PROFILING SMURFS AND BOOSTERS ON DOTA 2 USING K-MEANS

DING YING JIH

UNIVERSITI TUNKU ABDUL RAHMAN

PROFILING SMURFS AND BOOSTERS ON DOTA 2 USING K-MEANS

DING YING JIH

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Hons.) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

APRIL 2021

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :

Name : Ding Ying Jih

ID No. : 1904956

Date : 15/4/2021

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"PROFILING SMURFS AND BOOSTERS ON DOTA 2 USING K-MEANS"** was prepared by **DING YING JIH** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : kckhor

Supervisor : Dr. Khor Kok Chin

Date : 15/4/2021

ACKNOWLEDGEMENTS

I have to start by thanking my project supervisor, Dr. Khor Kok Chin. From welcoming the idea of the project to transforming it into a success, he was the most integral part of this paper. Without whom, this project would not exist.

Next, I would like to express my heartfelt gratitude towards Dr. Ng Oon-Ee who supported the idea of this project and introduced me to Dr. Khor. Also, I would like to thank Dr. Hoo Meei Hao for encouraging me to propose my own title and providing guidance in my report writing.

Thanks to my mother and my girlfriend, who supported me even when they had no idea what I was doing, and my friends Chong Wei Kang, Ng Yi Sheng and Shum Yngvi for helping me to mine the data.

Last but not least, I am so grateful and honoured to have Mushi, Ohaiyo, Juan and Winter in my paper despite their busy schedule.

ABSTRACT

Dota 2 is one of the most popular Multiplayer Online Battle Arena (MOBA) game and it also holds the grandest e-Sports tournament in the world —— The International. However, the game is experiencing a continuous decline in its player count. This is because the existence of smurfs/boosters in Dota 2 is ruining the game experience for all other Dota 2 players. Hence, this project aims to identify the smurfs/boosters and analyse their skills. The data were collected from OpenDota API and a data set was created after cleaning and pre-processing. To identify the smurfs and boosters in the data set, K-Means was used to divide the players into groups. To identify the high-skill players group, feature values of the data were examined. Interquartile Range (IQR) method was then used on the high skill players group to identify and profile smurfs/boosters. The resulted profile was reviewed by two game experts and one active player. A 95% accuracy score was achieved using majority voting. It is hoped that this work can be furthered for identifying the different skill levels of the smurfs/boosters after identifying them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF SYMBOLS / ABBREVIATIONS

GPM	Gold Per Minute
XPM	XP Per Minute
HD	Hero Damage
TD	Tower Damage
HH	Hero Healing
LHPM	Last Hit Per Minute
HHPM	Hero Healing Per Minute

LIST OF EQUATIONS

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Dota 2 is one of the most popular Multiplayer Online Battle Arena (MOBA) video game, with an average of 454,594 concurrent players per month (Steamcharts.com, 2020). Moreover, the Dota 2 tournament, The International 9, holds the largest e-Sports tournament in the world with a total prize pool of US$34,330,068.

The attractive prize pool has sparked the youngsters to jump on the bandwagon of e-Sports. Many business models such as coaching services, statistics websites and replay analysis services revolving around Dota 2 have been created to help enthusiastic players improve on their gameplay.

Dota 2 is enormously complex (Franco, Henrique Fonseca Ribeiro and Comarela, 2019; Demediuk et al., 2019). Millions of lines of codes are written just to implement the game logic (Berner et al., 2019, p.2). For every match of Dota 2, ten players are involved: five players on the Radiant team versus five players on the Dire team. Each player chooses a game character, known as hero, from 119 unique heroes and play the hero for the whole duration of the match which lasts for 40 minutes on average. Then, each player chooses if the hero is played as a support role or core role. The core role is to take in-game resources and become the strongest ones in game. The support role is to make sure that the cores can achieve that. Some heroes in games that are more suitable to be played as supports while than cores. To win a match, players from the same team have to work together to destroy the other's Ancient before the opponent team does so.

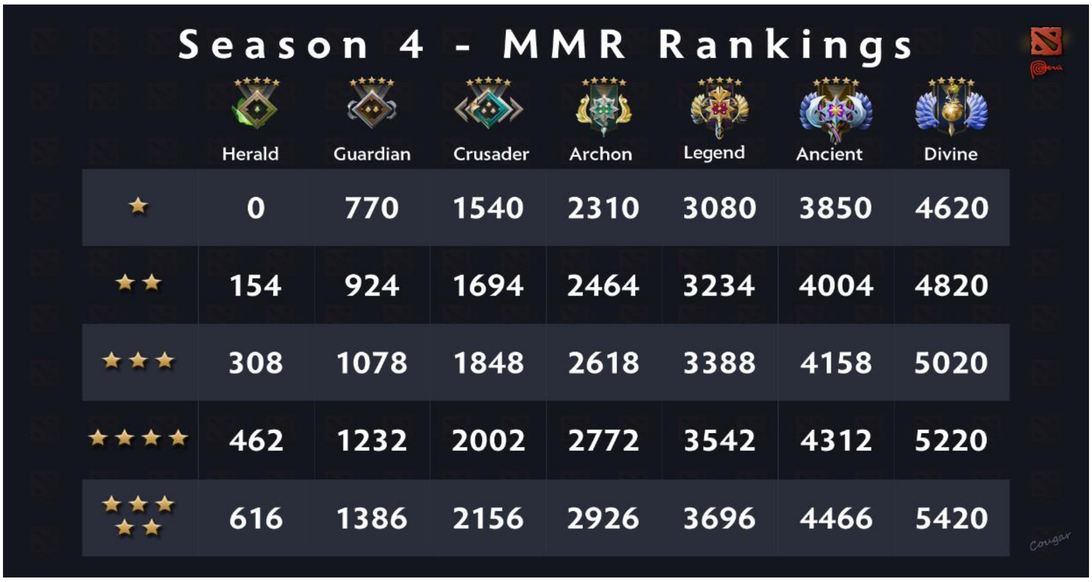
Figure 1.1 Radiant's Ancient (Left) and Dire's Ancient (Right)

Dota 2 is complex because it requires not only mechanical skills (reactions time, mouse click precision, etc.) but also a series of in-game decision-making. The examples of in-game decision making are tabulated in Table 1.

Table 1 Example of decision making required in a Dota 2 game.

Decision-making	**Description**
Hero drafting	A good hero draft could determine the outcome of the game even before the game starts. Each hero has unique skill sets and plays styles. Hero drafting is to pick heroes that counter the opponents’ heroes and synergises well with teammates’ heroes.
Resources allocation	Gold and Experience are two of the most important resources in a Dota 2 game. Players have to choose how to spend their gold and how they spend their skill points.
Farm or Fight	Each team has to constantly evaluate if they should earn more resources to become stronger or they are strong enough to start a fight with the enemy. If a team misses a good timing, the enemy team will take advantage of it and win the game.
Target Priority	Each team has to decide whom they should target and whom they shouldn’t target before a fight starts.

Dota 2 is a zero-sum game, whereby every player in the winning team earns X Matchmaking Rating (MMR). The value of X depends on whether the player is joining a match alone or joining a match with friends. The MMR decides which medal (tier) a player possesses. The higher the MMR of a player, the higher medal the player possesses.



Season 4 - MMR Rankings

	Herald	Guardian	Crusader	Archon	Legend	Ancient	Divine
★	0	770	1540	2310	3080	3850	4620
★★	154	924	1694	2464	3234	4004	4820
★★★	308	1078	1848	2618	3388	4158	5020
★★★★	462	1232	2002	2772	3542	4312	5220
★★★★★	616	1386	2156	2926	3696	4466	5420

Figure 1.2 Illustration of MMR required for each medal excluding Immortal medal (CougarDota, 2019)

In Figure 1.2, we can see that there are seven medals and a maximum of five stars for each medal. When a player gains 154 MMR, he gains a star. When he has five stars for his medal, he advances to the next level of medal. A Herald player with one star is addressed as Herald I, a Herald player with two stars is addressed as Herald II and so on.

The MMR of a player determines the players he companion and against with. The system will automatically assign players with similar MMR to play together in a match.

Despite the rising trend of e-Sports and the popularity of Dota 2, the number of average concurrent players experienced a consecutive drop for five years from 2016 to 2021(Steamcharts.com, 2021). This is because smurfs and boosters are ruining the game experience for the majority of the players.

This project strives to categorise outliers, smurfs/boosters, who play very differently and have considerable differences in statistics, i.e., kill/death/assist (KDA) score and win rate with players at the same skill level.

Smurfs and boosters are skilful players who camouflage themselves among lower MMR players. An analogy to illustrate this would be an NBA star player who plays against a group of fifth-grade students. The extreme imbalance of skill levels has caused the matches very one-sided and unenjoyable. The ordinary players will be mercilessly bullied by the smurfs/boosters during matches. There are different reasons behind their camouflage. Smurfs intend to have the fun of bullying others, and boosters intend to receive payment by winning games.

It is hoped that by the end of this research, the outliers (smurfs and boosters) can be identified accurately, and the findings can be implemented in automating the identification of the smurfs and boosters in the vast player pool. The Knowledge Discovery in Databases (KDD) approach shall assist in achieving the research aim.

1.2 Problem Background

As Salen and Zimmerman (2003) pointed out, one of the core principles of game conflict is fairness. When a player searches for a game to play, by default, the matchmaking algorithm of Dota 2 shall match a player with others who possesses the same skill level. This is to make sure both teams have equal chances of winning the game. As player’s experience determines an online game’s success (Sai and Maguluri, 2017; Korhonen, 2016), fair matchmaking ensures players’ enjoyment and the game’s lifespan.

There are too many smurfs and boosters in the game. According to Yan and Choi (2002, p.3), online cheating is an action that creates unfair advantages. By this definition, smurfing and boosting are considered cheating. Unbalanced matchmaking ruins player’s experience because games often end in a heavily one-sided manner and other players cannot change the outcome of a game in any way.

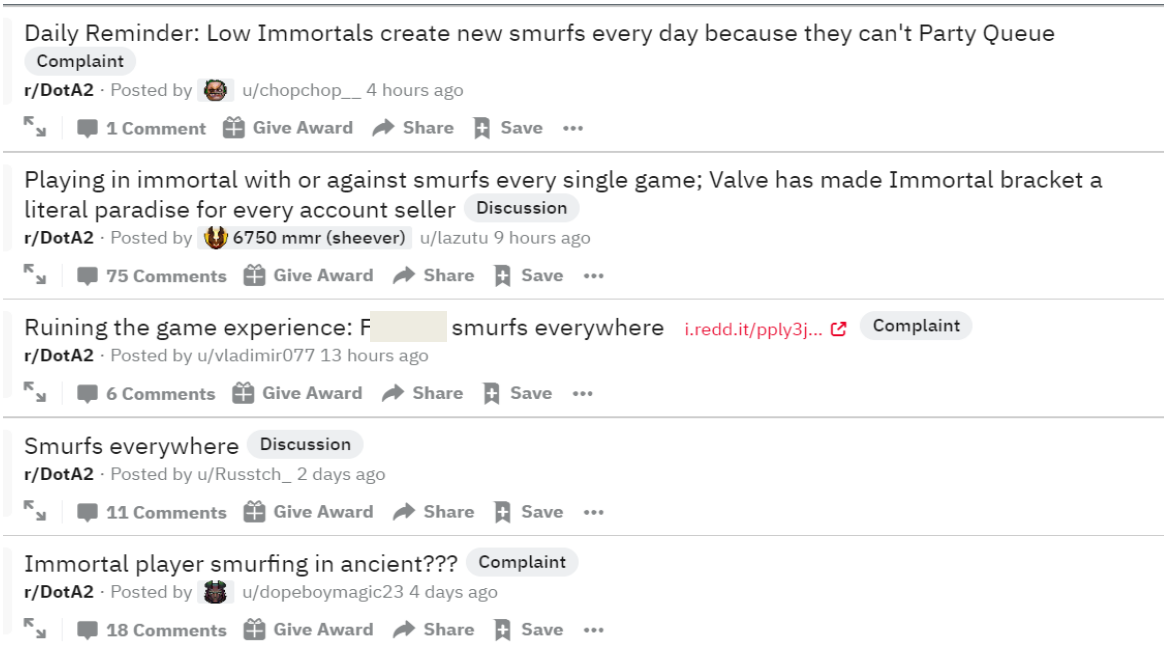


Figure 1.3 Smurfing issues voiced out in r/Dota2, a popular Dota 2 Community.

1.3 Problem Statement

1.3.1 The lack of smurf and booster indicator in the current statistics portals

The current statistics portals only provide raw statistics and features of players and matches but do not have any indicators to let the viewers know whether the player is a real smurf/booster or an ordinary player. The viewers often have to make judgement based on their perceptions, and the judgement made may be uninformed. Some ordinary players may be reported as smurfs/boosters, while the real smurfs/boosters are roaming freely.

1.3.2 The lack of a data science module to automatically detect smurfs and boosters

The current method of detecting smurfs/boosters is through community reports. The method is ineffective and inefficient as not all players would report the smurfs/boosters, and not all of them will report accurately. Moreover, some players' data are blocked from public access meaning that only the game developers can justify whether a player is a smurf/booster or just an ordinary player. The lack of an automated way of detecting smurfs/boosters in the game client will just leave the actual smurfs/boosters unpunished as we public do not have data to make judgment.

1.4 Project Objectives

The objectives of this research are:

i. To group players using the K-means algorithm.
ii. To profile the resulted group for identifying smurfs and boosters.

1.5 Research Questions

The questions that this research is trying to solve is:

i. What are the groups created by the K-means algorithm?
ii. How to profile smurfs/boosters using statistics?

1.6 Research Approach

In relation to the research questions identified, Knowledge Discovery in Databases (KDD) is used as this research approach. KDD aims to extract knowledge from the databases after going through a series of processes.

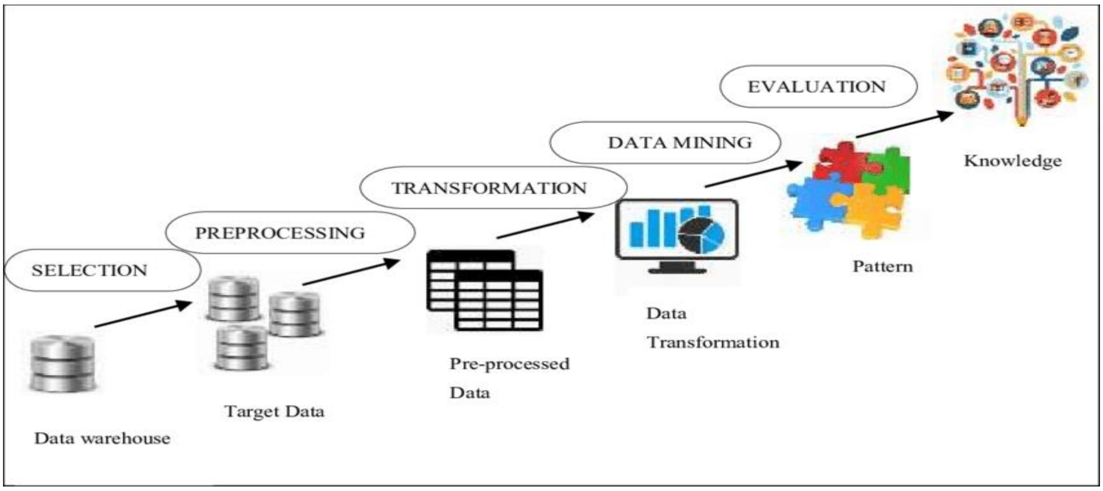


Figure 1.4 A graphical overview of the KDD process (Ahmad Sabri et al., 2019)

There are six steps involved in the process to find the answers to the research questions:

i. A data set is created by fetching necessary and relevant player data and match data from OpenDota API. Other relevant sources that provide Dota 2 related data will also be looked into and stored together in a data warehouse.

ii. The data set is then cleaned and pre-processed to select features, remove noises, duplicate data and missing values so that the data in the dataset are helpful in the research analysis.

iii. After that, data transformation is performed to transform the existing features of the data set into new features that are more useful to analyse a player's skill level.

iv. K-means clustering is then performed to cluster the data and create groups of players for profiling and analysis.

v. The resulted groups after the clustering process are then analysed statistically for the profiling of smurfs and boosters.

vi. The profile of smurfs and boosters are then sent to experts and normal player for review. Once approved, the knowledge is formed.

1.7 Scope and limitation of the Study

The research aims to collect 77,000 data (550 players x 20 Matches Data x 7 Medals) as the OpenDota API has a rate limit and a call limit for free tiers. Besides that, processing a large data set would be too taxing to the computer due to the limitation of computational resources.

Besides that, this research looks into players in 7 medals excluding the Immortal medal. This is to have a conclusive overview of smurfs in different medals. However, the Immortal medal is excluded because the MMR range in Immortal medal players is very different from the MMR range in other medal players. To illustrate, the MMR range for Herald I to Herald V is 0 – 616, while the MMR range for low Immortal to high Immortal is 5500-11000. The processing and analysing of the data for Immortal players would be very different. Hence, it is excluded from our research.

1.8 Contribution of the study

The project findings will benefit the players who are passionate towards the game and increase the game's life span. Moreover, identifying smurfs and boosters can lead to a better gaming experience as the system can automatically ban their accounts.

1.9 Outline of the report

This report contains five chapters: Introduction, Literature review, Methodology, Results and Discussion, and Conclusion and Future Work.

CHAPTER 2

Literature Review

This literature review will focus on reviewing literatures that address the ways to solve the problems stated in section 1.3. To tackle the difficulties faced, the areas below are explored and discussed:

i. Grouping players using unsupervised learning
ii. Profiling
iii. Evaluation method to review results

2.1 Unsupervised Learning and Clustering

Unsupervised learning is one of the four popular methods of how a machine learns. In unsupervised learning, the machine learns by receiving only the input data but without supervised target labels (Ghahramani, 2004). As without the need for manually labelled data, the benefits would be keeping away the risk of biased target label and can be used in more areas. It is widely used to find patterns in the provided input data (Wang, 2016; Ghahramani, 2004). Unsupervised learning is also crucial in dealing with contents that are in the form of pictures, videos and images without class labels (Greene, Cunningham and Mayer, 2008). With unsupervised learning, a machine may identify specific objects in the multimedia contents. Next, we are looking into popular unsupervised learning techniques.

Clustering is categorising data with similar features into groups (Bataineh, Naji and Saqer, 2011). Clustering has been widely used and researched in computer vision (Caron et al., 2018) to identify objects in the visual world. After the clustering process, the cluster elements are similar to each other but dissimilar to the other clusters' elements. Clustering is considered useful to construct a model to discover the natural groupings in a huge data set, which shows hidden pattern in the data (Bataineh, Naji and Saqer, 2011).

2.1.1.1 K-means Clustering

K-means clustering is the most popular partitional clustering algorithm (Greene, Cunningham and Mayer, 2008; Xu et al., 2014). The technique decomposes and groups data into k clusters, with the value of k pre-determined, based on a geometric standard (Caron et al., 2018; Greene, Cunningham and Mayer, 2008). K-means is generally used to reduce the distortion measure (Ghahramani, 2004). This is further explained by Greene, Cunningham and Mayer (2008) in their paper that discusses the K-means algorithm. The algorithm uses an iterative relocation scheme to create k clusters. K-means algorithm uses hard clustering, which means that each data or element is only assigned to one cluster. This is to reduce the distortion in a cluster between elements and representatives of a cluster, centroid. A centroid is the mean vector of all elements in a cluster. Also stated by Greene, Cunningham and Mayer (2008), the Euclidean distance is normally used to measure the distortion to minimises the sum-of-squared error (SSE) between the data elements and cluster centroids.

Greene, Cunningham and Mayer (2008) states that the very first step of the algorithm is to assign each data/element to its closest cluster centroid before updating centroid vectors to show the new assignments of cluster. The paper mentions that this process is continued until no changes in the assignment of data/elements to clusters.

Input: Data $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, the order k, MAX number of allowed iterations
Output: A partition $\mathcal{P} = \{\mathcal{C}_1, ..., \mathcal{C}_K\}$
$t = 0, \mathcal{P} = \emptyset$
Randomly initialize $\boldsymbol{\mu}_i, i = 1, ..., K$
loop
    $t += 1$
    Assignment Step: assign each sample $\mathbf{x}_j$ to the cluster with the nearest representative
    $\mathcal{C}_i^{(t)} = \{\mathbf{x}_j : d(\mathbf{x}_j, \boldsymbol{\mu}_i) \leq d(\mathbf{x}_j, \boldsymbol{\mu}_h) \text{ for all } h = 1, \ldots, K\}$
    Update Step: update the representatives
    $\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|\mathcal{C}_i^{(t)}|} \sum_{\mathbf{x}_j \in \mathcal{C}_i} \mathbf{x}_j$
    Update the partition with the modified clusters:
    $\mathcal{P}^t = \{\mathcal{C}_1^{(t)}, ..., \mathcal{C}_K^{(t)}\}$
    if $t \geq$ MAX OR $\mathcal{P}^t = \mathcal{P}^{t-1}$ **then**
        return $\mathcal{P}^t$
    end if
end loop

Algorithm 1 Pseudocode for K-means Clustering (Drakos, 2020)

To further explain the pseudo-code above (Algorithm 1), the input of the algorithm would be the data points in the cleaned and pre-processed data set, the pre-determined number of clusters **k** that we want to create and the maximum number of iterations which acts as a stop function for the algorithm.

The output of the algorithm would be **k** partitions with **k** centroids. The first step of the algorithm is to initialise the **t** value used to count the number of iterations and an empty set ***P*** (line 1)**.** Then, the k number of centroids are determined randomly (line 2).

Inside the loop (line 3 to line 13), each data point will be assigned to the nearest clusters by using the Euclidean metric to find the distance between a data point and its nearest centroid. After that, the centroids will be updated and reallocated by using an objective function that is based on the distance and the membership value of the data point in the cluster (Syakur et al., 2018). The following step is updating the partition with the newest clusters. This loop will continue until the maximum number of iterations is reached or the updated partition is the same as the previous partition.

2.1.2 Fuzzy C-Means clustering algorithm

Fuzzy c-means algorithm (FCM), or Fuzzy ISODATA, is one of the most popular fuzzy clustering algorithms in objective function based (Bataineh, Naji and Saqer, 2011). FCM is a generalisation of the k-means algorithm, which allows one data/element to belong to more than one cluster to certain degrees as determined by probabilistic weights (Greene, Cunningham and Mayer, 2008). Like the k-means algorithm, the FCM algorithm also needs the user to specify the number of clusters, **c** beforehand.

Parameters that are used in the FCM algorithm:

i. number of clusters **c**,
ii. fuzziness exponent **m**,
iii. termination tolerance **ε**,
iv. norm-inducing matrix **A**.
v. fuzzy partition matrix **U**

According to Bataineh, Naji and Saqer (2011), the most important parameter is still the number of clusters, k. They also added that one has to assume the number of underlying clusters logically when there is no fundamental understanding of the structure of the data set. The main approaches to determine the number of clusters are the validity measure approach and the iterative insertion approach (Bataineh, Naji and Saqer, 2011).

