. Finally, we show that player assessments can be made by applying dynamic regression models, taking roles into account that were derived from the hierarchical clustering process.

Contents

2 Introduction

In this thesis we will consider tasks related to optimizing win probability and improving a team's play style to positively affect the outcome of a team sports game.

Team sports have been around a long time. However, the popularity of sport events are continuously increasing as seen in the *Rio 2016 Olympic Games International Federations Report* [Com13]. The report states that a minute of soccer television coverage at the *2012 London Olympic Games* was watched by more than 19 million people globally on average. Also, the revenue of just the *English Premier League* was no less than 5297 million euro in the season of 16/17 [Del18].

With increasing popularity and revenue, a demand for optimal performance is expected for a team to compete and stay at the top of their league. Multiple scientific papers on things like aerobic endurance training for improving performance [HEWH01], and nutritional guidance [Cla94] already exist. A scientific paper on maximizing the likelihood of winning a penalty shootout was even written by McGarry and M. Franks [MF00]. However not much research seems to utilize the potential of data analysis and machine learning in order to make general models that can be used in a variety of team sports.

In this thesis we take a look at three major components that can be formalized for many team sports; team composition and roles, estimating winning probability, and player performance. Throughout the thesis, we will work with data sets from the popular online team sports game *Dota 2*.

Team composition, or formation of roles is the aspect of players having different key objectives within the game. A game like soccer makes it possible to have many different formations, with varying number of attackers, mid-fielders and defenders. We will investigate whether it is possible to distinguish between worse or better team compositions, in order to maximize the winning probability.

Live estimation of winning probability can help teams in time to make up decisions and test strategies to see if they approach a favourable outcome. We will investigate whether its possible to make models that output the probabilities of winning games based on live data.

An assessment of the players' performance can help coaches and players to improve their individual traits by pointing out which aspects that need improvement. Taking several factors that affect the game into consideration, we will investigate whether its possible to make models that give players fair assessments whether the resulting outcome was a win or a loss.

3 Description of the game and data set

To find ways to optimize the win probability in team sports, we will focus on data from the online game Dota 2. Each Dota 2 match takes place on a playing field that is compromised of two sides with two teams of five players placed on each side of the map (Radiant and Dire). They must try to gain an advantage over the other team with the goal of destroying the opposing team's base, named the "Ancient". Each base is protected by towers that attack any enemy who gets within its range and can be seen as checkpoints that the opposing team has to get through before reaching the base.

Figure 1: A view of the playing field which is seen as a miniature-map in the game [pla18].

The green and red squares represent the teams' towers and the one closest to the lower left and upper right corner represent the base. The lighter parts of the playing field depicts places where a player has vision. When enemies move around in the light parts, the player will be able to see the enemies in the miniature map. To gain extra vision, a player can buy and place utility items called *observers* and *sentries* on the playing field to gain vision in a certain radius around the items.

Each player controls a character and each character has its own unique abilities and role to fulfill. A role can be seen as the expected play style that a character has to accommodate to. A character's unique abilities provides tactical advantages and disadvantages at different situations in the game and this often reflects in the role. For example, a character with healing abilities or abilities that disable enemies would be put in a role where they have to support their team members [Dot].

A character can improve its abilities by earning experience points and gold by for example destroying towers or killing players from the opposing team. The gold can then be spent to buy items that provide new abilities or upgrades the current ones. Besides items, there are many other factors that decide the outcome of the game such as tactics, team coordination, and in what order a character's abilities are upgraded.

3.1 Game telemetry

A lot of information is saved about every match as data, since the collection of data is relevant for both game development and game research. By collecting and analyzing relevant data, we can gain a better insight of the game and how to improve a team's performance. The collection of data is also known as game telemetry and involves tracking attributes and objects that can help understand how the game is played. Game telemetry is an automated process of recording and receiving data remotely

from somewhere, such as the game client that submits information about how a player interacts with the game or the online payment system that submits data about a player's transactions.

Data can be split into different types such as player activity analytics that focuses on data for a single player or player group. This includes information about the behavior of players, such as their interaction with the game and other users, their physical movement in the game, or spending habits on virtual items. Another type of data is game system analytics that focuses on game mechanics instead of individuals. This includes information about things such as resources in the game, game economy, and issues of balance in the game. For example, if statistics show that some characters are chosen significantly more often than other characters, it could indicate that there is an issue with balance in the game.

To record the data, objects and their attributes must first be defined. For a match, objects could for example be the individual players or each team, while attributes would be information about the objects, such as their location, net worth, or experience gained. After defining objects and their attributes, the attributes must then be operationalized. That is, the attributes are turned into trackable variables or features which are measured in a specific way, such as in numbers, time measurements such as min/sec, or currency [Wal13].

After the data has been recorded, the data is available for collecting. While we have not been able to gain access to everything that the Dota 2 game client tracks for each match, we managed to gather some of the attributes using OpenDota; an open source data platform where advanced match data can be collected through calls to their API [Ope18]. While we will focus on how to optimize the win rate, the data could also be used for other purposes such as research about user behavior, game-balance, or how popular specific characters or items are.

3.2 Gathering the data

To begin with, we retrieved a list of randomly sampled public matches by sending a GET request to OpenDota for "/publicMatches". Each calls gives a list of 100 recent matches and we can then specify in each request what the match ids should be less than, by comparing with the lowest match id from the previous call. The matches who fit some specific conditions were then saved to the data set. The first condition was that the matches had to be categorized as ranked match of type 'All Pick' or 'Captains mode', as these are the game modes most often played for more serious players.

The casual-hardcore spectrum is a way of describing the many kinds of players in a game. The spectrum indicates a player's level of interest and skill in the game and how much time they have committed to the game [Wal13]. For example, the interested and casual players may know about the game and play a few times a month, while the other end of the spectrum contains hardcore players that are very skilled and know every little detail of a game. To ensure that the matches in the data set contain more serious and competitive players, the second condition was that the average skill level of players in a match should be around the 75% percentile [Dot18d].
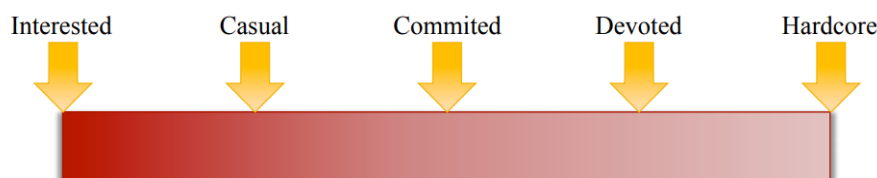
Figure 2: Casual-Hardcore spectrum [Wal13].

The match ids of the matches fulfilling both conditions were selected and after collecting the match ids, the detailed match data was sampled for each match id by sending a GET request to OpenDota for "/matches/{match_id}". The detailed match data contains several features which can be seen at docs.opendota.com. Some of the features were either not relevant or sometimes empty/null, so we only saved some specific features for each match. The "players" feature was too long for our program to be able to write it fully to a csv file, so we had to split that feature up into 10 features; one for each respective player and his/her character.

3.3 Features in the data set

The data set that was gathered using OpenDota API contains data from 117.615 Dota 2 matches that all took place in 2018. The matches were set to a recent time period as online games tend to change over time and the win rate optimization model could get affected negatively by using outdated data. When collecting the detailed match data, there were also a lot of different features where some are more useful than others for our problem. We therefore did some feature selection using our domain knowledge of the game, and selected features that seemed useful or relevant. For example, data about cosmetic items that characters were wearing was excluded from the data set, since cosmetic items only affect characters visually and have no effect on a player's mechanics or game play in a match.

The feature selection resulted in 19 features that includes details such as which team won, the duration of the game, and the characters chosen on each team. In some of the matches, there is also some live data available in form of gold and experience advantage/disadvantage, which is measured every minute in the game.

Feature	Description
radiant_win	True if Radiant team has won, else false
duration	Duration of the match
skill	Average skill level of the players
radiant_score	Number of player kills for Radiant team
dire_score	Number of player kills for Dire team
radiant_gold_adv	Amount of gold Radiant team is ahead/behind with
radiant_exp_adv	Amount of experience Radiant team is ahead/behind with
patch	The version of the game
region	Which regional server the match took place in
player0	*data about player 0*
player1	*data about player 1*
player2	*data about player 2*
player3	*data about player 3*
player4	*data about player 4*
player5	*data about player 5*
player6	*data about player 6*
player7	*data about player 7*
player8	*data about player 8*
player9	*data about player 9*

Table 1: Description of the features in the data.

Player Feature	Description
player_slot	A number that indicates which team the player belongs to
hero_id	ID of the character that the player has chosen
kills	How many times the player killed an enemy
deaths	How many times the player died
assists	How many times the player assisted in an enemy kill
hero_healing	How much a player increased their allies' current health
last_hits	Number of kills on basic units
gold_per_min	Average gold earned per minute
exp_per_min	Average experience earned per minute
tower_damage	Damage dealt to towers
hero_damage	Damage dealt to enemy players
obs_placed	Number of observers placed on the field
sen_placed	Number of sentries placed on the field

Table 2: Description of the features for each player.

4 Learning from data

Learning from data is used in situations where we do not have an analytic solution to a problem where the exact outcome can be determined by following a set of logical steps. Instead, we can use the data to make an empirical solution where we construct a model and predict the outcomes for new incoming data [YMH12a]. We can thus solve tasks with no exact solution using e.g. clustering, classification and regression.

4.1 The learning problem

The learning problem is the task of finding some model, using training samples, that faithfully produces an output given some new input (test samples). Faithfully means that the predicted output satisfactorily approximates the output of the function that maps the training samples to their outputs. To formalize the *basic* setup of the learning problem we must first define the components of the setup.

In this definition of the learning problem we are given some training data set $\mathcal{D}$ containing input and output samples $(x_1, y_1), \ldots, (x_n, y_n)$ where each x_i is an input vector containing d features. The number of features, that is the size of the input vector, is often referred to as the dimensionality of the data. Each y_i is the output often referred to as a label or class. Note that we can have cases in which there are no recorded output-samples. This is briefly described in section 4.3.3.

We have an unknown target function $f : X \to Y$ that perfectly maps the input data to some output, where X is the input space (set of all possible values of x), and Y is the output space (set of all possible values of y), such that $f(x_i) = y_i$ for $i = 1, \ldots, n$.

We also have a learning algorithm $\mathcal{A}$ that uses the data set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ to pick a formula $g : X \to Y$ that approximates f. The algorithm $\mathcal{A}$ chooses g from a set of candidate formulas which is called the hypothesis set $\mathcal{H}$ [YMH12a]. Now when we get new incoming test data $\{x_1, \ldots, x_n\}$, we use $g \in \mathcal{H}$, the hypothesis produced by $\mathcal{A}$, in order to e.g. classify or predict the outputs $\{y_1, \ldots, y_n\}$.
The outputs produced by our model g will be appropriate if g approximates f faithfully.

To determine whether g is a proper model, the performance can be evaluated by for example measuring the ratio of correct predictions. This is described further in section 4.5.

4.2 Types of analyses on data

Data analysis is a huge field, and in order to distinguish the different types of data analyses that can be done on a data set, one can consider professor in bio statistics, Jeffrey Leek's way of dividing the field into 6 sub-categories: Descriptive, exploratory, inferential, predictive, causal, and mechanistic data analysis [Lee15][J13].

- **Descriptive analysis**
  The descriptive analysis seeks to summarize and describe some data set in numbers without further interpretation. Typically, this is done by performing measures of central tendency or measures of dispersion.

  In measures of central tendency, we estimate the "center" of the distribution for the data set, i.e. what a typical value of a variable is. This is for example done by computing the mean or median [UoSAa]. As well as having an insight into what a typical value is for a variable, we would also to know how spread out the data is around that typical value. In measures of dispersion, we therefore compute numbers such as the range, quartiles, variance and standard deviation [UoSAb]. An example of this could be the FIFA-rankings of soccer-clubs as seen in figure 3.

Name	League	ATT	MID	DEF	OVR	Team Rating
FC Bayern	Bundesliga	91	85	86	86	★★★★★
Real Madrid	Liga BBVA	86	86	86	86	★★★★★
FC Barcelona	Liga BBVA	93	85	84	85	★★★★★
Juventus	Serie A TIM	87	84	84	85	★★★★★
Manchester City	Barclays PL	85	87	83	84	★★★★★
PSG	Ligue 1	86	82	82	83	★★★★★
Spurs	Barclays PL	85	83	82	83	★★★★★
Manchester Utd	Barclays PL	85	83	81	83	★★★★★

Figure 3: Snapshot of FIFA-rankings June 2018 [Fif18].

These FIFA-rankings ranks soccer-clubs by their overall rating, with features regarding the club name, playing league, attack rating, midfield rating, defence rating and overall rating. The interpretation of what this ranking tells you is then left to the viewer.

- **Exploratory analysis**
  The exploratory data analysis build on a descriptive analysis by searching for e.g. discoveries, trends or relationships the measurements of multiple variables. The analysis can be done by using graphical techniques such as box plots, histograms, scatter plots and dimensionality reduction in order to visually represent the data. The potential discoveries can then be used to define future study and questioning. A case of an exploratory analysis could be the following example, plotting 10 students rank in tennis versus their rank in soccer. The scatter plot in figure 4 could suggest that there is a positive relationship between an individuals skill in tennis versus their skill in soccer.
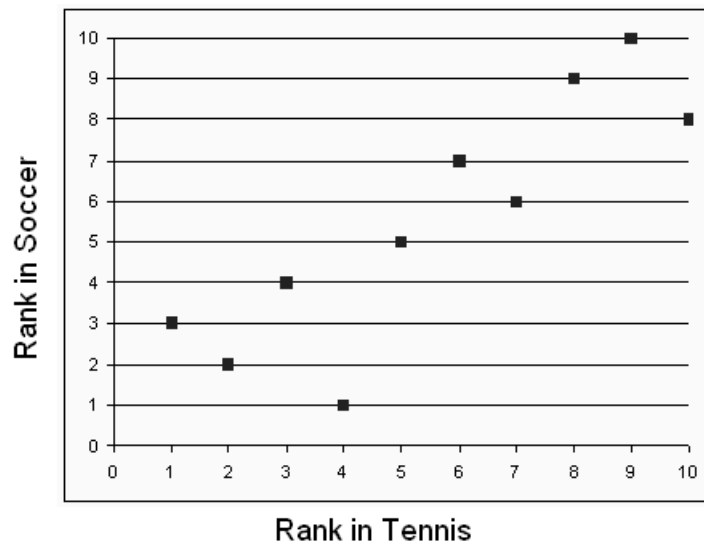
Figure 4: Students rank in tennis vs their rank in soccer [SAT].

- **Inferential analysis**
  It is usually difficult or impossible to measure every item in a population. Thus we have to make inferences about the population based on a sample. In the inferential analysis, we go beyond the exploratory analysis and quantify if the observed patterns are likely to extend beyond the sample at hand. This is only possible if the sample is representable of the population, and even then the sample is unlikely to represent the population in all aspects. There is thus a uncertainty in how much the sample represents the population and to quantify whether a relationship exists, we can use statistical models such as testing hypotheses [uwe18].

- **Predictive analysis**
  While the inferential analysis makes inferences about the population based on the sample, the predictive analysis uses the sample data set to predict some outcome of a future event, using methods such as classification.

- **Causal analysis**
  The causal analysis seeks to find out what happens to one measurement if you change another measurement. An example would be to randomize the type of serve a badminton player would make, and then measuring how it changes the performance of winning the point. Unlike predictive and inferential analysis, the causal analysis identifies both the magnitude and the direction of the relationship between variables.

- **Mechanistic analysis**
  While the causal analysis seeks to identify the average effects between often noisy variables, the mechanistic analysis aims to not only understand that there is an effect, but to demonstrate how that effect operates and has deterministic behaviour between some variables. That is understanding how exact changes in one measurement exclusively leads to changes in another measurement. An example would be of how kicking the soccer-ball would change the forces applied to the ball, ultimately determining the path and slope of the ball.

4.3 Machine Learning

Machine learning algorithms can be divided into four different types: supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning. Which type to use depends on many factors, such as computational time, the size and quality of the data, and what the data should be used for [Hui17]. In the following subsections, the four types will be explained.

4.3.1 Supervised learning

Supervised learning is about using labeled training data as examples of input-output pairs to learn a function that can predict labels for future input. Labels are typically added by some 'supervisor' and these labels guide the algorithm to find out which features are important for the prediction making and what the most optimal parameters for the function are [Cas18]. If the label has a limited number of values, it is defined as a classification task, while if the label is a real number, the task is defined as a regression task[Kor17].

4.3.2 Semi-supervised learning

Semi-supervised learning algorithms use a combination of labeled and unlabeled data to train the prediction model. This can be beneficial for cases containing large amounts of unlabeled data, since labeling all the data for supervised learning is likely to be very time-consuming. There is also a risk of establishing cognitive bias on the model when labeling all the data, since human mistakes can happen when making a judgment of what the labels should be. With semi-supervised learning, the final model can therefore be more accurate than in supervised learning [Cas18].

4.3.3 Unsupervised learning

In unsupervised learning, the training data does not contain any outputs at all, so there are no informative outputs to guide the machine learning algorithm. Instead, the algorithm trains on unlabeled data to learn something from the inputs themselves. The goal is to discover patterns and structures within the data; for example to identify and categorize samples that have similar properties together [YMH12a].

4.3.4 Reinforcement learning

In reinforcement learning, the training data does not explicitly contain the correct output for each input. Instead, the training data contains some possible output along with a measure of how good that output is. However, this does not explain how good other outputs would have been for that given input. Reinforcement learning is particularly useful for learning how to play a game [YMH12a].

For example, it can be difficult to decide what the most optimal action is in a game of chess at a given time of the game. Therefore we just take some action and make a training example where we report how well things went for that action. The task for the reinforcement learning method is then to figure out the best way to play based on the different training examples [YMH12a].

4.4   Normalization

Learning by using distance metrics can be extremely biased if the dimensions are of much varying units and variance (i.e. price be in range $[0, \infty)$ dollars and rating be in range $[0, 5]$ stars). Thus normalization by standardization can be applied to make the training part less sensitive to scaling.

Let $f_j = (x_1, \ldots, x_n) \in \mathbb{R}^n$ be a feature containing n data points. Standardization by z-score is the act of mapping the features of the data set $X = (f_1, \ldots, f_k) \in \mathbb{R}^{n \times k}$ to $X_{norm} = (f_{1,norm}, \ldots, f_{k,norm}) \in \mathbb{R}^{n \times k}$, making the data have zero mean and unit variance along all the features of the data set.
Let the empirical mean for a feature f_j be denoted by

$$\bar{f}_j = \frac{1}{n} \sum_{i=1}^{n} f_{j,i}$$

Let the empirical variance for a feature f_j be denoted by

$$v_j = \frac{1}{n} \sum_{i=1}^{n} (f_{j,i} - \bar{f}_j)^2$$

Then we set

$$f_{j,norm} = \left(\frac{f_{j,1} - \bar{f}_j}{\sqrt{v_j}}, \ldots, \frac{f_{j,n} - \bar{f}_j}{\sqrt{v_j}} \right)$$

for each feature of the data set to obtain [SS14a]

$$X_{norm} = (f_{1,norm}, \ldots, f_{k,norm})$$

4.5   Performance measures for supervised learning methods

Performance measures are important for testing the strength of the learning algorithm. In this section we briefly present and explain some of the widely known performance measures who we may apply or consider throughout this thesis.
Considering classes of winning/losing matches we primarily focus on binary classification, and thus define the following 4 terms, that is, possible outcomes of testing a binary classification algorithm:

- **(TP) True Positive**:
  The algorithm predicts win and the game results in a win.

- **(TN) True Negative**:
  The algorithm predicts loss and the game results in a loss.

- **(FP) False Positive**:
  The algorithm predicts win and the game results in a loss. Also known as a type I error.

- **(FN) False Negative**:
  The algorithm predicts loss and the game results in a win. Also known as a type II error.

4.5.1   Confusion Matrix

A confusion matrix, despite the name, gives an overview of how many of the different cases the classification algorithm produced, represented by a matrix. It gives the ability to examine quickly the performance at a per-label level. Each (row, column) in the confusion matrix shows the number of times, and fraction of times that row was classified as column. Thus all ratios per row sum to 1. The diagonal represents the true rates of the classifier, and thus hopefully most of the probability mass lies here, indicating a good classification model [CS16a].

	Predicted positive (win)	**Predicted negative (loss)**
Actual positive (win)	$|TP|$, $\frac{|TP|}{|TP|+|FN|}$	$|FN|$, $\frac{|FN|}{|TP|+|FN|}$
Actual negative (loss)	$|FP|$, $\frac{|FP|}{|FP|+|TN|}$	$|TN|$, $\frac{|TN|}{|FP|+|TN|}$

Table 3: Binary classification confusion matrix. Each (row, column) shows the count of classified cases and the ratio with respect to actual class.

4.5.2 Precision & Recall

Precision $\in [0, 1]$ is a ratio measure that describes how many matches that the algorithm correctly predicted to class A to how many matches that the algorithm predicted to class A. That is a high precision tells us that the algorithms choice of class A tends to be actual class A's. For the binary classification, the prediction on the winning class can be computed by

$$Precision_{win} = \frac{|TP|}{|TP| + |FP|}$$

and

$$Precision_{loss} = \frac{|TN|}{|TN| + |FN|}$$

for the losing class.

While precision tells us how precise it classifies a specific class, it doesn't tell us something about whether its good at capturing all the instances of the actual specific class.

Recall $\in [0, 1]$ is a ratio measure that describes how many matches that the algorithm correctly predicted to class A to how many matches that are actually of class A. That is a high recall tells us that the algorithm is a good at identifying class A. For the binary classification, the recall for the winning class can be computed by

$$Recall_{win} = \frac{|TP|}{|TP| + |FN|}$$

and

$$Recall_{loss} = \frac{|TN|}{|TN| + |FP|}$$

for the losing class.

A good evaluation may thus often require both precision and recall to be high, in order for the classification the be good. This tandem ability can for instance be considered via. the $F\beta$ score [CS16b].

4.5.3 $F\beta$ score and $F1$ score

There tends to be a tradeoff between precision and recall and so its natural to often combine them. One measure that does this is the $F\beta$ score $\in [0, 1]$ that combines precision and recall and weights precision higher or lower than the recall depended on the chosen $\beta \in \mathbb{R}^+$. The $F\beta$ score can be computed by

$$F\beta \text{ score} = \frac{1}{\frac{\beta^2}{\beta^2+1}\frac{1}{Recall} + \frac{1}{\beta^2+1}\frac{1}{Precision}} = \frac{(\beta^2 + 1)Precision \cdot Recall}{\beta^2 Precision + Recall}$$

In order to penalize precision and recall equally we can use a special version of $F\beta$ score called $F1$ score where we set $\beta = 1$ and compute the harmonic mean between the precision and recall. The $F1$ score can thus be computed by [CS16b]

$$F1 \text{ score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

4.5.4 Accuracy score

Accuracy $\in [0, 1]$ is a ratio measure that describes how accurate the classification algorithm classifies all classes in tandem. That is, it is the number of correct predictions to the number of total predictions and is thus the sum of the diagonal in the confusion matrix to the sum of the whole matrix.

$$Accuracy = \frac{|\text{Correct predictions}|}{|\text{Total predictions}|}$$

In the binary setup we can compute the accuracy by

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}$$

However, the accuracy score assumes equal distributions, and wouldn't represent a fair performance if extreme difference in the number of actual classes is present.

Sometimes the term 'error rate' or 'misclassification rate' may also be used and i simply defined by $1 - Accuracy$.

4.5.5 R^2 score

R^2 measures the proportion of variation in the dependent variable y that is explained by the linear regression model and is conveniently scaled to be between 0 and 1.

Let TSS be the total squared variation of y_i's from their mean. This is also known as the total sum of squares

$$TSS(y_i) = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

Let SSR be the sum of squared residuals

$$SSR = \sum_{i=1}^{n}(y_i - \bar{y}_i)^2$$

The fraction $\frac{SSR}{TSS}$ then displays the proportion of variation in y that is not explained by the linear regression model. The R^2 score is then obtained by subtracting the fraction from 1, as that gives the proportion of variation in y explained by the linear regression model [Gru15a]:

$$R^2 = 1 - \frac{SSR}{TSS}$$

If the sum of squared residuals happens to be equal to the model's total sum of squares, the model performs no better than just predicting the mean. And in this case, the R^2 score would be zero. Our model should be at least as good as a model that just predicts the mean, which means that R^2 should

be at least zero [Gru15a]. In practice, the score is not always at least 0, but may be negative when the model is bad at predicting and *SSR* is large. In these cases, we can opt for using the mean value as response instead of the response derived from the regression calculation. This would make the score zero and the assumption of R^2 being at least zero would therefore still hold [Cos].

The sum of squared residuals must be at least zero, where a value of zero indicates that the predicted responses are exactly the same as the actual responses. In this case, the fraction will also be zero which means the R^2 score can be at most one. The higher the R^2 score is, the better our model fits the data [Gru15a].

5 Using clustering to group characters into roles

5.1 Motivation

As of June 2018, there are currently 115 playable characters available that players can choose from in each match. A character may also be referred to as a *hero*. The number of possible character combinations for a team of five are therefore enormous and choosing a favorable team composition can be difficult, especially if being under time pressure. Instead, the challenge of selecting a composition of heroes that increase the probability of winning can be simplified to what roles a team could benefit from. This can be done by looking at which role(s) a specific character usually represents within the game.

To group heroes into general but distinct roles, we may consider a cluster analysis, also known as clustering. It is an unsupervised learning method for dividing data samples into a number of groups (clusters) such that samples in one cluster are more similar to each other than the samples in other clusters, given some criteria. In this case, a cluster is equal to a role and by performing clustering, we aim to group heroes together that are similar such that the roles can be used to simplify the task of forming a team composition [SS14b].

5.2 Data preprocessing and selecting an appropriate clustering algorithm

In the data set containing Dota 2 matches, the heroes are identified by a unique ID. This is not useful as a feature, since a unique number does not describe much about a hero's attributes or role in the game. Instead, we want to segment the different heroes into what roles they belong to. For the cluster analysis, we sample only the features specifically related to the ten players from the detailed match data, as described in section 3.3. We then strip the data for rows where details about features such as *obs_placed* and *sen_placed* are missing. This gave us 36.653 rows in total. The rows were then expanded into a new data set, where the nested features within each player-field was turned into columns. Each row of the data set then displays a record of a player as seen below. The new data set totaled 366.514 rows. (This is not exactly a factor ten of the number of matches, due to some corrupted rows in the data set).

	hero_id	kills	deaths	⋱
player 1 from match 1	51	0	2	⋱
player 2 from match 1	41	0	1	⋱
⋮	⋮	⋮	⋮	⋱
player 10 from match 1	90	2	1	⋱
player 1 from match 2	68	3	2	⋱
⋮	⋮	⋮	⋮	⋱

Table 4: Structure of the data set containing player samples obtained from all matches.

To cluster the data, one can choose between many clustering algorithms and the two most popular methodologies are connectivity-based clustering and centroid-based clustering [TKP+15]. In connectivity-based clustering, the similarity of data points is based on the distance between the data points. The closer the data points are to each other, the more similar they are. The advantage is that the number of clusters does not need to be known beforehand, but can be decided by looking at the output. The disadvantage is however that connectivity-based clustering is very sensitive to outliers

and does not scale well for large data sets [San16]. Connectivity-based clustering is also known as hierarchical clustering.

In centroid-based clustering, the number of clusters to detect has to be specified in advance in this algorithm. The algorithm then runs iteratively to estimate the optimal mean of the different clusters, which is also known as cluster centroids. The data points are then assigned to the cluster whose centroid they lie the closest to.

To choose an appropriate method, one must look at the data available. In this case, we do not know how many roles the heroes should be split into, which makes the number of clusters unknown. Hierarchical clustering would therefore be a viable choice here for grouping the data into different clusters, since it is not necessary to specify the number of clusters beforehand. However, the data set contains 366.514 rows and can therefore be considered too large for hierarchical clustering due to computation time/memory limitations. To overcome this issue, the data points were grouped by hero id and for each id, the estimated arithmetic mean was computed for the features shown below to get an average statistics of the specific character. This reduced the data set to 115 rows; one for each character in the game. This also prevents heroes from being grouped into multiple clusters which can occur due to wide different performance between individual players.

Feature	Minimum	Max	Mean	Std
kills	2.93	12.50	7.34	2.67
deaths	5.38	10.88	7.94	1.21
assists	6.16	23.70	13.80	3.27
hero_healing	5.34	11 121	999.26	2 049.32
last_hits	31.99	379.50	163.92	86.56
gold_per_min	275.33	703.00	432.01	97.76
exp_per_min	375.86	860.49	519.63	83.58
tower_damage	122.88	8 713.53	2 316.10	1 927.19
hero_damage	7 830.46	43 617.87	20 287.35	6 940.14
obs_placed	0.05	11.53	3.05	3.50
sen_placed	0.09	11.75	2.80	3.49

Table 5: Descriptive statistics of the different features.

The table displays some descriptive statistics about the features of the player data set. It is seen that for example the mean and standard deviation is vastly different for the different features since they are being measured in different units. To avoid that the clustering result is being too dependent on features with large difference in mean and variance, the data was normalized to have mean zero and standard deviation one.

Potential outliers in the data set were not removed prior to the average statistics computation, since it cannot be proven that outliers are due to measurement or typing errors rather than unusual player behavior. At the time the data was gathered (June 2018), there was a pool of 115 available characters that players could select from in the game [Dot18c]. Assuming that the occurrence of each character is roughly the same in the data set of 366.514 rows, there will be approximately $366.514/115 \approx 3187$ observations of each character in the data set. It can therefore also be argued that a couple of outlier points for a specific character does not have a big statistical impact on the mean if the number of observations for each character is large.

5.3 Hierarchical clustering

Connectivity-based clustering is also known as hierarchical clustering and is a cluster analysis method that builds a hierarchy of clusters based on connectivity. The most common approach is agglomerative (bottom-up), where each data point is assigned to its own cluster. To move up the hierarchy, the two most similar clusters are merged into the same cluster, and the merging continues until only one cluster is left. The clustering is usually visualized with a dendrogram that shows how the clusters were merged together. Each merge between two clusters is displayed with a horizontal line and its height represents the distance between these two clusters. In other words, the higher the horizontal line stands, the lower the similarity is. After clustering, the dendrogram can be cut at a chosen height since hierarchical clustering does not partition the data set into a prespecified amount of clusters. When a dendrogram is cut at a certain height, each connected component forms a cluster [noa][Sau16].
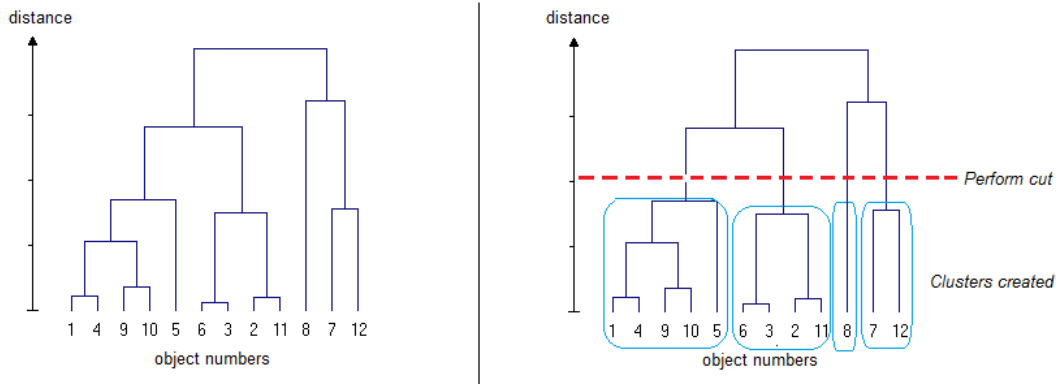
Figure 5: Example of a dendrogram and how clusters are found.

The hierarchical clustering was performed using the built-in functions *linkage* and *dendrogram* from scipy.cluster.hierarchy [Sci18a], which performs hierarchical clustering in an agglomerative approach and plots the result as a dendrogram. The similarity of data points were found by using Euclidean distance which is one of the most commonly used metrics for numeric data. And Ward's method was chosen as the linkage criteria for determining which pair of clusters are merged, since Ward's method is suitable for problems where data has to be categorized into types. At each iteration, a new cluster is made that minimizes variance.

Let u be a newly formed cluster of clusters s and t. Let v be any cluster that is not u. And let T be the number of clusters, $T = |v| + |s| + |t|$. The distance between u and each v is then calculated by [Sci18b]:

$$d(u, v) = \sqrt{\frac{|v| + |s|}{T} d(v, s)^2 + \frac{|v| + |t|}{T} d(v, t)^2 - \frac{|v|}{T} d(s, t)^2}$$

Algorithm: Hierarchical Clustering (agglomerative method)

Input: Set of data points $X = \{x_1, \ldots, x_n\}$, Distance function $d(u, v)$.

1 Let each data point be in its own cluster and let C be the set of clusters.

2 **While** $C.size > 1$**:**

3   Combine two clusters u and v with the smallest distance in C into a new cluster $w = \{u, v\}$.

4   Remove u and v from C and add w to C.

Output: A hierarchy depicting the history of the cluster formation.

The resulting dendrogram is depicted in figure 6 and shows the different characters each starting in their own cluster, identified by an id at the x-axis. When increasing the distance, clusters who are

closest are merged together until all of the characters are contained in the same cluster. Choosing the number of clusters based on the dendrogram gives us the number of roles that characters can be grouped into, where characters of similar characteristics are grouped into the same role. By knowing what roles there exists in the game, we will be able to analyze team compositions and investigate what roles a team can benefit from. However, it is not intuitive how many clusters would be a good selection by looking at the dendrogram. 2, 3, 6 and 7 all seem to be some great values for k, but we would like to use a mathematical argument for the optimal choice of clusters and thus how many roles there might be.
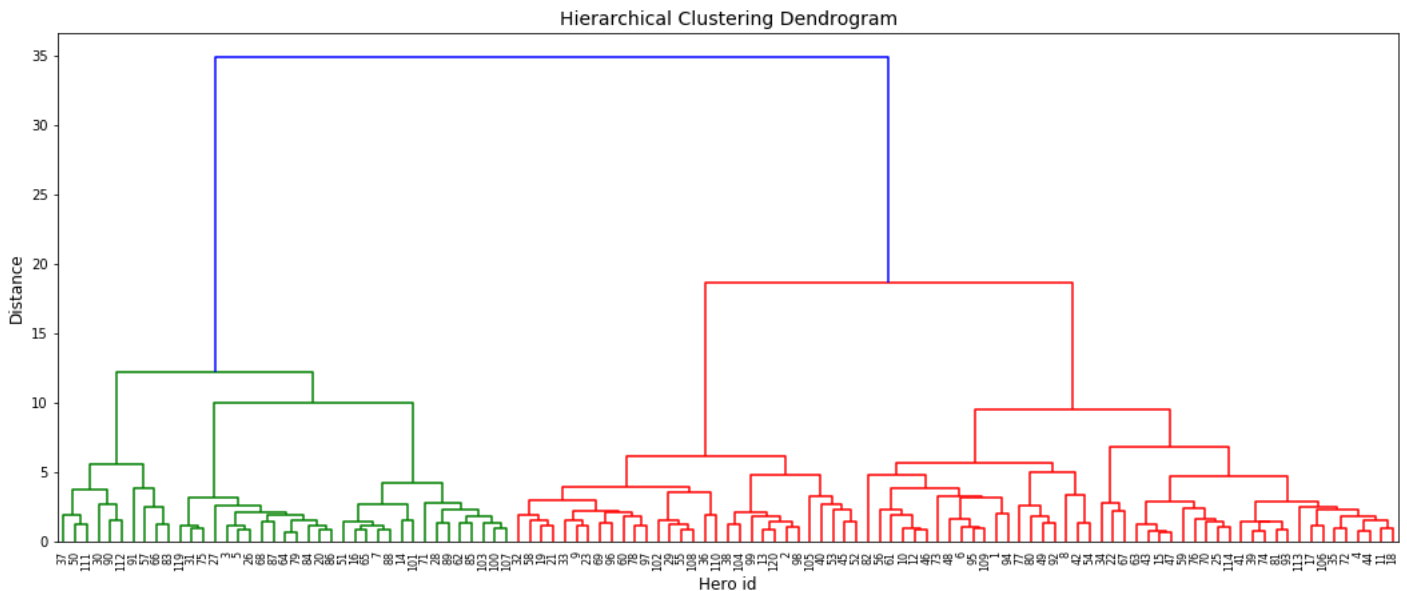
Figure 6: Dendrogram result.

5.4 Using the gap statistic to choose the number of roles

Analyzing the number of clusters is a bit ambiguous, since the number of clusters is really subjective and subject to the problem. One way of estimating the "optimal" number of clusters can be achieved through the gap statistic method by Tibshirani, Walther and Hastie. The gap statistic can be used for any clustering method (i.e. k-means clustering, hierarchical clustering) and a simulation study shows that gap statistic usually outperforms other methods, such as the Elbow method or the average silhouette method [RWH00].

In the implementation of gap statistic, the maximum number of clusters to consider k_{max} was set to 10. This is due to the optimal number of clusters seeming to be 7 clusters or less by looking at the dendrogram result. By setting a limit on how many clusters to split the data into, it can be avoided that there are many roles who share similar characteristics.

The gap statistic works by comparing the observed total variation within a cluster with the expected value for reference data where there is no obvious clustering. The comparison is done for different values of k (number of clusters) and the reference data set is generated using Monte Carlo simulations of the sampling process. That is, we generate samples where their values are randomly selected between the min and max values of the data samples in the original data set [RWH00].

Algorithm: Estimate best k for clusters

Input: $k_{max} \in \mathbb{Z}^+, B \in \mathbb{Z}^+, X \in (\mathbb{R}^n \times \mathbb{R}^d)$, clustering-algorithm $\mathcal{A}$.

1 Cluster X using $\mathcal{A}$ with varying number of clusters from
$k = 1, \ldots, k_{max} \implies \{C_1\}, \{C_1, C_2\}, \ldots, \{C_1, C_2, \ldots, C_{k_{max}}\}$ and compute their corresponding W_k's.

2 Generate B reference data sets and cluster them using $\mathcal{A}$ with varying number of clusters from
$k = 1, \ldots, k_{max} \implies \{C_1\}, \{C_1, C_2\}, \ldots, \{C_1, C_2, \ldots, C_{k_{max}}\}$ and compute their corresponding W_{kb}^*'s.

3 For each $k = 1, \ldots, k_{max}$ compute $Gap(k) = \frac{1}{B} \sum_b ln(W_{kb}^*) - ln(W_k)$.

4 For each $k = 1, \ldots, k_{max}$ compute $s_k = \sqrt{\frac{1}{B} \sum_b \left(ln(W_{kb}^*) - \frac{1}{B} \sum_b ln(W_{kb}^*) \right)^2} \cdot \sqrt{1 + \frac{1}{B}}$.

Output: Smallest k such that $Gap(k) \geq Gap(k+1) - s_{k+1}$.

Let $X \in (\mathbb{R}^n \times \mathbb{R}^d)$ be a matrix containing n samples of vectors $x_i \in \mathbb{R}^d$ having d features each. Let $d_{ii'}$ denote the distance between sample i and i'. One distance metric, and the most common choice for $d_{ii'}$ is the squared Euclidean distance that places progressively greater weights on samples that are farther apart. For our purpose we use the standard Euclidean distance since our usage of Ward's method for hierarchial clustering uses standard Euclidean distance. We define $d_{ii'}$ to be

$$d_{ii'} = \sqrt{\sum_{j=1}^{d}(x_{ij} - x_{i'j})^2}$$

Now suppose that we have clustered the data into k clusters $C_1, C_2, \ldots, C_k$ with C_r denoting the indices of samples in cluster r. Let

$$D_r = \sum_{i,i' \in C_r} d_{ii'}$$

be the sum of the pairwise distances for all points in cluster r. That is the sum of either the upper or lower triangle in C_r's distance matrix (or the full sum divided by 2). Now set

$$W_k = \sum_{r=1}^{k} \frac{1}{2|C_r|} D_r$$

and we call W_k the "pooled within-cluster Euclidean distances around the cluster means".

The idea is to standardize the graph of $ln(W_k)$ by comparing it with its expectation under an appropriate null reference distribution of the data.
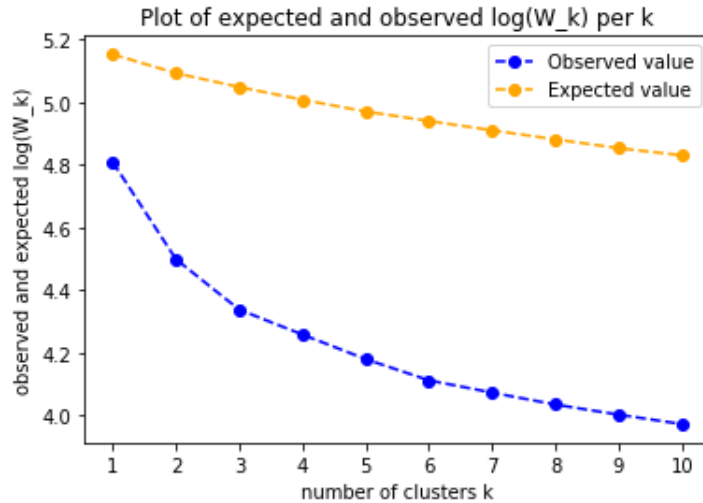
Figure 7: Observed and expected $ln(W_k)$ value per k for the clusters produced by our hierarchical clustering process.

The gap statistic for a given k is defined by

$$Gap_n(k) = E_n^*(ln(W_k)) - ln(W_k)$$

where $E_n^*(ln(W_k))$ denotes the expectation of $ln(W_k)$ under a sample size n from the reference distribution, obtained from creating B reference data sets using Monte Carlo samples. Let $b = 1, 2, \ldots, B$, we can then formalize $Gap_n(k)$ to be

$$Gap(k) = \frac{1}{B} \sum_b ln(W_{kb}^*) - ln(W_k)$$

After the estimated gap statistic was computed for each $k = \{1, \ldots, 10\}$, the standard error was also computed for each k in order to choose the optimal number of clusters. We now define

$$\bar{l} = \frac{1}{B} \sum_b ln(W_{kb}^*)$$

and the standard deviation of k to be

$$\sigma_k = \sqrt{\frac{1}{B} \sum_b (ln(W_{kb}^*) - \bar{l})^2}$$

and the standard error to be

$$s_k = \sigma_k \sqrt{1 + \frac{1}{B}}$$

Finally we can choose our number of clusters k to be [RWH00]

$$\hat{k} = \text{smallest } k \text{ such that } Gap(k) \geq Gap(k + 1) - s_{k+1}$$

The smallest k thus that $Gap(k) \geq Gap(k + 1) - s_{k+1}$ was true, were found to be $k = 6$. The gap statistic therefore estimates that the optimal number of roles to divide the 115 characters into is 6.
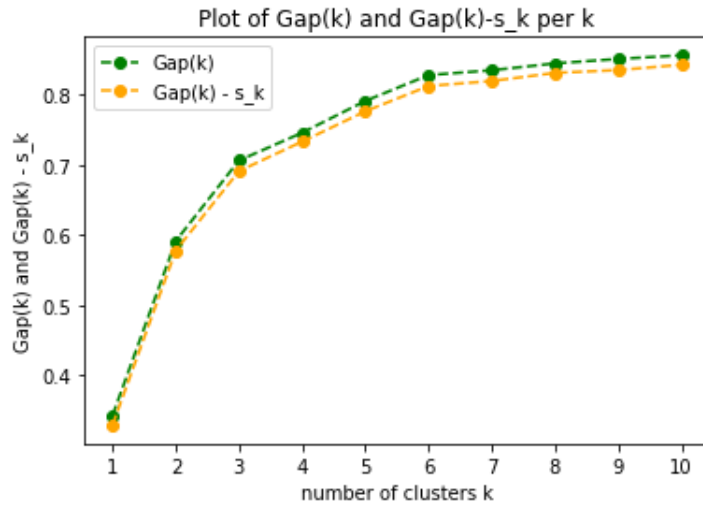
Figure 8: Plot of $Gap(k)$ and $Gap(k) - s_k$ per number of clusters, k.

5.4.1 Choosing the value *B* for number of Monte Carlo simulations

When *B* reference data sets had to be generated for gap statistic, it is hard to find exact mathematical arguments for the number of Monte Carlo simulations that should be done for the sampling process. In our case we have generated the normalized mean-features for each game-character, for all 115 characters. That is our data $X \in \mathbb{R}^{115 \times 11}$. We bound the number of clusters to be at max 10, since we want no more than 10 possible classes of this unsupervised learning process to describe the different types of roles in this particular setup. Given our data X and $k_{max} = 10$, we simulate the following Monte Carlo simulations, with varying number of reference sets *B*, plotting $E_n^*(ln(W_k))$ from $k = 1, \ldots, 10$.
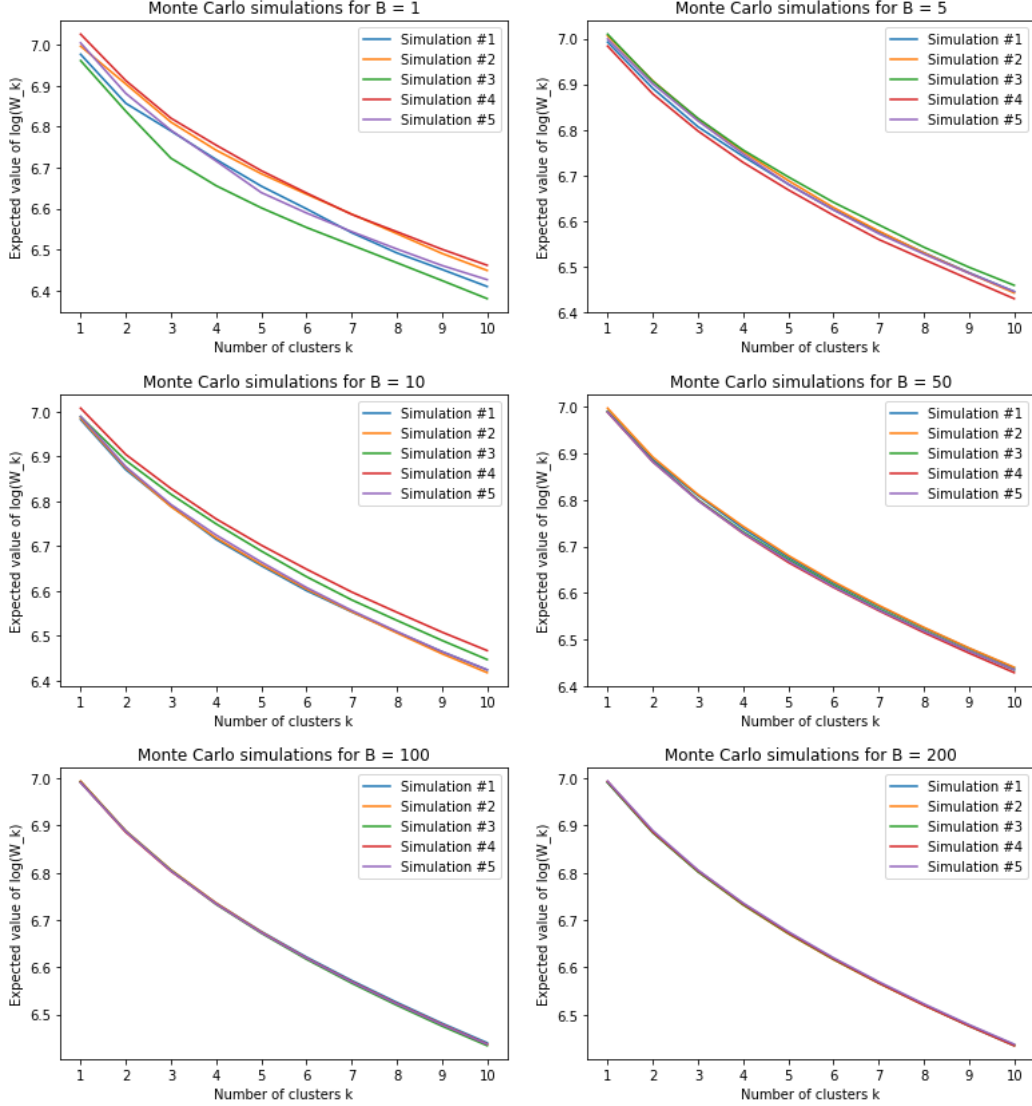
Figure 9: Simulation results of different values of reference data sets *B*.

We see that with low values of *B*, the random sampling processes differs noticeably from each other. Thus we choose $B = 100$ Monte Carlo simulations for the gap statistic in order to suppress the outliers of this random sampling process, but still limiting the cost of the computation. We finally have $100 \times \mathbb{R}^{115 \times 11}$ Monte Carlo samples that can be used to compute a satisfactory value of $E_n^*(ln(W_k))$. Having a high value of e.g. $B = 500$ gives quite precise results so that the gap plot is basically unchanged after an another run [Sta], however that also increases the computation time by alot.