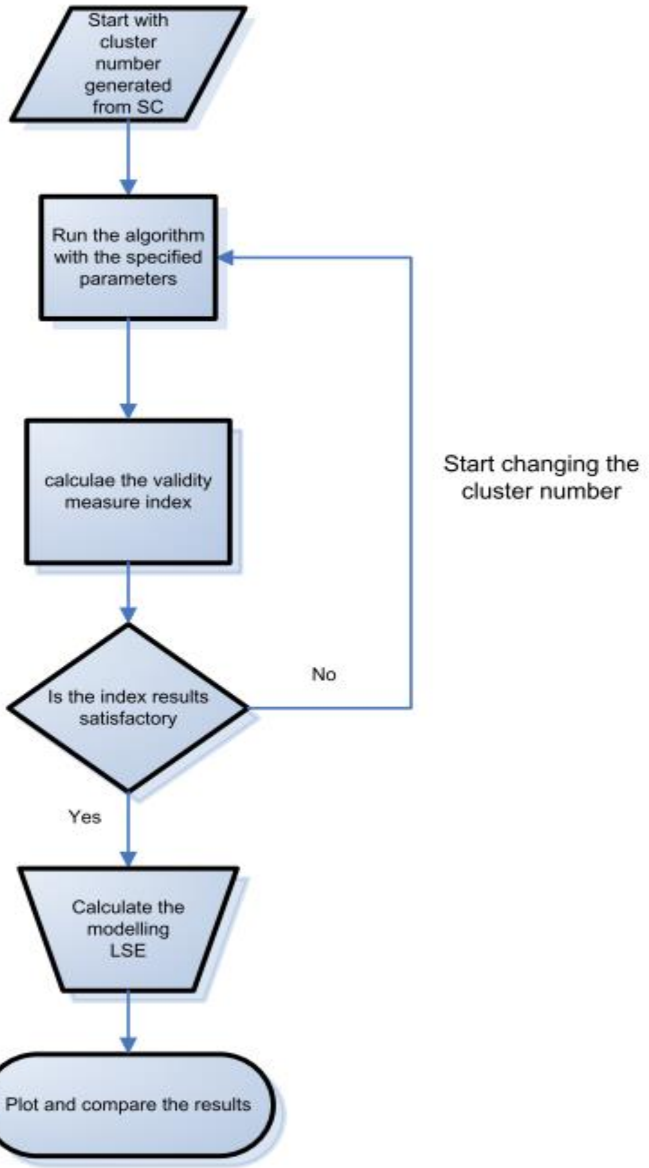


Figure 2.1 Process of Fuzzy C-Means (Bataineh, Naji and Saqer, 2011)

2.1.3 K-means++

K-means++ is first proposed to solve the issue that K-means has, which is the high sensitivity to the initial k value, by choosing the optimal centre (Xu et al., 2016). K-means++ is a simple and fast algorithm (Xu et al., 2016) which overcomes the difficulty of randomly selecting the initial cluster centre (Kapoor and Singhal, 2017) and improves considerably in the final error of k-means (Agarwal, Yadav and Singh, 2012). K-means++ achieves a lower potential value and has a faster running time compared to K-means (Agarwal, Yadav and Singh, 2012).

In the paper of Xu et al. (2016) and Agarwal, Yadav and Singh (2012), the algorithm is explained in details:

i. The first centre is constructed by choosing from the data points uniformly and randomly.
ii. The remaining data points with the probability proportional to its squared distance from the existing closest cluster centre are then chosen to be the following centres.
iii. Step 2 is repeated until a total of k centres are selected.
iv. Step 2 to 4 is repeated with the standard k-means algorithm.

However, the limitations of k-means++ are still there. The work of Xu et al. (2016) states that the algorithm becomes inefficient using enormous data size like terabytes. This is because of the large number of clusters and the data has to be split into several machines. The work of Öztürk, Cavusoglu and Zengin (2015) also proves that that k-means++ does not display good performance on large data set due to the need for iterations. Besides that, k-means++ initialisation's sequential nature when the initial centres are chosen is another issue, which means that whether a point is chosen to be a centre is very dependent on the previous centre (Xu et al., 2016).

2.1.4 Expectation Maximisation (EM)

The expectation maximisation algorithm enables parameter estimation probabilistic models with incomplete data (Do and Batzoglou, 2008; Tzoreff and Weiss, 2017). The EM algorithm is popular at providing an excellent benchmark in various machine learning areas such as natural language processing, speech recognition and image retrieval (Tzoreff and Weiss, 2017). When the suitable latent variables are chosen, the EM can effectively decouple search problems with high dimensionality into smaller subproblems of one-dimensional search, which will drastically reduce the search complexity.

The EM is explained clearly in the work of Li et al. (2019). $X = \{x1, x2,\cdots, xN\}$ is denoted as the data set which contains N number of observed samples and for each data point x_i , there is a corresponding latent variable z_i.$\{X, Z\}$ is the complete data and its approximation function is in the form of ln p $(X, Z|\theta)$, where **θ** is the set of all parameters of the model. Posterior distribution p $(Z|X, \theta)$ provides the knowledge of latent variables in Z (Li et al., 2019).

Li et al. (2019) further explain the EM algorithm in the paper. There are two steps, E step and M step, to maximise the likelihood ln p $(X, Z|\theta)$. The EM algorithm alternates the execution between step E and step M until the convergence criterion is satisfied.

2.1.5 Other work on Dota 2 regarding unsupervised learning

The comparison on other work on Dota 2 regarding unsupervised learning is tabulated in Table 2.

Table 2 Comparison on Other Work on Dota 2 Regarding Unsupervised Learning

Paper	(Looi et al., 2018)	(Drachen et al., 2015)	(Franco, Henrique Fonseca Ribeiro and Comarela, 2019)	(Demediuk et al., 2019)
Clustering Objectives	Used to improve the accuracy of logistic regression system	Find the movement behaviour of players and the factors that shape the behaviour	Develop new metrics to analyse skill level of players	Identify the role of Players in game
Algorithm	k-medoids algorithm	k-medoids and fuzzy clustering algorithms	k-means++ heuristic	k-means algorithm, Means-shift and DBSCAN
Evaluation method for number of k	silhouette method	silhouette method	Elbow method	Ensemble clustering

The work by Looi et al. (2018) is on developing an item recommendation system Dota 2 using three different systems, including a clustering system. The clustering system is used together with a logistic regression system to improve the item recommendation accuracy. The clustering method used is the k-medoids algorithm that chooses k players to be the medoids of clusters. The sum of Jaccard distances between players and their closest medoid is minimised. This showed the purchasing strategies of the players. The silhouette method is implemented together to find the number of clusters k. The average silhouette lengths and the cluster medoids for k values from 2 to 20 are calculated for each hero to choose the k value with the highest average silhouette length.

Time-series clustering is one of the methods used in the paper of Drachen et al. (2015) to determine the difference of players behaviour at different skill levels. The aims are finding matches where players display identical movement and finding the factors that lead to a certain movement pattern. Permutation Distribution (PD) is used as a distance measure to measure the complexity of a time series, where the divergence between the distributions of two time series determines the similarity. The resulting distance matrix is then applied k-medoids and fuzzy clustering algorithms. Silhouette width is then used to evaluate the clusters generated by the two algorithms.

In the paper of Franco, Henrique Fonseca Ribeiro and Comarela (2019), the k-means clustering algorithm is chosen as the unsupervised learning approach and used together with the k-means++ heuristic to create new metrics to analyse the skill levels of players. To choose the k value, the elbow method is used in the research. Notable mention from the paper of Franco, Henrique Fonseca Ribeiro and Comarela (2019) is that genetic algorithm (GA) is used together with the k-means algorithm by using the score achieved from k-means algorithm as the fitness of chromosome in GA.

On the other side, the work of Demediuk et al. (2019) is to identify the role of a player using unsupervised learning to avoid the difficulty of labelling the data manually. In their work, three unsupervised learning approaches are explored to choose the best one. The three approaches are the k-means algorithm, Means-shift and DBSCAN. Based on the explanation of Demediuk et al. (2019), these are the descriptions of the two new algorithms:

Means-shift: It is similar to k-means except that it trades scalability with the ability to find a proper number of clusters automatically.

DBSCAN: It is non-centroid based and creates clusters by choosing a random initial point. Then, datapoints in the specified range with any datapoints in the cluster will be added into that cluster.

For the k-means algorithm, ensemble clustering is used to manually combine separate classifications from various metrics performed on the same data set.

2.1.6 Evaluation and justification on unsupervised learning method

Expectation Maximisation is good at filling missing data and value and most importantly discovering the values of latent variables. Although discovering the values of latent variables is useful, this algorithm is not used in any literature regarding Dota 2 as per our knowledge. As for FCM, the ability to classify a datapoint into multiple clusters is redundant because the aim of this research is to classify a player into one accurate cluster. On the other hand, the k-means algorithm is more straightforward and can associate with other approaches such as gap statistics and the elbow method to determine the best number of clusters for profiling. The elbow method is proven in the above-mentioned literatures to be able to help in choosing the number of clusters with good accuracy. K-means++ as an improved version of the k-means algorithm has high speed and low complexity. It is able to solve the problem of k-means which does not require the pre-determined number of k value. However, both k-means and k-means++ are distance-based metrics, which means that the distance of every data point from its respective cluster centre can be obtained to analyse outliers, which are smurfs/boosters in this project. In this research, k-means along with other cluster number evaluation methods, are the preferred algorithms because the combination is proven in the literatures to produce good result.

2.2 Profiling grouped players

2.2.1 What is profiling?

Profiling is frequently used in different sectors with multiple meanings. In the sector of criminal psychology, several work are explored to give a deeper understanding of the usefulness of profiling. Kocsis (2007) describes profiling as the technique to analyse the behavioural pattern of a crime so that a descriptive template of a suspect can be modelled. This is further supported by Warikoo (2014) who says profiling may discover the suspect’s behaviour.

In the sector of gaming, profiling enables us to evaluate players in a concrete and quantifiable way so that we can understand the players behaviour and the games they play (Sifa, Drachen and Bauckhage, 2018).

2.2.2 How does profiling work?

Profiling aims to evaluate each cluster quantitatively (Rajagopal, 2011). The reason for that is provided by Cecere et al. (2010), which is to understand the quantitative value of the variables in the clusters. To further illustrate, in the work of Alawi, Shaharanee and Jamil (2017), profiling is done by analysing each cluster quantitatively using the features of gender, types of school (public/private school), place of stay (urban/rural area), age group and study performance. During the analysis, each cluster's statistics such as mean, median, mode, minimum value and maximum value are looked into to understand the cluster. A similar method using different features is shown when profiling in other work (Halim et al., 2019; Rajagopal, 2011; Cecere et al., 2010).

A more interesting and descriptive work about profiling is presented in the work of Sifa, Drachen and Bauckhage (2018). They described two approaches to perform profiling (Refer to Table 3).

Table 3 Approaches of Profiling (Sifa, Drachen and Bauckhage, 2018)

Bottom-up	Top-down
Explorative – Discover hidden pattern which is previously unknown.	**Feature Intensive** – Test defined hypothesis to prove the validity.

2.2.3 Other work on Dota 2 regarding profiling

Table 4 Comparison of Other Work on Dota 2 Regarding Profiling.

Paper	(Drachen et al., 2015)	(Demediuk et al., 2019)
Profiling Objectives	To analyse the time-series and player movement across different skill level.	Find position/role for each cluster
Features used	Player movement and time series	ability build, resource priority and map movement

In the paper of Demediuk et al. (2019), profiling is done after the ensemble clustering to find the position/role for each cluster. The main features used are ability build, resource priority and map movement while map movement is used to validate the profiling. Demediuk et al. (2019) uses map movement feature to create a label for position/role to do it in a supervised leaning way. Then, they manually analyse the effect of resource priority and skill build on the target label (position/role) to do the profiling.

In the paper of Drachen et al., (2015) , profiling is done after the clustering to find the hidden pattern lay beneath. The main features used are time series and distance between in-game heroes. The findings are that the statistics of professional players when playing the heroes are more likely to be similar compared to lower skilled players. Besides that, professional players play most of the short matches, which indicates that they are more objectives-oriented or trying to win the game.

2.2.4 Summary of profiling and implication on current research

Profiling is done after cluster analysis and usually presented at the “Results and Discussion” section to analyse each cluster quantitatively so that knowledge can be formed. Top-down is the preferred approach here because the metrics and attributes to define a player’s skill have been mentioned and introduced in the other work on Dota 2.

2.3 Evaluation method

In this section, the evaluation methods to verify and validate resulted clusters are produced. El-Mandouh et al. (2019) and Tibshirani, Walther and Hastie (2001) stated that determining the accurate number of clusters is a critical challenge in clustering. Methods and evaluation of the clustering are further discussed below.

2.3.1 Elbow Method

The elbow method finds the number of clusters by interpreting and analysing a graph with a cost function plotted against the number of clusters (Kodinariya and Makwana, 2013). Syakur et al. (2018) consider the elbow method easy because the optimal number of the clusters can be found by locating the elbow on the ideal k value graph.

Input: $iris = datasets.load\_iris(), X = iris.data\ [:, 2:]$

Output: d, k

1: $d = [];$

2: **for** $k = 1, k\ in\ rang\ (1, 9)$ **do**

3: $d = \sum_{i=1}^{k} \sum dist(x, c_i)^2;$

4: return $d, k;$

Algorithm 2 Pseudocode for Elbow Method (Yuan and Yang, 2019)

The pseudocode (Algorithm 2) takes all the features data set as the input and outputs the Sum of Squared Errors (SSE) **d** by iterating the value of k from one to ten. As mentioned, the elbow method is a visual method to determine the optimal k value. The SSE is then plotted against the number of clusters k so that the elbow can be observed from the graph.

Kodinariya and Makwana (2013) provide a detailed explanation on working with the elbow method and here we summarise the explanation provided. The elbow method works by starting with the number of clusters, k=2, and continuously increase k by 1 it in each step by 1. The cost for each cluster is calculated and plotted against k. At a certain value of k, the cost drops drastically, and plateaus out after a further increase on k value. That certain value of k is chosen as the best number of clusters. In Figure 2.2 below, the cost function drops a lot from k=1 to k=3, and then the cost function does not have a big drop after k=3. An “elbow” can be seen at k=3. Hence the best number of clusters is three.

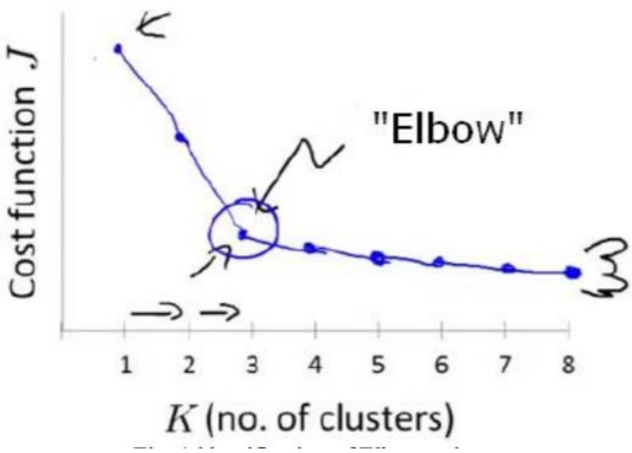


Figure 2.2 Example of elbow graph (Kodinariya and Makwana, 2013)

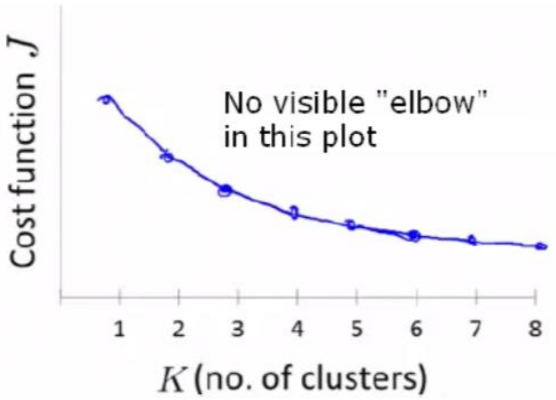


Figure 2.3 Example of elbow graph with no visible elbow (Kodinariya and Makwana, 2013)

However, there is a flaw in using the elbow method which is illustrated in Figure 2.3 above. Sometimes. There may not be a visible “elbow” formed on the graph, making it hard to determine the best number of clusters (Kodinariya and Makwana, 2013)

2.3.2 Gap Statistics

Gap statistics or “gap” method was first proposed by Tibshirani, Walther and Hastie (2001). It is one of the widely used method to finds the best cluster’s number (Mohajer, Englmeier and Schmid, 2011). Gap statistics can estimate the best number of clusters on any distance metrics and any clustering algorithm (El-Mandouh et al., 2019). Gap statistics compare the dispersion within a cluster to its expected value under a suitable null reference distribution (Tibshirani, Walther and Hastie, 2001).

A gap means the logarithmic difference between the dispersion of the original data set and the mean dispersion of reference data sets (E.P and K.A, 2016). Applying the minimum value of k will maximise the gap (El-Mandouh et al., 2019). This approach is to standardise the comparison of $logWk$ with a null reference distribution of the data (Kodinariya and Makwana, 2013). The value for which $logWk$ is the highest is the best number of k (El-Mandouh et al., 2019).

```
Input: iris = datasets.load_iris(), X = iris.data [:, 2 :]
Output: k
def SampleNum, P, MaxK, u, sigma;
SampleSet = [];
size (u) = [uM, ];
for i = 1 : uM do
    SampleSet =
[SampleSet; mvnrnd(u(i, :), sigma, fix(SampleNum/uM))];
W_k = log(CompuW_k(SampleSet, MaxK));
for b = 1 : P do
    W_kb = log(CompuW_k(RefSet(:, :, b), MaxK));
for k = 1 : MaxK, OptimusK = 1 do
    Gap_k = (1/P) Σ_{b=1}^{P} log(W*_kb);
    Gap_k <= Gap_{k−1} + s(k), OptimusK == 1;
    OptimusK = k − 1;
returrn k;
```

Figure 2.4 Sample Pseudocode for Gap Statistics (Yuan and Yang, 2019)

Gap Statistics uses reference measurements to calculate the sum of squares of the Euclidean distance between two classes (Yuan and Yang, 2019). From the research of Tibshirani, Walther and Hastie (2001), they explained the process of gap statistics:

i. Cluster the observed data.
ii. Generate referenced data sets and cluster each data set using within-dispersion measures. Then, compute the estimated gap statistic.
iii. Compute the standard deviation.

iv. Choose the smallest k value as the optimal number of clusters.

2.3.3 Summary of Evaluation methods

In evaluating the number of clusters, two methods are: the elbow method and gap statistics. The elbow method uses a visual approach by plotting cost function, which is usually Sum of Square Error (SSE), against the number of clusters and choosing the cluster that has a significant drop in cost function in the graph as the best number of clusters. In the work of Franco, Henrique Fonseca Ribeiro and Comarela (2019) and Demediuk et al. (2019), the elbow method is used together with k-means clustering to achieve their research objective due to the nature of the method being easy and effective. In the case of no visible elbow found, gap statistics will be looked into to determine the number of clusters. The gap statistics calculates the $logWk$ and the cluster that maximises the value of $logWk$ is chosen to be the best number of clusters.

CHAPTER 3

Methodology

3.1 Introduction

This section contains the summary of the workflow, the detailed workflow and the research tools used in the research

3.2 Summary of the Workflow

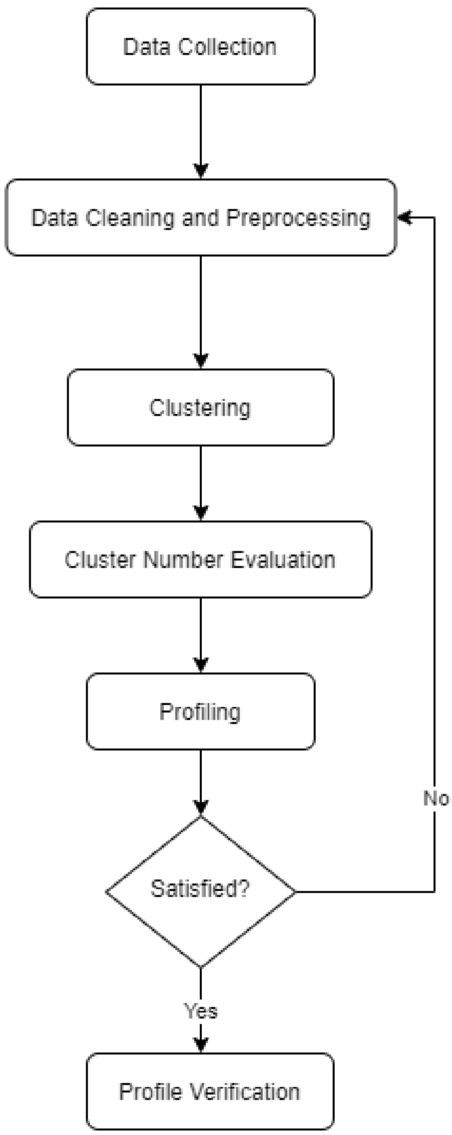


Figure 3.1 Profiling Players for Dota 2 Workflow

3.2.1 Steps for collecting player IDs.

Due to the limitation of OpenDota API, we could not directly retrieve enough player IDs. To solve the problem, we collected a few player ID from the OpenDota website and use them on the API to retrieve their past 50 matches data. Then, we looked into each match data to collect another 10 players' id. We repeated the above steps to roll out more player IDs.

However, when repeating above steps, we selected the player IDs based on their medal (ranking or skill level in the game). The same amount of player IDs from each medal was collected to ensure equal distribution. The distribution refers to the rank tier distribution from the OpenDota website (refer to Figure 3.2). We collected the IDs from Herald V, Guardian V, Crusader V, Archon V, Legend V, Ancient V and Divine V. Eventually, we managed to get 550 player IDs from each medal, thus forming a total IDs of 3850.

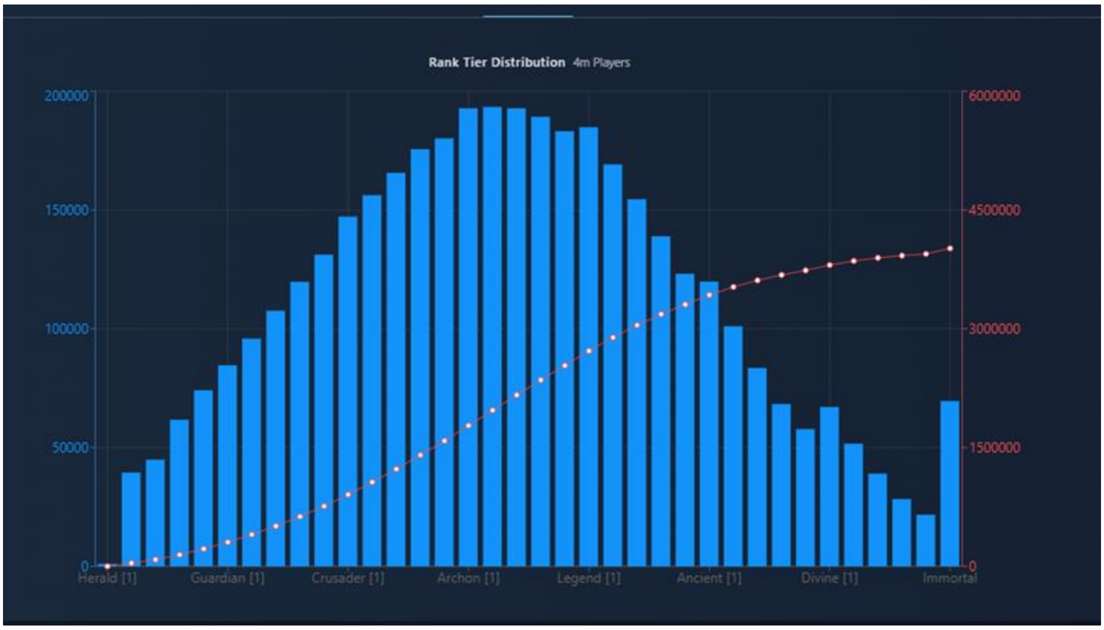


Figure 3.2 Rank Tier Distribution (OpenDota, 2021)

3.2.2 Steps for collecting match data.

With the 3850 player IDs, we used the API to retrieve their 20 matches data. It is a standard form the Dota 2 community to look into past 20 matches to determine a player's performance. However, a player might leave early in a match. Such data were removed, and we would find full match data of the player for replacement. Besides that, some players did not have 20 matches played. Therefore, some players were dropped. In total, we have 76,520 match data with 10,000 number of features.