5.5 Analysis of the result / different roles

The optimal number of clusters using the gap statistic was found to be six clusters. By grouping many characters into specific roles, coaches can for example switch a character out with another character of same role, while maintaining the same team composition. This could be useful in matches where certain characters are banned and alternatives must be found.

Now that the clusters have been found, and thus the different roles, the next problem is to find out what each cluster represents. By looking at the descriptive statistics of the features for the six different roles, it is possible to characterize each role. Recall from section 3.3 that hero healing is the act of increasing team members' health, while observers and sentries are utility items that provide extra vision on parts of the playing field. Generally, the first three roles appear to be roles that supports other characters, while the the fourth to sixth role are more fighter-oriented.

Feature	Role 1	Role 2	Role 3	Role 4	Role 5	Role 6
kills	4.00	4.69	5.52	6.70	8.96	10.60
deaths	8.25	9.29	8.98	8.28	6.31	7.35
assists	15.60	16.85	17.44	14.10	10.23	11.75
hero_healing	6 387.63	255.43	607.28	811.01	602.73	75.69
last_hits	69.91	59.03	75.15	157.86	264.472	237.54
gold_per_min	318.76	317.16	337.67	416.85	553.36	515.99
exp_per_min	416.30	422.17	448.20	509.05	611.12	594.88
tower_damage	913.69	806.19	520.25	1 739.81	5 424.62	3 016.84
hero_damage	10 917.59	13 438.34	15 297.37	20 742.85	22 343.24	28 350.82
obs_placed	7.93	9.88	3.70	1.72	0.45	0.66
sen_placed	7.67	9.58	3.27	1.51	0.32	0.39

Table 6: Mean values for different traits corresponding to a specific role.

For the supporting roles, role 1 is characterized by a high amount of hero healing and also many observers and sentries placed, but with low scores in most of the other features. Role 1 can therefore be seen as the most supporting role by providing much vision on the field and healing other characters. Role 2 also seems to contain supporting characters that provides vision on the field, but differs in that more damage is applied to enemies but significantly less healing to allies on average. The second role can therefore be seen as a more offensive support. Role 3 can be viewed as the middle ground between a support and fighter with characteristics such as more more kills and hero damage than the other two support roles, while still placing a few observers and sentries on the field.

For the fighter-roles, role 4 can be seen as a semi-fighter with moderately high scores in for example last hits and gold/experience earned per minute. Role 5 can be categorized as a fighter with specialty in dealing high amounts of tower damage. Lastly, role 6 is the most fighter-oriented with considerably more kills and hero damage than the other roles, while also having a high gain of gold and experience per minute.

Whether the clusters created are good or bad is subjective. We wanted to find roles that are distinguishable from each other, which can be said to be a success in this case. There is a reasonable amount of dissimilarity between data points belonging to different clusters. To see how similar data points within the same cluster are to each other, one possible method is to compute the standard deviation.

Another indicator of that a good cluster result has been achieved is by looking at the size of the different clusters. As seen in figure 10, the size of the clusters varies between 8.7% and 26.1% of the data points with role 1 containing the fewest characters and role 4 containing the most. This shows that the proportion of data points are well spread among the clusters, and indicates that there are no extreme outliers in the data.
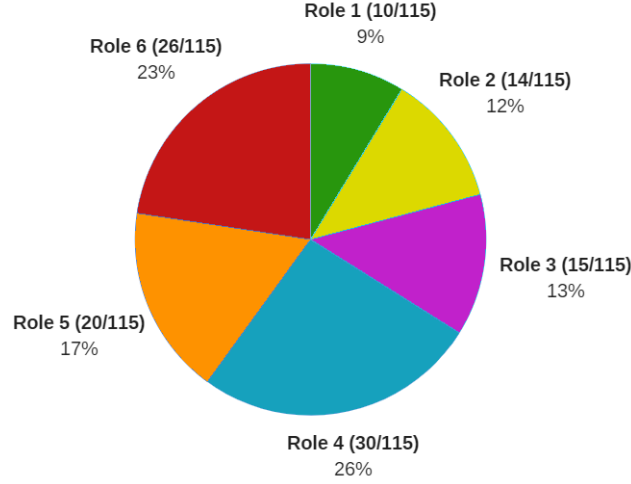
Figure 10: Distribution of cluster sizes.

5.6   Exploratory data analysis of team-composition

Using the result from the gap-statistic, we cut out the six clusters from the hierarchical clustering process and retrieved the role-distribution of each individual match. The goal is then to see how roles affect a team's win rate in general, and how well a specific team composition works both independent and dependently of the opposing team. The number of possible team-compositions, that is, the number of multisubsets of size k, when a composition consists of $k = 5$ players whom each can choose between $n = 6$ different roles, becomes the number of non-negative integer solutions to the Diophantine equation

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 5$$

which is the number of combinations with replacement/repetition, but where the order of which player has which role does not matter. This can more conveniently be computed by

$$C^R(n, k) = \binom{n}{k} = \binom{n + k - 1}{k} = \frac{(n + k - 1)!}{k!(n - 1)!}$$

In our case we thus have

$$C^R(6, 5) = \binom{6}{5} = \binom{10}{5} = \frac{10!}{5!5!} = 252$$

number of team compositions for a 5-player team. As some compositions are more often played than others, there are some compositions that are either not found in our data set or played very rarely. To avoid analyzing matches where the win rate is largely due to randomness because of low match count, we have decided to only look at compositions that have been played in a reasonable amount of matches. This bar was set at 50 matches.

To see how the win rate is affected by how many heroes a team has from a specific role, the average win-rate was computed for games where the team contained from 0 to 5 heroes from that role respectively. Some team compositions were either not found or rarely found in the data set, which results in the following plots having incomplete data for the six roles. This indicates that teams prefer to have a diversity of roles when making a team combination.
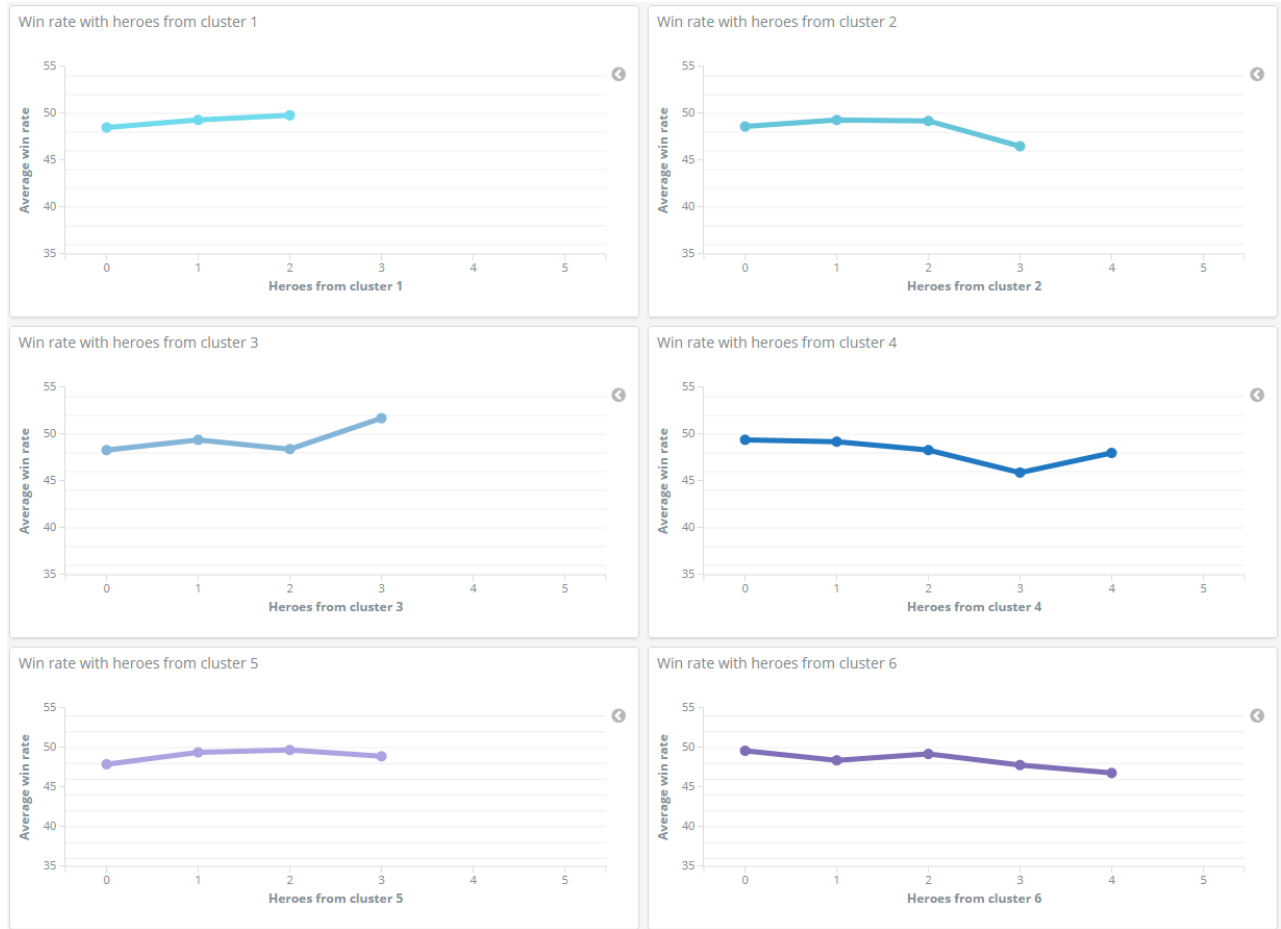
Figure 11: Win rate for team combinations containing heroes from different clusters.

Looking at the plots, there also seems to be a tendency of a declining win rate when a team contains many players of the same role, with the exception of cluster 3. For example, when looking at the win rate with heroes from cluster 6, the win rate is at its lowest when there are four heroes of role 6 in a team composition. Role 6 was the most fighter-oriented role and the low win rate could be a result of not getting enough gold and experience to fight effectively, since teams containing other roles are less dependent on items and experience. Overall, the win rate seems to be relatively stable for teams containing 0-2 players of the same role, indicating that it is more optimal to have a variety of roles, but that there is not always much difference between whether a team has 1 or 2 from a specific role.

After clustering the classes, we also computed a team composition matrix of size 252 x 3 to see if there is a correlation between teams that generally do well or do badly, independently of the team faced in a match. The matrix contains each possible composition, how many matches the composition has been seen in, and its win-rate independent of the opposing teams compositions.

Feature	Description
composition	The roles in the team composition
match_count	How many matches the composition is found in
win_rate	The % win rate of the team composition

Table 7: Features in the data set, where win rate is independent of the opposing team's composition.

Role 1 (healing support)	Role 2 (support)	Role 3 (support/ fighter)	Role 4 (semi- fighter)	Role 5 (building damaging)	Role 6 (fighter)	Win rate
2	0	0	0	0	3	59.6%
2	0	0	2	1	0	58.6%
0	1	3	0	0	1	58.2%
1	2	0	1	1	0	58.1%
0	1	1	0	3	0	57.7%

Table 8: Top-five team combinations based on win rate.

Role 1 (healing support)	Role 2 (support)	Role 3 (support/ fighter)	Role 4 (semi- fighter)	Role 5 (building damaging)	Role 6 (fighter)	Win rate
0	1	2	2	0	0	29.7%
0	2	1	2	0	0	34.5%
2	1	0	0	1	1	36.7%
0	0	0	1	3	1	38.8%
0	0	0	1	1	3	39.0%

Table 9: Bottom-five team combinations based on win rate.

Looking at the top five team compositions, there is a good balance in roles with supporting players that do not require many items to be effective, and fighters who get stronger as gold/experience increases. For the five combinations with the lowest win rate there seem to be either a lack of supports or fighters. For example, the first two compositions contain neither strong fighters or healing supports, while the last two compositions contain none of the support roles. Based on the team composition matrix and the results of the top and bottom five team compositions, we can therefore give suggestions to what roles a team is lacking and what roles to avoid based on what characters have been picked so far when assembling a team.

Unfortunately, we only have data for about 117k matches. If we had even more match data available, it would also be possible to create an average win-percentage for each team-composition versus every possible enemy team-composition, creating a 252×252 matrix. With this matrix, it would be possible to more accurately suggest what team compositions works well against a specific match up. For example by displaying the team compositions with the highest success rate against the specific match up. Having data about the win rate of each possible match up could also be used to predict a game's outcome based on the composition of the two teams in a match.

6 Live analysis of win probability

6.1 Motivation

The task of predicting the winning probability of a team in real-time can serve numerous purposes. A field like live-betting requires thoughtful work on computing probabilities given several factors that affects the current game. A bookmaker wants to put the right odds for given outcomes, and the bettor wants to put their money on some outcome, based on a prediction. Also, a team that is allowed to have such live insights in their game play can improve their performance by pointing out which factors that currently satisfies the outcome positively or negatively.

Live analysis can also help by spotting and pointing out the specific points in a match where the probability of winning decreases with a certain magnitude. A team can then evaluate their performance live, or after end of match, at which time-stamps their winning probability decrease(d), in order to have a greater understanding on what affects the outcome. By having this overview a team can adapt and modify their strategy into something more beneficial both while competing, and for their next match.

In this section we try to come up with methods that can be used for making minute-wise predictions for the outcome of the match as the match progresses using live data.

6.2 Exploratory data analysis

Using an approach called exploratory data analysis, we want to gain an insight into the underlying structure of the live data by the use of visualizations. In our data set, the live features *exp_advantage, gold_advantage* are available as lists, with one value measured every minute of the game. This can be used along with the actual winner of the match. In this case, there are 3 dimensions of features to plot (game minute, *gold_advantage, exp_advantage*), while the fourth feature (win/loss) can be visualized by color-coding the data points. We thus color code a win as green and a loss as red.
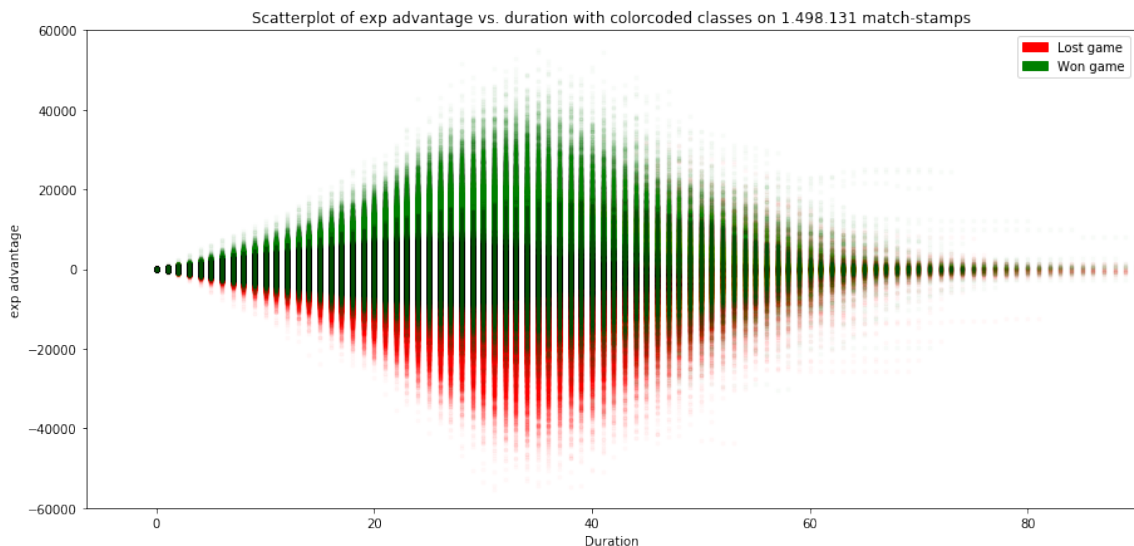
Figure 12: Scatter plot of single exp_advantage values per minute.
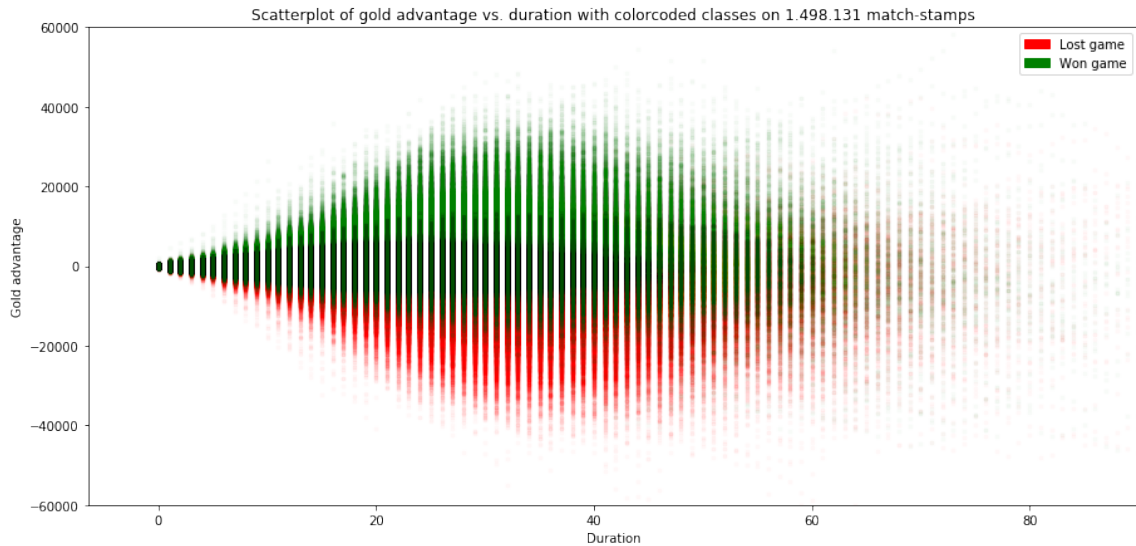
Figure 13: Scatter plot of single gold_advantage values per minute.

By visually examining these plots we see that an early advantage in either gold or experience does not quite map whether a game is won or lost. We also see that through minute 10-60, a huge advantage seems to correlate with winning games. In the experience plot we see that the advantage is near nonexistent when the duration exceeds 60 minutes. This is most probably due to an upper bound on *exp_advantage* that is set by the game rules. This makes the difference on *exp_advantage* between teams eventually go to zero when duration increases. We also see that an advantage in gold seems to dominate the outcome of the match less when the duration exceeds 60 minutes.

In this case it will be meaningful to see if there is a tendency for live features in tandem. In general, it is easier to visualize and spot tendencies when data is in 2D-space, but this is rarely the case in real life. We therefore resolve to principal component analysis; a graphical technique often used in exploratory data analysis that reduces the dimensionality of the data.

6.2.1 Principal Component Analysis

Very often, data sets have a high dimensionality with a large amount of features. This makes it hard to visualize the data and gain an overview of patterns or tendencies in the data. To reduce the dimensionality of our data set, we use principal component analysis (PCA).

In our case, the data set is in 3D-space. It can be difficult to spot tendencies with the large amount of data points we have, since data points may visually cover large volumes of other data points in the dimension of depth. PCA is will also be useful if more live features are made available in the future, both in terms of visual interpretation and the computation time for many classification algorithms.

PCA is a procedure that converts a number of possibly correlated variables into a smaller number of linearly uncorrelated variables known as principal components. The smaller set captures most of the information from the original data set. The principal components are ordered such that the first principal component captures as much of the variability in the original data set as possible, the second principal component capturing the second most variability, and so on [Jef07].

PCA is about maximizing variance as the procedure projects the original data onto directions that maximize the variance. It is therefore sensitive to outliers or data where features vary much in variance, since features with high variance will influence the principal components the most [Gle13]. The scales of our features differs a lot from each other. While a duration of a game could be 30 minutes, the gold and exp advantage could be tens of thousands in either positive or negative direction. Before using PCA, the data is thus normalized to have zero mean and unit variance.

Algorithm: Principal Component Analysis [Chr18]

Input: Data set $S = \{x_1, \ldots, x_d\} \in \mathbb{R}^{n \times d}$, number of dimensions k for new dataset.

1 Compute the empirical mean vector, $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$.

2 Compute the empirical covariance matrix, $Cov = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$.

3 Compute the $d \times k$ matrix U_k composed of the first k eigenvectors of Cov, where eigenvectors are ordered by decreasing eigenvalue.

4 Project data onto smaller dimensional subspace, $z_i = U_k^T (x_i - \bar{x})$ for $i = 1, \ldots, n$.

Output: Mean vector $\bar{x}$, principal components U_k, projected data $\{z_1, \ldots, z_n\}$.

The principal components are also known as eigenvectors, while the variance that each component captures are known as eigenvalues. The eigenvectors determine the directions of the new dimensional subspace, i.e. they will form the axes [Seb]. Each eigenvector is associated with an eigenvalue that measures the variance in the variables that the eigenvector is accounting for. The eigenvalue can also be interpreted as the magnitude of the eigenvector. A small eigenvalue means that the eigenvector does not contribute much to the explanation of variances in the features. It may thus be considered less informative compared to other eigenvectors with higher eigenvalues. Eigenvectors are therefore sorted by decreasing eigenvalue in PCA, as one may then consider only the k eigenvectors that preserve most of information from the original data when forming the smaller data set [Jef07].

6.2.2 Cumulative variance captured by applying PCA

We want to verify whether the affine transformation from applying PCA on our data set maps to a proper low-dimensional representation of the original data. To do this we can take a look at the eigenvalues belonging to each eigenvector and see how much variance that is captured per added principal component.
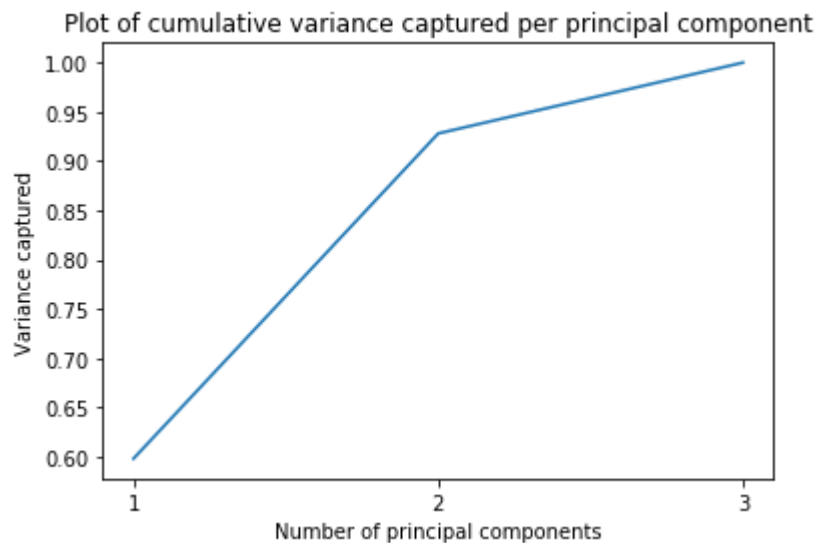
Figure 14: Cumulative variance captured per principal component.

In figure 14 we see that the 1st principal component captures around 60% of the variance, the cumulative variance by including both the 1st and the 2nd principal component is computed to be 0.928 = 92.8%. There is no threshold for how much variance that should be captured by PCA, but 92.8% of the variability is reasonably high and we thus redeem the variance captured satisfying for this task.

6.2.3 Exploratory data analysis on PCA

Upon projecting the data onto a smaller dimensional subspace and plotting it with a color based on the outcome, we get the following plots scattered by the 1st and 2nd principal components.
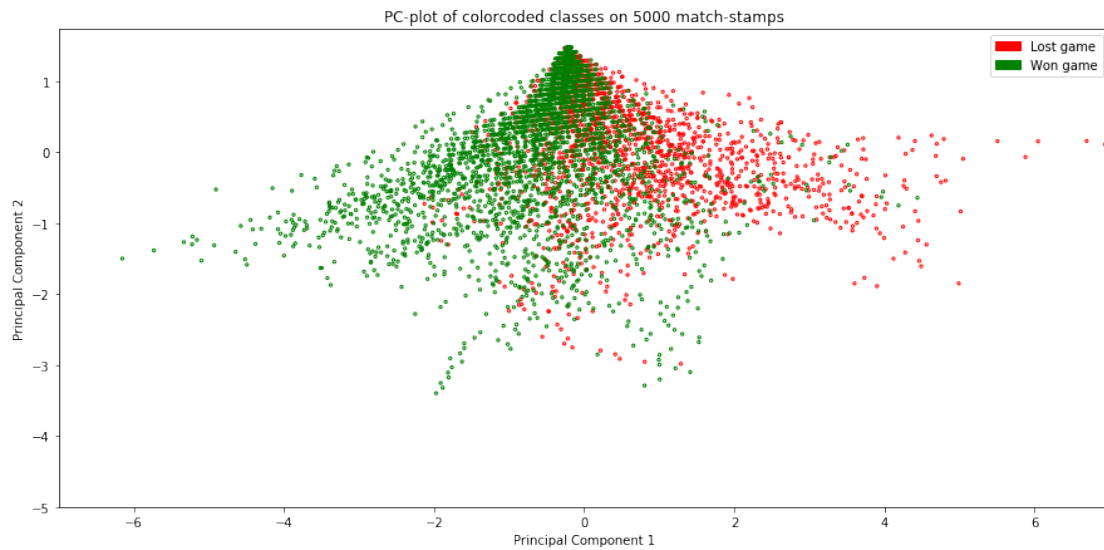
Figure 15: Scatter plot displaying a subset of the projected data.
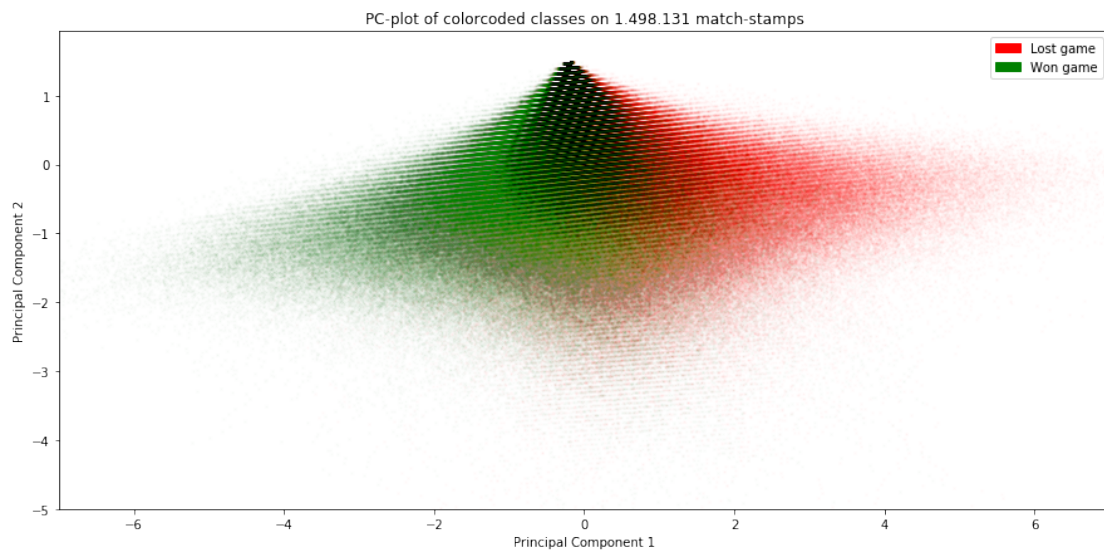
Figure 16: Scatter plot displaying the projected data.

In figure 15 and 16 the green colored data points displays games where the game was won, while the red colored data points displays games where the game was lost. If we have a look at these plots, we definitely see a tendency in the result of the outcome based on the value of the principal components. A higher contrast means a higher concentration of data points. The place with the highest concentration of data points is near the center at $(0, 0)$, where the number of lost and won games appears to be roughly equal. As the value of especially the first principal component changes, the outcome of the game appears to be largely from a specific label. It therefore seems that experience and gold advantage in tandem may have an actual effect on the outcome of the game. In order to investigate this hypothesis, we will resort to hypothesis testing.

6.3 Inferential data analysis

In order to predict whether a team will win, or even output a winning probability, we need some sort of prediction model. We saw in the exploratory data analysis that some visual tendencies were present, but in order to deem the data fit for predictive analysis, we want to test for inference between the live-features to verify the usefulness of applying prediction models on the data. Hypothesis testing is a popular method for testing inference between two or more variables. In this section, we want to verify whether the visual tendencies of the previous exploratory data analysis are actual present in the population with a certain probability.

6.3.1 Hypothesis testing

In this hypothesis testing, we want to investigate if the gold advantage and exp advantage plays a significant role in the probability of winning a match. To see if the advantage is the same for games where the team wins, compared to games where the team loses, we will perform a two-sampled t-test. Two-sampled (or two-tailed) t-test is a procedure that tests if there is a difference between the means of two different populations. Here, the two populations consists of matches where the team wins, and matches where the team loses.

The t-statistic $\in \mathbb{R}$ is used to decide if the null hypothesis should be supported or rejected and is defined as

$$t = \frac{(\bar{X} - \bar{Y}) - (\mu_X - \mu_Y)}{\sqrt{\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}}}$$

where $\bar{X}$ and $\bar{Y}$ are the sample means of the two sample populations. μ_X and μ_Y are the means of their respective underlying distributions. s_X and s_Y are the standard deviations of the two samples. n_X and n_Y are the number of elements in sample X and Y.

The only unknown variables here are μ_X and μ_Y since we only have subsets of the population. We can omit these from the formula by making the assumption that $\mu_X = \mu_Y$. That is, we construct a null-hypothesis where we assume that their population means are the same, and then an alternative hypothesis, such that they are mutually exclusive. If the null-hypothesis is true, the other hypothesis is false, and vice versa [Ste03]. We thus have the following hypotheses

$$H_0 : \mu_X = \mu_Y$$

$$H_A : \mu_X \neq \mu_Y$$

Following from the formula before we are now left with the following t-test (also known as Welch's t-test)

$$t = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}}}$$

This t-statistic approximately follows a t-distribution with ν degrees of freedom (df), where ν is approximated using the Welch-Satterthwaite equation [Ste14]

$$\nu = \left| \frac{\left(\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}\right)^2}{\frac{s_X^4}{n_X^2(n_X-1)} + \frac{s_Y^4}{n_Y^2(n_Y-1)}} \right|$$

Let t be our observed t-value associated with the two sample populations X and Y. Under the assumption that the two populations come from distributions with equal means, the probability of observing an extreme value t_0 is given by the integral

$$P(t < -|t_0|) + P(t > |t_0|) \quad = \quad \int_{-\infty}^{-|t_0|} f_t(x)dx + \int_{|t_0|}^{\infty} f_t(x)dx$$

where the extreme value is selected from a t-distribution table with ν degrees of freedom (see appendix figure 28). The probability of observing a value beneath or above the extreme values $-t_0$ or t_0 are mutually exclusive since $\forall t_0 \in \mathbb{R}$ we have that $(-|t_0| \leq |t_0|)$. Due to the symmetry of the t-distribution we have the equivalence

$$P(|t| > |t_0|) \quad = \quad 2\int_{|t_0|}^{\infty} f_t(x)dx$$

which computes the coloured area of the t-distribution shown in figure 17 [AC15]
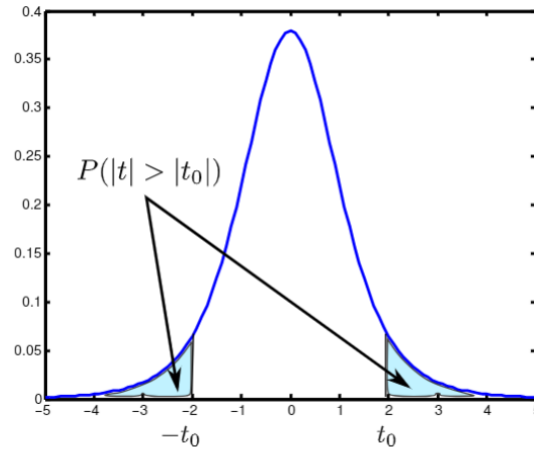
Figure 17: Critical t_0 values of a two-tailed hypothesis test. In this case $t_0 = \pm 1.96$ when conducting the test with confidence 0.95 and degrees of freedom $\nu \to \infty$.

We evaluate the result of the hypothesis test based on a level of confidence. A typical confidence level is 0.95 which corresponds to $100(1 - \alpha)\% = 95\%$, with an α value 0.05. Another often used confidence level is 0.99 which corresponds to $100(1 - \alpha)\% = 99\%$, with an $\alpha = 0.01$.

The p-value is the probability of the null hypothesis H_0 being true. If we obtain a p-value that falls below our critical value of α we say that we reject our null-hypothesis H_0. Thus we can claim with a probability of $1 - \alpha$ that the observed difference between μ_X and μ_Y is not due to chance alone. That is, we have $\mu_X \neq \mu_Y$ with probability $1 - \alpha$, and $\mu_X = \mu_Y$ with a probability p.

6.3.2 Assumptions when conducting a *t*-test

The following assumptions has to be met in order to conduct a proper *t*-test [Doc18]:

- **Normality:** The population should have a normal distribution.

- **Sample size:** Should be large enough to approximately follow the underlying distribution.

- **Independence:** The sample should be randomly drawn from the population.

6.3.3 Algorithm

If all assumptions are correct, we can proceed to the *t*-testing. Instead of computing $2 \int_{|t_0|}^{\infty} f_t(x)dx$ from before, we can compute $2 \int_{-\infty}^{-|t_0|} f_t(x)dx$ (again due to symmetry). From this, we can utilize a neat trick of computing the cumulative distribution function (*CDF*) of $2 \cdot P(-|t| \leq -|t_0|)$ to get the probability p.

Algorithm: Two-tailed Hypothesis Testing of $H_0 : \mu_X = \mu_Y$

Input: Sample $X \in \mathbb{R}^n$, sample $Y \in \mathbb{R}^m$, significance level α.

1   Compute Welch's *t*-value, $t = \dfrac{\bar{X}-\bar{Y}}{\sqrt{\frac{s_X^2}{n_X}+\frac{s_Y^2}{n_Y}}}$.

2   Compute degrees of freedom, $\nu = \left\lfloor \dfrac{\left(\frac{s_X^2}{n_X}+\frac{s_Y^2}{n_Y}\right)^2}{\frac{s_X^4}{n_X^2(n_X-1)}+\frac{s_Y^4}{n_Y^2(n_Y-1)}} \right\rfloor$.

3   Compute the probability via CDF, $p = 2 \int_{-\infty}^{-|t|} f_{t_{df=\nu}}(x)dx$.

Output: Reject H_0 or fail to reject H_0 by evaluating $p < \alpha$.

6.3.4 Hypothesis testing of live analysis features

We want to test for inference in gold advantage and exp advantage between won and lost matches, as we saw in the exploratory data analysis that a high advantage tends to lead to a win and vice versa. I.e., we want to statistically investigate whether the gold advantage and exp advantage plays a significant role for the chances of winning the game. Since the goal is to conduct a live analysis, we do not just want to test whether there is a difference in the winning and losing teams' gold and exp advantages, but also look at each individual minute mark to test whether the advantage plays a significant role for different periods of the match. Figure 18 displays the number of data points for each minute-mark. I.e., for a game that lasts 87 minutes, we have 88 data points from that match, where each data point displays the advantage for a specific minute mark starting from minute 0. Looking at the graph, a game tends to be less than 40 minutes on average.
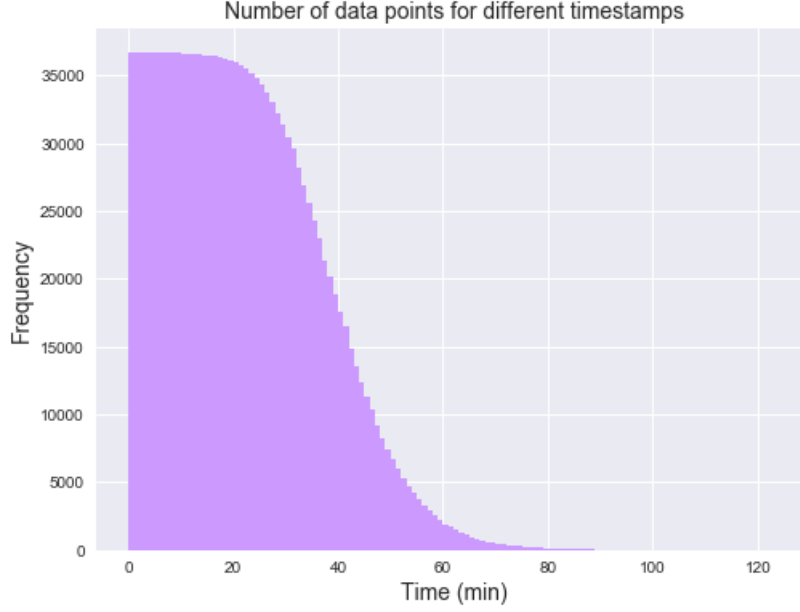
Figure 18: Number of data points for different minute marks.

We now define the following iterative hypothesis testing setup.
Let

$$X_{gold,i}, Y_{gold,i}$$

be samples of gold advantage for games that were won X, and games that were lost Y, at minute i.
Let

$$X_{exp,i}, Y_{exp,i}$$

be samples of exp advantage for games that were won X, and games that were lost Y, at minute i.

Following the assumption about t-testing, we want to have a considerable amount of data points for a minute mark. Thus we define a threshold k to be the i'th minute where X_i or Y_i has less than 20 counts. The two iterative hypotheses can then be stated as

$$H_{0,gold,i} : \mu_{X_{gold,i}} = \mu_{Y_{gold,i}} \quad \text{for } i = 1, \ldots, k$$

$$H_{A,gold,i} : \mu_{X_{gold,i}} \neq \mu_{Y_{gold,i}} \quad \text{for } i = 1, \ldots, k$$

and

$$H_{0,exp,i} : \mu_{X_{exp,i}} = \mu_{Y_{exp,i}} \quad \text{for } i = 1, \ldots, k$$

$$H_{A,exp,i} : \mu_{X_{exp,i}} \neq \mu_{Y_{exp,i}} \quad \text{for } i = 1, \ldots, k$$

with a confidence level of 0.95 ($\alpha = 0.05$). For each minute mark we plot the corresponding p-value by conducting the t-tests of respectively $H_{0,gold}$ and $H_{0,exp}$. I.e., the probabilities of observing the null hypotheses in the population for gold and exp advantages.

When computing the p-value in Python, there were some p-values that fell below the smallest value > 0 representable in Python. These p-values were set to be the smallest positive value > 0 representable in Python.
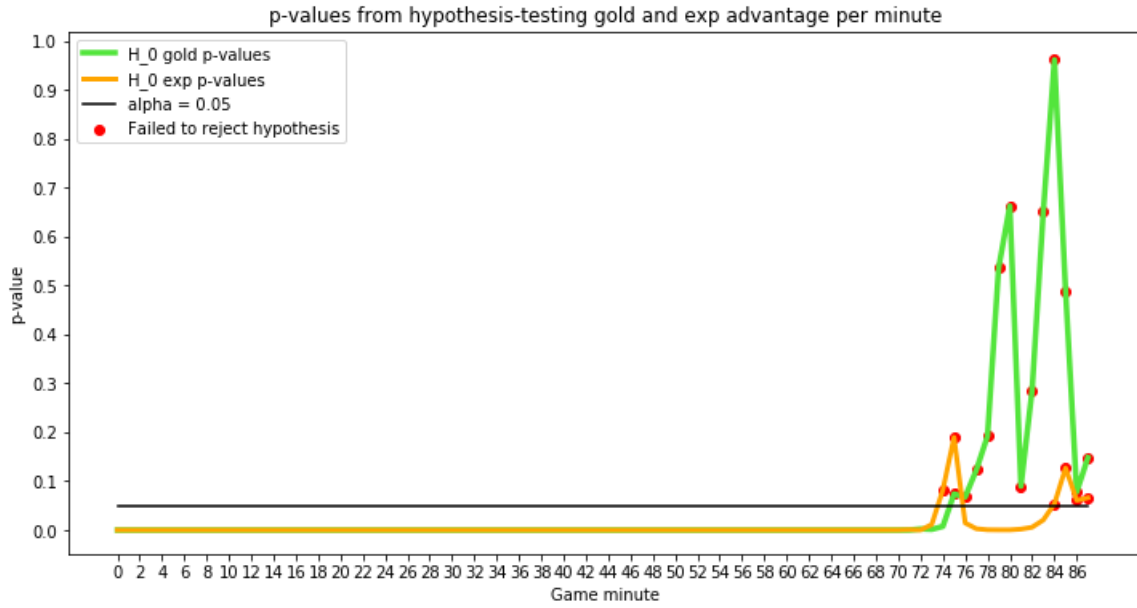
Figure 19: Plot of *p*-values for minute 0 to 87.

We see all the null hypotheses being rejected up to minute 74. Beyond that point we see some of the null hypotheses failed to be rejected, indicating that the gold and exp advantage are significant until this minute mark. That is, with more than 95% probability that the observed tendency between the advantages for won and lost games is not due to chance alone.

As the match progresses towards the late game minutes, we see that the advantage may play a lesser role in determining the outcome of the match. Following from figure 18, games also tend to end when gold and exp advantage seem to be a significant factor for determining the outcome.
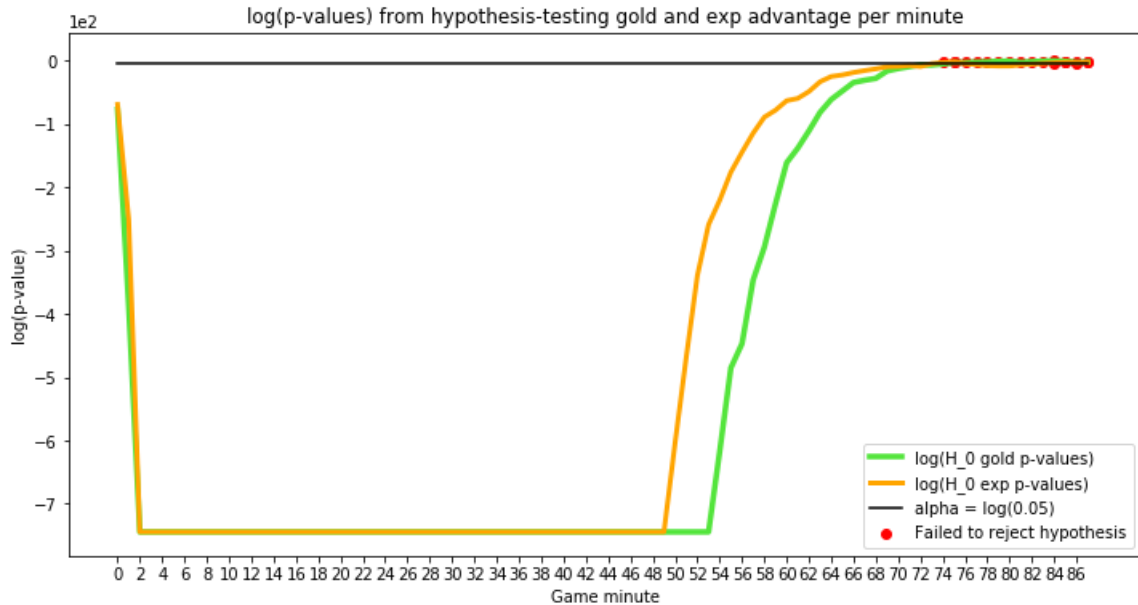
Figure 20: Log-plot of *p*-values for minute 0 to 87.

By plotting the same plot with $log(p)$-values instead, it is seen that many of the $log(p)$-values lie below -700, which is an extremely low $log(p)$-value. It is also seen that as the game progresses, the $log(p)$-value of the null hypotheses converges towards the critical value where the null hypotheses fails to be rejected.

By this inferential analysis we can thus conclude that gold and exp advantage tends to play a major role in terms of indicating the winner of the match for a large duration of the game. With these results obtained from the inferential analysis we can now proceed to use the data for predictive data analysis.

6.4 Predictive data analysis using logistic regression

There are multiple possible learning algorithms that can be used to predict the win probability of a match given a specific minute mark and some live features. We decided to perform *logistic regression* as it can be used for outputting a probability.

6.4.1 Logistic Regression

In a multivariate regression task, we use multiple explanatory independent variables $x \in \mathbb{R}^d$ in order to predict an outcome $y \in (-\infty, \infty)$, which is also known as the dependent variable. Given a data set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathbb{R}^d \times \mathbb{R})$ we try to estimate the weights $\beta \in \mathbb{R}^d$ and intercept $\alpha \in \mathbb{R}$ that describes the linear model

$$y = \beta_1 x_1 + \cdots + \beta_d x_d + \alpha + \epsilon$$

where ϵ is the residual. However, in logistic regression we let y be a dichotomous variable, in which there are only two possible outcomes (e.g. true/false, pregnant/not pregnant, win/loss), usually denoted by a class -1 or a class 1. That is, $y \in \{-1, 1\}$. The individual predictions of y can be interpreted as probabilistic values in range $[0, 1]$. I.e., a value of 0.25 means $P(y = 1) = 0.25$.

The immediate problem is however that the output from the above linear model can vary from negative numbers to huge positive numbers. This can be avoided by applying a logistic (sigmoid) function on the linear model [Gru15b]

$$\theta(s) = \frac{e^s}{1 + e^s}$$

where s is a scalar, and $\theta(s)$ maps the probability $P(y = 1)$ to the regression formula $\hat{y} = \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_d x_d + \hat{\alpha}$, thus that

$$\hat{y} = \theta(\hat{\beta}_1 x_1 + \cdots + \hat{\beta}_d x_d + \hat{\alpha}) = \frac{e^{\hat{\beta}_1 x_1 + \cdots + \hat{\beta}_d x_d + \hat{\alpha}}}{1 + e^{\hat{\beta}_1 x_1 + \cdots + \hat{\beta}_d x_d + \hat{\alpha}}}$$

If we let $w = (\beta_1, \beta_2, \ldots, \beta_d, \alpha)$ and $x = (x_{i1}, x_{i2}, \ldots, x_{id}, 1)$ be a sample, we can formalize the model to

$$\hat{y}_i = \theta(w^\intercal x_i) = \frac{e^{w^\intercal x_i}}{1 + e^{w^\intercal x_i}}$$

6.4.2 Deriving the likelihood of logistic regression

In order to solve a logistic regression problem we start by deriving its maximum likelihood.
Let $w^\mathsf{T} = (\beta_1, \ldots, \beta_d, \alpha)$, $x_i^\mathsf{T} = (x_{i1}, \ldots, x_{id}, 1)$. Given some weights w, our model states that each y_i should equal 1 with probability $f(w^\mathsf{T} x_i)$, and -1 with probability $1 - f(w^\mathsf{T} x_i)$. In other words, we have for each y_i a Bernoulli distribution with probability distribution function [Gru15b]

$$P(y_i|w, x_i) = f(w^\mathsf{T} x_i)^{y_i}(1 - f(w^\mathsf{T} x_i))^{1-y_i}$$

that is

$$P(y_i = 1|w, x_i) = f(w^\mathsf{T} x_i)$$

and

$$P(y_i = -1|w, x_i) = 1 - f(w^\mathsf{T} x_i)$$

If we use our two probability spaces given by $f(w^\mathsf{T} x)$ and $1 - f(w^\mathsf{T} x)$, and the definitions of $\theta(s)$, we can prove that $1 - \theta(s) = \theta(-s)$:

$$
\begin{aligned}
1 - \theta(s) &= \theta(-s) \\
1 &= \theta(-s) + \theta(s) \\
&= \frac{e^{-s}}{1 + e^{-s}} + \frac{e^s}{1 + e^s} \\
&= \frac{e^{-s}}{1 + e^{-s}}\frac{e^s}{e^s} + \frac{e^s}{1 + e^s} \\
&= \frac{1}{e^s + 1} + \frac{e^s}{1 + e^s} \\
&= \frac{1 + e^s}{1 + e^s} \\
&= 1
\end{aligned}
$$

in order to cover both cases of y

$$P(y|x) = \theta(yw^\mathsf{T} x)$$

Since the data points of $\mathcal{D} = (x_1, y_1), \ldots, (x_n, y_n)$ are independent and identically distributed, the probability of getting all the y_i's from their corresponding x_i's, that is the maximimum likelihood of y is given by

$$MLE(y) = \prod_{i=1}^{n} P(y_i|x_i)$$

when data is generated by some hypothesis h with parameters w, equivalent to

$$\underset{w}{\arg\max} \prod_{i=1}^{n} P(y_i|x_i; h)$$

Selecting the hypothesis h that maximizes the likelihood corresponds to minimizing the log-likelihood of

$$-\frac{1}{n} ln\left(\prod_{i=1}^{n} P(y_i|x_i) \right)$$

since $-\frac{1}{n} ln(x)$ is a monotonically decreasing function. Also we have that

$$-\frac{1}{n} ln\left(\prod_{i=1}^{n} P(y_i|x_i) \right) = \frac{1}{n} \sum_{i=1}^{n} ln\left(\frac{1}{P(y_i|x_i)} \right)$$

Since we have that $P(y|x) = \theta(yw^\mathsf{T}x)$ we would be minimizing

$$\frac{1}{n}\sum_{i=1}^{n} ln\left(\frac{1}{\theta(y_i w^\mathsf{T} x_i)}\right)$$

with respect to the weights w. Replacing $\theta(y_i w^\mathsf{T} x_i)$ with its functional form yields us

$$\frac{1}{n}\sum_{i=1}^{n} ln\left(\frac{1}{\frac{e^{y_i w^\mathsf{T} x_i}}{1+e^{y_i w^\mathsf{T} x_i}}}\right) = \frac{1}{n}\sum_{i=1}^{n} ln\left(\frac{1+e^{y_i w^\mathsf{T} x_i}}{e^{y_i w^\mathsf{T} x_i}}\right)$$

Multiplying both parts of the fraction by the conjugate of $e^{y_i w^\mathsf{T} x_i}$ gives us

$$\frac{1}{n}\sum_{i=1}^{n} ln\left(\frac{1+e^{y_i w^\mathsf{T} x_i}}{e^{y_i w^\mathsf{T} x_i}}\frac{e^{-y_i w^\mathsf{T} x_i}}{e^{-y_i w^\mathsf{T} x_i}}\right) = \frac{1}{n}\sum_{i=1}^{n} ln\left(\frac{1+e^{-y_i w^\mathsf{T} x_i}}{1}\right)$$

What we are left with is the sample residual error for logistic regression, when using our hypothesis h with weights w

$$s_\epsilon = \frac{1}{n}\sum_{i=1}^{n} ln\left(1 + e^{-y_i w^\mathsf{T} x_i}\right)$$

We can thus use a numerical optimization algorithm (e.g. stochastic gradient descent) on this derived log-likelihood, in order to approximate the w that makes the log-likelihood 0 [YMH12b].