3.2.3 Steps for collecting players’ win and loss count

The 20 matches data did not include the important information of win and loss count. To get the win and loss count, we called the API directly.

3.2.4 Dimensional reductions

We then reduced the data dimension by removing features that contained a lot of null values. This left with 68 features for the experts to verify. After the discussion with the experts, we identified 9 useful features for profiling smurfs and boosters.

3.2.5 Data cleaning

The data set was then explored to identify missing values in the remaining features. Two player’s data with missing values were dropped, thus making the current size of data 3824 players.

3.2.6 Data transformation

The features of the reduced data set were transformed using scaling and Principal Component Analysis (PCA). Scaling was conducted so that all the features fell in the same range of values. PCA was used to further reduce the dimension of the data set. After the scaling and PCA, we sorted the 76,480 match data by player ID, thus resulting a 3824 groups of players’ match data. For each group of player data, we took the average of their feature values. At the end, we had 3824 players’ match data.

3.2.7 Clustering

K-means clustering was then used to cluster this 3850 players’ match data into different groups for profiling purpose.

3.2.8 Cluster number evaluation

The gap statistics and elbow method were used to determine the best number of clusters for the k-means clustering.

3.2.9 Profiling

With the clusters formed, we profiled the clusters according to the average statistics. Then, the cluster with the high average feature values was likely to contain the smurfs/boosters. To identify the actual smurfs/booster, we then apply Interquartile Range method on that cluster.

3.2.10 Profile Verification

We randomly picked 20 smurfs/boosters and 20 normal players we identified from the previous steps. Their data were sent to two game experts and one active player for verification. To determine whether these 40 players are smurfs/boosters, the two game experts and the active player voted. Decisions were made based on majority voting. Their decisions were then compared with our research result for calculating the accuracy score.

3.3 The detailed workflow

The subsequent subsections shall explain the details of the above workflow.

3.3.1 Data Collection

For the data collection, this research used the API endpoints of OpenDota. OpenDota API was chosen after comparing it with other data sources like PandaScore API, Steam API and Kaggle Dota 2 data sets. OpenDota API was the better choice in terms of charges, documentation and features of data provided. It offered a free tier for pulling data but with a rate limit set at 60 API calls per minute and an API call limit of 50,000 per month. Then, Python and Jupyter notebook were used to fetch the data from the API and store it in a CSV file for further processing. Due to the limitation of the call rate, only data of 550 players were collected and they possess different medals.

For each player, their past 20 matches data were collected. Such 20-match analysis was a standard to judge the recent performance of a player.

Figure 3.3 An example of 20-matches analysis from OpenDota (2021). Screenshot by author.

Figure 3.4 Another example of 20-matches analysis from official Dota 2 client. Screenshot by author.

The same amount of player data from each skill level was collected to ensure equal data distribution. The data size we aimed to collect was 77,000 rows.

There were three steps involved in creating the data set: manually searching for data head, fetching player data, and fetching match data for each player.

3.3.1.1 Manually Searching for Player IDs

The OpenDota API did not have an API endpoint for us to collect player data with specific medals directly. Hence, we had to develop a workaround in order to collect player data for each medal. Since we were aiming to find smurfs across seven medals (Herald, Guardian, Crusader, Legend, Ancient, Divine), we had to first find a player for each medal. Then, we used the three players' ID to get into their matches history to find players with a specific rank tier (an indicator for player's medal and number of stars). It was very important to find three players from different regions so that there will be lesser repeating seeds. For consistency, only medals with five-star were collected.

First, we searched randomly across the OpenDota site to find three player who had a rank tier close to the rank tier we wanted. For example, we came across a Guardian I player and we recorded the player's ID. Due to the matchmaking system which matches players with similar rank tier in a game together, Due to how the system's matchmaking algorithm works, Guardian I player had a very high chance to play with a Herald V player which we aimed to collect for the next step. Repeating the same process, 21 players' data were collected.

3.3.1.2 Fetching Player IDs

To automate the process of fetching player data, a **findPlayersByRank** python function was written. Below are the steps and the logics in the function.

Input: **account_id**, **idList**, **n**, **medal**

i. Connect to the OpenDota API **GET /players/{account_id}/matches** endpoint using **account_id** as path parameter to get 200 matches played by players in JSON format.
ii. Save the 200 matches basic information as **matchData**.
iii. Extract the match id from each of the **matchData** and save in an array **MatchId.** There is a total of 200 match id.
iv. Connect to the OpenDota API **GET /matches/{match_id}** using the match id in the array **MatchId**.
v. Save the JSON response as **detailMatchData.**
vi. Iterate through **detailMatchData** to get all player id in the match.
vii. Append the player id to **idList** if the medal of the player id matches the **medal** and the player id is not already in **idList**. Display the appended **idList** as console output.
viii. Repeat step 4 to step 7 for 50 times with different match id in the array **MatchId** to avoid the exhaustive search.
ix. Increment **n** by 1.
x. Set the **n**-th element of **idList** as the new **account_id**.
xi. If the list has lesser than 550 elements, call **findPlayersByRank** again with new values of **account_id**, **idList** and **n** input parameters to repeat step 1 to step 10. Recursion and automation are achieved.

In the function above, we manually specify the medal through the input parameter. By theory, a better way would be to get the medal of the first player’s ID and use it to find players with the same medal. Hence, the process could be automated and faster. However, there was a time interval between getting the data head and fetching the player data. For example, when we first obtained the player data, the player’s medal could be Herald I. The next time we used the player’s ID, his rank tier could already be Herald II or Herald III and the function would collect players with Herald II or Herald III instead of Herald I. Hence, it was very crucial to specify the medal manually to ensure data consistency.

In order to collect a significant amount of data, the better way to do it was to automate the process as much as we could so that it was faster and required lesser supervision. However, there were multiple issues during the collection process that needed manual handling. The examples of issues happened were handshake operation timeout, call limit reached, connection error and getting blocked by the API provided. Most of the issues were not able to be handled using exception handling in the function except call limit reached issue which was handled using **time.sleep()** to pause the function for a while. To tackle the issue, we often had to manually restart the function using the latest **idList** printed in Step 7 and tweaked **account_id** and **n** as the new input parameters.

3.3.1.3 Fetching Match Data for Each Player

With the 3850 players’ ID, this step used them to get their matches to analyse their performance. For the collection of match data, a function **findMatchByPlayers** was written as a Python function by using the idList generated previously.

Input: **idList**, **df, n**

i. Connect to the OpenDota API **GET /players/{account_id}/matches?limit=20&lobby_type=7** endpoint using n-th player id in **idList** as path parameter and limit and lobby_type as query parameter to get 20 matches with ranked game mode played by player in JSON format.

ii. If there are 20 match data for the player account, save the 20 matches basic information as **match**.

iii. Connect to the OpenDota API **GET /matches/{match_id}** endpoint using match id in **match** as path parameter to get details of match data.

iv. Save each match data details as **matchDetails.**

v. Transform **matchDetails** to a data frame row and append it to **df** data frame.

vi. Repeat step 3 to step 5 using another match id in **match** for 19 times.

vii. Save the **df** as a CSV file.

viii. Increment **n** by 1. Display **n**.

ix. Repeat step 1 to step 8 using another player id in **idList** for 549 times.

The output of the function will be a list of comma-separated values (CSV) files saved locally. Using the CSV files and the **n** counter**,** any issues and errors occurred during the data collecting which interrupted it could be amended easily. We could just use the **n** displayed in step 8 to set it as a new **n** input parameter to the function and the function would continue to get data from where it left off. All the previous data were already saved. Therefore, we wouldn't need to restart the whole collection in the event of an issue or error occurred. After all match data for each player had been collected, they were all combined to seven CSV files grouped by medals. After a complete run of this function, we discovered that we no longer had 550 player match data for each medal. We analysed the issue and found that some players did not pass the checking in step 2. It was weird at first because by default, a player had to play at least 20 ranked matches to have their medals. Since we had data of their medals, it meant that they must already had played for 20 ranked matches. In the end, we deduced that the most possible cause was that the players changed their privacy setting to block public access to their data. Hence, there were some minor differences in the player count for each medal.

After the collection, we checked that every ID had 20 matches so that it would be consistent for our analysis.

```
#check every account has 20 match data
heraldDF['account_id'].value_counts().loc[lambda x: x!=20].index
```

```
Int64Index([], dtype='int64')
```

Figure 3.5 Missing match data in Herald data set

```
#check every account has 20 match data
guardianDF['account_id'].value_counts().loc[lambda x: x!=20].index
```

```
Int64Index([], dtype='int64')
```

Figure 3.6 Missing match data in Guardian data set

```
#check every account has 20 match data
crusaderDF['account_id'].value_counts().loc[lambda x: x!=20].index
```

```
Int64Index([], dtype='int64')
```

Figure 3.7 Missing match data in Crusader data set

```
#check every account has 20 match data
archonDF['account_id'].value_counts().loc[lambda x: x!=20].index
```

```
Int64Index([], dtype='int64')
```

Figure 3.8 Missing match data in Archon data set

```
#check every account has 20 match data
legendDF['account_id'].value_counts().loc[lambda x: x!=20].index
```

```
Int64Index([], dtype='int64')
```

Figure 3.9 Missing match data in Legend data set

```
#check every account has 20 match data
ancientDF['account_id'].value_counts().loc[lambda x: x!=20].index
```

```
Int64Index([], dtype='int64')
```

Figure 3.10 Missing match data in Ancient data set

```
#check every account has 20 match data
divineDF['account_id'].value_counts().loc[lambda x: x!=20].index
```

```
Int64Index([], dtype='int64')
```

Figure 3.11 Missing match data in Divine data set

From the Figure 3.5 to Figure 3.11 above, it could be seen that empty lists were returned for each medal, indicating that there were 20 match data for each player.

3.3.1.4 Drop matches that the player left during the game

The next thing that we looked into was the **abandon** and **leaver_status** features. The **abandon** and **leaver_status** indicated whether the player left the game whether willingly or unwillingly before either Ancient fell. The difference between the two features was that the **abandon** recorded whether the player was the first to quit the game while the **leaver_status** tracked whether the player left the game and how the player left including abandon. Hence, we only needed to take care of the **leaver_status** as it already contained information about the **abandon**. We had to remove the row of match data that the **leaver_status** was not zero. In other words, we wanted to analyse matches that the player played until the very end because those games contained complete information for our skill level analysis. For example, if we did not remove them, the value for gold gained would be zero and the player would be identified as a low skill player. To compensate for dropping the rows of data that had **leaver_status** not equal to zero so that we still had a good amount of match data for the analysis, we modified the **findMatchByPlayers** function as below:

i. Receive the combined data frame as input parameter and save all the match id in the data frame as **matchID**.
ii. Find which account id has **leaver_ status** not equal to zero and save the count as **n**.
iii. Use the account id to find new match id that is not already in the **matchID** and has **leaver_ status** equal to zero.
iv. Repeat step 3 until **n.**
v. Repeat step 2 until no more account id has **leaver_ status** not equal to zero.

After that, it returned a new data set which we appended it to the combined data set. The rows in the combined data set which had **leaver_ status** not equal to zero were dropped.

3.3.1.5 Fetching Win and Loss Data for Each Player

For a better skill analysis of a player, we also created another function to get the total wins and losses for every player in **idList.** The total wins and losses could be transformed into a player's total win rate and total matches.

The python function **findWinLossByPlayers** was a much simpler function that requires lesser steps.

Input: **idList**

i. Create an empty data frame **df**.
ii. Connect to the OpenDota API **GET /players/{account_id}/wl?limit=20&lobby_type=7** endpoint using player id in **idList** as path parameter and lobby_type as query parameter to get win and losses of an account id in JSON format.
iii. Save the JSON data as **wl.**
iv. Add the current pointer in **idList** as the **wl** key.
v. Convert the **wl** to a data frame row and append it to **df**.
vi. Repeat step 2 to step 5 until every player in idList has been iterated.
vii. Save it to a CSV file.

After done running the **findWinLossByPlayers** function which only took a very short time compared to the previous functions, the CSV file was added to the aforementioned seven CSV files using the players' id key in all the CSV files as additional features. The final output for the data collection was seven CSV files with two new wins and losses features. The seven CSV files would be combined in the data preprocessing steps using python function later. All the match data that we collected were from ranked matches, which was a more challenging and popular game mode in which players gain and lose MMR every time they win or lose a game (Chen et al., 2017).

3.3.2 Data Cleaning and Pre-processing

The data cleaning and preprocessing was a crucial step to create a quality data set to be input into the clustering in the later steps. Since we collected our data for the purposes of this research, it was important for us to assess the data quality and perform cleaning on the data set so that we could perform quality analysis later. The function we wrote for data collection might have missed out something, or the connection to the OpenDota API might have lost during the data collection, resulting in duplicated or missing values in our data set.

3.3.2.1 Reducing the size of data sets

For the seven data sets that we created, each contained more than 10,000 rows and 10,000 features, which was considered huge. Loading and processing huge data sets required a lot of computational resources. We had to look into reducing their size. Since the number of rows of data was required for the significance of our analysis, we tried to look into the features in the data set. Of the match data that we collected, some of the data was parsed match data while the other was not. Parsed match data refers to matches that were parsed upon request to obtain highly detailed match data. Due to the parsed match data, some rows had up to 1,000 features. While parsed match data would be able to provide extra information to analyse skill level of a player, the extra information had to be removed from our data sets for consistency. To explain further, with only some rows of data contained value in the extra features while the other rows had null value in those extra features, we could not compare them equally.

To reduce the size by filtering parsed match features, a very simple and straightforward approach was used. In every data set, parsed matches features were dropped. With that, only around 80 features remained. To further reduce the number of features, the sequence of upgrading abilities were dropped. It was because the sequence was unique to hero used and it also required much other specific information which we did not have, to justify whether the sequence was better. Specific information meant here refers to the lane match-up, the specific in-game situation, the position/role of the player, the resources priority and etc. The number of features remained was 68.

3.3.2.2 Feature Selection

For the feature selection, we had our two game experts, Mushi (see Appendix A) and Ohaiyo (see Appendix B), involved. They responded in a survey form for the selection of the features. From the feedback they provided, we evaluated and picked the final selection for our clustering. The selection and feedback from the experts were tabulated in Table 5 along with our own evaluation.

Table 5 Expert Feedback and Final Selection of Features

Features	M	O	Selected	Reason
Kill Count		Y	N	It is correlated to **Benchmark: Kills Per Minute.**
Kills per minute		Y	N	It is correlated to **Benchmark: Kills Per Minute.**
Death Count		Y	Y	A smurf/booster may have lower death count due to having better gameplay.
Assist Count		Y	Y	A smurf/booster may have higher assist count due to involving in the deaths of many players.
KDA (Kill Death Assist)	Y	Y	Y	A very well-known metrics to measure a player's in-game performance.
Hero used by player			Y	Some heroes are preferred by smurfs and boosters because the heroes have more in-game impact.
XPM (experience per minute)	Y		N	It is correlated to **Benchmark: Experience Per Minute.**

Table 5 Expert Feedback and Final Selection of Features (Continued)

Items they have in the end			N	Item choices are unique and are on a game-by-game basis. This feature does not tell a player's skill level.
Total last hit count			N	It is correlated to **Benchmark: Last Hit Per Minute.**
Last hit per minute			N	It is correlated to **Benchmark: Last Hit Per Minute.**
Deny count			N	This feature does not tell a player's skill level.
Net Worth		Y	N	It is correlated to **Benchmark: Gold Per Minute.**
Game Duration			N	The length of a game is determined by many factors. It is not very relevant to a player's skill level.
Win rate in last 20 matches		Y	Y	A smurf/booster may have higher 20 games win rate because he/she is consistently better.
Tower Damage		Y	N	It is correlated to **Benchmark: Tower Damage Per Minute.**
Hero Damage		Y	N	It is correlated to **Benchmark: Gold Per Minute.**

Table 5 Expert Feedback and Final Selection of Features (Continued)

Hero Healing			N	Hero healing is related to the hero that a player chooses. It does not tell a player's skill level.
Player's Total Match Count			N	It does not tell a player's skill level.
Player's Overall Win rate	Y	Y	Y	A high win rate means the player is consistently better in all the games he/she plays.
Benchmark: GPM (Gold Per Minute)	Y		Y	A high GPM percentile indicates a good farming skill of a player.
Benchmark: XPM (XP Per Minute)	Y		Y	A high XPM percentile indicates a good farming skill of a player.
Benchmark: Kills Per Minute			Y	A high kills per minute percentile indicates a player's high capability to dominate other players.
Benchmark: Hero Healing Per Minute			N	Hero healing is related to the hero that a player chooses. It does not tell a player's skill level.
Benchmark: Last Hit Per Minute			Y	A high last hit per minute percentile indicates a player is able to earn gold and experience faster.
Benchmark: Hero Damage Per Minute		Y	Y	A high hero damage per minute percentile indicates a player contributes a lot to kill opponents.
Benchmark: Total tower damage done			Y	A high tower damage percentile indicates a player focuses taking objectives to win the game.

Note. **M** means that Mushi selected the feature to be put into consideration. **O** means that Ohaiyo selected the feature to be put into consideration.

All the features with Y in column **Selected** were tried out in the clustering section to find the best smurf/booster cluster. We used the benchmark features instead of raw features because benchmark told us directly how well a player performs compared to all other players using the same hero. This removed the hero factor that existed in the raw features. For example, some heroes have unique abilities that allow them to have either more gold per minute, more hero damage or more tower damage. If we used raw features, players who played those heroes would get categorised as smurfs/boosters instead of players who were way better in terms of gameplay skills. Besides that, most benchmark features considered the match duration as well, which nullified the impact of game length on the features. Hence, benchmark features were used to replace most of the raw features.

The features in the final data set and their descriptions were tabulated in Table 6. Note that not all the features in the table below were used in the clustering. Extra features were mentioned in the table for better clarification on their correlative benchmark features.

Table 6 Features in the final data set.

Feature	**Description**
account_id	A series of number that uniquely identifies a player.
deaths	The number of times a player dies during a game.
assists	The number of times a player deals damage but not the final blow to an enemy player which leads to the enemy player's eventual death.

Table 6 Features in the final data set (Continued)

gold_per_min	The amount of gold earned per minute. Gold can be earned by killing creeps, enemy players and enemy towers. Gold can be used to buy items that will increase the chance of winning.
hero_damage	The amount of damage dealt to enemy players during a game.
smurf_hero	Heroes that are difficult to play well and stronger in the right hands.
kills	The number of times a player deals the last blow on an enemy player.
last_hit	The number of creeps that a player kills during a game.
tower_damage	The number of damage that a player deals to enemy tower during a game.
xp_per_min	The amount of experience that a player gains during a game. Experience can be earned by killing creeps, enemy players and enemy towers. Experience are required to level up a player’s hero so that the player’s hero is stronger.
winrate_20matches	The percentage of games won by a player in 20 matches

Table 6 Features in the final data set (Continued)

kda	A common metric used in Dota 2 to identify a player's skill level. The formula is $(Kills + Assist)/Deaths$.
kills_per_min	The number of kills that a player gets per minute.
rank_tier	The medal of a player. Higher number indicates a higher ranking.
benchmarks_gold_per_min_pct	The gold per minute of the player's hero in the form of percentile. A value of 0.90 means the player's hero earns more gold per minute in that game than 90% of all players using the same hero.
benchmarks_xp_per_min_pct	The experience per minute of the player's hero in the form of percentile. A value of 0.90 means the player's hero earns more experience per minute in that game than 90% of all players using the same hero.
benchmarks_kills_per_min_pct	The kills per minute of the player's hero in the form of percentile. A value of 0.90 means the player's hero earns more kills per minute in that game than 90% of all players using the same hero.
benchmarks_last_hits_per_min_pct	The last hit per minute of the player's hero in the form of percentile. A value of 0.90 means the player's hero earns more last hits per minute in that game than 90% of all players using the same hero.
benchmarks_hero_damage_per_min_pct	The hero damage dealt per minute of the player's hero in the form of percentile. A value of 0.90 means the player's hero deals more hero damage per minute in that game than 90% of all players using the same hero.

Table 6 Features in the final data set (Continued).

benchmarks_tower_damage_pct	The tower damage dealt per minute of the player’s hero in the form of percentile. A value of 0.90 means the player’s hero deals more tower damage per minute in that game than 90% of all players using the same hero.
total_matches	The total number of matches that the player has played using the account.
total_winrate	The total number of won matches over the total number of matches that the player has played using the account.

The nine features in Table 9 below were the features used in the clustering as the combination was proven to create the best clusters.

Table 7 Nine features selected for clustering

Features	**Type**
winrate_20matches	Float
total_winrate	Float
kda	Integer
benchmarks_gold_per_min_pct	Float
benchmarks_xp_per_min_pct	Float
benchmarks_kills_per_min_pct	Float
benchmarks_last_hits_per_min_pct	Float
benchmarks_hero_damage_per_min_pct	Float
benchmarks_tower_damage_pct	Float

3.3.2.3 Check for Duplicate Rows

Duplicate records would make our analysis inaccurate because they could misinterpret the actual skill of a player. Since the **match_id** feature in our data frame was not unique, we had to use both **match_id** and **account_id** features to find duplicate rows. It was because one player could only appear once in a match.

```
completeDF[completeDF.duplicated(subset=['match_id','account_id'], keep=False)].sort_values(by=['match_id'])
```

	match_id	player_slot	account_id	assists	backpack_0	backpack_1	backpack_2	deaths	denies	gold	...	benchmarks_hero_healing_per_min_raw	benchmarks_hero_healing_pe
75484	5688997676	3	851623143	7	0.0	244.0	38.0	10	1	968.0	...	0.000000	
76466	5688997676	3	851623143	7	0.0	244.0	38.0	10	1	968.0	...	0.000000	
76474	5715946759	4	148436006	15	0.0	38.0	0.0	4	13	703.0	...	0.000000	
68805	5715946759	4	148436006	15	0.0	38.0	0.0	4	13	703.0	...	0.000000	
76480	5723207156	132	210683302	14	0.0	0.0	0.0	6	1	1728.0	...	10.425717	
...	...	...	...	...	...	...	...	...	...	...	...	...	
71551	5788607062	129	217310065	20	0.0	0.0	0.0	3	3	1508.0	...	0.000000	
76359	5790128509	1	1062800212	12	0.0	0.0	0.0	13	19	2273.0	...	0.000000	
66478	5790128509	1	1062800212	12	0.0	0.0	0.0	13	19	2273.0	...	0.000000	
76514	5790239580	1	100945263	20	73.0	0.0	0.0	8	9	3419.0	...	0.000000	
65834	5790239580	1	100945263	20	73.0	0.0	0.0	8	9	3419.0	...	0.000000	

112 rows × 70 columns

Figure 3.12 Duplicated rows found in data set

Unfortunately, there were some mistakes in the data collection process which resulted in 56 duplicated rows as shown in Figure 3.12 To amend this, the 56 duplicated rows were dropped.

```
completeDF[completeDF.duplicated(subset=['match_id','account_id'], keep=False)].sort_values(by=['match_id'])
```

match_id	player_slot	account_id	assists	backpack_0	backpack_1	backpack_2	deaths	denies	gold	...	benchmarks_hero_healing_per_min_raw	benchmarks_hero_healing_per_min_pct

0 rows × 70 columns

Figure 3.13 Verify the duplicated rows had been dropped.

We tried to ensure 20 match data for each player but dropping the duplicated rows reduced some of the match data for some players. To make sure that there were still enough match data for each player to analyse their skill level, we checked for the least number of match data for a player in the data set.

```
: completeDF['account_id'].value_counts()

: 201869303     20
  357413539     20
  1112593359    20
  178295687     20
  75537286      20
                ..
  142418243     17
  200838365     17
  419018997     16
  196427746     15
  166486744     15
  Name: account_id, Length: 3826, dtype: int64
```

Figure 3.14 Least match data count for players.

In Figure 3.14, the least number of match data count is 15, which was considered acceptable since we were going to perform group by function to obtain the mean.

3.3.2.4 Handle Missing Value

Missing values in the data set were looked into so that every player could be accurately analysed. Upon checking the null values on each feature in the data set. It appeared that four features had missing values in them. They were tabulated in Table 8.

Table 8 Missing Values and their count

Features	Missing value count
Tower_damage	4
Hero_damage	4
Benchmarks_tower_damage_pct	4
Kills_per_min	2204

A deeper analysis into this issue revealed that there were four rows which had missing values in **tower_damage**, **hero_damage** and **benchmarks_tower_damager_pct** . The four rows were from two different players. Since there were only two players with missing data, we decided to drop the two players from the data set. To verify the features were free of missing values, verifications were done as shown in the figures below.

```
completeDF[completeDF['hero_damage'].isnull()]
```

account_id	deaths	assists	gold_per_min	hero_damage	hero_id	kills	last_hits	tower_damage	xp_per_min	...

0 rows × 23 columns

Figure 3.15 **Hero_damage** with no missing value

```
completeDF[completeDF['tower_damage'].isnull()]
```

account_id	deaths	assists	gold_per_min	hero_damage	hero_id	kills	last_hits	tower_damage	xp_per_min	...

0 rows × 23 columns

Figure 3.16 **Tower_damage** with no missing value

```
completeDF[completeDF['benchmarks_tower_damage_pct'].isnull()]
```

account_id	deaths	assists	gold_per_min	hero_damage	hero_id	kills	last_hits	tower_damage	xp_per_min	...

0 rows × 23 columns

Figure 3.17 **Benchmarks_tower_damage_pct** with no missing value

For **kills_per_min,** it appeared that the number of missing values was quite large. After an explorative analysis into the issue, it was found that the missing value was caused by the number of kills being zero.

```
#confirm that the null in kpm is caused by 0 kills
nullKill = completeDF[['kills', 'kills_per_min']]
nullKill =nullKill[nullKill.isnull().any(axis=1)]
nullKill[nullKill['kills']!=0]
```

kills	kills_per_min

Figure 3.18 Confirming the zero kills caused the missing value

The Figure 3.18 was to confirm that in every row with missing **kills_per_min** value, the **kills** value was zero. There were no cases where the **kills_per_min** value was missing, and the **kills** value was not zero. This was probably due to the

kills_per_min value was created from $\boldsymbol{kills/match\_duration}$ and zero divided by any number returned empty value instead of zero. To solve this, zero value was assigned to replace all the missing values in **kills_per_min** feature**.**

After all the processing and cleaning of players, the player count for each medal was looked into and tabulated into Table 9.

Table 9 Player count for each medal

Medal	Player Count
Herald	546
Guardian	548
Crusader	545
Archon	548
Legend	545
Ancient	550
Divine	542

From Table 9, it can be seen that the distribution of players for each medal is pretty even. This was to ensure that the clustering in the later section would not categorise player based on the medal itself instead of outliers in each medal.

3.3.2.5 Feature Transformation

In the expert's survey, we also collected experts' opinion on which heroes were the smurfs' heroes. Smurf's heroes refer to heroes who are harder to use and have a higher chance of winning a game. The smurfs' heroes according to the experts were Lycan, Broodmother, Lone Druid, Morphling, Meepo, Huskar, Arc Warden, Visage, Templar Assassin and Riki. For our feature transformation, we transformed the **Hero_id** feature to **smurf_hero** to record whether a player played a smurf's hero.
Besides that, **total_winrate** was created by using the Formula 1.

$$\frac{total\ games\ won}{total\ games\ won\ +\ total\ games\ lost}$$

Equation 1 Formula for **total_winrate**

3.3.2.6 Group player data

In this step, we combined all rows of data that were played by the same player and obtained the mean value. This was done so that every player's skill could be described by their 20 matches data. Each row of data contained the skill level data of a player. We could then use the data set for further analysis and clustering on the players' skill level. There was a total of 3824 rows of players in our dataset after the grouping.

3.3.2.7 Features Scaling

Features scaling was necessary to be performed on our data set because of the difference in scales for our features. For example, **kda**, **deaths, assists and kills** features had values ranged from zero to more than ten while benchmark features ranged from zero to one. If scaling was not done, raw features would carry greater weight and dwarf the impact of the benchmark features on the clustering results. It was especially important to scale the features when we were going to use distance - based clustering method K-means clustering. Hence, the normalisation and standardisation for our data were both looked into in this preprocessing step.

Normalisation is scaling the values so that they all range from zero to one. We used MixMaxScaler to achieve normalisation on our data.

Standardisation is scaling the value so that the values have a mean of zero and a standard deviation of zero. We used StandardScaler to achieve standardisation on our data.

The scaling technique that we chose in the end is Normalisation as it produced better clusters in our clustering.

3.3.2.8 Principal Component Analysis

Principal Component Analysis (PCA) is a technique to reduce the dimensionality of our data set while retaining as much information as possible. PCA was considered in our preprocessing step as our data set had a large dimension. The purpose of reducing the number of features was to enable us to explore and visualise the data easily. Besides that, the reduction of dimensionality enabled a faster clustering process.

For choosing the number of PCA components, explained variance graph was used to determine the number of components.

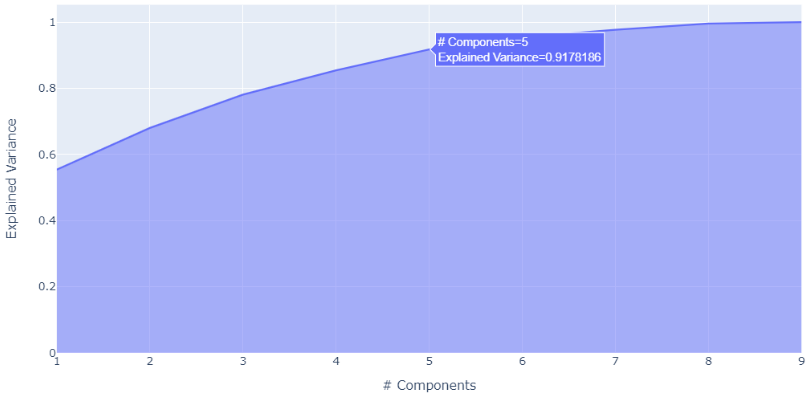


Figure 3.19 Explained variance for each number of components

Using the graph, we determined the number of components that had above 80% explained variance and with less than six components. The reasons for that were to lower the computational resources required for clustering while retaining most of the information from our data set. In Figure 3.19, we chose five components as they had a 90% explained variance and an acceptable number of components.

3.3.3 Clustering

K-means clustering was used to cluster smurfs and boosters. K-means was proved to be able to produce good results. Furthermore, we were able to obtain the distance from every data point to its respective cluster centre to further identify the outliers.

3.3.4 Cluster number evaluation

K-means clustering requires a predefined number of clusters. To compute the optimal number of clusters $\boldsymbol{k}$, we used both the elbow method and gap statistics. For the elbow method, we chose the cluster number that forms an "elbow" shape on the elbow graph using an inertia value, which is the average value of the squared distances of the data points from their respective clusters.

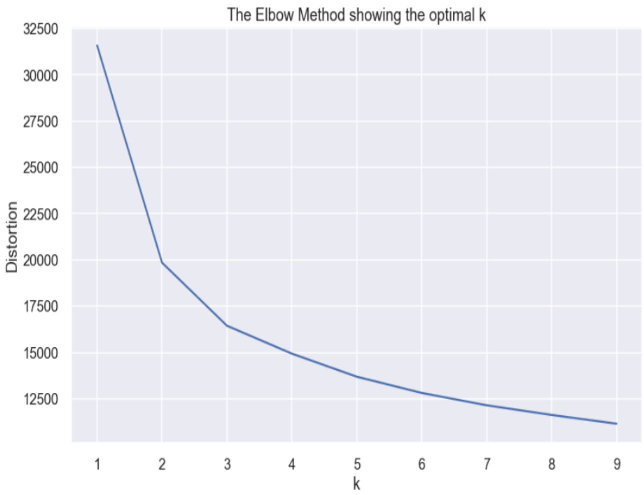


Figure 3.20 The elbow graph using inertia.

It can be seen from Figure 3.20 that an elbow formed vaguely at ***k* = 3.** To further verify ***k*,** gap statistics were used. For gap statistics, we plotted a graph to show the gap values for each cluster count.

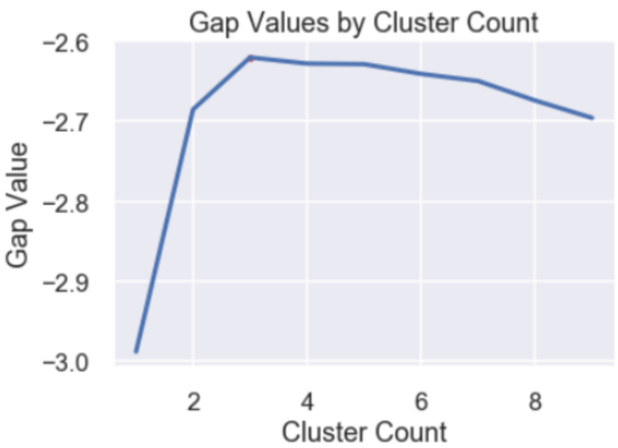


Figure 3.21 Gap values computed for each cluster count.

From Figure 3.21, it can be seen that ***k* = 3** had the highest gap value, which made the optimal ***k*** to be three.

Since both cluster number evaluation methods determined **k = 3**, we chose $\boldsymbol{k = 3}$ to be the optimal number of clusters for the k-means clustering.

3.3.5 Profiling

3.3.5.1 Profiling Steps

Using top-down profiling, we labelled the resulted clusters as based on the skills level by examining the statistical values of each features in every cluster. In matches where every player has an equivalent skill ranking, the players who exhibit exceptionally high skill for 20 matches are most probably smurfs/boosters. Therefore, applying the popular outlier detection method, the IQR method, on the distance values, we could get the data points that were relatively far away from their respective cluster than the other data points. The IQR method, or more precisely the 1.5 × Interquartile Range method, calculates the difference between the third quartile Q_3 and the first quartile Q_1 , then multiplies the difference by 1.5 to get the 1.5 × IQR value, x . Any values that fall below $Q_1 - x$ or above $Q_3 + x$ are labelled as outliers.

IQR method was applied on the high-skill players' profile to capture the players with exceptional high statistical values. A new profile was created and labelled as smurfs/boosters.

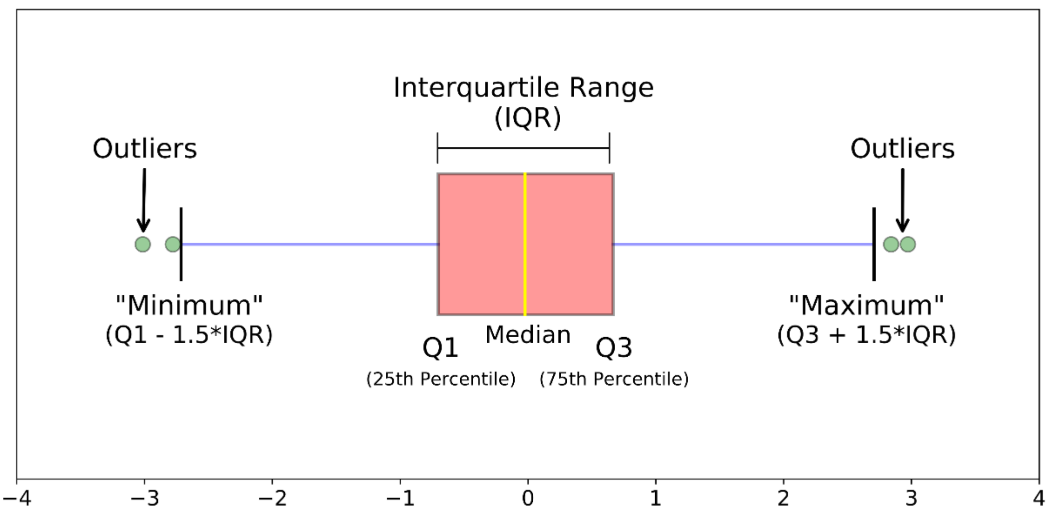


Figure 3.22 Illustration of IQR method (Galarnyk, 2018)

3.3.5.2 Profile Verification by Expert & Player

To verify the profiling results comprehensively and intensively, one professional Dota 2 player Ohaiyo, one professional Dota 2 coach/analyst Winter (see Appendix C), and one normal Dota 2 player Mr. Loh were invited to participate in the profile review. The profile review was done using questionnaire along with an additional reference document. In the questionnaire, there were 40 players to be reviewed. 40 players were chosen so that the result would have a statistical significance while not overwhelming the reviewers. The 40 players consisted of 20 players randomly picked from smurfs/boosters' profile and 20 players randomly picked from all the non-smurfs/boosters' profile. The players from smurfs/boosters' profile were labelled as one and the non-smurfs/boosters' profile were labelled as zero.

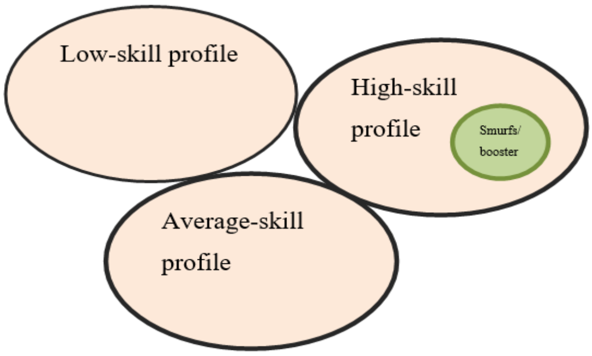


Figure 3.23 Illustration of the selection of the 40 players.

For the questionnaire (see Appendix D), the average statistics of players based on 20 matches were presented and the options "Smurf/Booster" and "Normal Player" were presented to the reviewers to choose either one. However, not all of the average statistics that we displayed in the questionnaire were the ones from the features used in the clustering. Instead, the correlatives of the benchmark features were displayed in the questionnaire for better clarification because the reviewers were more familiar with the raw features as raw features are presented in the Dota 2 game client (see Appendix E) and the benchmark features might cause unnecessary confusion. Table 10 explains the features included in the questionnaire.

Table 10 Explanation for features included in questionnaire

Features used in the questionnaire	Reason(s)
Medal	To let the reviewers know what skill level to expect from the players based on their medal ranking.
Total Matches played	To let reviewers compare it with Medal. Normally a player with low matches played but high medal ranking is smurfs/boosters.

Table 11 Explanation for features included in questionnaire

Overall win rate	One of the features used to cluster.
Win rate in 20 matches	One of the features used to cluster.
Average kill count	Raw feature of benchmark kills per minute.
Average death count	One of the most popular in-game raw features.
Average KDA	One of the features used to cluster.
Average GPM	Raw feature of benchmark GPM.
Average XPM	Raw feature of benchmark XPM.

Benchmark LHPM, benchmark HD and benchmark TD were omitted due to the space limitation and they could be more or less represented by average GPM, average kill count and win rate in 20 matches respectively.

To achieve that, the clustering results were mapped to the original data set without any dropping of columns.

An additional reference document (see Appendix F) was made for the reviewers to make their judgements. The additional reference documents contained screenshots of each of the 40 players' profiles in both OpenDota and Dota 2 game client. The profiles in Opendota contained recent performance of the players including the average statistics, the maximum statistics and the heroes used. On the other hand, the profiles in the Dota 2 game client contained information of players that could not be retrieved from any other places. The information was the rampage information, aegis snatches information and all-time most successful heroes.

Table 12 Information in Dota 2 game client and their description

Information	Description
Rampage	The rampage is a very rare in-game occurrence where a player gets five consecutive kills in a short time frame. The number of recent rampages shown may indicate the player's skill level. A higher number indicates that the player is likely a smurf/booster.

Table 12 Information in Dota 2 game client and their description (Continued)

Aegis Snatch	The aegis snatch is a very rare in-game occurrence where a player steals an important objective from the opponents. The number of recent aegis snatches shown may indicate the player's skill level. A higher number indicates that the player is likely a smurf/booster.
All-time most successful heroes	This information reveals the win rate and win streak of their all-time most successful heroes. The win rate, win streak and whether the player plays smurf heroes may indicate the player's skill level. A higher number of win rate, a higher number of win streak or the player's most successful heroes are among the smurf heroes indicates that the player is likely a smurf/booster.

However, it was stated explicitly that in case of conflict, the statistics shown in the questionnaire should carry a greater weight towards their decision-making as there was a time gap between the data that we used in the analysis and the data shown in the additional reference document.

After collecting their responses, majority voting was applied. The answers with two out of three approvals were accepted. Then, the answers were compared to the labels generated. A confusion matrix was created, and the accuracy score was computed using $(true\ positive\ +\ true\ negative)\ /\ 40$.

3.4 Research Tool Used

3.4.1 Jupyter notebook + Python

Python had many libraries that were useful for conducting this research. It allowed us to use created and defined functions and saved us time from writing functions from scratch. Among the libraries that we used were matplotlib for data visualisation, Scikit for pre-processing and clustering, and pandas for presenting and handling the data set. Python allowed us to streamline the data collection processes and to convert the JSON data to a proper data set. All other processes used in our methodology were also able to be done easily using Python.

Jupyter notebook was selected to run the python code because it allows cell-by-cell code running. It provides instant output based on a snippet of code, which we enjoyed very much when experiments were done repeatedly. Last but not least, our code had high-readability and our output was easily documented for reference.

3.4.2 OpenDota API

OpenDota API was used to collect the data needed to create the data set needed for our research. It was the better choice in terms of charges, documentation and attributes of data provided. Besides that, the community was provided through discord channel where questions could reach the developers。

3.4.3 Questionnaire

The questionnaires were sent during both feature selection phase and profile review phase to collect their feedback. A questionnaire was used to allow the experts to have control over the time to respond. Although face-to-face interview might have been able to brief them properly and extract more insights, their tight tournaments schedule did not allow the option. To compensate that, reference documents were made in addition to the questionnaires for both feature selection and profile review briefing to ensure they had a sufficient understanding of the instructions.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Clustering Result

This section presents the clustering result after using the k-means algorithm to create three clusters. Due to space limitations, only the scatter plot with principal component I against principal component II, principal component III, principal component IV and principal component V are shown.

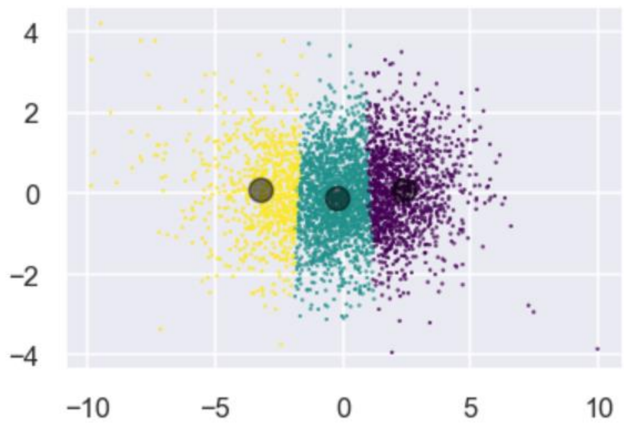


Figure 4.1 The clustering result of principal component I plotted against principal component II

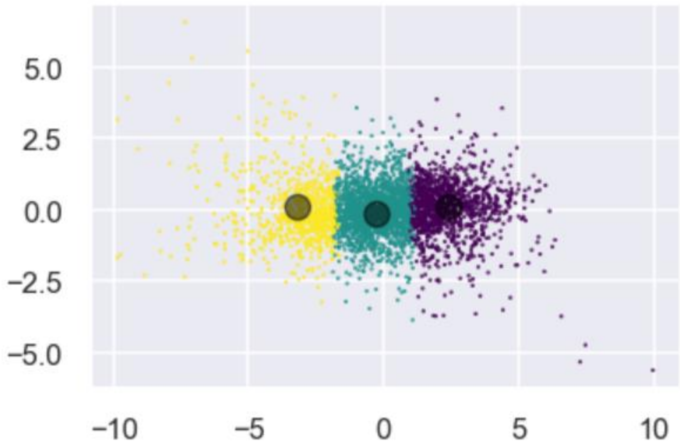


Figure 4.2 The clustering result of principal component I plotted against principal component III

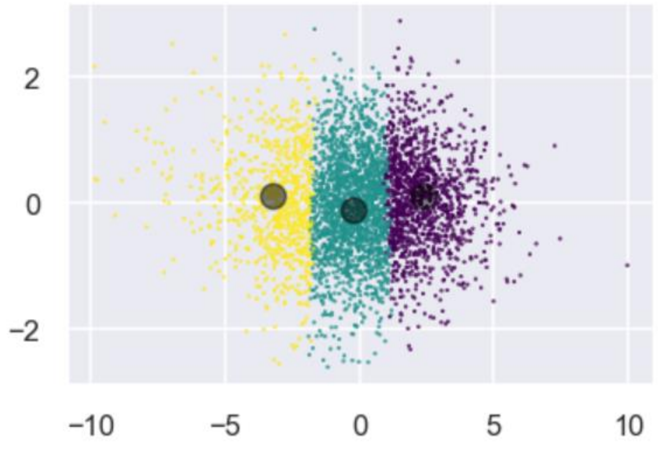


Figure 4.3 The clustering result of principal component I plotted against principal component IV

Figure 4.4 The clustering result of principal component I plotted against principal component V

The principal component I was chosen to be plotted against other principal components as the first principal component contained the higher number of explained variance. Hence, the plot would be more meaningful to readers.

The cluster centres (black dots) aligned horizontally and the three clusters were formed on the left, right and middle.

Table 13 The average statistics for each cluster based on 20 matches

Cluster Number	Win rate in 20 matches	Benchmarks (percentile)							Total win rate
		KDA	GPM	XPM	KPM	LHPM	HDPM	TD	
1	0.43461	2.26382	0.38399	0.40849	0.40791	0.37513	0.40266	0.45823	0.49993
2	0.51092	3.00137	0.52105	0.52204	0.50720	0.52446	0.51965	0.53139	0.50659
3	0.61502	4.37626	0.66062	0.63931	0.61892	0.65554	0.63198	0.61177	0.52434

Table 14 The player count for each cluster

Cluster Number	Player Count
1	1238
2	1816
3	770

A deeper look into the three clusters gave us the results tabulated in Table 13. The table shows that the first cluster had the lowest values in all of the features used. The second cluster had the ordinary values while the third cluster had the highest values in all features. Hence, the first cluster was labelled as low-skill players, the second cluster was labelled as average-skill players and the third cluster was labelled as high-skill players.

However, it was unclear that the high-skill players were smurfs/boosters as they might be just normal players who were learning and improving. The average statistics for the high-skill players' profile were not absurdly high as the difference between the average statistics of low-skill players' profile and the average statistics of average-skill players' profile matched that of the difference between the average statistics of average-skill players' profile and the average statistics of high-skill players' profile. Moreover, the high player count of high-skill players' profile further suggested that the high-skill profile was not equivalent to smurfs/boosters' profile. It was because the high player count, or approximately 20% of the sample size, was illogical. It meant that in every match of Dota 2 there would be two smurfs/boosters in the match. To further investigate it, a kernel density plot was created to show the distribution of the distance between each data point in the high-skill players' profile and the cluster centre.