6.4.3 Using logistic regression to predict game outcomes

Upon having found the weights w that maximizes the maximum likelihood, we can now use these in our sigmoid function

$$\theta(s) = \frac{e^s}{1 + e^s}$$

By using our regression model

$$\frac{e^s}{1 + e^s} = \frac{e^{w^\mathsf{T} x}}{1 + e^{w^\mathsf{T} x}} = \frac{1}{1 + e^{-w^\mathsf{T} x}} = P(y = 1|x)$$

which we can then interpret as the conditioned probability of observing that the dependent variable y belongs to class 1, given the independent input variables x. Thus $\frac{1}{1+e^{-w^\mathsf{T} x}} \in (0, 1)$ is the probability of winning the game.

6.4.4 Algorithm

Algorithm: Logistic regression using stochastic gradient descent [Ige17a]

Input: Set of data points $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathbb{R}^d \times \{-1, 1\})^n$, learning rate η.

1   initialize $w = \{0\}^{d+1}$.
2   **repeat**
3   pick $(x, y) \in \mathcal{D}$.
4      $w = w + \eta\frac{yx}{1+e^{yw^\mathsf{T} x}}$.
5   **until** stopping criterion is met.

Output: Weights w describing the linear model $h(x) = w^\mathsf{T} x$.

A stopping criterion could e.g. be based on a number of iterations, the length of the data set, or if the residual of applying w to x falls below some threshold.

6.4.5 Performance and evaluation

Using logistic regression turned out to not work as well as intended as the prediction accuracy was always near 50%. As we saw from figure 16 in the exploratory data analysis, many data points are concentrated around $(0, 0)$. The logistic regression model may therefore not put enough weight on the data points where gold and exp advantage is large.

We therefore resolve to another classification algorithm; a modified version of *k-nearest neighbors* that can be used to output probabilities instead of labels.

6.5 Predictive data analysis using modified *k*-nearest neighbors

6.5.1 Modified *k*-nearest neighbors

Instead of logistic regression, we opted for another learning algorithm named *k*-nearest neighbors. It is a supervised learning algorithm that predicts the label a data sample belongs to, by looking at the labels of its *k* nearest neighbor-samples. Based on the exploratory data analysis, this algorithm appears like a better candidate for live predictions, as samples who share the same label tend to lie close to each other in the visualizations. Whereas the logistic regression model tries to separate the data through a hyperplane in the *d*-dimensional Euclidean space.

When classifying the label for a sample, the label from its neighbors with largest occurrence is chosen as the label. Normally, this version of *k*-nearest neighbors algorithm is used for classification, but since we want to predict win probability, the algorithm is modified to instead return a probability based on the frequency of the labels from the neighbors [Tav18].

Algorithm: Ordinary *k*-nearest neighbors

Input: Train data set $X = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathbb{R}^d \times \{-1, 1\})^n$, test data set
$Y = \{x_1, \ldots, x_m\} \in \mathbb{R}^{m \times d}$, number of neighbors to consider $k \in \mathbb{Z}^+$.

1 **For each sample *s* in test data set:**
2   Calculate the Euclidean distance between the sample *s* and each of the training data samples *t*,
$d(s, t) = \sqrt{\sum_{i=1}^{d} (s_i - t_i)^2}.$
3   Sort the computed distance values in ascending order.
4   Select the *k* nearest neighbors based on distance values.
5   Find the most frequent class of these neighbors and return that as predicted class.
Output: The predicted classes for each data sample.

Algorithm: Modified *k*-nearest neighbors

Input: Train data set $X = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathbb{R}^d \times \{-1, 1\})^n$, test data set
$Y = \{x_1, \ldots, x_m\} \in \mathbb{R}^{m \times d}$, number of neighbors to consider $k \in \mathbb{Z}^+$.

1 **For each sample *s* in test data set:**
2   Calculate the Euclidean distance between the sample *s* and each of the training data samples *t*,
$d(s, t) = \sqrt{\sum_{i=1}^{d} (s_i - t_i)^2}.$
3   Sort the computed distance values in ascending order.
4   Select the *k* nearest neighbors based on distance values.
5   Count the frequency of the neighbor labels.
Output: The class probabilities for each test sample *s*.

6.5.2 Estimating model performance using cross-validation

To use the modified k-nearest neighbors algorithm, one must first decide what k should be. To estimate a good value for k, we can perform a technique called cross-validation. This technique estimates how well a model predicts in practice by giving an estimate of the out-of-sample error E_{out} which can then be computed for a range of k-values.

In cross-validation, the training data with size N is partitioned randomly m times into a validation set with size K and a training set with size $N - K$. Like a test set, a validation set is also not used during the learning process. The difference however lies in that the validation set affects how we want to train a model, which a test set must not do. In each iteration $1, \ldots, m$, a model is trained with the training set and the validation set is then used to calculate the validation error $e_i = 1 - \text{Accuracy score}$, as defined in section 4.5.4. The cross-validation estimate is then found by computing the average value of the validation errors $e_1, \ldots, e_m$ [YMH12c].

$$E_{cv} = \frac{1}{m} \sum_{i=1}^{m} e_i$$

This expected value of E_{cv} is equal to the out-of-sample error E_{out} for data sets of size $N - K$. How the size K of the validation set is then set will affect the estimate of E_{out}. If K is large, there is less training data to work with, which may lead to problems such as risk of overfitting or outliers/noise affecting the model to a bigger degree, and thus an increase in the expected validation error. On the other hand, a K that is too small may give a biased validation error, so it is important to find a middle ground. In general, a rule of thumb is to let 20% of the data be used for validation, which is also what we intend to do [YMH12c].

Algorithm: Cross-validation [Bro18]

Input: Set of training data points $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^{n \times d}$, number of groups $m \in \mathbb{Z}^+$, model $\mathcal{M}$.
1 Shuffle the data set randomly and split the data set into m groups.
2 **for each group i:**
3   Let i be the validation data set. Let the remaining groups be the training data set.
4   Fit the model $\mathcal{M}$ on the training set and compute the validation error e_i.
5   Save the validation error and discard the model.
6 Compute and return the average of the m validation errors.
Output: An estimate of the out-of-sample error E_{out}.

By getting an estimate of the out-of-sample error from cross-validation, we can then resolve to hyperparameter optimization in order to pick a good value of k. A hyperparameter is a parameter where the value has to be defined before the learning algorithm is started. In this case, the hyperparameter is the number of neighbors k to consider. Hyperparameter optimization is then the problem of finding the hyperparameter k that results in an optimal model that minimizes the classification error on our training data. For different values of k, we perform cross-validation on the training data set and check which k-value gives the most optimal training accuracy (i.e. lowest out-of-sample error). This k is then used when applying the modified k-nearest neighbors algorithm.

A common way to perform hyperparameter optimization is by going through a manually specified list of hyperparameters, and measuring the performance for each hyperparameter, which is also known as grid search. To measure the performance, we want to use the ordinary k-nearest neighbors algorithm, as the label predictions can be used to output an accuracy which we can then base our choice of k on. Given the live features as mentioned earlier, the goal is to predict the probability of a team winning the match, which is a binary classification problem where the result is either -1 (team Dire win)

or 1 (team Radiant win). For each sample being classified, we are looking at the k neighbors that resemble the sample the most. Only odd values of k are considered in the grid search, since an even k-value may result in an even distribution of the labels -1 and 1. Without a majority of either -1's or 1's, a label cannot be decided for the sample.

Algorithm: Grid search

Input: Set of data points $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^{n \times d}$, specified list of hyperparameters
$\quad\quad K = \{k_1, \ldots, k_{max}\} \in \mathbb{Z}^{+,max}$.

1 **for each** k **in** K**:**
2 $\quad$ Define $\mathcal{M}$ to be an ordinary k-NN classification model with parameter k.
3 $\quad$ Get the cross-validation error E_{cv} by performing **_Cross-validation_** on $\mathcal{M}$.
4 $\quad$ Save $1 - E_{cv}$ as the accuracy score a.
Output: List of accuracy scores for each hyperparameter, $\{a_1, \ldots, a_n\}$.

6.5.3   Choosing hyperparameter k for modified k-nearest neighbors algorithm

We want to find a good value of k that takes the following factors into consideration:

- High training accuracy.

- A sufficiently high value of k thus that the distribution of sampling the k nearest neighbors for a test sample is not affected too much by local anomalies. In figure 21, it is seen that when picking a low value of k, anomalies may influence the outcome of the test sample. For $k = 3$, the outcome of the test sample is heavenly influenced by local anomalies (represented as red).
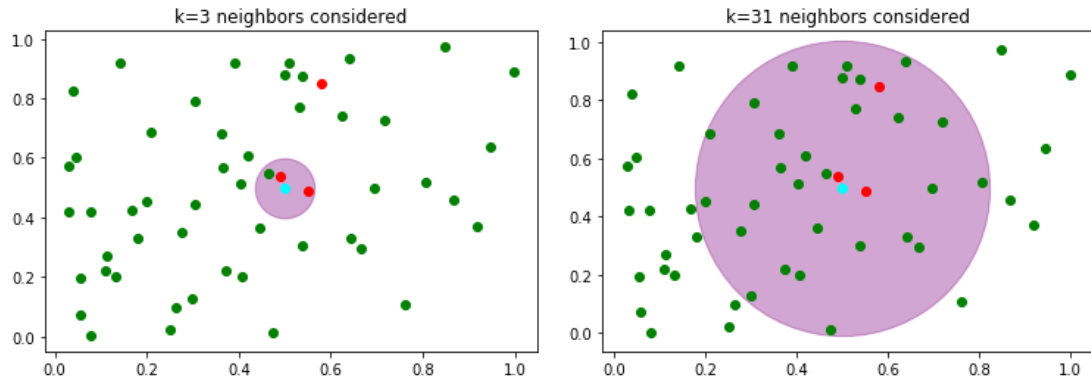
Figure 21: $k = 3$ versus $k = 31$ neighbors considered.

- Low enough value of k to avoid overfitting. When k approaches a very high value, the outputted probability will resemble the distribution of the data set. A low k-value will also reduce training time for cross-validation and limit the number of iterations when searching for k.

Considering these constraints we resolve to hyperparameter tuning using a grid search and test the training accuracy score for $k = 1, 3, \ldots, 201$. Accuracy score is a good performance measure as we focus on being able to predict both wins and losses equally well. Also, the accuracy score is viable here because the distribution of won and lost matches are somewhat even. This is explained further on in section 6.5.4. The range of k-values we grid search through may seem high, but considering the live data set has roughly 1.5 million samples, the k_{max} value is a tiny fraction of the data set. The live data set is split randomly into 80% training and 20% test, and the following plot shows the training accuracy score for different values of k.
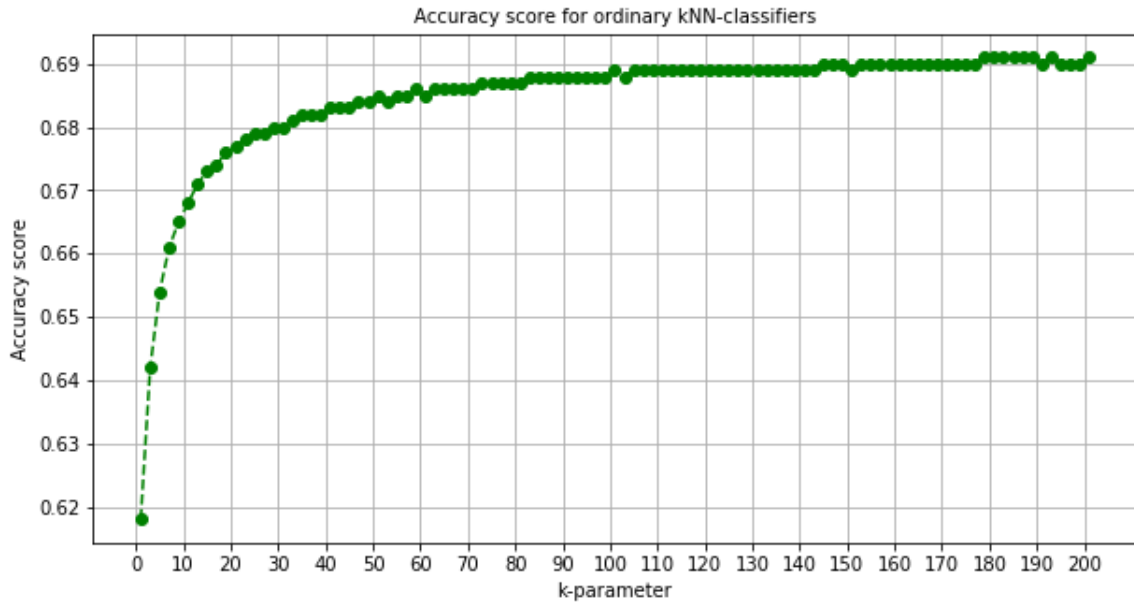
Figure 22: Accuracy score for ordinary *kNN* classifiers.

Looking at the graph, the training accuracy increases as k increases. The lowest training accuracy is for $k = 1$ with an accuracy score of $\approx 62\%$, while the highest accuracy score is at $k \geq 201$. It is possible that a $k > 201$ would have had a higher accuracy score, but we have set a threshold at $k_{max} = 201$ to avoid a significant increase in the computation time.

It is also seen that the increase in accuracy score is most significant for low k-values, while the accuracy score barely increases for large k-values. Based on these results and the constraints defined before, a reasonable k-value to use for our modified *kNN* algorithm could be $k = 101$ as this gives a good balance between high training accuracy and a reasonable computation time. Choosing a higher k-value such as $k = 201$ would only give a tiny increase in training accuracy, while the computation time may increase since each test sample has to look at twice as many nearest neighbors to find the label probabilities.

6.5.4 Performance and evaluation

The ordinary k-nearest neighbors algorithm was trained on the training set with $k = 101$ to measure the performance. The model was then used for predicting labels on each of the 292,296 samples from the test set, where each sample contains live data for a random minute mark in a match and a label with the actual match outcome. Label 1 is team Radiant win and label -1 is team Radiant loss. The predicted labels were compared with the actual labels and the confusion matrix below gives an overview of the number of correct and incorrect predictions.

	Predicted positive (win)	Predicted negative (loss)
Actual positive (win)	111 036 (0.729)	41 233 (0.271)
Actual negative (loss)	48 699 (0.348)	91 328 (0.652)

Table 10: Confusion matrix for the trained ordinary *kNN* classifier.

The predictions for both wins and losses generally match the actual outcomes quite well with 72.9% of the predicted wins, and 65.2% of the predicted losses being correct. It is however interesting to see that the model is better at predicting wins than losses with a difference of 7.7%. Looking at the test data's distribution, the number of data points with label 1 (win) accounts for 52.1% of the data points

$$\frac{\text{Actual wins}}{\text{Total labels}} = \frac{111036 + 41233}{292296} \approx 0.521$$

The samples in the training and test set were randomly assembled from the original data set of live features. It can therefore be assumed that the training set used for training the classification model contains roughly the same distribution. When the label for a data point is being predicted by looking at its k nearest neighbors, the higher prevalence of data points with label 1 will therefore increase the chance of labeling the data point as 1, even though the actual label may be 0.

As discussed in section 4.5, there are different performance measures for finding out how well the classification model is actually performing. To see how accurately the classification model classifies all classes in tandem, the test accuracy can be computed by dividing the number of correct predictions (true positives and true negatives) with the total number of predictions

$$Accuracy_{test} = \frac{|TP| + |TN|}{|TP| + |FP| + |FN| + |TN|} = \frac{111036 + 91328}{292296} \approx 0.692$$

The test accuracy displays that the classification model predicts a label correctly 69.2% of the time. Recall that we in the live data set take each match in the original data set and split it into $n+1$ samples where n is the duration of the match in minutes. In the early stages of matches there is often no large advantage/disadvantage established yet, as seen in figure 12 and 13. It is maybe close to 50/50 which team will win, so predicting the outcome correctly for early stages in the game can be difficult. The samples depicting later stages in the game are less common, as we also saw from figure 18 in the inferential analysis. Roughly half of the matches end at 40 minutes or before. The test accuracy and the algorithm's ability to predict the outcome can therefore be seen as relatively high, considering that the live data set is dominated by data points depicting the early stages of the matches.

In our case, the accuracy is not fully enough for judging our classification model since the distribution of actual wins and losses deviates slightly. Thus, we resolve to also compute the average F1 score for the model to see if it matches up against the accuracy score. This is done by first computing the precision and recall

$$Precision_{win} = \frac{|TP|}{|TP| + |FP|} = \frac{111036}{111036 + 48699} \quad , \quad Recall_{win} = \frac{|TP|}{|TP| + |FN|} = \frac{111036}{111036 + 41233}$$

$$Precision_{loss} = \frac{|TN|}{|TN| + |FN|} = \frac{91328}{91328 + 41233} \quad , \quad Recall_{loss} = \frac{|TN|}{|TN| + |FP|} = \frac{91328}{91328 + 48699}$$

Since we want our model to equally weight predictions for wins and losses, we simply compute the mean of the two F1 scores

$$
\begin{aligned}
F1\ score_{avg} &= \frac{F1\ score_{win} + F1\ score_{loss}}{2} \\
&= \frac{1}{2} \cdot \left(\frac{2 \cdot Precision_{win} \cdot Recall_{win}}{Precision_{win} + Recall_{win}} + \frac{2 \cdot Precision_{loss} \cdot Recall_{loss}}{Precision_{loss} + Recall_{loss}} \right) \\
&= \frac{1}{2} \cdot \left(\frac{2 \cdot \frac{111036}{159735} \cdot \frac{111036}{152269}}{\frac{111036}{159735} + \frac{111036}{152269}} + \frac{2 \cdot \frac{91328}{132561} \cdot \frac{91328}{140027}}{\frac{91328}{132561} + \frac{91328}{140027}} \right) \\
&= \frac{\frac{111036}{159735} \cdot \frac{111036}{152269}}{\frac{111036}{159735} + \frac{111036}{152269}} + \frac{\frac{91328}{132561} \cdot \frac{91328}{140027}}{\frac{91328}{132561} + \frac{91328}{140027}} \\
&\approx 0.691
\end{aligned}
$$

The average F1 score is 69.1% which comes very close to the accuracy score. Now after evaluating the performance using both accuracy score and F1 score, we can conclude that the classification model is relatively good at predicting the correct outcome, but there is still room for improvement. Unfortunately, there is not much live data in the data set. Only the gold and experience advantage/disadvantage is available as live data, so it is limited what the model takes into consideration when live predicting. The model is however predicting the correct outcome relatively often, which indicates that gold and experience generally has a large influence on which team will win in the end.

The following table displays twelve different samples from the test set ordered by minute mark in the game, where the advantage is seen from the viewpoint of team Radiant. Looking at how the model predicts, the win probability in the early stages of the matches seem to be in favor of team Radiant in general. For example, sample number two depicts a negative gold and exp advantage for team Radiant, but the model still predicts that they have a $\approx 61.4\%$ chance of winning. This is likely due to them have a higher prevalence of data points in the sample. As the game progresses and advantage/disadvantage increases, the leading team often has a considerably higher chance of winning. However, the advantage's effect on the win probability seems to slowly decrease again when duration gets high enough. It was also seen from the hypothesis tests that especially the gold advantage tended to matter less for large minute marks.

Sample	Minute mark	Gold advantage	Experience advantage	Votes	Radiant win probability	True winner
1	1	258	338	76/101	75.248 %	Radiant
2	1	-1852	-160	62/101	61.386 %	Radiant
3	5	-4710	-5938	30/101	29.703 %	Dire
4	5	4	375	75/101	74.257 %	Radiant
5	15	-15171	4727	67/101	66.337 %	Radiant
6	25	11074	15811	96/101	95.050 %	Radiant
7	35	-23832	-33072	7/101	6.931 %	Dire
8	45	38862	594	99/101	98.020 %	Radiant
9	50	8205	-3212	72/101	71.287 %	Dire
10	75	499	235	53/101	52.475 %	Radiant
11	90	-40707	-146	34/101	33.663 %	Dire
12	119	-38876	166	26/101	25.743 %	Dire

Table 11: Different test samples with team Radiant's probability of winning.