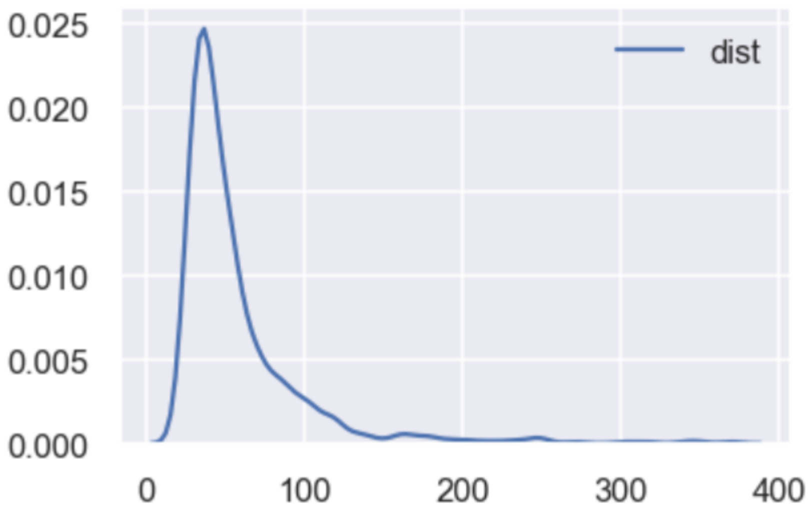


Figure 4.5 Kernel density plot of the distance between data point to the cluster centre.

From Figure 4.5, we were able to observe the distribution of the distances in the high-skill profile. The data points were cluttered around 0 to 150 squared distance while the few other data points had squared distances scattered around 150 to 400 square distance. Running a deeper check, the value of the median obtained was 44.55 squared distance while the value of the mean was 58.31. The analysis showed that there were players who did not belong to the profile like the other players in the same cluster. There were the outliers of the profile and most likely to be the smurfs/boosters. To find the cut-off point to identify outlier, the IQR method was applied.

4.2 IQR Result

After applying IQR method, a new cluster with 60 data points was formed.

Table 15 The average statistics of the cluster formed after IQR method.

Cluster Number	Win rate in 20 matches		Benchmarks (percentile)						Total win rate
		KDA	GPM	XPM	KPM	LHPM	HDPM	TD	
4	0.71963	6.36083	0.80076	0.76703	0.75950	0.77584	0.75594	0.69497	0.57739

The values in each of the features were high. To compare it side by side with other clusters, a table with all the clusters were made.

Table 16 The average statistics for each cluster including new cluster based on 20 matches.

Cluster Number	Win rate in 20 matches		Benchmarks (percentile)						Total win rate
		KDA	GPM	XPM	KPM	LHPM	HDPM	TD	
1	0.43461	2.26382	0.38399	0.40849	0.40791	0.37513	0.40266	0.45823	0.49993
2	0.51092	3.00137	0.52105	0.52204	0.50720	0.52446	0.51965	0.53139	0.50659
3	0.61502	4.37626	0.66062	0.63931	0.61892	0.65554	0.63198	0.61177	0.52434
4	0.71963	6.36083	0.80076	0.76703	0.75950	0.77584	0.75594	0.69497	0.57739

From Table 16, it can be observed that the four clusters had a similar gap in feature values between them. The newly created cluster had the highest values in all features. The distribution of the smurfs was acceptable with 0.015 smurfs/boosters (three smurfs/boosters in 20 matches). The new cluster was visualised in the plots below.

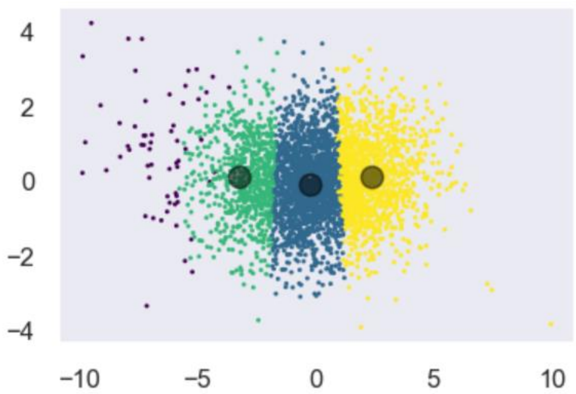


Figure 4.6 The clustering result of principal component I plotted against principal component II (with new cluster added).

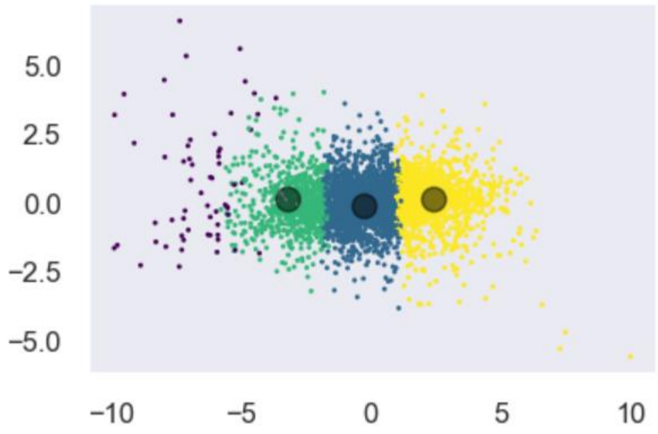


Figure 4.7 The clustering result of principal component I plotted against principal component III (with new cluster added).

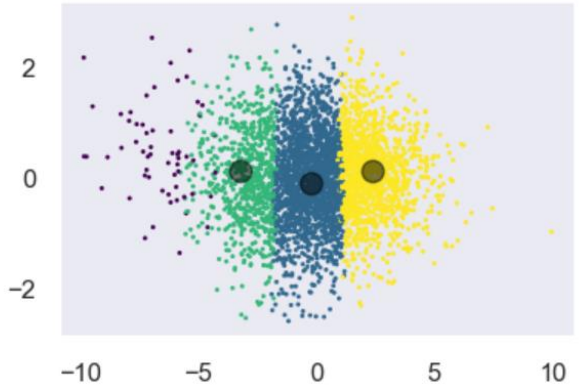


Figure 4.8 The clustering result of principal component I plotted against principal component IV (with new cluster added).

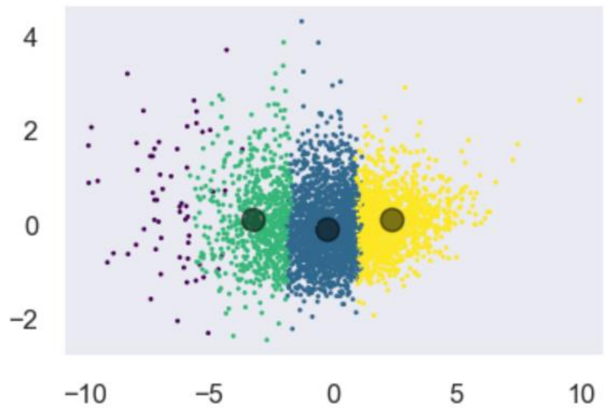


Figure 4.9 The clustering result of principal component I plotted against principal component V (with new cluster added).

From the figures above, it can be seen that the data points in the fourth cluster were far away from the third cluster (high-skill players' profile) centre. Hence, the outliers in the high skill players' profile were captured and labelled as smurfs/boosters' profile.

4.3 Profile Verification Result

A questionnaire with 20 smurfs/boosters and 20 normal players randomly picked were sent to the professional Dota 2 player Ohaiyo, the professional Dota 2 analyst/coach Winter and a normal player Mr. Loh. Their feedback was collected (see Appendix G) for further analysis.

Using majority voting, the feedback from three viewers was combined into one. Then, it was compared to our research result to generate the confusion matrix in Table 17.

Table 17 The confusion matrix generated based on the majority voting on the profile

	Research Result	
Expert Result	Normal Player	Smurf/Booster
Normal Player	20	2
Smurf	0	18

From Table 17, the accuracy score was calculated by getting the matching expert result and research result. With 38 correct matches out of 40, the accuracy score achieved was 95%. While the accuracy score was very promising, a deeper analysis was done to analyse the two mismatches. It appeared that the two smurfs/boosters identified by our research were identified as normal players.

Table 18 Features of the mismatches compared to the mean of all other correctly identified smurfs/boosters.

Player	Medal	Total Matches	Overall win rate	Winrate in 20 Matches	Average Kills	Average Deaths	Average KDA Ratio	Average GPM	Average XPM
1	Archon	379.0	56.20	55.0	15.25	7.25	5.65	650.30	764.9
2	Crusader	1775.0	50.54	55.0	8.40	6.35	2.75	478.10	687.30
Others	**-**	**802.05**	**59.20**	**73.88**	**14.10**	**4.47**	**7.11**	**658.93**	**767.87**

From Table 18, the statistics of both players could be compared to the statistics of other confirmed smurfs/boosters to check the variance.

Player 1 was Archon with 379 matches played. With the lower matches count, Player 1 was able to get a decent medal. Even so, it was still not exceptional enough to judge. The overall win rate and win rate in 20 matches were below the mean values. However, the average number of kills of Player 1 was very high, with an average of 15 kills per game. The average number of deaths and the average KDA ratio were worse than the mean values while the average GPM and XPM were pretty close to the mean values. In conclusion, whether Player 1 was a smurf/booster remained unclear. The player had some smurfs/boosters' statistics and some average player's statistics. It could be Player 1 was a smurf/booster but not as good as other smurfs/boosters.

Player 2 was Crusader with 1775 matches played. It was reasonable for a player to achieve the third medal after a considerable amount of match played. For all the other features, the statistics of Player 2 were considerably worse than the mean values. To conclude, experts were right about the player being a normal player and our research misjudged the player.

To further into the reasons Player 2 was included in the smurfs/boosters' profile, the player's features that were used in the clustering was looked into.

Table 19 Statistics of Player 2 that were used in the clustering.

Win rate in 20 matches		Benchmarks (percentile)						Total win rate	Distance
	KDA	GPM	XPM	KPM	LHPM	HDPM	TD		
0.55	2.75	0.83377	0.8265	0.7480	0.8843	0.7571	0.6636	0.5053	115

From Table 19, the values of the benchmark features were very high. The distance from its cluster centre was closer to the median of the high-skill players' profile distance distribution than the other smurfs/boosters. Another check on all 20 matches and the features (see Appendix H) revealed that the player played 17 out of 20 matches of Winter Wyvern, a hero typically used as a support hero. The Player 2 played it as a core hero, resulted in the high benchmark values. Hence, the Player 2 was wrongly labelled as smurf/booster by our methods.

All in all, the good accuracy score showed that our research can profile smurfs/boosters accurately using K-means.

CHAPTER 5

Conclusion and Future Work

The project achieves the objectives of grouping players using K-means and profiling the resulted groups for identifying the smurfs/boosters. With 95% accuracy score using majority voting on the feedback provided by one normal player and two domain experts Ohaiyo and Winter, it proves that the methodology used is effective. We also learned that the smurfs/boosters' distribution is approximately 0.015 or 3 out of 20 games. However, we are inclined to believe the actual distribution of smurfs/boosters is larger because smurfs/boosters tend to block third-party access to their data. Hence, only some smurfs/boosters' data were included, making the distribution inaccurate.

` To further improve on identifying smurfs/boosters, features selection has to be done more carefully so that the players who play support heroes as core would not be categorised as smurfs/boosters.

The methodology can be implemented in the official Dota 2 game to automatically ban the smurfs/boosters automatically. In the future, both the data collection technique and the data set we created can be used in further research. The skills level of the verified smurfs/boosters can be furthered to differentiate the skill level between smurfs/boosters.

REFERENCES

Agarwal, S., Yadav, S. and Singh, K., 2012. K-means versus K-means ++ Clustering Technique. (February 2010), pp.379–384.

Ahmad Sabri, I.A., Man, M., Abu Bakar, W.A.W. and Mohd Rose, A.N., 2019. Web Data Extraction Approach for Deep Web using WEIDJ. *Procedia Computer Science*, 163(July), pp.417–426.

Alawi, S.J.S., Shaharanee, I.N.M. and Jamil, J.M., 2017. Profiling Oman education data using data mining approach. *AIP Conference Proceedings*, 1891(October).

Bataineh, K.M., Naji, M. and Saqer, M., 2011. A comparison study between various fuzzy clustering algorithms. *Jordan Journal of Mechanical and Industrial Engineering*, 5(4), pp.335–343.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P.P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pondé, H., Pinto, D.O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F. and Zhang, S., 2019. Dota 2 with Large Scale Deep Reinforcement Learning

Caron, M., Bojanowski, P., Joulin, A. and Douze, M., 2018. Deep clustering for unsupervised learning of visual features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11218 LNCS, pp.139–156.

Cecere, W., Abreu, D., Mccarthy, J. and Jacob, T., 2010. A Method for Improving List Building: Cluster Profiling. [online] (Cml), pp.1–10. Available at: <papers://c0f50874-1acc-4627-879e-caecd2247efe/Paper/p899>.

Chen, Z., Sun, Y., El-nasr, M.S. and Nguyen, T.-H.D., 2017. Player Skill Decomposition in Multiplayer Online Battle Arenas. [online] Available at: <http://arxiv.org/abs/1702.06253>.

CougarDota, 2019. *r/DotA2 - Season 4 Matchmaking Medal Rankings, by Dota 2 Wiki - Last revised on October 11, 2019*. [online] reddit. Available at: <https://www.reddit.com/r/DotA2/comments/dgnwxx/season_4_matchmaking_medal_rankings_by_dota_2/> [Accessed 25 Mar. 2021].

Demediuk, S., York, P., Drachen, A., Walker, J.A. and Block, F., 2019. Role Identification for Accurate Analysis in Dota 2. *Aiide*, pp.130–138.

Do, C.B. and Batzoglou, S., 2008. What is the expectation maximization algorithm? *Nature Biotechnology*, 26(8), pp.897–899.

Drachen, A., Yancey, M., Maguire, J., Chu, D., Wang, I.Y., Mahlmann, T., Schubert, M. and Klabajan, D., 2015. Skill-based differences in spatio-Temporal team behaviour in defence of the Ancients 2 (DotA 2). *Conference Proceedings - 2014 IEEE Games, Media, Entertainment Conference, IEEE GEM 2014*, 2(DotA 2).

Drakos, G., 2020. *Implementation of K-means from scratch in Python (9 lines)*. [online] GDCoder. Available at: <https://gdcoder.com/implementation-of-k-means-from-scratch-in-python-9-lines/>.

E.P, S. and K.A, A.N., 2016. A Hybrid K-Harmonic Means with ABC Clustering Algorithm Using an Optimal K Value for High Performance Clustering. *International Journal on Cybernetics & Informatics*, 5(2), pp.51–59.

El-Mandouh, A.M., Mahmoud, H.A., Abd-Elmegid, L.A. and Haggag, M.H., 2019. Optimized K-means clustering model based on gap statistic. *International Journal of Advanced Computer Science and Applications*, 10(1), pp.183–188.

Franco, G., Henrique Fonseca Ribeiro, M. and Comarela, G., 2019. Towards an interpretable metric for DOTA 2 players: An unsupervised learning approach. *Proceedings - 2019 Brazilian Conference on Intelligent Systems, BRACIS 2019*, pp.341–346.

Galarnyk, M., 2018. *Understanding Boxplots*. [online] Medium. Available at: <https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51>.

Ghahramani, Z., 2004. LNAI 3176 - Unsupervised Learning. *Machine Learning*, pp.72–112.

Greene, D., Cunningham, P. and Mayer, R., 2008. *Unsupervised learning and clustering*. *Lecture Notes in Applied and Computational Mechanics*, .

Halim, Z., Atif, M., Rashid, A. and Edwin, C.A., 2019. Profiling Players Using Real-World Datasets: Clustering the Data and Correlating the Results with the Big-Five Personality Traits. *IEEE Transactions on Affective Computing*, 10(4), pp.568–584.

Kapoor, A. and Singhal, A., 2017. A comparative study of K-Means, K-Means++ and Fuzzy C-Means clustering algorithms. *3rd IEEE International Conference on* , pp.1–6.

Kocsis, R.N., 2007. Skills and accuracy in criminal profiling. *Criminal Profiling: International Theory, Research, and Practice*, 47(2003), pp.365–382.

Kodinariya, T.M. and Makwana, P.R., 2013. Review on determining number of Cluster in K-Means Clustering. *International Journal of Advance Research in Computer Science and Management Studies*, 1(6), pp.2321–7782.

Korhonen, H., 2016. *Hannu Korhonen Evaluating Playability of Mobile Games with the Expert Review Method Hannu Korhonen Evaluating Playability of Mobile Games with the Expert Review Method*.

Li, X., Zhong, Z., Wu, J., Yang, Y., Lin, Z. and Liu, H., 2019. Expectation-maximization attention networks for semantic segmentation. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob, pp.9166–9175.

Liquidpedia, 2021a. *Mushi*. [online] Liquipedia Dota 2 Wiki. Available at: <https://liquipedia.net/dota2/Mushi> [Accessed 13 Apr. 2021].

Liquidpedia, 2021c. *WinteR*. [online] Liquipedia Dota 2 Wiki. Available at: <https://liquipedia.net/dota2/WinteR> [Accessed 13 Apr. 2021].

Liquidpedia, 2021b. *Ohaiyo*. [online] Liquipedia Dota 2 Wiki. Available at: <https://liquipedia.net/dota2/Ohaiyo> [Accessed 13 Apr. 2021].

Looi, W., Dhaliwal, M., Alhajj, R. and Rokne, J., 2018. Recommender System for Items in Dota 2. *IEEE Transactions on Computational Intelligence and AI in Games*, 11(4), pp.396–404.

Mohajer, M., Englmeier, K.-H. and Schmid, V.J., 2011. A comparison of Gap statistic definitions with and without logarithm function. [online] (096). Available at: <http://arxiv.org/abs/1103.4767>.

OpenDota, 2021. *OpenDota - Dota 2 Statistics*. [online] Opendota.com. Available at: <https://www.opendota.com/> [Accessed 13 Apr. 2021].

Öztürk, M.M., Cavusoglu, U. and Zengin, A., 2015. A novel defect prediction method for web pages using k-means++. *Expert Systems with Applications*, 42(19), pp.6496–6506.

Rajagopal, S., 2011. Customer Data Clustering Using Data Mining Technique. *International Journal of Database Management Systems*, 3(4).

Sai, N. and Maguluri, N., 2017. Multi-Class Classification of Textual Data : Detection and Mitigation of Cheating in Massively Multiplayer Online Role Playing Games MULTI-CLASS CLASSIFICATION OF TEXTUAL DATA : DETECTION AND MITIGATION OF CHEATING IN.

Sifa, R., Drachen, A. and Bauckhage, C., 2018. Profiling in Games : Understanding Behavior from Telemetry. In: *Social Interactions in Virtual Worlds: An Interdisciplinary Perspective*. Cambridge University Press.pp.337–374.

Steamcharts, 2021. *Dota 2 - Steam Charts*. [online] steamcharts.com. Available at: <https://steamcharts.com/app/570#All> [Accessed 6 Apr. 2021].

Syakur, M.A., Khotimah, B.K., Rochman, E.M.S. and Satoto, B.D., 2018. Integration K-Means Clustering Method and Elbow Method for Identification of the Best Customer Profile Cluster. *IOP Conference Series: Materials Science and Engineering*, 336(1).

Tibshirani, R., Walther, G. and Hastie, T., 2001. *Estimating the number of data clusters via the gap statistic*. *Journal of the Royal Statistical Society: Series B*, .

Tzoreff, E. and Weiss, A.J., 2017. Expectation-maximization algorithm for direct position determination. *Signal Processing*, 133(October 2016), pp.32–39.

Wang, L., 2016. Discovering phase transitions with unsupervised learning. *Physical Review B*, 94(19).

Warikoo, A., 2014. Proposed Methodology for Cyber Criminal Profiling. *Information Security Journal*, 23, pp.172–178.

Xu, S., Qiao, X., Zhu, L., Zhang, Y., Xue, C. and Li, L., 2016. Reviews on determining the number of clusters. *Applied Mathematics and Information Sciences*, 10(4), pp.1493–1512.

Xu, Y., Qu, W., Li, Z., Min, G., Li, K. and Liu, Z., 2014. Efficient k-means++ approximation with MapReduce. *IEEE Transactions on Parallel and Distributed Systems*, 25(12), pp.3135–3144.

Yuan, C. and Yang, H., 2019. Research on K-Value Selection Method of K-Means Clustering Algorithm. *J*, 2(2), pp.226–235.

Zimmerman, E. and Salen, K., 2003. *Rules of play : game design fundamentals*. Cambridge, Mass.: Mit Press.

APPENDICES

APPENDIX A: EXPERT PROFILE (MUSHI)

Chai "Mushi" Yee Fung is one of the most famous and successful Dota 2 figures in the world. The figure below shows the best achievement from the veteran player.

Date	Place	Tier	Tournament	Team	Result	Prize
2018-08-21	9 - 12th	Tier 1	The International 2018		0 : 2	$382,983
2018-04-07	1st	Tier 1	Dota 2 Asia Championships 2018		3 : 2	$370,000
2017-10-22	1st	Tier 2	PGL Open Bucharest		2 : 0	$130,000
2016-08-12	4th	Tier 1	The International 2016		0 : 2	$1,453,932
2016-06-10	5 - 6th	Tier 1	The Manila Major 2016		0 : 2	$202,500
2016-03-05	5 - 6th	Tier 1	The Shanghai Major 2016		0 : 2	$202,500
2014-07-20	4th	Tier 1	The International 2014		0 : 2	$819,298
2014-04-20	1st	Tier 1	StarLadder StarSeries Season 9		3 : 0	$85,000
2014-01-01	1st	Tier 1	2013 WPC ACE Dota 2 League		4 : 3	$165,179
2013-08-11	3rd	Tier 1	The International 2013		1 : 2	$287,438

Source: Liquidpedia (2021a)

APPENDIX B EXPERT PROFILE (OHAIYO)

Khoo “Ohaiyo” Chong Xin is also one of the most famous and successful Dota 2 figures in the world. The figure below shows the best achievement from the veteran player.

Date	Place	Tier	Tournament	Team	Result	Prize
2017-12-17	2nd	Tier 2	DOTA Summit 8		1 : 3	$60,000
2016-08-12	4th	Tier 1	The International 2016		0 : 2	$1,453,932
2016-07-24	3rd	Tier 1	StarLadder i-League StarSeries Season 2		0 : 2	$37,500
2016-06-10	5 - 6th	Tier 1	The Manila Major 2016		0 : 2	$202,500
2016-04-24	3 - 4th	Tier 1	ESL One Manila 2016		1 : 2	$25,000
2016-03-05	5 - 6th	Tier 1	The Shanghai Major 2016		0 : 2	$202,500
2015-06-07	3rd	Tier 1	joinDOTA MLG Pro League Season 2		2 : 0	$25,087
2015-05-23	3rd	Tier 1	i-League Season 3		0 : 2	$55,590
2015-01-06	1st	Tier 1	Dota 2 League Season 5		3 : 1	$26,679
2013-08-11	3rd	Tier 1	The International 2013		1 : 2	$287,438

Source: Liquidpedia (2021b)

APPENDIX C EXPERT PROFILE (WINTER)

Chan “Winter” Litt Binn is a professional Dota 2 player, a professional coach and a famous panel commentator/analyst. He was invited to numerous official Valve events as a broadcast talent. The figure below shows his participation in some major Valve events.

Date	Tier	Tournament	Position	Partner List
2018-08-25	Tier 1	The International 2018	Commentator/Analyst	[show]
2017-08-12	Tier 1	The International 2017	Commentator	[show]
2016-12-10	Tier 1	The Boston Major 2016	Analyst	[show]
2016-08-13	Tier 1	The International 2016	Analyst	[show]
2016-06-12	Tier 1	The Manila Major 2016	Analyst	[show]
2016-03-06	Tier 1	The Shanghai Major 2016	Commentator/Analyst	[show]
2015-11-21	Tier 1	The Frankfurt Major 2015	Off-site Commentator	[show]
2015-08-08	Tier 1	The International 2015	Commentator	[show]
2015-02-09	Tier 1	Dota 2 Asia Championships 2015	Analyst	[show]
2014-07-21	Tier 1	The International 2014	Analyst	[show]

Source: Liquidpedia (2021c)

APPENDIX D QUESTIONNAIRE FOR FEATURE SELECTION

1. Based on your experience and in-game knowledge, what are the attributes that a smurf/booster has? (For example, more than 30 kill count per match, less than 5 deaths in 40min matches.) Please be as specific as possible.