Now that a classification model has been created, it will be possible to visualize how the win probability changes throughout a given match by giving the model values of gold and experience advantage/disadvantage for a given minute. The following visualizations display two randomly selected matches that were played shortly after the matches that we gathered data for. The first game displays live analysis for a game where team Radiant lost and the second game shows live analysis of a game where team Radiant won. For every fifth minute, the gold and experience advantage/disadvantage was depicted for team Radiant.
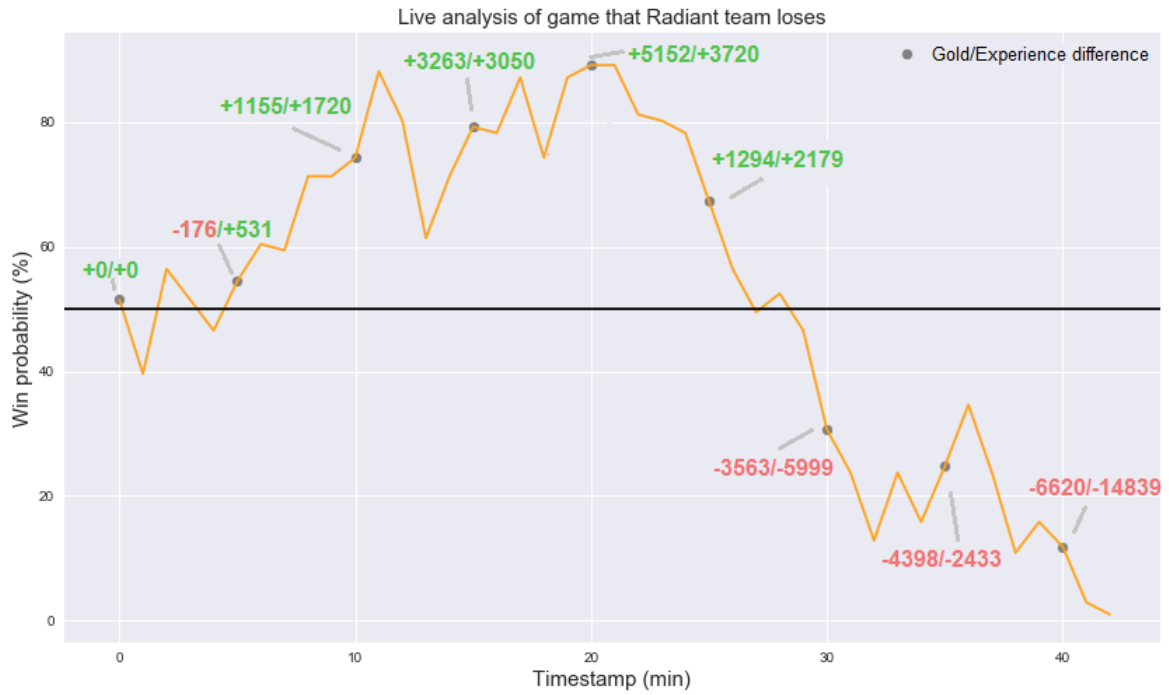
Figure 23: Live analysis of game that team Radiant loses.
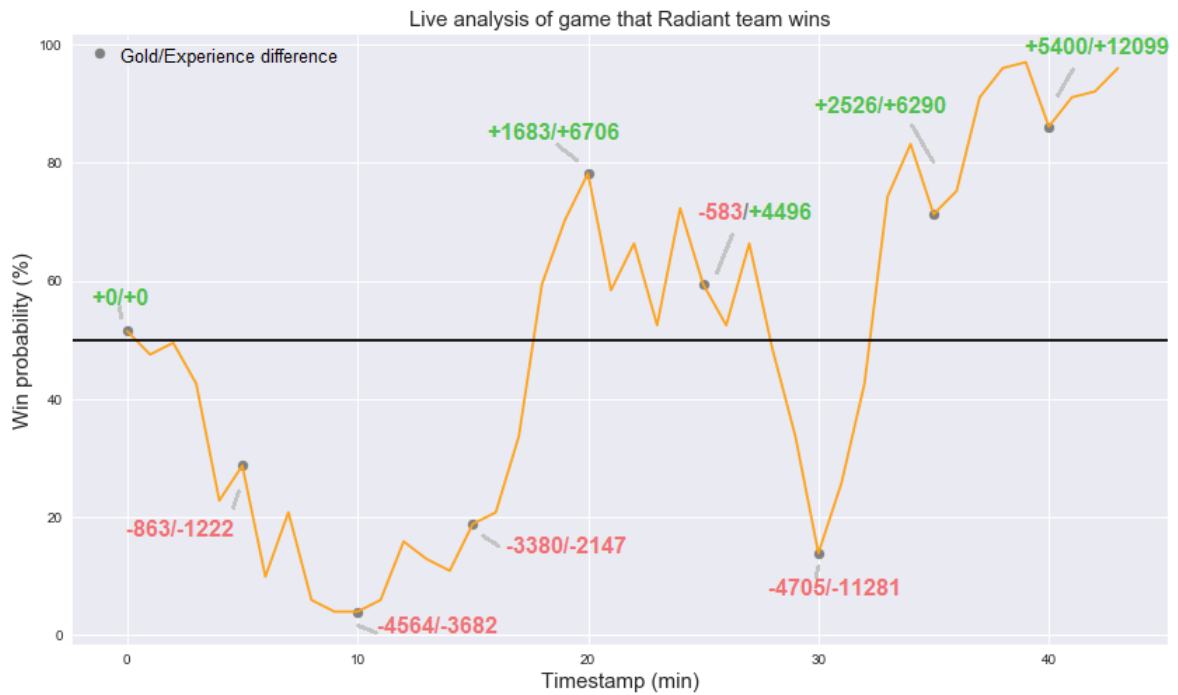
Figure 24: Live analysis of game that team Radiant wins.

It is seen that the win probability increases as the gold and experience advantage increases, and vice versa. For both games, the win probability at the start of the match is slightly higher than 50% for team Radiant even though no advantage has been established. The model is therefore slightly biased, which was also seen from the previous data table.

Graphs such as these can have a useful effect in terms of players and teams being able to identify and adjust for sudden changes in play styles that affect the outcome of the game. Bookmakers can also make more precise judgments in form of live adjusting the odds by looking at the probability based outcomes from our model.

To improve this classification model further, one could also have taken other features into consideration such as the team compositions of each team. From the clustering and roles chapter it was concluded that team compositions have an impact on the winning potential in a match. It could for example be interesting to include live features such as number of kills on enemy heroes, or the number of objectives cleared in the playing field. If further live related features had been available, it would be interesting to test if the model would be more effective as other factors would be taken into consideration besides only gold and experience.

7 Assessment of player performance using regression-analysis

7.1 Motivation

In the live analysis chapter, we saw that the win probability for a team can change a lot over the course of a game. The gold and experience advantage clearly has an impact on the chance of winning, but what else could the individual players have done to increase those chances? In this chapter, we therefore want to investigate multiple parameters for each player. Based on the resulting game parameters, we construct a model that gives the player an assessment of their performance in the match and on what areas they can improve.

A problem with some player performance assessments is their naive approach to assess a player to be underperforming or overperforming by simply comparing the result with some average value.
For instance, lets consider a popular handball player such as the Dane Mikkel Hansen. As of December 2018, Mikkel Hansen on average scores $\approx$ 4.75 goals each time he plays a national game [Wik18]. A simple approach to assess the performance in goal score would be to compare the number of goals Mikkel Hansen does in some game, with his average. The problem with this approach is that it does not take in factors such as how much the opponent team is focusing on defending particularly against Mikkel Hansen, how many passes he receives from his teammates, how many screenings the line players make etc.
By taking multiple factors into consideration that changed the state of the game, positively and negatively, we want to assess a player based upon these factors, and not by a naive comparison to a mean value.

To do this we have decided to use multivariate linear regression in order to predict the expected performance in a variety of aspects, based on factors that influence the game.

7.2 Preparing data for regression modeling

Before performing multivariate regression, the following features were extracted from the original data set which resulted in $n = 366.514$ data points with player details.

All features except for *hero_id* and *my_team_win* are used in the regression task. The two features *hero_id* and *my_team_win* are instead used for modifying the data set and training of the linear regression models.

In the data set, the hero IDs are replaced with the role that the hero was classified as in the cluster analysis (see section 5.5). By having information about the role of each player, the data set can be further divided into six subsets; one for each role. This allows us to make more customized regression models so the assessment of a player's performance is based on what role the player takes. For example, we saw from the cluster analysis that support roles generally deal less hero damage than fighters. It would therefore be unfair to primarily assess a support's performance based on hero damage.

Each of the subsets were then split into a training and test set. In the training set, the feature *my_team_win* was used to remove data points for players that lost their match. By removing lost matches from the training set, the regression model will not just estimate the expected value for the given match in general, but try to predict the expected value for a match that should result in a victory.

Feature	Description
hero_id	ID of the hero that the player chose
kills	How many times the player killed an enemy
deaths	How many times the player died
assists	How many times the player assisted in an enemy kill
hero_healing	How much a player increased their allies' current health
last_hits	Number of kills on basic units
gold_per_min	Average gold earned per minute
exp_per_min	Average experience earned per minute
tower_damage	Damage dealt to towers
hero_damage	Damage dealt to enemy players
obs_placed	Number of observers placed on the field
sen_placed	Number of sentries placed on the field
my_team_win	Whether the player's team won or not
my_team_score	Total enemy kills made by the team
enemy_team_score	Total kills made by the enemy team
my_team_gold_adv	The team's gold advantage at end of game
my_team_exp_adv	The team's experience advantage at end of game

Table 12: Features used in assessing a player's performance.

7.2.1 Z-Score and Modified Z-score for outlier detection

A linear regression model can be sensitive to outliers [STE]. To combat this, we can consider the **Z-score**.

The **Z-score** can be used to screen a data set X for outliers. It is known by the property of the normal/Gaussian distribution that if X is normal distributed with mean μ and variance σ^2

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

then the Z-score can be computed by subtracting the mean from X and dividing it with the standard deviation σ. Then the resulting vector Z is normal distributed with mean zero and unit variance

$$Z = \frac{X - \mu}{\sigma} \sim \mathcal{N}(0, 1)$$

Normalization by Z-score tries to label outliers by computing the Z-scores for observations $x_1, x_2, \ldots, x_n$ and comparing them with a threshold D. The Z-score for a single sample x_i is defined by

$$z_i = \frac{x_i - \bar{x}}{s}, \quad \text{where } s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n - 1}}$$

One popular rule is that data points with Z-scores that exceed $D = 3$ in absolute value is deemed outliers. The problem is however that $\bar{x}$ and s are not resistant. That is, $\bar{x}$ and s can be affected greatly by one single outlier, since their values are computed from all data points of a feature. To confirm this we can consider $n - 1$ observations to have similar values, and 1 observation that differs greatly from the rest. For example

$$x_1 = x_2 = \cdots = x_{n-1} = x \quad and \quad x_n = x + yn$$

Then $\bar{x} = x + y$ and for $y > 0$, we have that $s = y\sqrt{n}$. Thus $\bar{x}$ and s goes to ∞ as $y \to \infty$. As a consequence the largest Z-score for this case is given by

$$Z_n = \frac{x + yn - (x + y)}{y\sqrt{n}} = \frac{yn - y}{y\sqrt{n}} = \frac{y(n - 1)}{y\sqrt{n}} = \frac{n - 1}{\sqrt{n}}$$

50

which is the same as the maximum possible Z-score which does not depend on data values, but only on the number of observations [BD93]. A **Modified Z-score** for outlier labeling was therefore developed by Iglewicz & Hoaglin [BD93]. For a multidimensional data set $X \in \mathbb{R}^{n \times d}$, we compute the median vector

$$median_X = (median(f_1), \ldots, median(f_d)) \in \mathbb{R}^d$$

where each $median(f_j)$ denotes the median of feature j. For each sample i, we also define the summed deviations about the median, $diff_i$ to be

$$diff_i = \sum_{j=1}^{d} x_{i,j} - median_{X,j}$$

and

$$diff = (diff_1, \ldots, diff_n) \in \mathbb{R}^n$$

Now we define the estimator $MAD = median(|diff|) \in \mathbb{R}$ that describes the median of the absolute summed deviations about the median. The $diff$ and MAD then yield the modified Z-score for a sample x_i to be

$$m_i = \frac{0.6745 \cdot diff}{MAD}$$

For each data point in our regression data, we computed the modified Z-score and compared it with a threshold D. The value $D = 3.5$ was chosen since Iglewicz & Hoaglin recommends this threshold for outlier labeling based on a simulation study [BD93]. A data point x_i was thus labeled and seen as an outlier if $|m_i| > 3.5$. For the regression data, a few of the data points had a high modified Z-score, but none surpassed $D = 3.5$. No outliers were thus detected which may be due to the large dimensionality of the data set. One or few features deviating from the sample median will only have a limited effect on the modified Z-score.

7.3 Multiple Variable Linear Regression

Linear regression is a supervised learning method that attempts to explain the relationship between one or more features and the response. This is done by fitting a linear equation to the observed data where the value of an independent feature x is associated with a value of the dependent response y [YU97]. In the simple linear regression model, we try to fit a model using one feature for predicting the response

$$y_i = \beta x_i + \alpha + \epsilon_i$$

For the multivariate case, we try to utilize more parameters to predict the response. Let $\beta \in \mathbb{R}^d$, $\alpha \in \mathbb{R}$ and $x_i \in \mathbb{R}^d$ be a d-dimensional vector containing d independent variables, we construct the linear model [Gru15c]

$$y_i = \beta_1 x_{i1} + \cdots + \beta_d x_{id} + \alpha + \epsilon_i$$

For convenience, lets define the weights $w^\intercal = (\beta_1, \ldots, \beta_d)$ thus that

$$y_i = w_1 x_{i1} + \cdots + w_d x_{id} + \alpha + \epsilon_i$$

If we then formalize and extend $\tilde{w}^\intercal = (\beta_1, \ldots, \beta_d, \alpha)$ and $\tilde{x}_i^\intercal = (x_{i1}, \ldots, x_{id}, 1)$, we have that

$$
\begin{aligned}
y_i &= \beta_1 x_{i1} + \cdots + \beta_d x_{id} + \alpha + \epsilon_i \\
&= \sum_{k=1}^{d} w_k x_{ik} + \alpha + \epsilon_i = w^\intercal x_i + \alpha + \epsilon_i \\
&= \sum_{k=1}^{d+1} \tilde{w}_k \tilde{x}_{ik} + \epsilon_i = \tilde{w}^\intercal \tilde{x}_i + \epsilon_i
\end{aligned}
$$

That is, we have the regression formula $y_i = \tilde{w}^\intercal \tilde{x}_i + \epsilon_i$ and for a general y, $y = \tilde{w}^\intercal \tilde{x} + \epsilon$.
Our goal is then to find a solution that minimizes the residual ϵ [Ige17b].

7.3.1 Using Moore-Penrose pseudoinverse to estimate weights

The problem of minimizing the residuals corresponds to estimating optimal weights $\tilde{w}$. We resolve to use a classical approach called Moore-Penrose pseudoinverse in order to minimize the residual. This approach isolates $\tilde{w}$ on matrix-form which makes the weight estimation of $\tilde{w}$ easy to implement and interpret.

One way of handling this problem could be to minimize the empirical risk under the squared loss $\ell(y, \hat{y}) = (y - \hat{y})^2$, where $\hat{y}$ is our expected response of y. That is, we want to minimize the mean-squared error. Let n be the number of samples contained in some data set $\tilde{X} \in \mathbb{R}^{n \times (d+1)}$ with the extended 1-column, and $Y \in \mathbb{R}^n$ be the responses. The mean-squared-error is then defined by

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \tilde{w}^\intercal \tilde{x}_i) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{w}^\intercal \tilde{x}_i)^2$$

> **Fermat's Theorem:**
> If $f(\tilde{w})$ has a local extremum at $\tilde{w} = \tilde{w}^*$ and f is differentiable at $\tilde{w}^*$, then $f'(\tilde{w}^*) = 0$.

If we utilize Fermat's Theorem and set the partial derivative to be 0, we can obtain the optimal weights of $\tilde{w}$ by minimizing the mean-squared-error. Thus we have [Ige17b][Gui]

$$
\begin{aligned}
0^\intercal &= \frac{\partial}{\partial \tilde{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{w}^\intercal \tilde{x}_i)^2 = \frac{\partial}{\partial \tilde{w}} \frac{1}{n} \sum_{i=1}^{n} y_i^2 - 2y_i \tilde{w}^\intercal \tilde{x}_i + (\tilde{w}^\intercal \tilde{x}_i)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} 2\tilde{w}^\intercal \tilde{x}_i^\intercal \tilde{x}_i - 2\tilde{x}_i^\intercal y_i = \frac{1}{n} \sum_{i=1}^{n} -2(y_i - \tilde{w}^\intercal \tilde{x}_i)\tilde{x}_i^\intercal \\
&= -\frac{2}{n} \sum_{i=1}^{n} (y_i - \tilde{w}^\intercal \tilde{x}_i)\tilde{x}_i^\intercal
\end{aligned}
$$

and since $\frac{2}{n} f(\tilde{w})$ has the same optimum at $\tilde{w}^*$ as $f(\tilde{w})$, we can omit the scaling. We therefore have

$$0^\intercal = - \sum_{i=1}^{n} (y_i - \tilde{w}^\intercal \tilde{x}_i)\tilde{x}_i^\intercal$$

The negative sign can be eliminated, since it does not make a difference whether the residual is minimized from the positive or negative direction. We then extend it further and get

$$0^\intercal = \sum_{i=1}^{n} (y_i - \tilde{w}^\intercal \tilde{x}_i)\tilde{x}_i^\intercal = \sum_{i=1}^{n} y_i \tilde{x}_i^\intercal - \tilde{w}^\intercal \sum_{i=1}^{n} \tilde{x}_i \tilde{x}_i^\intercal$$

On matrix form we now have that
$$0^\intercal = y^\intercal \tilde{X} - \tilde{w}^\intercal \tilde{X}^\intercal \tilde{X}$$

We can now derive it on matrix form:

$$
\begin{aligned}
0^\intercal &= y^\intercal \tilde{X} - \tilde{w}^\intercal \tilde{X}^\intercal \tilde{X} \\
\tilde{w}^\intercal \tilde{X}^\intercal \tilde{X} &= y^\intercal \tilde{X} \\
(\tilde{X}^\intercal \tilde{X})^\intercal \tilde{w} &= \tilde{X}^\intercal y
\end{aligned}
$$

Assuming $\tilde{X}^\intercal \tilde{X}$ has full rank we can isolate the weights and get the maximum likelihood estimate [Ige17b]

$$\tilde{w} = (\tilde{X}^\intercal \tilde{X})^{-1} \tilde{X}^\intercal y$$

7.3.2 Deriving the maximum likelihood from minimizing the MSE

To convince that we in fact reach the optimal weights w by minimizing the mean-squared-error, we can consider the maximum likelihood estimate.

The likelihood of some weights w given the training data $\mathcal{D} = \{(x_1, y_1), \ldots (x_n, y_n)\}$ is the probability of observing $\mathcal{D}$ when the data is generated by hypothesis h with parameters w. The likelihood for y with independent and identically distributed random variables is given by

$$MLE(y) = \prod_{i=1}^{n} P(Y = y_i | X = x_i; h) \text{ or just } \prod_{i=1}^{n} P(y_i | x_i)$$

Expressed in another way, our task is to find some weights w applied to X that maximizes the probability of observing y given x. That is we solve the maximum likelihood estimator

$$MLE(y) = \arg\max_{w} \prod_{i=1}^{n} P(y_i | x_i; h)$$

Normal/Gaussian distributions are often used to model observed data. If we consider a deterministic real-valued target function perturbed by zero-mean additive Gaussian noise with variance σ^2. That is, the regression errors are normally distributed with mean 0 and some known σ^2, we have

$$P(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-f(x))^2}{2\sigma^2}}$$

Given our dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, the likelihood of the model $h(x) = w^\mathsf{T} x$ is given by

$$P((y_1, \ldots, y_n) | (x_1, \ldots, x_n), w, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w^\mathsf{T} x_i)^2}{2\sigma^2}}$$

If we take the natural logarithm on both sides, we can log-minimize

$$\ln P(y|x, w, \sigma^2) = \ln \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w^\mathsf{T} x_i)^2}{2\sigma^2}}$$

and utilize the fact that

$$\ln \prod_{i=1}^{n} P(y_i | x_i) = \sum_{i=1}^{n} \ln P(y_i | x_i)$$

in order to derive that

$$\ln P(y|x, w, \sigma^2) = \sum_{i=1}^{n} \ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w^\mathsf{T} x_i)^2}{2\sigma^2}}$$

If we now derive it further

$$
\begin{aligned}
\ln P(y|x, w, \sigma^2) &= \sum_{i=1}^{n} \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w^\mathsf{T} x_i)^2}{2\sigma^2}} \right) \\
&= \sum_{i=1}^{n} \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \ln \left(e^{-\frac{(y_i - w^\mathsf{T} x_i)^2}{2\sigma^2}} \right) \\
&= n \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \sum_{i=1}^{n} -\frac{(y_i - w^\mathsf{T} x_i)^2}{2\sigma^2} \\
&= n \ln(1) - n \ln \left((2\pi\sigma^2)^{\frac{1}{2}} \right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^\mathsf{T} x_i)^2 \\
&= -\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^\mathsf{T} x_i)^2
\end{aligned}
$$

We have some terms $-\frac{n}{2}\ln(2\pi)$ and $-\frac{n}{2}\ln(\sigma^2)$ whom are independent of w and thus they can be omitted. Also scaling the function, in this case by $\frac{1}{2\sigma^2}$ can be omitted, since neither changes the w that maximizes the function [Ige17b]. We are left with

$$\ln P(y|x, w, \sigma^2) = -\sum_{i=1}^{n}(y_i - w^\intercal x_i)^2$$

It is seen that the closer we get to 0 the better our maximum likelihood estimate becomes. A perfect fit would be if $\ln P(y|x, w, \sigma^2) = 0$ which implies $P(y|x, w, \sigma^2) = 1$. That is we have a hypothesis h using w that perfectly minimizes the error of our guess on y, given x, weighted by w. Thus we see that minimizing the mean-squared error towards 0 corresponds to maximizing the maximum likelihood of w.

7.3.3 Assumptions

Assumptions when applying multiple regression [STE]

- *Input*: Let y be an interval variable $y \in (-\infty, +\infty)$, or ratio variable $y \in [0, +\infty)$ that is dependent on x. Let x be two or more interval, ratio, or dichotomous variables that is independent of y. For multiple regression, no specific distribution of x and y is assumed [Fox15].

- *Sample size*: It is possible to train the model on a small sample size. However, to get reliable results, the sample size should be at least $n > 50 + 8d$ where d is the number of independent variables [R. 07].

- *Outliers*: Outliers can have considerable impact on the modelled solution, and their inclusion should therefore be carefully considered. Multivariate outliers can be adjusted for by considering the modified Z-score for each sample, as described in 7.2.1.

- *Multicollinearity and singularity*: Multicollinearity and singularity may let the regression model put too much weight on features that contain the same information. Thus multicollinnearity and singularity is assumed to not be present when applying regression. This will be explained further in section 7.4.1.

- *Normality*: The residuals about the predicted responses should be normally distributed around 0. This can for example be checked by plotting the frequency of the residuals to see the distribution.

- *Linearity and homoscedasticity*: The residuals should have a straight line relationship with the predicted responses. That is, the response variables should somewhat follow the regression line. This can be checked through a scatter plot of standardized residuals vs the standardized predicted responses. In this plot, a relationship should not be seen between the standardized predicted responses and standardized residuals.

  The residuals should also be somewhat rectangular distributed around 0. That is, data points should follow a horizontal line with a roughly constant variance along the vertical line. The variance of the residuals about predicted responses should be roughly the same for all predicted responses.

  Homoscedasticity is assumed. An example of when homoscedasticity is not present (heteroscedasticity) is when the residuals have larger or smaller variance when the predicted values increase. A sideways trapezoid could thus indicate lack of homoscedasticity.

- *Independence of residuals*: Independence of residuals can be assumed if one has collected cross-sectional data randomly [Ban].

7.4 Defining the regression models

The goal is to create an assessment of a player's performance within a variety of aspects, where the role is also taken into consideration. For each of the 6 roles we defined in section 5.5, we therefore create a regression model for each of the 15 features that were described in section 7.2. All the regression models were only trained on matches that were won in order to predict the responses using a model that considers a winning outcome. Each regression model consists of an interception of the regression line, along with 14 regression coefficients that weight the explanatory variables.

The total number of regression models created is $6 \cdot 15 = 90$. A player with some role thus gets assessed on 15 different aspects in the match. Using the assessment, the player can then compare their actual stats with the expected stats for a match and see if something can be improved.

7.4.1 Avoiding bias in the regression model from multicollinearity & singularity

When performing multiple regression, one assumption is that the data set does not contain multicollinearity or singularity, as stated in 7.3.3. When multicollinearity or singularity exists in a data set, we risk that the regression models will be biased when predicting the response. Before performing the multiple regression, we therefore check if this exists.

Multicollinearity exists when there is a high correlation between some explanatory variables, while singularity exists when there is a perfect correlation between some explanatory variables. A larger amount of information are therefore covered by the same variables and their presence will therefore affect the response variable by putting too much importance on multiple features that cover the same information. To combat this, we can make an interaction model. One property of the linear model

$$y = \beta_1 x_1 + \cdots + \beta_d x_d + \alpha + \epsilon$$

is that β_i is the change in y when x_i increases by 1 and all the other independent variables are held fixed. This is the same regardless of the values of the other variables. However, not all real-world situations work like that. When the effect of one variable on the response is dependent by the level of another variable, we can say that they interact (multicollinearity). Given a regression model of some degree, we can instead construct a new model with a weighted interaction variable that jointly explains some change in y per change in the interaction model. Consider an example of a multivariate regression model

$$y = \beta_1 x_1 + \beta_2 x_2 + \alpha + \epsilon$$

If x_1 and x_2 interact we introduce a new variable $x_3 = x_1 x_2$ and consider the following regression model instead [Int14]:

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \alpha + \epsilon$$

To see if an interaction model is necessary, we detect the presence of collinearity and singularity by examining the correlation matrix. Using a threshold of e.g. $t = \pm 0.9$, one can detect if there is a pair of variables that appear to be highly multicollinear, and if so, introduce a new variable for the regression model. If $t = 1$, the pair of features is instead singular, and in this case we can remove one of them, since the information is preserved in the other. Let $f_i = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ be the i'th feature of the data set. The correlation matrix is then defined by

$$\begin{bmatrix} r_{f_1,f_1} & r_{f_1,f_2} & \cdots & r_{f_1,f_d} \\ r_{f_2,f_1} & r_{f_2,f_2} & \cdots & r_{f_2,f_d} \\ \vdots & \vdots & \ddots & \vdots \\ r_{f_d,f_1} & r_{f_d,f_2} & \cdots & r_{f_d,f_d} \end{bmatrix}$$

where each entry r_{f_x,f_y} denotes the sample Pearson's correlation coefficient $\in [-1, 1]$ defined by

$$r_{f_x,f_y} = \frac{\sum_{i=1}^{n}(f_{x,i} - \bar{f}_x)(f_{y,i} - \bar{f}_y)}{\sqrt{\sum_{i=1}^{n}(f_{x,i} - \bar{f}_x)^2}\sqrt{\sum_{i=1}^{n}(f_{y,i} - \bar{f}_y)^2}}$$

A value of -1 means that the pair of features have perfect anti-correlation, while a value of 1 means that the pair of features have perfect correlation. A value such as 0.20 would therefore represent a relatively weak positive correlation [Gru15d]. The following figure displays the correlation matrix for our regression data, where only the bottom triangle is color coded due to the correlation matrix being a symmetric matrix. The entries are colored red if $|r_{f_x,f_y}| \in [0, 0.25)$, yellow if $|r_{f_x,f_y}| \in [0.25, 0.75)$, and spring green if $|r_{f_x,f_y}| \in [0.75, 1)$. Finally, an entry is colored green if the feature pair is singular. No collinearity or singularity was found between the pairs of different features and it is therefore not necessary to remove any of the features before we perform linear regression.

	kills	deaths	assists	hero healing	last hits	gold per minute	exp per minute	tower damage	hero damage	obs placed	sen placed	my team score	enemy team score	my team gold adv	my team exp adv
kills	1	-0.114	0.148	-0.061	0.539	0.727	0.708	0.463	0.737	-0.265	-0.223	0.521	0.11	0.28	0.283
deaths	-0.114	1	0.203	-0.001	-0.092	-0.326	-0.215	-0.315	0.086	0.252	0.251	0.164	0.731	-0.385	-0.39
assists	0.148	0.203	1	0.198	0.013	0.047	0.223	0.01	0.326	0.292	0.26	0.724	0.187	0.356	0.391
hero healing	-0.061	-0.001	0.198	1	-0.072	-0.071	-0.016	-0.009	-0.052	0178	0.155	0.111	0.001	0.078	0.091
last hits	0.539	-0.092	0.013	-0.072	1	0.805	0.666	0.505	0.671	-0.295	-0.245	0.519	0.073	0.36	0.387
gold per minute	0.727	-0.326	0.047	-0.071	0.805	1	0.849	0.671	0.618	-0.412	-0.342	0.403	-0.063	0.42	0.381
exp per minute	0.708	-0.215	0.223	-0.016	0.666	0.849	1	0.515	0.617	-0.295	-0.245	0.519	0.073	0.36	0.387
tower damage	0.463	-0.315	0.01	-0.009	0.505	0.671	0.515	1	0.307	-0.251	-0.21	0.281	-0.182	0.413	0.373
hero damage	0.737	0.086	0.326	-0.052	0.671	0.618	0.617	0.307	1	-0.232	-0.191	0.515	0.356	0.113	0.116
obs placed	-0.265	0.252	0.292	0.178	-0.412	-0.412	-0.295	-0.251	-0.232	1	0.787	0.125	0.102	0.018	0.018
sen placed	-0.223	0.251	0.26	0.155	-0.332	-0.342	-0.245	-0.21	-0.191	0.787	1	0.112	0.134	-0.015	-0.011
my team score	0.521	0.164	0.724	0.111	0.313	0.403	0.519	0.281	0.515	0.125	0.112	1	0.22	0.529	0.536
enemy team score	0.11	0.731	0.187	0.001	0.238	-0.063	0.073	-0.182	0.356	0.102	0.134	0.22	1	-0.529	-0.536
my team gold adv	0.28	-0.385	0.356	0.078	0.094	0.42	0.36	0.413	0.113	0.018	-0.015	0.529	-0.529	1	0.866
my team exp adv	0.283	-0.39	0.391	0.091	0.073	0.381	0.387	0.373	0.116	0.018	-0.011	0.536	-0.536	0.866	1

Table 13: Correlation matrix for all features used in the regression models.

7.4.2 Algorithm

Algorithm: Multiple Variable Linear Regression

Input: Data set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathbb{R}^d \times \mathbb{R})^n$.

1 Let y be the vector of responses, $y = (y_1, \ldots, y_n)^\intercal$.

2 Let $\tilde{X}$ be the matrix containing explanatory variables with an added 1-column:

$$\tilde{X} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1d} & 1 \\ x_{21} & x_{22} & \ldots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nd} & 1 \end{bmatrix}.$$

3 Compute $(\tilde{X}^\intercal \tilde{X})^{-1} \tilde{X}^\intercal y$ and return the estimated coefficients and intercept $(\hat{\beta}_1, \ldots, \hat{\beta}_d, \hat{\alpha})$.

Output: $(\hat{\beta}, \hat{\alpha})$, defining the affine linear model $h(x) = \hat{\beta}^\intercal x + \hat{\alpha}$.

7.4.3 Verifying that data and regression model satisfies assumptions

- **Input:** For all regression models, the output Y, and features contained in the data set X satisfies the assumption of belonging to the set of real numbers.

- **Sample size:** For each regression model we should have that their training set has sample size $> 50 + 8d$. In our case we have $d = 14$ explanatory variables for each regression model. The smallest training set is that of role 1 which has $n = 20816 > 50 + 8 * 14$ samples. Thus the sample sizes are indeed satisfied.

- **Outliers:** We removed all $n = 0$ outliers identified by the modified Z-score. Assumptions about outliers are thus satified.

- **Multicollinearity and singularity:** In order to check for multicollinearity and singularity we computed the correlation matrix in section 7.4.1. We saw from the correlation matrix that we do not need to craft any interaction models, and thus the correlations satisfies our assumptions about multicollinearity and singularity.

- **Normality:** In our setup we may predict above/below the actual value since the regression model is trained on won games. The expected values are therefore not always close to the actual values, but should still yield normal distributions. We see from the residual plots in appendix figure 29 that the residuals seem normal distributed around 0, which satisfies the assumption about normality.

- **Linearity and homoscedasticity:** From the residual plots in appendix figure 30, it is seen that the points are primarily symmetrically distributed around a horizontal line, where the variance is roughly constant. For some plots, we see that the predicted output is bounded by the negative value of itself. This happens when the observed response of a ratio variable is zero since the residual ϵ is defined as the difference between the observed response y and predicted response $\hat{y}$, $\epsilon = y - \hat{y}$ which doesn't violate the assumption. Thus the assumptions about linearity and homoscedasticity are met.

- **Independence of residuals:** We have cross-sectional data, where the data set contains at most 1 record of each played match, and at most 1 record of each player in that match. Also all matches are of equal importance for the data set. We can thus assume the independence of residuals to be met.

7.5 Results

The regression models were tested on 2 randomly selected players that played role 1 and role 4 respectively. The results are seen below in colorcoded terms, where a red value represents under-performance (value differs more than 1% negatively from the expected value), and a green value represents overperformance (value differs more than 1% positively from the expected value). A grey value represents that the player value was within 1% of the expected value or the player value was equal to the nearest integer of the expected value. Note: For deaths and enemy team score, the color is inverted since these features represent negative impact on the game.

A player from role 1	Actual	Expected
kills	6	6.91
deaths	10	8.2
assists	17	21.57
hero healing	10522	7093.52
last hits	26	65.36
gold per min	357	326.74
exp per min	524	456.01
tower damage	1759	1025.26
hero damage	10516	13306.24
observers placed	17	14.35
sentries placed	20	15.56
team score	51	45.23
enemy team score	38	42.83
team gold advantage	9710	12963
team exp advantage	4917	12385.93

A player from role 4	Actual	Expected
kills	9	7.91
deaths	8	9.35
assists	16	17.15
hero healing	0	1197.91
last hits	42	91.77
gold per min	339	356.96
exp per min	540	498.58
tower damage	0	906.53
hero damage	20805	20811.74
observers placed	1	1.05
sentries placed	0	1.35
team score	42	43.15
enemy team score	43	40.71
team gold advantage	-13067	4271.91
team exp advantage	1532	253.68

Table 14: Assessment of player performance for data point #57 from role 1, where the player won.

Table 15: Assessment of player performance for data point #24 from role 4, where the player lost.

Recall from chapter 5.5 that role 1 was a supporting role characterized by high values in e.g. *hero healing* and *observers placed*. Meanwhile, role 4 was a semi-fighter characterized by e.g. high values in *last hits*. It is seen that the performance assessments do take the roles in consideration by showing different expectations for the different roles. For example, the player from role 1 is expected to have more *hero healing* than the player from role 4.

Looking at the performance of the two randomly selected players, the player from role 1 performed well in most of the aspects, with room for improvement in for example the number of kills made. This player also ended up winning the match. For the player from role 4 that lost a match, the player lived up to some of the expected values, but most were below the expected values. It is seen that the difference between the actual and expected gold advantage for the player's team was large, which indicates that it was not only this player having difficulties with living up to the expected performance. It was generally a difficult match for the team. So overall, the player from role 4 did alright, but of course with room for improvement.

By giving a player an assessment of their performance in a match, the player can thus evaluate what areas they need to focus on improving. This is based upon the expectations from the regression model that takes multiple factors into consideration, rather than a naive assessment based on descriptive statistics such as average values.

7.6 Performance and evaluation

To measure the goodness of fit for our linear regression models, there are different methods available, such as root-mean-square error (RMSE) or coefficient of determination (R^2). RMSE displays how much our predictions deviate from the actual responses on average. This can however be difficult to evaluate without having a different regression model (using the same data set) to compare with, as the interpretation of the RMSE depends on comparison. We use unique data sets for each regression model, so we opted for R^2 score instead to measure the performance. It measure the variance captured and is easy to interpret since the value lies in the interval [0, 1]. Recall from section 4.5.5 that the R^2 score was given by

$$R^2 = 1 - \frac{SSR}{TSS}$$

where *TSS* is the total sum of squares and *SSR* is the sum of squared residuals. To compute the R^2 score for each regression model, we separated the test data into won and lost matches. We then computed the R^2 score for won matches since these matches more likely represent the value we want to predict and the results are seen below.

	Role 1	Role 2	Role 3	Role 4	Role 5	Role 6	Arbitrary role
kills	0.63	0.63	0.74	0.73	0.76	0.74	0.71
deaths	0.69	0.69	0.65	0.64	0.62	0.63	0.65
assists	0.62	0.58	0.55	0.55	0.53	0.62	0.58
hero healing	0.28	0.02	0.16	0.07	0.05	0.02	0.1
last hits	0.68	0.66	0.58	0.71	0.73	0.71	0.68
gold per min	0.76	0.74	0.73	0.8	0.81	0.83	0.78
exp per min	0.59	0.58	0.64	0.64	0.63	0.61	0.62
tower damage	0.2	0.22	0.2	0.29	0.25	0.34	0.25
hero damage	0.68	0.7	0.73	0.74	0.81	0.74	0.73
observers placed	0.54	0.49	0.43	0.56	0.36	0.3	0.45
sentries placed	0.51	0.46	0.43	0.55	0.34	0.25	0.42
team score	0.67	0.68	0.68	0.7	0.69	0.69	0.69
enemy team score	0.85	0.85	0.84	0.84	0.83	0.82	0.84
team gold advantage	0.52	0.49	0.49	0.48	0.5	0.47	0.49
team exp advantage	0.58	0.58	0.59	0.59	0.57	0.57	0.58

Table 16: R^2 scores for regression models on won matches.

We see that the goodness of the fit in terms of variance captured by the regression models differ for both the role-specific regression models and the feature-specific regression models. It is seen that the regression models do not predict *hero healing* very well. This could indicate that multiple latent variables are describing the hero healing. Something to notice though is that hero healing can be explained in the regression models for role 1 and 3 to a certain extent.

Tower damage also seems to be hard to predict properly based on the current features within in the data set. A feature like *sentries placed* would match the actual value better if the player is playing role 4 instead of role 6. Meanwhile, *gold per min* is the feature that is explained the best in the regression models, with a high amount of variance captured for all roles.

It can therefore be concluded that the expected value outputs of some regression models should be taken with a grain of salt, while other regression models may indeed give a reliable expected value.

For lost matches, the R^2 score will most likely show bad results, since we do not try to predict the actual value for a lost match, but rather the expected value for a winning match. The table below displays the average R^2 score for lost matches for each of the 15 features

	Arbitrary role
kills	0.57
deaths	0.49
assists	0.66
hero healing	-0.03
last hits	0.67
gold per min	0.78
exp per min	0.68
tower damage	-1.17
hero damage	0.64
observers placed	0.29
sentries placed	0.35
team score	0.72
enemy team score	0.66
team gold advantage	-1.25
team exp advantage	0.08

Table 17: R^2 scores for regression models on lost matches.

Some of the R^2 scores do match up against those of the R^2 score from table 16.
Otherwise the scores are as expected; not very great for the regression models. For example, the R^2 score for *observers placed* and *team exp advantage* is quite low, and some of the other features are even negative. As mentioned in section 4.5.5, a negative R^2 score can be interpreted as 0 here. The negative R^2 score in this case is caused by the sum of squared residuals *SSR* being large, which is expected when we do not try to predict the actual value for a lost match. It is therefore difficult to conclude whether the assessment of performance is reliable for players that lose their match. This is up for future work.

8 Implementation

To solve how win probability can be optimized in our thesis, we have implemented some programs in the programming language Python 3 using Spyder, an integrated development environment (IDE).

The code is located in a GitHub folder. Furthermore, the folder contains a sub-folder named *src* that contains source code, as well with code used for gathering the match data and pre-processing to make it suit different needs depending on the problem being solved.

The data sets are stored in Google Drive, which is a file storage service. The files are numbered to show the order in which the files were created and used in the thesis. Microsoft Excel was used to view the files and for computing descriptive statistics such as the average or standard deviation for different features.

9 Discussion

9.1 Historical data's effect on performance

Historical data refers to data about previous events and circumstances related to a particular subject, such as team sports. When learning from data, one should be cautious about the time frame in which the data was collected.

Using historical data may affect the models negatively, as some team sports are more subject to change than others. Dota 2 and other electronic sports games occasionally receive updates where new content is released or existing content is tweaked or balanced. This may alter the dynamics of the game by introducing new play styles and strategies. One example of a new play style is when Dota 2 added a new character named *Techies* to the game back in 2014. With its unique ability to place obstacles on the playing field that damage enemy players when they move close by, the opponents suddenly found themselves having to move much more carefully around the playing field.

The accuracy of our optimization model and prediction of win rate can therefore be impacted negatively if it was computed by using historical and outdated data. In order to give players valid suggestions, the optimization models should be up to date with the game. A potential problem with games that are very dynamic is that it may be difficult to find data that is up to date when new updates are released. It may take some time before enough data is available. A possible solution for this would be to incorporate the new data into the historical data, so the models gradually improve over time and take the new play styles or strategies into consideration.

For more established team sports such as soccer or handball, the use of historical data is not as problematic. Players may change, but the overall game and objectives remain the same. One possible benefit for established team sports is that data can be gathered over a larger time frame and it can thus be avoided that the sample size is too small. For example, in the assessment of a player's performance, it was assumed that the sample size had to be at least of some size in order to get reliable results. Another benefit with established team sports is also that the models do not have to be updated often, as large modifications in the game and its rules rarely happen.

9.2 Analysis of roles and team compositions

9.2.1 Expected Performance

In many team sports, it is generally assumed that the team composition, i.e. the distribution of roles, has an actual impact on the winning potential in a team game. This is for example seen as the coach of e.g. a soccer team thoughtfully distributes roles for the players.

From chapter 5.5 where team compositions in Dota 2 matches were analyzed, we saw that there can be a somewhat large difference in the probability of winning based on the team compositions chosen. Teams with a good variety and balance of roles generally performed well with a win rate probability of up to 59.6%. On the other hand, teams that lacked variety and balance in the team composition would have a win rate probability reaching as low as 39.0%.

One downside of our solution is that we have simplified the problem of assembling a team to consist of selecting heroes from specific roles, instead of viewing them as unique heroes that each have something to offer. With 115 available heroes to choose from and a team size of 5, the number of possible team compositions is equal to the binomial coefficient $C(115, 5) = \binom{115}{5} = 2,813,729,380$. With almost three billion possible team compositions, it would require an enormous amount of matches gathered to be able to compute statistics about each possible team composition. This is simply too difficult or even impossible.

To incorporate more information about the individual heroes into our model, we could instead have computed some statistics about which heroes generally perform well against some other hero. In the beginning of every Dota 2 match, the players from each team take turns to pick a hero to play. Our model could then give more specific suggestions about which hero to choose, based on both the roles that the team currently needs and on the heroes that the enemy players have chosen so far. This may have improved the performance even further.

9.2.2   Roles and team compositions in other team sports

The analysis of roles and team compositions can also be done for popular team sports such as soccer or handball. The hierarchical clustering process of pairwisely clustering characters together to define roles can easily correspond to hierarchically clustering players together. The table below shows some features that could be interesting to consider for players in soccer

Player Height
Player Weight
Player Age
Average running speed with ball possession
Average possession time with the ball per pass
Ball-pass accuracy
Ball steals per game
Average goals per game
Average shots on goal per game
Scoring accuracy
Average assists per game
Average fouls per game
Tackle accuracy

Table 18: Some features that could be considered in soccer games.

The hierarchical clustering process can make it possible to evaluate questions such as

- **How many types of players that are on the team.**
  By e.g. using the gap-statistic to find the optimal number of types. It would then be possible to investigate what kind of characteristics each type has.

- **Dividing the players into e.g. 3 or 4 classes that fit most soccer lineups.**
  Thus finding the players best suited for attackers, mid-fielders, and defenders.

- **Finding out which team formation to use for a given match.**
  The word *formation* is used in soccer to portray how the players in a team position themselves on the field. The position of a player generally defines if the player mostly has a defensive or attacking role. Similar to how team compositions can vary in Dota 2, the formations in soccer games can also vary from team to team. After clustering the players, it is then possible to compute team composition matrices so a coach can gain insight into which team formations generally perform well both independently and dependently on the opponent team's formation.

- **Getting a clear picture of which individual players are most similar to each other.**
  This could help a coach pick out, or opt out, the right players for each role in a match, given a large number of potential players. An example of this could be to consider the popular Danish soccer player Christian Eriksen who plays a key part of the Danish mid-field, but also a type of player whom the coach does not want more than 1 off in the team.

If you would go up against a team who is known to be weak against counter attacks in handball and the popular Danish wing player Casper U. Mortensen has proven to be an excellent choice for this task, and thus the coach's favourite choice for the left wing player. Then one could, by selecting the features that does a great counter attacker, for instance the features of table 19, find the players whom are most similar in play style, to that of Casper U. Mortensen, thus that you could put a similar player on the right wing spot. The table below shows some features that could be interesting to consider

Goal accuracy on 100% chances
Acceleration after failed opponent attack
Avg. running speed without the ball
Avg. running speed while dribbling
Ball-catch accuracy from >15 meter range

Table 19: Some features that may define a great counter attacking wing player.

9.3   Live analysis of win probability

From the live analysis of win probability, we saw that gold and exp advantage generally has a large influence on which team will win in the end. The live analysis predicted the correct winner 69.2% of the time, which is relatively good, considering that it is difficult to predict the outcome correctly for the early stages of a match.

To improve the model, we discussed that more live features could have been incorporated, if they had been available. Another possible thing to improve would be the live analysis graphs that depict the win probability over time. Overall, the graphs were fluctuating a lot, which is somewhat understandable since Dota 2 is a fast paced game where the advantage can change significantly throughout the match. However, many of the fluctuations also happened from minute to minute, which seems a bit extreme. To decrease the minute-wise fluctuations, we should perhaps have used even larger k-values. The figures below show one of the graphs from our live analysis section, along with the exact same graph with $k = 10001$ neighbors considered instead. It is seen that the new graph is more smooth, especially in the middle section, when comparing with the original live analysis graph. The higher k is, the more the probability will reflect the data's distribution.
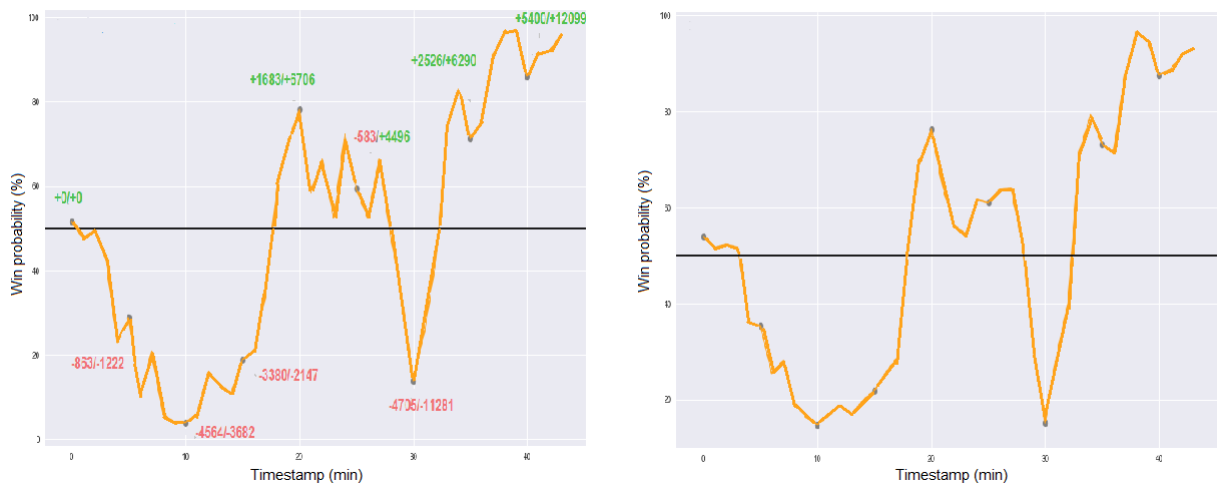

Figure 25: Live analysis graphs for $k = 101$ and $k = 10001$ respectively.

9.3.1 Live analysis of other team sports using logistic regression

For the particular game that we based the live analysis on, the logistic regression did not achieve great results. This was probably, as we discussed, due to a large amount of matches being equal in terms of advantage. Dota 2 is a game where the competing teams are often close to be on an equal footing in the early stages of the matches.

Since the classic logistic regression uses a hyperplane in order to segment the data linearly, it would perhaps work much better in team sports where the state of the game more easily favors one of the teams. This type of live analysis in general could also be applied in soccer by taking into account features such as ball possession, passing accuracy, the distance the ball is to the enemy goal, etc. This is presumably viable since traits such as ball possession and passing accuracy can make good indicators of who controls the flow of the game.

9.3.2 Sports betting market

The sports betting market is very dependent on statistics about all sorts of betting possibilities put out by the bookmaker. In the *Las Vegas* football betting market, the structure is relatively straight forward. The bookmaker ensures that bettors must be right 52.4 percent of the times on binary resulting bets in order to break even [K. 97].

However, for the bookmaker to set the proper odds to ensure that the return of investment is positive, the bookmaker relies on various kinds of predictions for game outcomes. In soccer betting, it is not uncommon to base many predictions on historical data for different leagues mapped into attacking strength and defending strength. These can then be used in i.e. Poisson distributions to predict the most likely goal score, who to score the first goal, etc.

Using some of the techniques we discussed in live analysis and clustering of roles, the betting market both in terms of the bookmaker and the bettor, can benefit from having additional variables to base their set of odds and/or prediction on.

9.4 Assessment of player performance

9.4.1 Optimizing performance of regression models

In the assessment of player performance, it was concluded that most but not all of the regression models gave a reliable expected value when testing on matches where the player won. We were however unable to measure if the player performance assessment was reliable for players that lose their match.

Another improvement that could be done is the computation time. In our data set from the player assessment section 7 we have 366.514 samples and 15 features to consider the regression formula from. If one were to consider more samples in the future, with even larger dimensionality, it would probably be faster to not compute $(\tilde{X}^\intercal \tilde{X})^{-1}$ in practice. Rather one could consider a QR-decomposition of $\tilde{X}$ and use e.g. back substitution in order to solve the system of linear equations $R\tilde{w} = Q^\intercal y$. QR is then equal to $\tilde{X}$, where Q is an orthogonal matrix, that is $Q^\intercal Q = I$, and R is an upper triangular matrix [Nic15].

9.5 Demand for statistics/guidance services

9.5.1 Dota Plus

Around the beginning of our thesis, a monthly subscription service named Dota Plus was launched for Dota 2. Similar to our learning goals, the service also has a purpose of showing players how they can improve themselves with the use of statistics and in-game suggestions for e.g. what items a player should buy or what hero to choose [Dot18a]. We do not know what exact methods or algorithms Dota Plus bases all their improvement suggestions on, so the methods we have used in our thesis have been selected independently of Dota Plus.