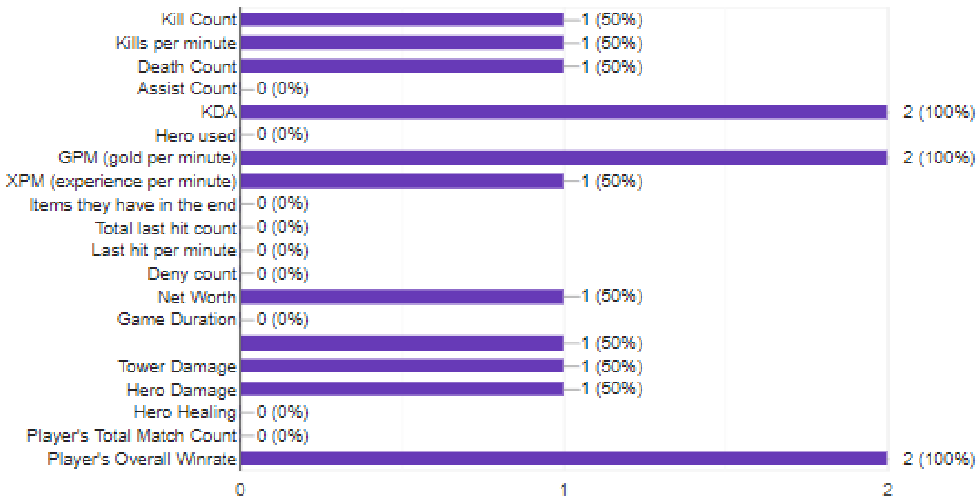
2 responses

High skill

High KDA + win rate match (example 18-2 to 20-0) least

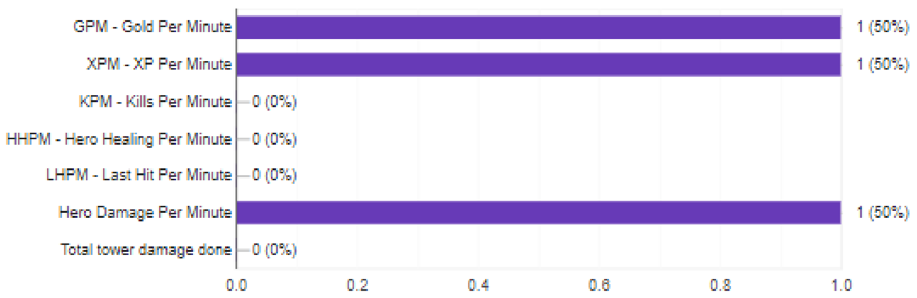
2. From the below choices, please pick at most 10 features to look at if we want to identify a smurf/booster.

2 responses



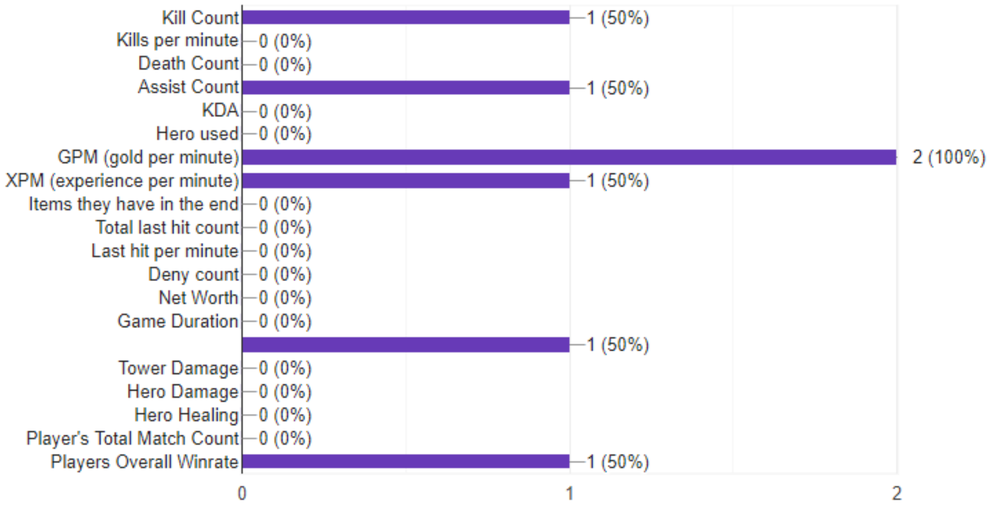
3. In conjunction with the above question, which benchmarks below should we look at when we are trying to identify a smurf/booster?

2 responses



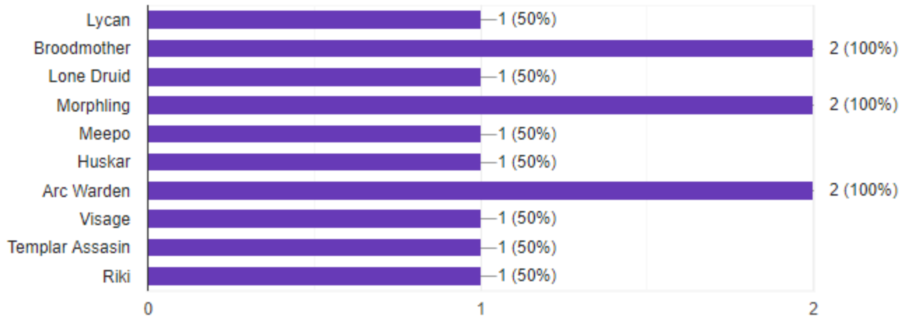
4. If we were to consider the smurfs/boosters who play as supports, what are the other things you would add?

2 responses



5. The following list is a list of smurf heroes that I create. Select the heroes that you agree. Add in more if you think there is any other.

2 responses



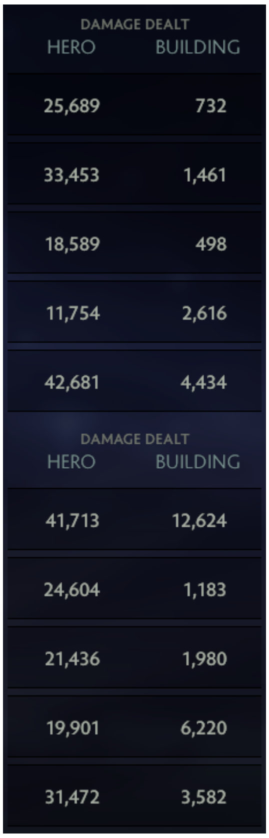
6. Based on your experience, do you have any other advice on identifying a player's skill level?

2 responses

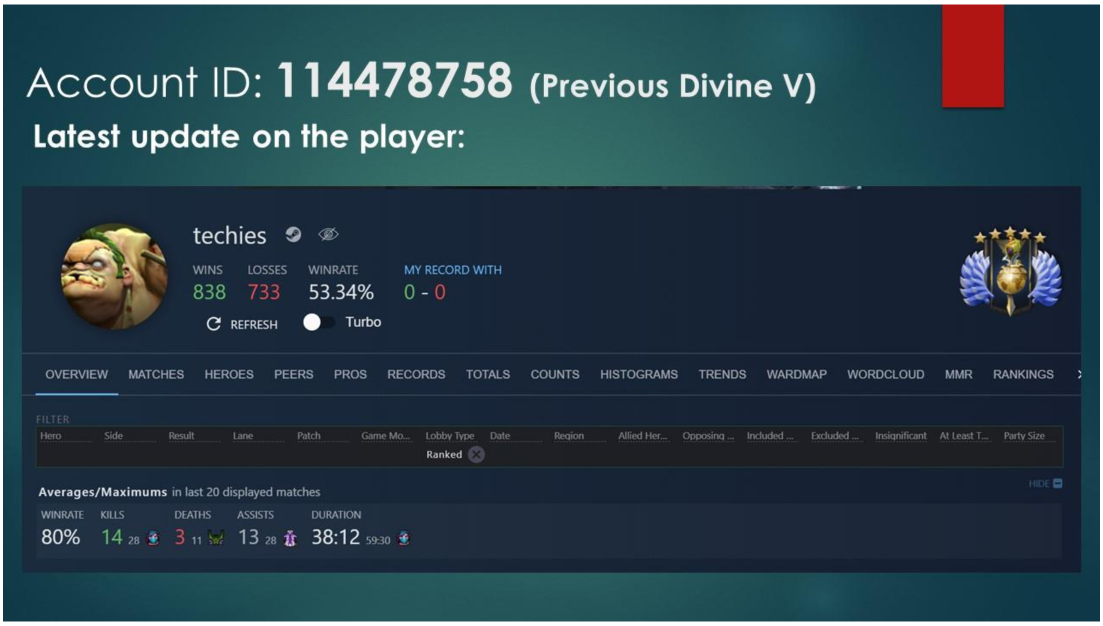
No

Just KDA first is the easiest one

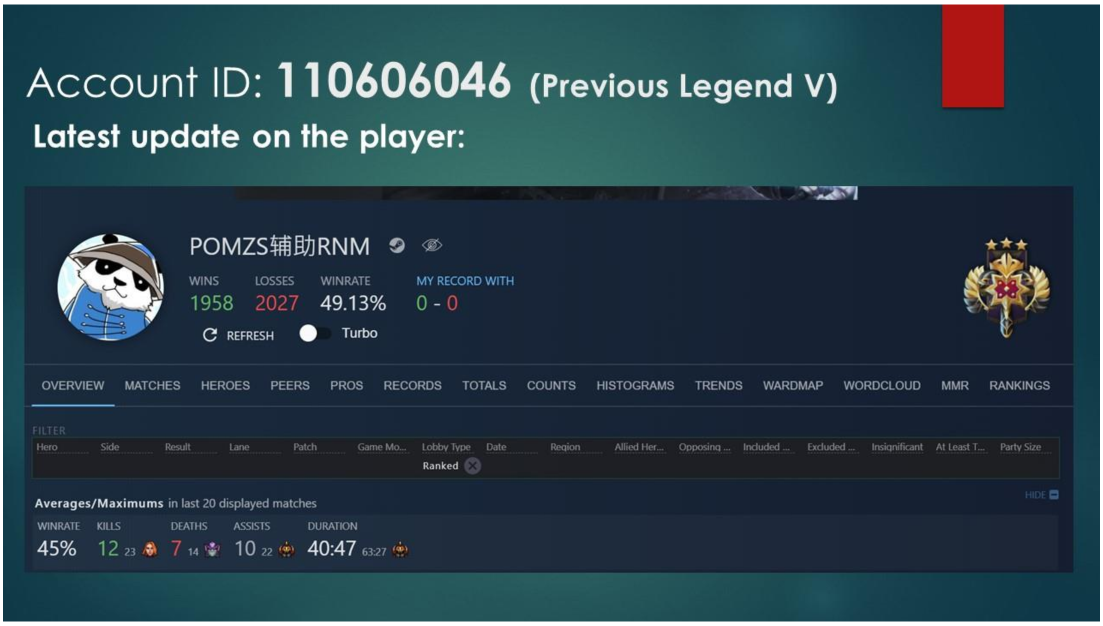
APPENDIX E RAW FEATURES SHOWN IN DOTA 2 GAME CLIENT



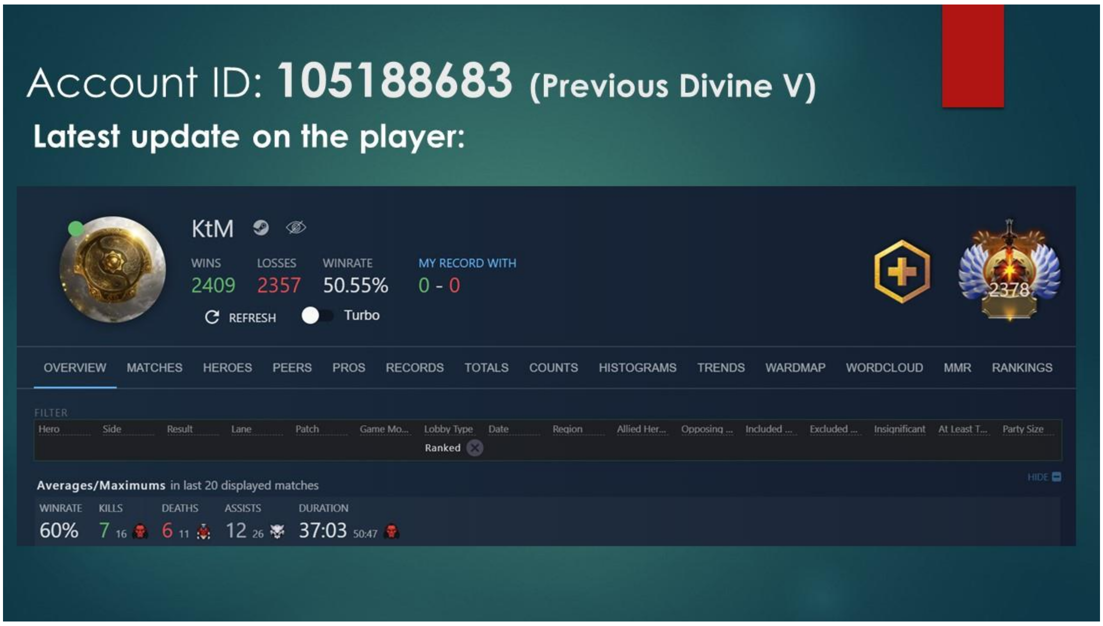
APPENDIX F ADDITIONAL REFERENCE DOCUMENT



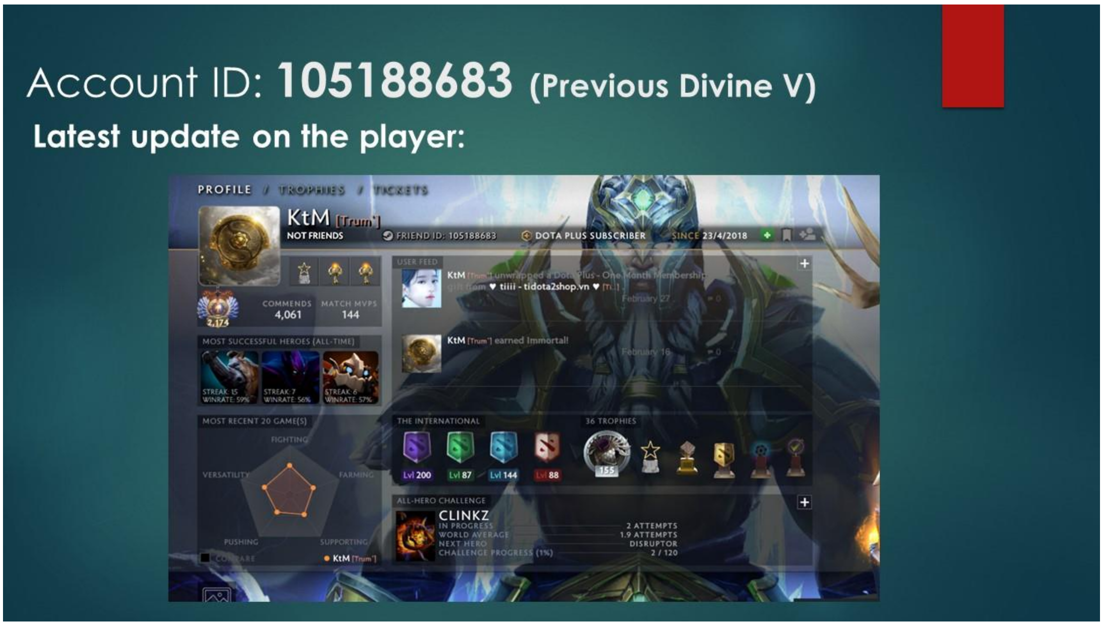
Source: OpenDota (2021)

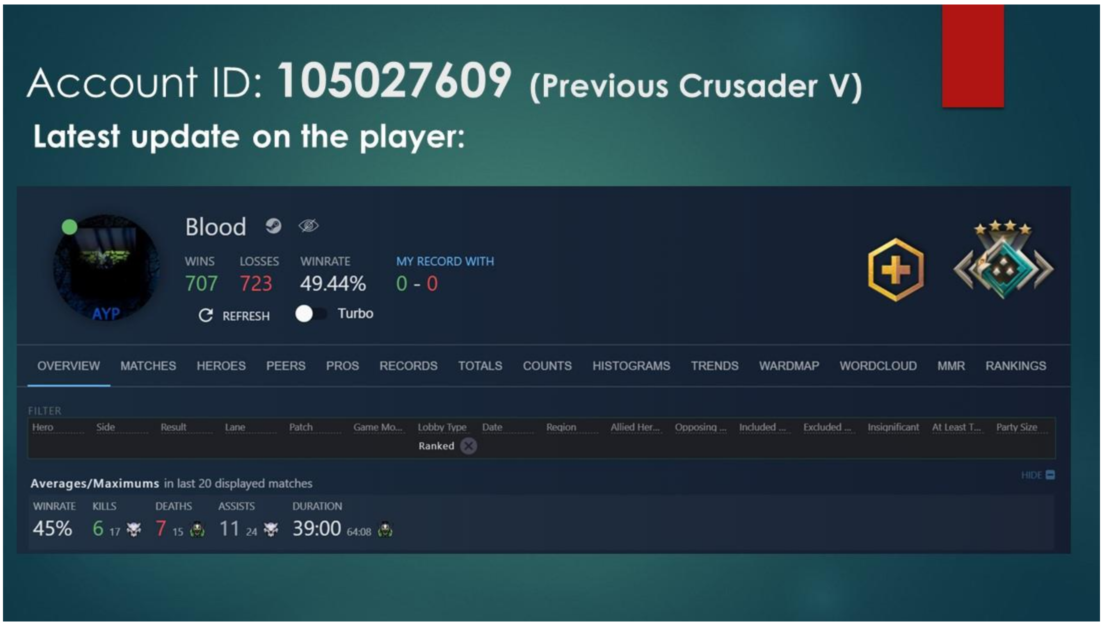

Account ID: 110606046 (Previous Legend V)
Latest update on the player:
POMZS辅助RNM
WINS 1958
LOSSES 2027
WINRATE 49.13%
MY RECORD WITH 0 - 0
REFRESH
Turbo
OVERVIEW MATCHES HEROES PEERS PROS RECORDS TOTALS COUNTS HISTOGRAMS TRENDS WARDMAP WORDCLOUD MMR RANKINGS
FILTER
Ranked
Averages/Maximums in last 20 displayed matches
WINRATE 45%
KILLS 12 23
DEATHS 7 14
ASSISTS 10 22
DURATION 40:47 63:27


Account ID: 110606046 (Previous Legend V)
Latest update on the player:
PROFILE / TROPHIES / TICKETS
POMZS辅助RNM [faman]
NOT FRIENDS
FRIEND ID: 110606046
PREVIOUS RANK
Ancient
MOST SUCCESSFUL HEROES (ALL-TIME)
MOST RECENT 20 GAME(S)
FIGHTING
FARMING
SUPPORTING
PUSHING
VERSATILITY
THE INTERNATIONAL
Lvl 497
Lvl 256
29 TROPHIES
ALL-HERO CHALLENGE
GYROCOPTER
IN PROGRESS 0 ATTEMPTS
WORLD AVERAGE 1.9 ATTEMPTS
NEXT HERO SLARDAR
CHALLENGE PROGRESS (4%) 5 / 120

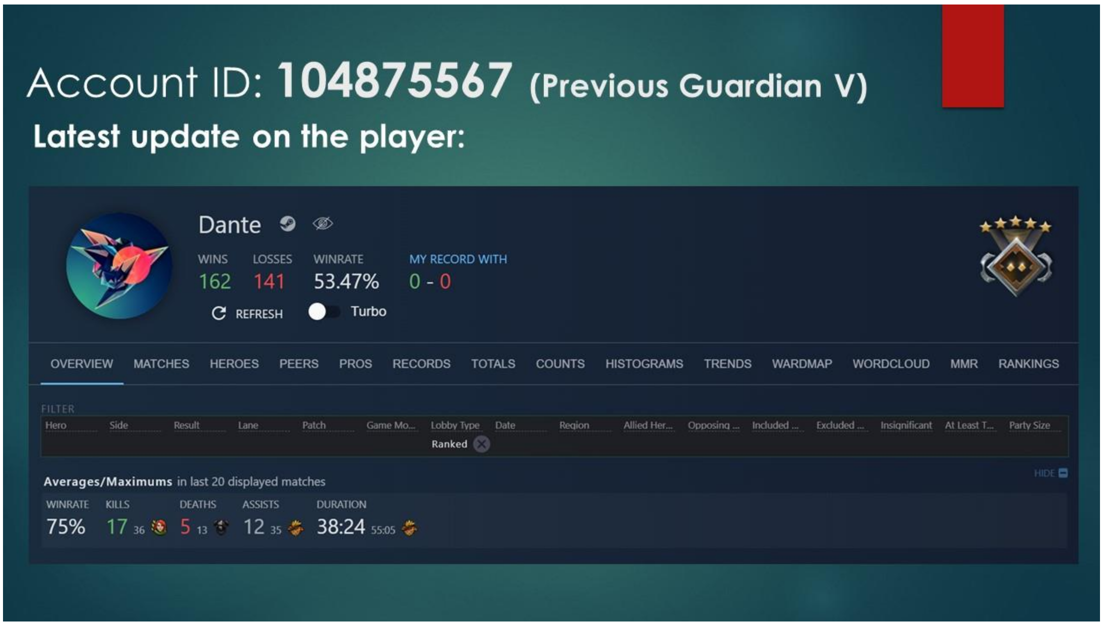


Source: OpenDota (2021)