When a match has finished, a player will be able to view different kinds of information about the match, such as the win probability of each team over the course of the match. In contrary to our live analysis of win probability, their computation of the odds of winning are not always around 50/50 at the start of the game. One possible reason may be that Dota Plus takes team compositions into consideration. From our analysis of team compositions, we saw that there could be a significant difference in win rate probability based on the team compositions chosen. Another possible reason could be that there is a gap in the overall skill level of the two teams facing each other. When searching for a match, the game tries to create teams based on different criteria such as matching players of similar skill level together and minimizing the discrepancy in skill between the most and least skilled players [Dot18b]. However, this may not always be possible due to few players currently being online or for other reasons.
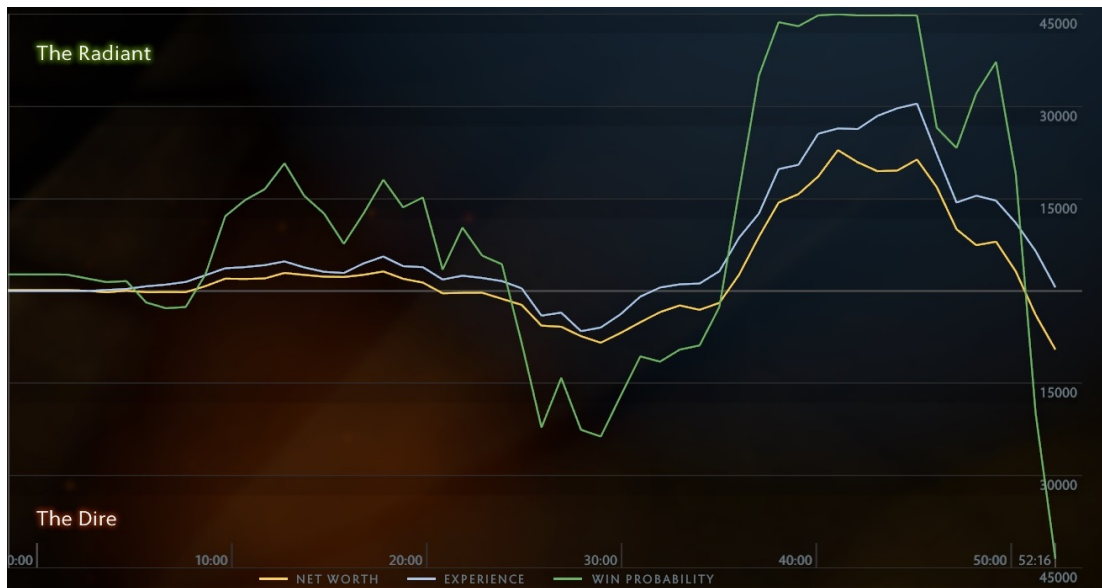

Figure 26: A Dota Plus graph of net worth, experience, and win probability over time [noa18].

Another thing that Dota Plus offers is an assessment of the player's performance. While our assessment was based on regression-analysis, theirs appear to be based on mean value statistics. When a match is completed, the player is able to see graphs that display their net worth and level over time and compares them with the expected values for players of a similar skill level. The graphs display a player's performance for fewer features than in our player assessment, but Dota Plus is however able to give an assessment of performance over time. This allows the player to see at what stages of the game the player may be underperforming. Thus they can try to improve their play style in e.g. the late stages of games in general.

The launch of this service illustrates that there is a market for services/models that can help players improve their skills; not only for professional players, but also for normal players. The number of frequent viewers and enthusiasts of electronics sports (eSports) was 143 million in 2017. By 2021, this number is expected to increase to 250 million. As electronics sports continues to gain more popularity, the demand for such services will most likely increase further in the future, as people seek out tools for increasing their skills [The18].
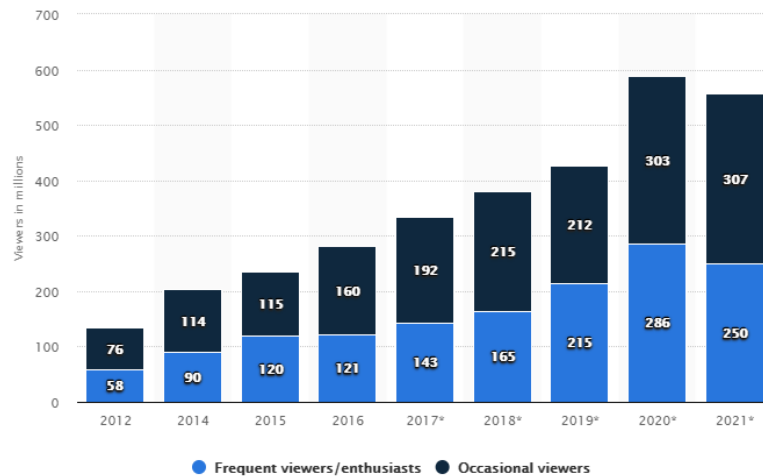
Figure 27: Global eSports audience size and estimated growth.

9.5.2 Customized guidance

The companies that develop these competitive eSports games most likely gather a lot of valuable data. Both about the matches that take place, but also data about the individual players. This could for example include what roles or characters they like to play, what kind of characters they perform well or bad against, and how often they play. This could be used to create customized guidance services that displays statistics about the user over time, so the user can see if they are heading in the right direction or if there are some weaknesses they have to focus on.

10 Conclusion

We saw that hierarchical clustering along with the *gap statistic* proved well for identifying roles. These roles were used in the analysis of team compositions in order to identify which team compositions that perform worse or better than an arbitrary team composition. We also discussed a variety of other problems that could be solved by considering this technique. I.e. identifying similar players, defining roles and their traits, dividing players into a specific number of roles, etc.

In the live analysis, we managed to predict winning probabilities by using a modified version of the *k-nearest neighbors* algorithm. A similar setup for exploratory data analysis and inferential data analysis can then be used to confirm the validity of performing a predictive data analysis on live data for other team sports.

For each role derived from the hierarchical clustering process, we managed to create dynamic regression models that took multiple variables into consideration. These regression models could then be used to display player performance assessments.

It can therefore be concluded that using some of the dynamic algorithms covered in this thesis can lead to optimizing the winning potential in team sports. Finally, better predictions could perhaps be obtained by combining the results of numerous predictors and models. That is the very essence of ensemble learning.

11   Learning Goals

- Present some of the newest literature within Game Data Analytics and Machine Learning

- Explain basic terms within Game Data Analytics

- Explain advanced methods in Machine Learning.

- Apply methods such as data pre-processing, dimensionality reduction, and supervised learning to analyze game data.

- Test complex hypotheses using statistical methods such as hypothesis testing and regression.

- Present results, using visual means such as graphs and diagrams and fitting descriptions.

- Analyze and discuss the results and improve the analytical methods derived from the partial conclusions.

- Reflect on possible uses of methods and applications used in the project to other team sports.

References

[AC15]   Aasa Feragen and Christian Igel, *Data Analysis Methods 2015 - Assignment 1*, 2015, original-date: 2016-06-24T09:57:30Z.

[Ban]   Gaurav Bansal, *What are the four assumptions of linear regression? | gaurav bansal, ph.d.*

[BD93]   Boris Iglewicz and David C. Hoaglin, *Volume 16: How to detect and handle outliers*, ch. Chapter 3 - Outlier Labeling, ASQC Quality Press, 1993.

[Bro18]   Jason Brownlee, *A Gentle Introduction to k-fold Cross-Validation*, May 2018.

[Cas18]   Nikki Castle, *What is Semi-Supervised Learning?*, February 2018.

[Chr18]   Christian Igel, *Principal Component Analysis*, Tech. report, University of Copenhagen - Department of Computer Science, January 2018.

[Cla94]   K. Clark, *Nutritional guidance to soccer players for training and competition*, S43–50.

[Com13]   Olympic Programme Commission, *Rio 2016 olympic games international federations report*, September 2013.

[Cos]   Adele Coster, *Goodness of fit statistics*.

[CS16a]   ChengXiang Zhai and Sean Massung, *Text data management and analysis*, ch. Chapter 15 - Text Categorization, ACM Books and Morgan & Claypool, June 2016.

[CS16b]   _____ , *Text data management and analysis*, ch. Chapter 9 - Search Engine Evaluation, ACM Books and Morgan & Claypool, June 2016.

[Del18]   Deloitte, *Annual review of football finance 2018 | deloitte UK*, 2018.

[Doc18]   MathWorks Documentation, *Hypothesis test assumptions - MATLAB & simulink - MathWorks nordic*, 2018.

[Dot]   *Role - Dota 2 Wiki*.

[Dot18a]   *Dota Plus Statistics*, October 2018.

[Dot18b]   *Dota Plus Statistics*, October 2018.

[Dot18c]   *Heroes by release*, September 2018.

[Dot18d]   *Dota Seasonal Rank distribution and Medals - November 2018*, 2018.

[Fif18]   *Team Stats Database - FIFA 19 - FIFA Index*, 2018.

[Fox15]   John Fox, *Applied regression analysis and generalized linear models*, p. 318, SAGE Publications Inc, May 2015.

[Gle13]   Glen_B, *normalization - Why do we need to normalize data before principal component analysis (PCA)?*, September 2013.

[Gra15]   Jim Grange, *Statistics tables: Where do the numbers come from?*, 2015.

[Gru15a]   Joel Grus, *Data science from scratch*, ch. Chapter 14 - Simple Linear Regression, O'Reilly Media, April 2015.

[Gru15b]   _____ , *Data science from scratch*, ch. Chapter 16 - Logistic Regression, O'Reilly Media, April 2015.

[Gru15c]   ————, *Data science from scratch*, ch. Chapter 15 - Multiple Regression, O'Reilly Media, April 2015.

[Gru15d]   ————, *Data science from scratch*, ch. Chapter 5 - Correlation, O'Reilly Media, April 2015.

[Gui]   David Guichard, *5.1 maxima and minima*, Department of Mathematics, Whitman College.

[HEWH01]   Jan Helgerud, Lars Christian Engen, Ulrik Wislff, and Jan Hoff, *Aerobic endurance training improves soccer performance*, 7.

[Hui17]   Hui Li, *Which machine learning algorithm should I use?*, April 2017.

[Ige17a]   Christian Igel, *Linear classification*, 2017.

[Ige17b]   ————, *Linear regression*, 2017.

[Int14]   *Multiple linear regression*, 2014.

[J13]   Dr J, *Six Types Of Analyses Every Data Scientist Should Know*, January 2013.

[Jef07]   Jeff Thorne, *Introduction to Principal Components and Factor Analysis*, Tech. report, North Carolina State University, October 2007.

[K. 97]   F. Gray, Stephen K. Gray, Philip, *Testing Market Efficiency: Evidence From The NFL Sports Betting Market*, The Journal of Finance (1997).

[Kor17]   Daniil Korbut, *Machine Learning Algorithms: Which One to Choose for Your Problem*, October 2017.

[Lee15]   Jeff Leek, *The Elements of Data Analytic Style*, Leanpub (2015), 98 (en).

[MF00]   Tim McGarry and Ian M. Franks, *On winning the penalty shoot-out in soccer*, no. 6, 401–409.

[Nic15]   Ian Nick, *Is there a relationship between qr decomposition and the normal equation solutions to linear regression?*, 2015.

[noa]   *Q. Differentiate between agglomerative and divisive clustering?*

[noa18]   *Dota plus win probability graph*, April 2018.

[Ope18]   *OpenDota API*, 2018.

[pla18]   *Dota 2 playing field*, 2018.

[R. 07]   L. Morgan, Betsy R. Wilson VanVoorhis, Carmen, *Understanding Power and Rules of Thumb for Determining Sample Sizes*, Tutorials in Quantitative Methods for Psychology (2007).

[RWH00]   Robert Tibshirani, Walther, Guenther, and Hastie, Trevor, *Estimating the number of clusters in a data set via the gap statistic*, Tech. report, Stanford University, November 2000.

[San16]   Marina Santini, *Advantages & Disadvantages of k-Means and Hierarchical clustering (Unsupervised Learning)*, Tech. report, Uppsala University, December 2016.

[SAT]   *Scatter plot for SAT Prep*.

[Sau16]   Saurav Kaushik, *An Introduction to Clustering & different methods of clustering*, November 2016.

[Sci18a]   SciPy.org, *Hierarchical clustering (scipy.cluster.hierarchy) SciPy v1.1.0 Reference Guide*, May 2018.

[Sci18b]   _____, *scipy.cluster.hierarchy.linkage SciPy v1.1.0 Reference Guide*, May 2018.

[Seb]   Sebastian Raschka, *Principal Component Analysis*.

[SS14a]   Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning: From theory to algorithms*, ch. Chapter 25 - Feature Selection and Generation, Cambridge University Press, May 2014.

[SS14b]   _____, *Understanding machine learning: From theory to algorithms*, ch. Chapter 22 - Clustering, Cambridge University Press, May 2014.

[Sta]   *Determining the optimal number of clusters: 3 must known methods*.

[STE]   STEPS, *Multiple regression*.

[Ste03]   Steve DeLoach, *Two-sample test of a hypothesis*, Tech. report, Elon University, August 2003.

[Ste14]   Stephanie, *Satterthwaite formula for degrees of freedom*, 2014.

[Tav18]   Tavish Srivastava, *Introduction to KNN, K-Nearest Neighbors : Simplified*, March 2018.

[The18]   *eSports audience size worldwide from 2012 to 2021, by type of viewers (in millions)*, 2018.

[TKP+15]   Olga Tanaseichuk, Alireza Hadj Khodabakshi, Dimitri Petrov, Jianwei Che, Tao Jiang, Andrey Santrosyan, and Yingyao Zhou, *An efficient hierarchical clustering algorithm for large datasets*, 6.

[UoSAa]   Research Methodologies and Statistics University of South Australia, *Descriptive statistics: Measures of central tendency*.

[UoSAb]   _____, *Descriptive statistics: Measures of dispersion*.

[uwe18]   *Inferential statistics*, 2018.

[Wal13]   Walker M. White, *Lecture notes in Game Analytics*, April 2013.

[Wik18]   Wikipedia, *Mikkel hansen*, 2018.

[YMH12a]   Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin, *Learning from data*, ch. Chapter 1 - The Learning Problem, March 2012.

[YMH12b]   _____, *Learning from data*, ch. Chapter 3 - The Linear Model, March 2012.

[YMH12c]   _____, *Learning from data*, ch. Chapter 4 - Overfitting, March 2012.

[YU97]   Department of Statistics & Data Science Yale University, *Multiple linear regression*, 1997.

12 Appendix

12.1 T-distribution table

Degrees of freedom	Significance level					
	20% (0.20)	10% (0.10)	5% (0.05)	2% (0.02)	1% (0.01)	0.1% (0.001)
1	3.078	6.314	12.706	31.821	63.657	636.619
2	1.886	2.920	4.303	6.965	9.925	31.598
3	1.638	2.353	3.182	4.541	5.841	12.941
4	1.533	2.132	2.776	3.747	4.604	8.610
5	1.476	2.015	2.571	3.365	4.032	6.859
6	1.440	1.943	2.447	3.143	3.707	5.959
7	1.415	1.895	2.365	2.998	3.499	5.405
8	1.397	1.860	2.306	2.896	3.355	5.041
9	1.383	1.833	2.262	2.821	3.250	4.781
10	1.372	1.812	2.228	2.764	3.169	4.587
11	1.363	1.796	2.201	2.718	3.106	4.437
12	1.356	1.782	2.179	2.681	3.055	4.318
13	1.350	1.771	2.160	2.650	3.012	4.221
14	1.345	1.761	2.145	2.624	2.977	4.140
15	1.341	1.753	2.131	2.602	2.947	4.073
16	1.337	1.746	2.120	2.583	2.921	4.015
17	1.333	1.740	2.110	2.567	2.898	3.965
18	1.330	1.734	2.101	2.552	2.878	3.922
19	1.328	1.729	2.093	2.539	2.861	3.883
20	1.325	1.725	2.086	2.528	2.845	3.850
21	1.323	1.721	2.080	2.518	2.831	3.819
22	1.321	1.717	2.074	2.508	2.819	3.792
23	1.319	1.714	2.069	2.500	2.807	3.767
24	1.318	1.711	2.064	2.492	2.797	3.745
25	1.316	1.708	2.060	2.485	2.787	3.725
26	1.315	1.706	2.056	2.479	2.779	3.707
27	1.314	1.703	2.052	2.473	2.771	3.690
28	1.313	1.701	2.048	2.467	2.763	3.674
29	1.311	1.699	2.043	2.462	2.756	3.659
30	1.310	1.697	2.042	2.457	2.750	3.646
40	1.303	1.684	2.021	2.423	2.704	3.551
60	1.296	1.671	2.000	2.390	2.660	3.460
120	1.289	1.658	1.980	2.158	2.617	3.373
∞	1.282	1.645	1.960	2.326	2.576	3.291

Figure 28: T-distribution table [Gra15].

12.2 Distribution of residuals for the regression models
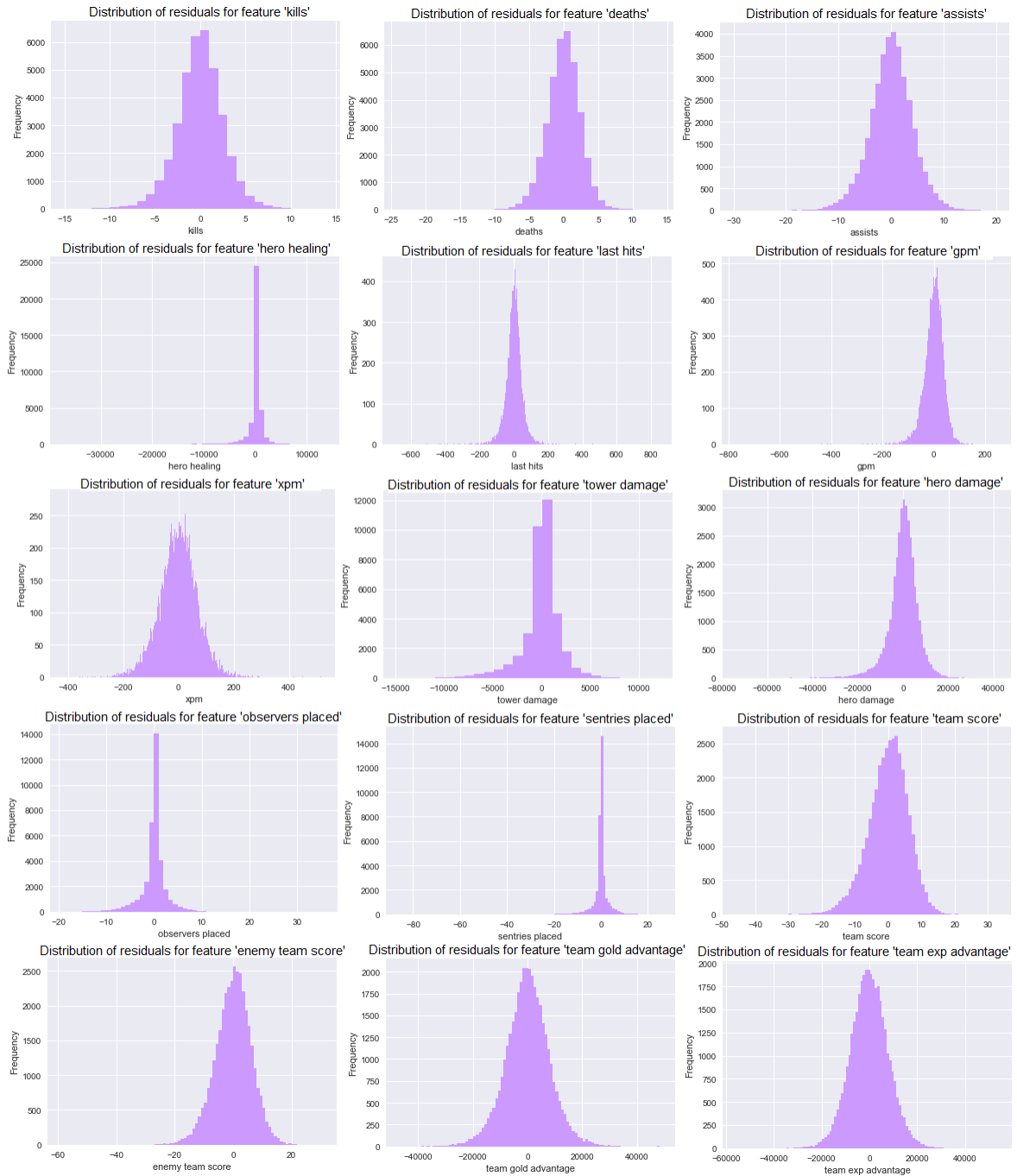

Figure 29: Distribution of residuals for the features used in linear regression.

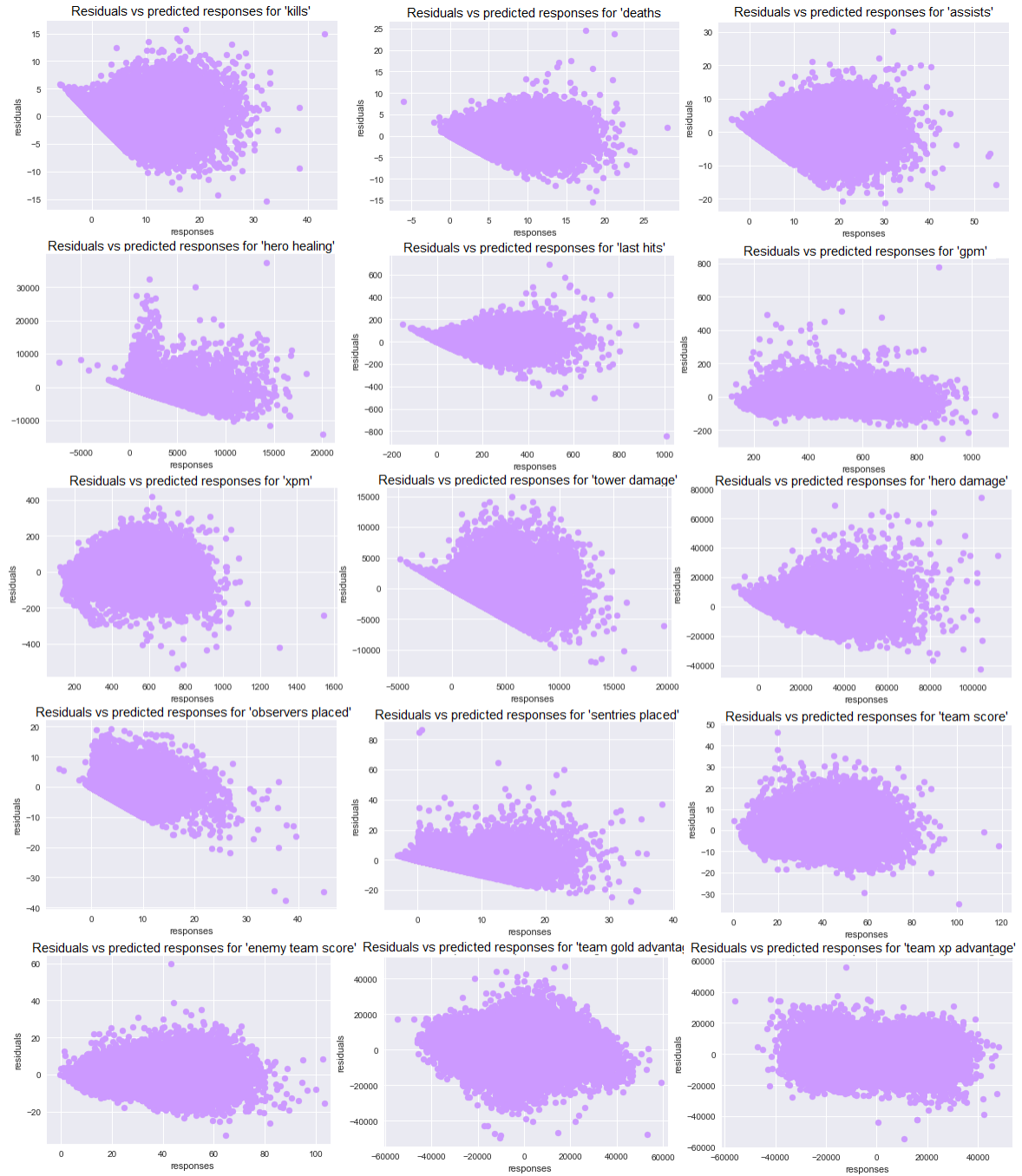## 12.3 Residuals vs the predicted responses for the regression models

Figure 30: Scatter plots of the residuals vs the predicted responses.