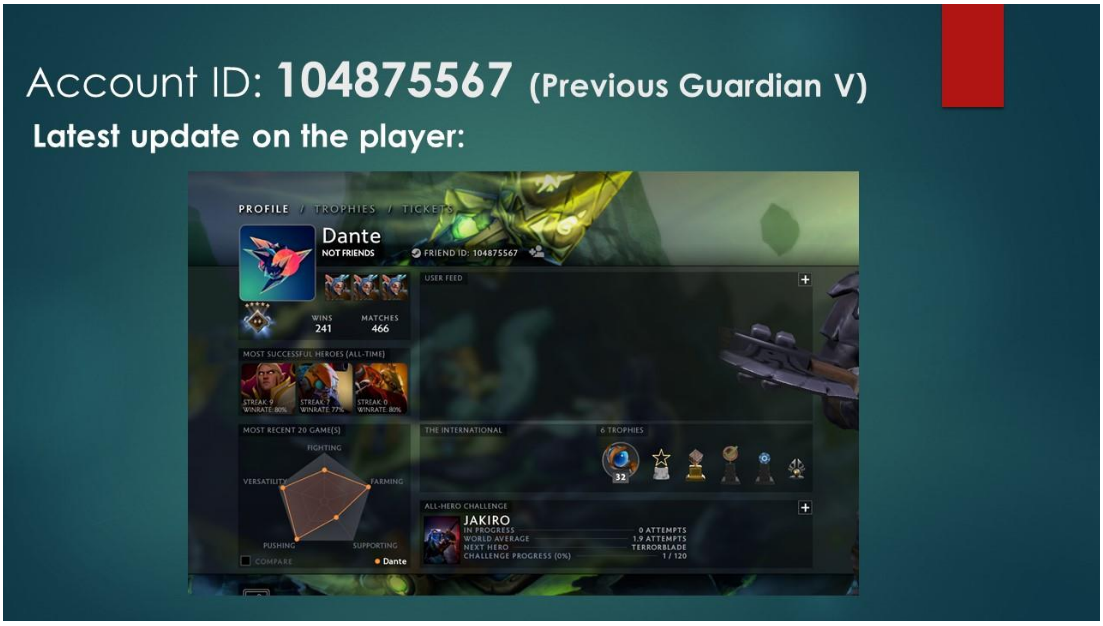


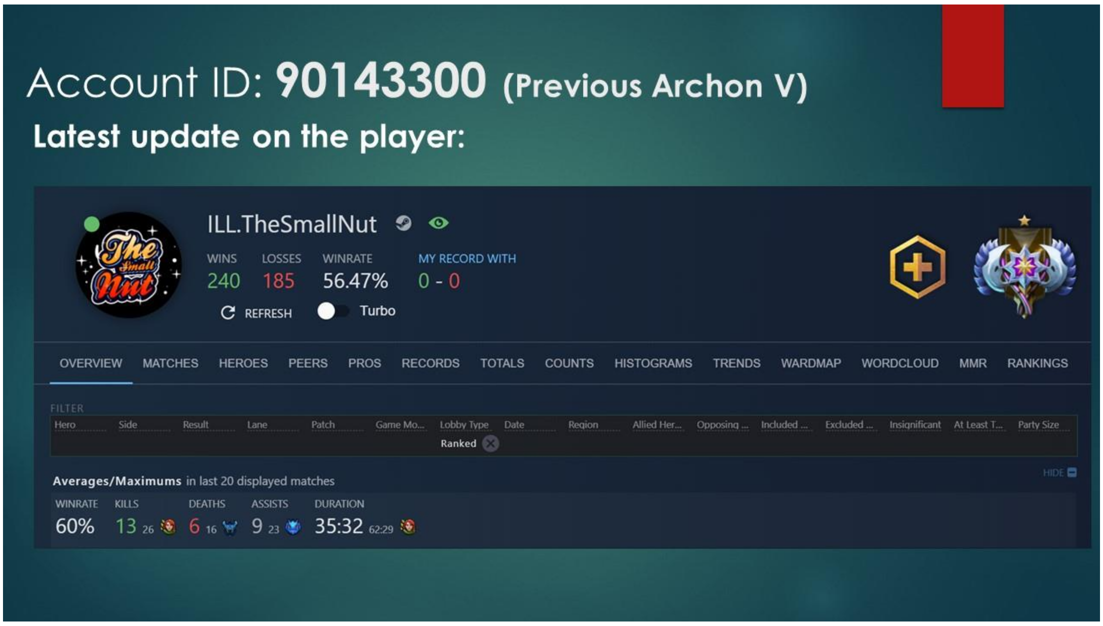


Source: OpenDota (2021)

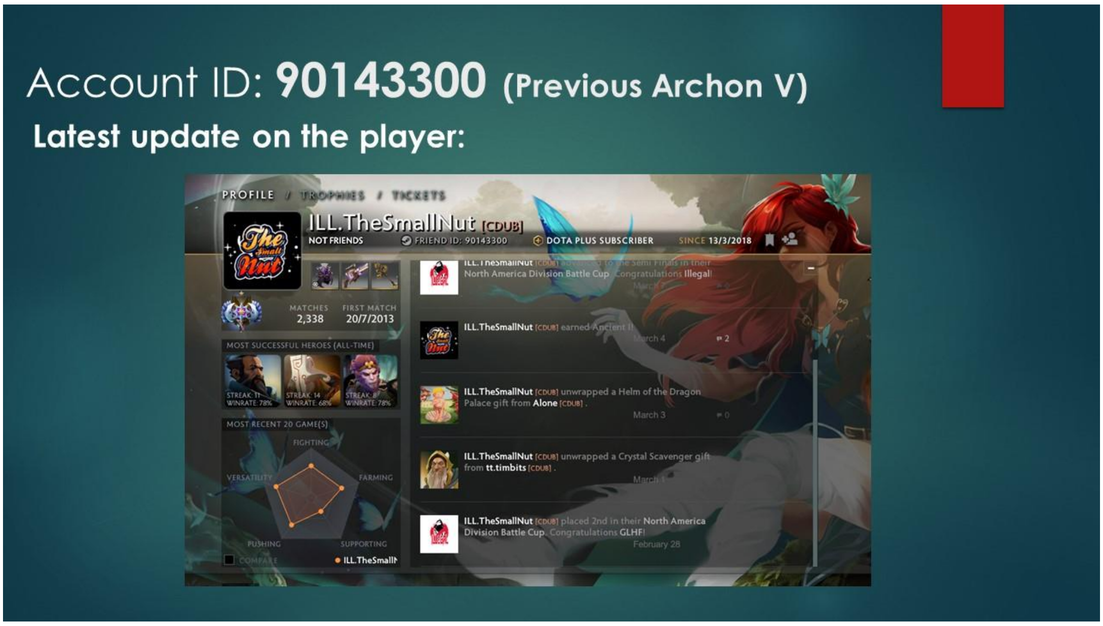


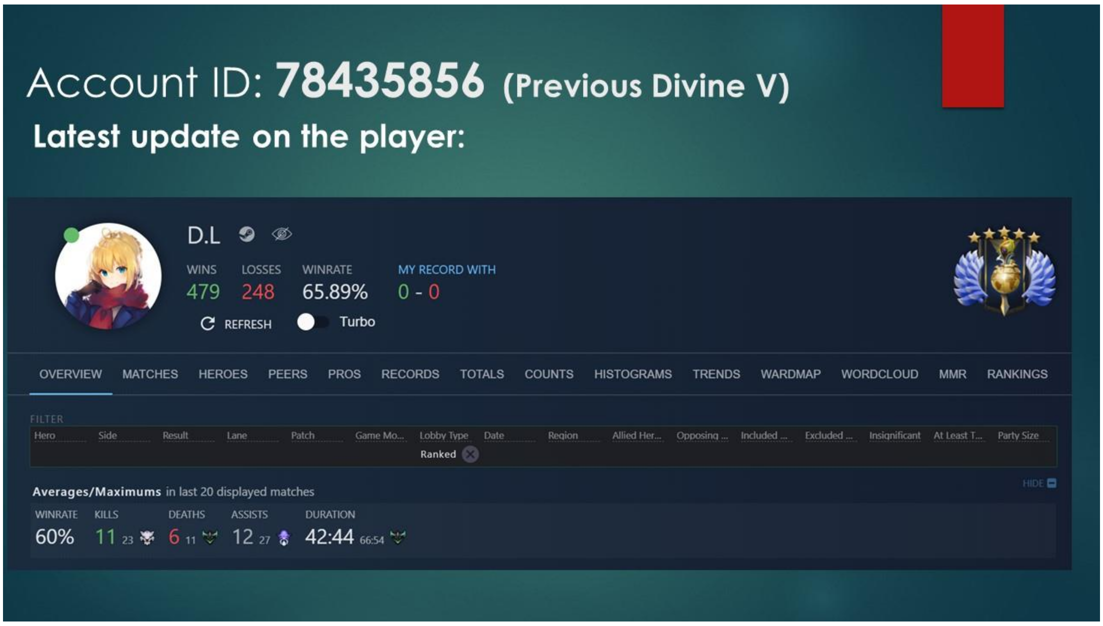
Source: OpenDota (2021)



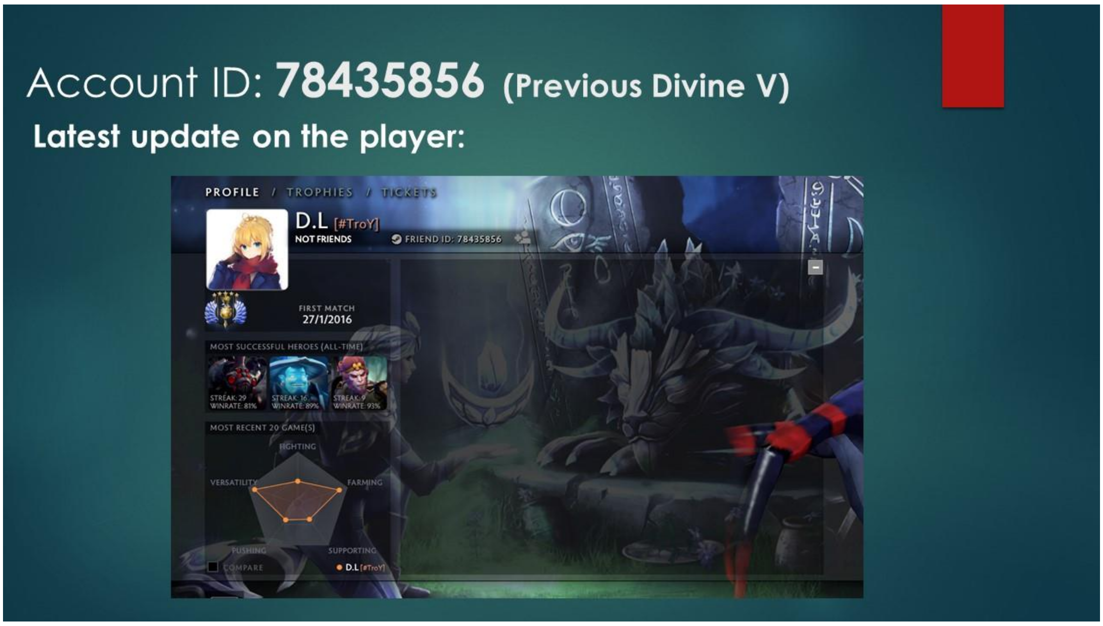


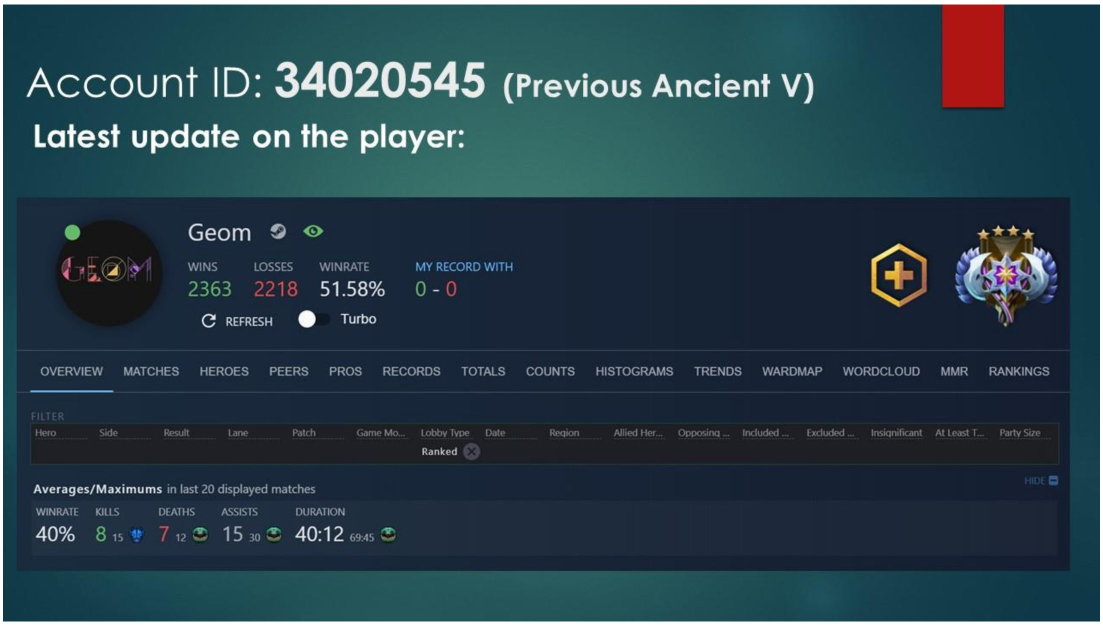
Source: OpenDota (2021)



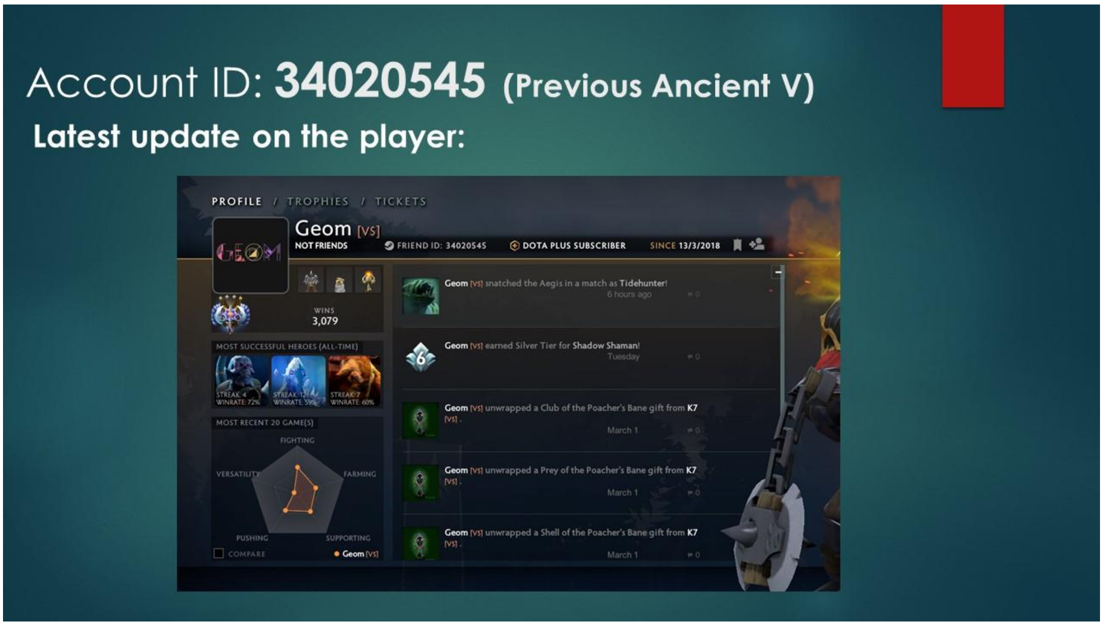


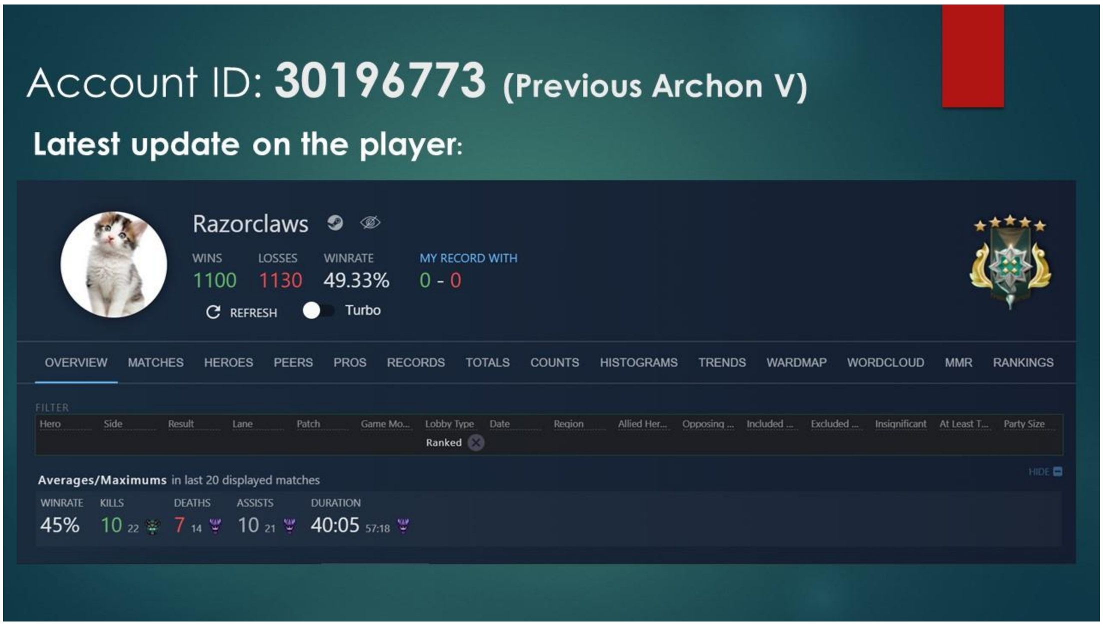
Source: OpenDota (2021)



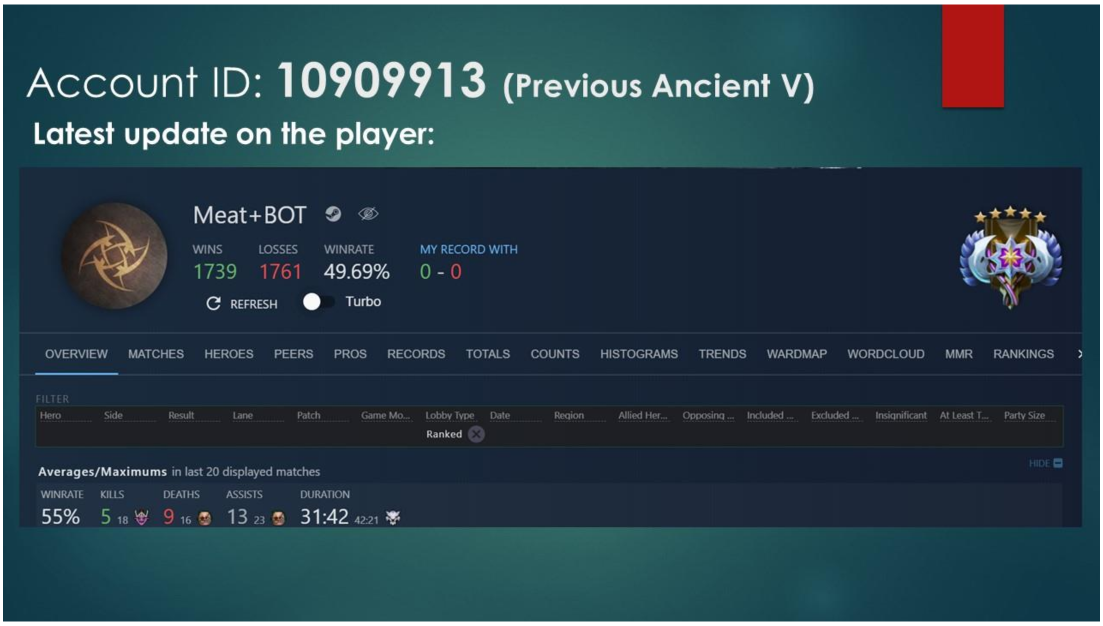


Source: OpenDota (2021)

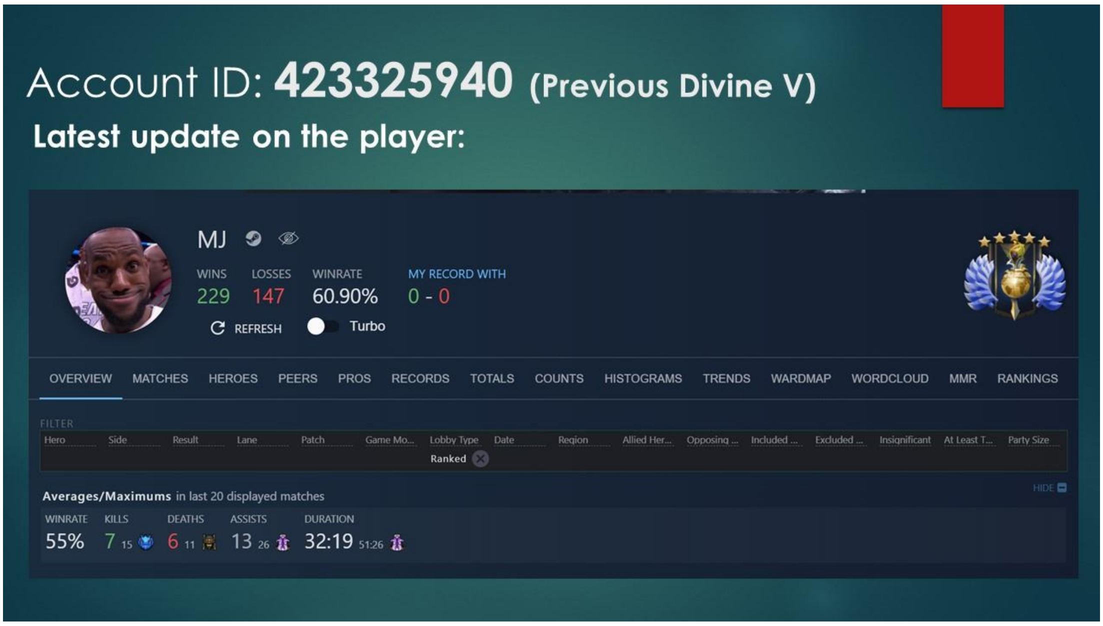




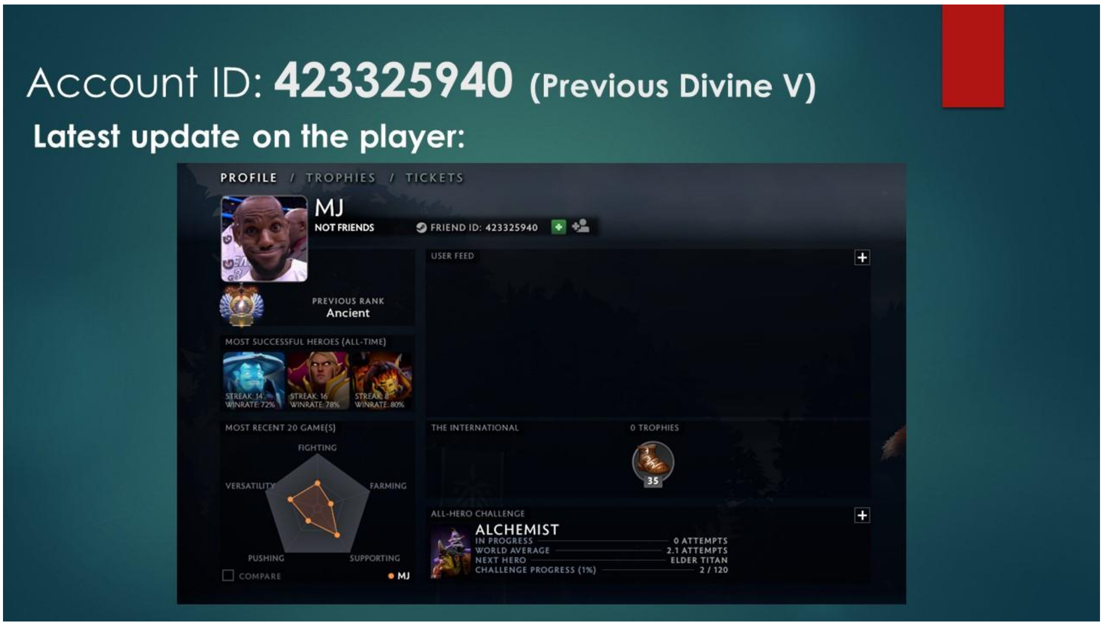
Source: OpenDota (2021)

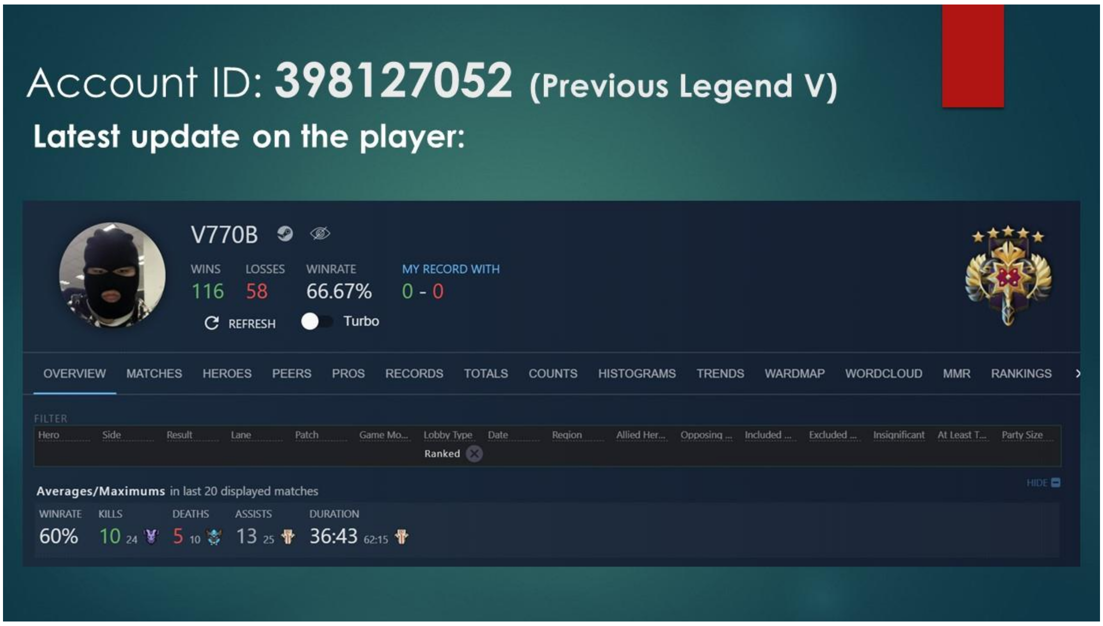


Source: OpenDota (2021)

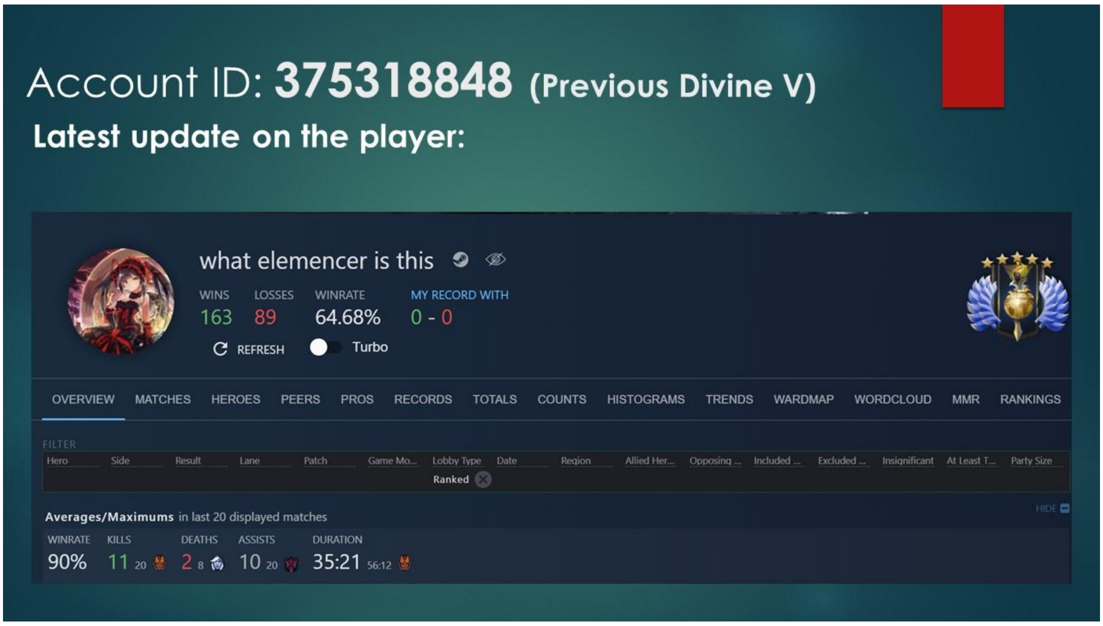


Source: OpenDota (2021)

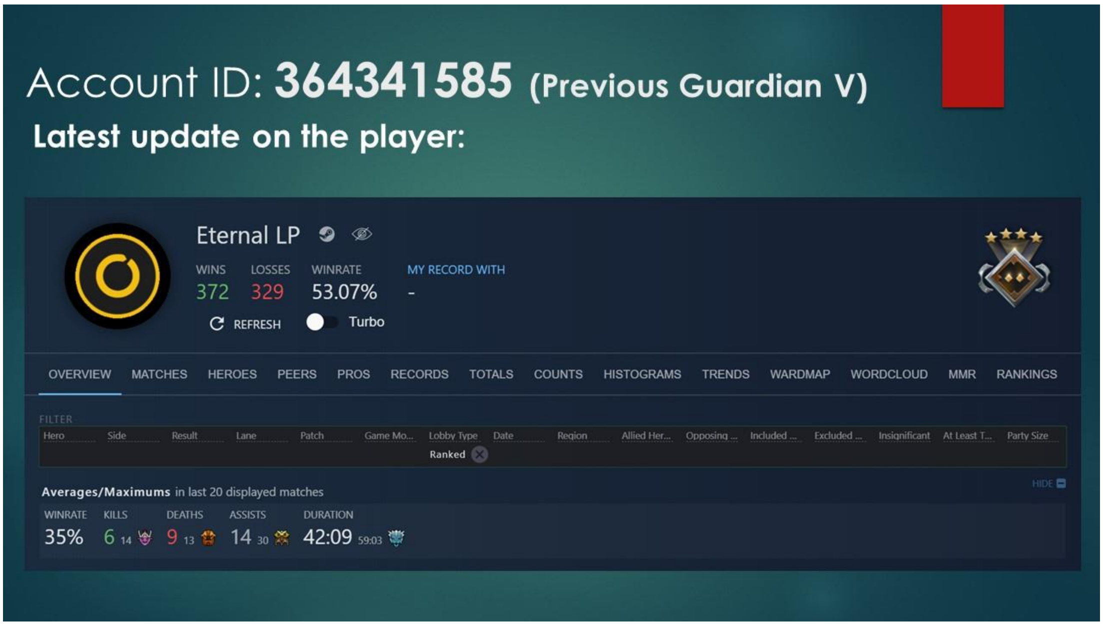




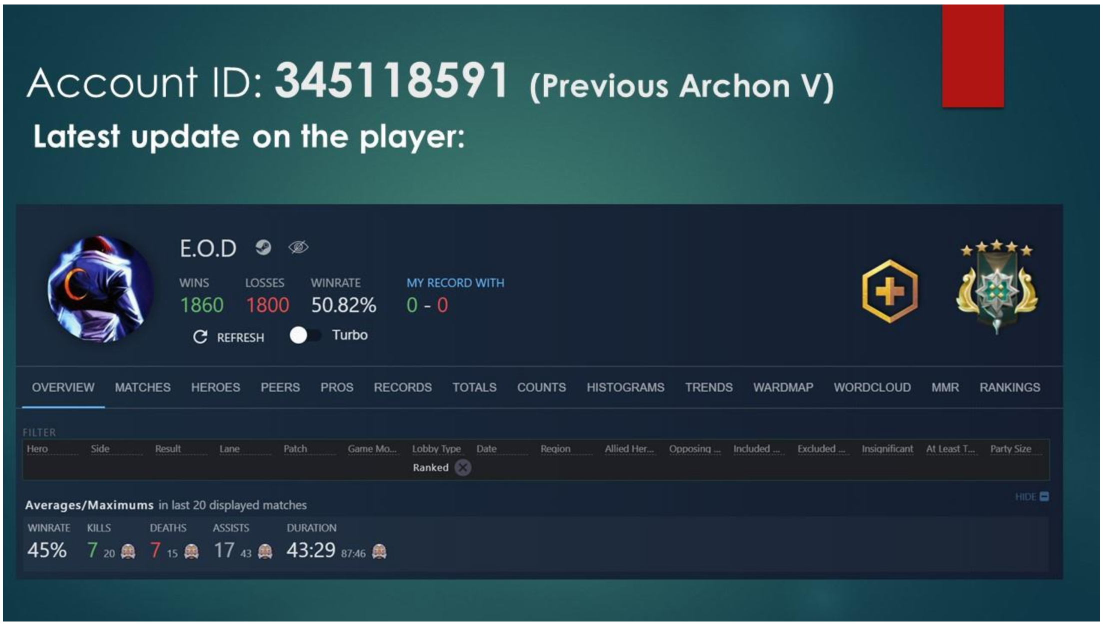
Source: OpenDota (2021)



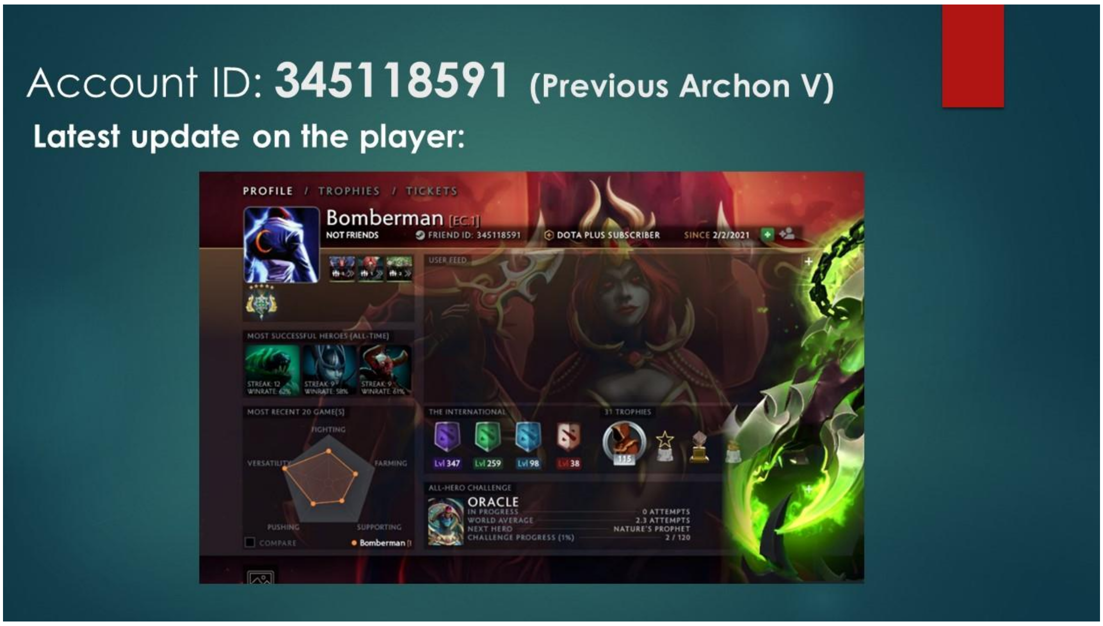
Source: OpenDota (2021)

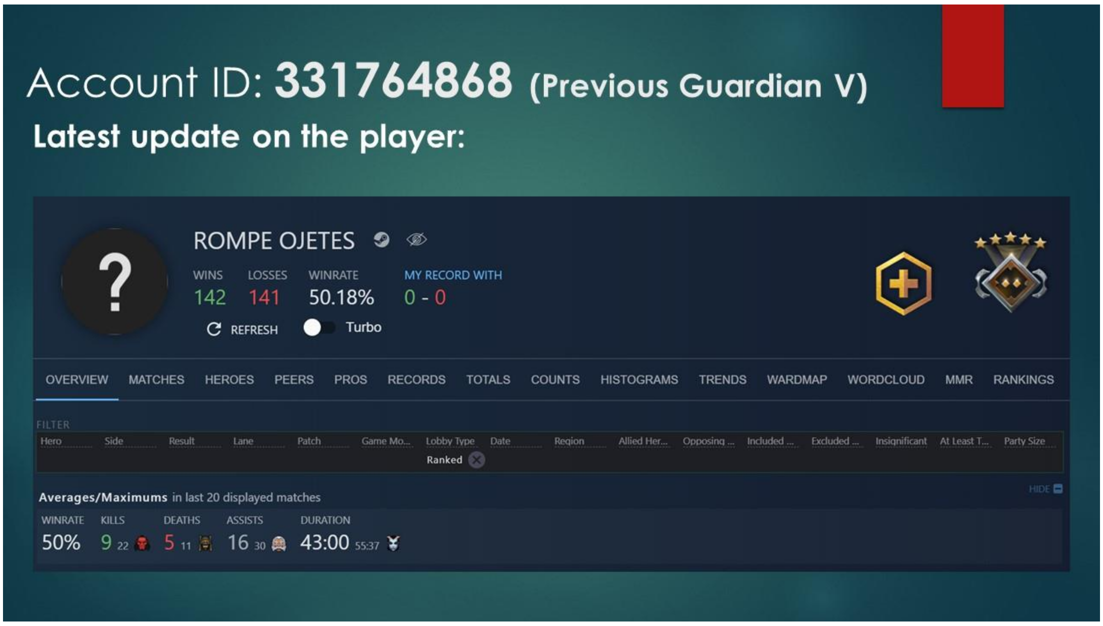


Source: OpenDota (2021)

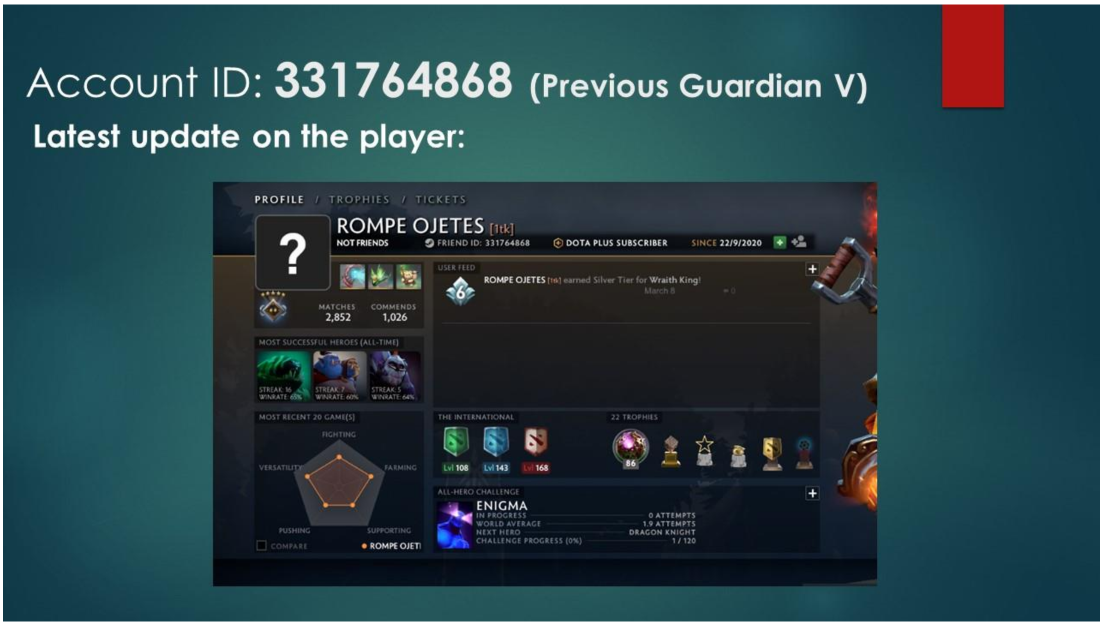


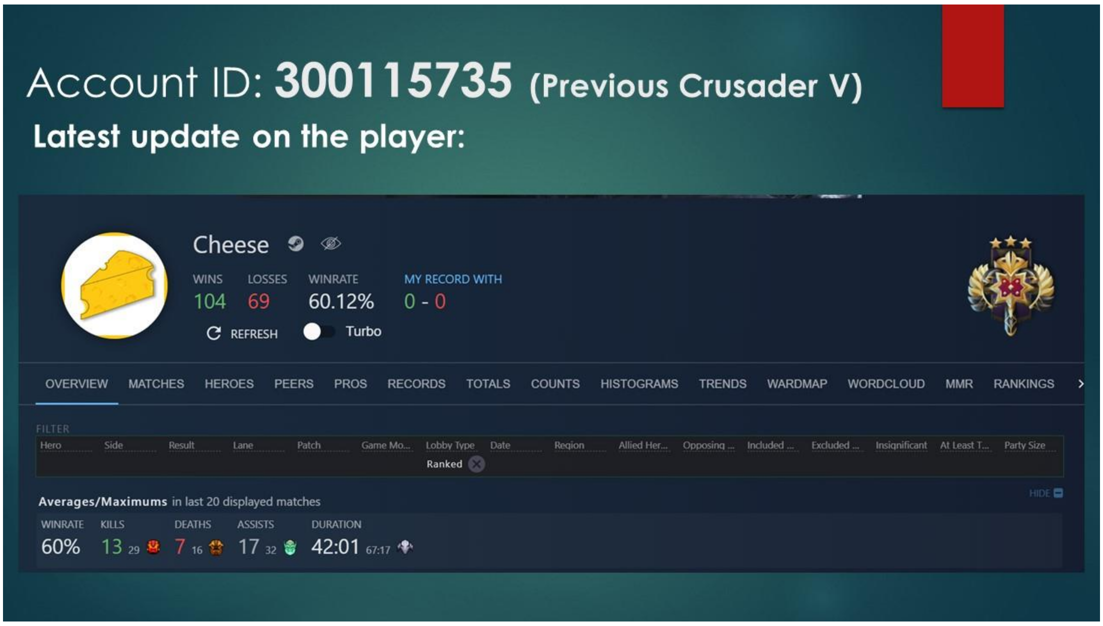
Source: OpenDota (2021)



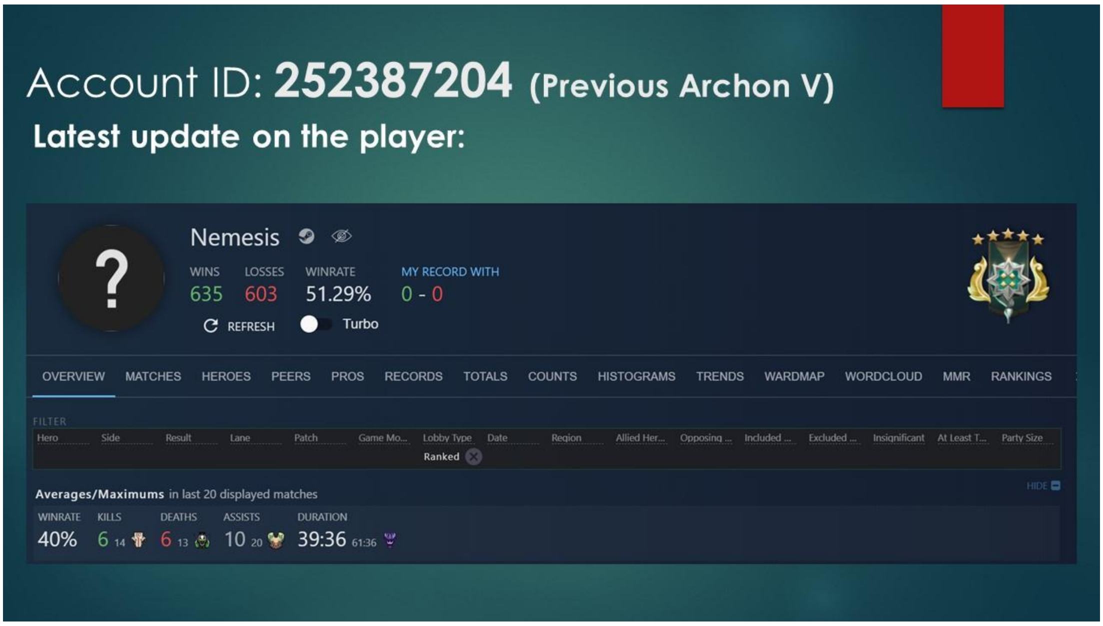


Source: OpenDota (2021)

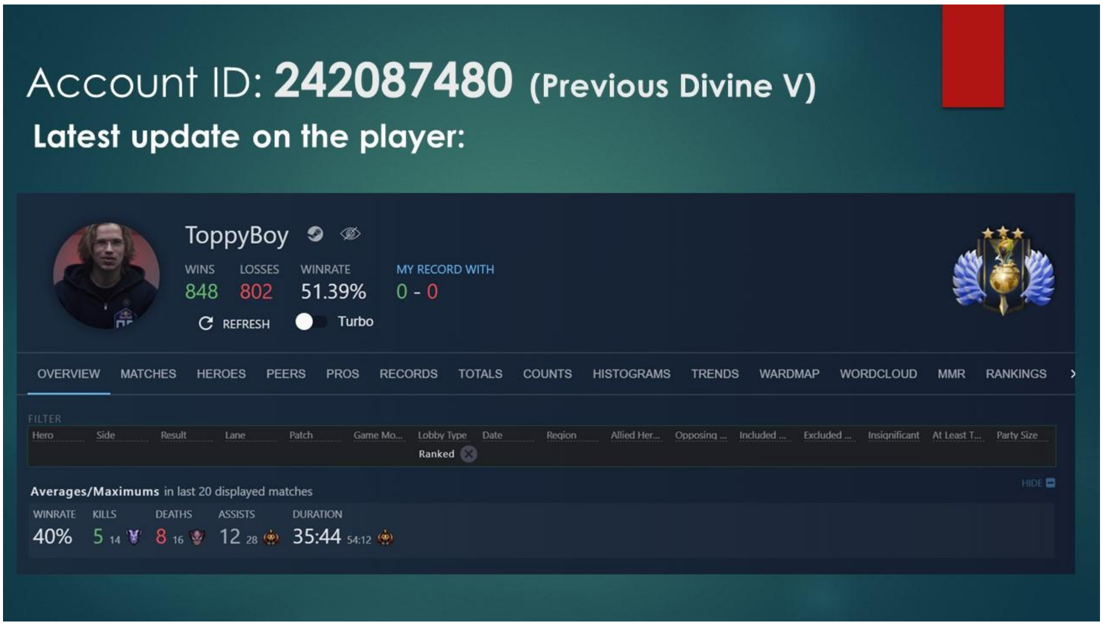




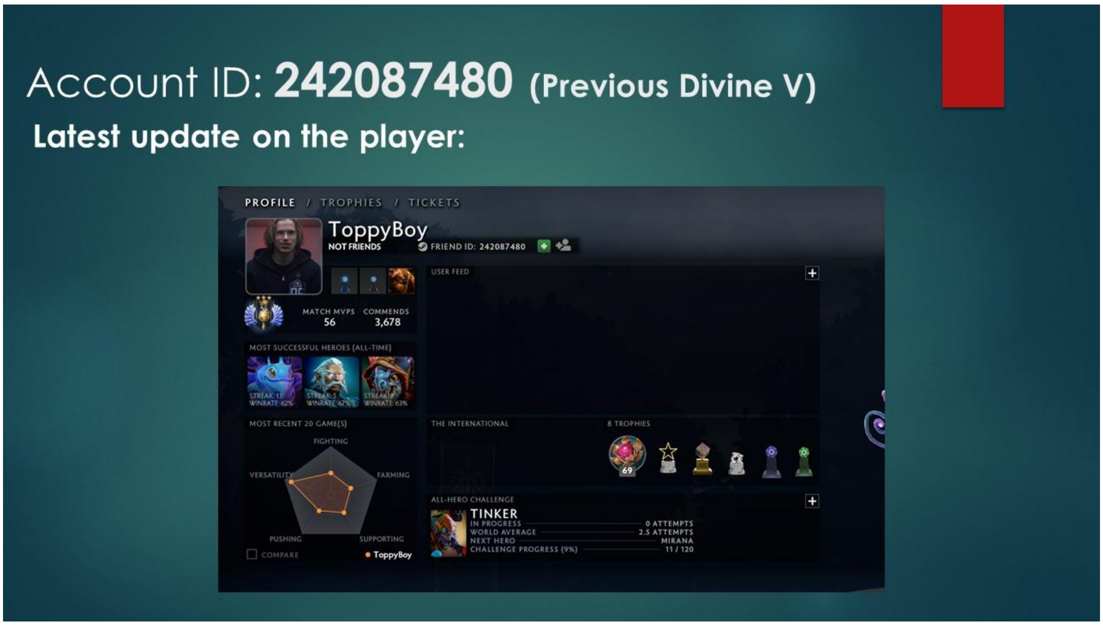
Source: OpenDota (2021)

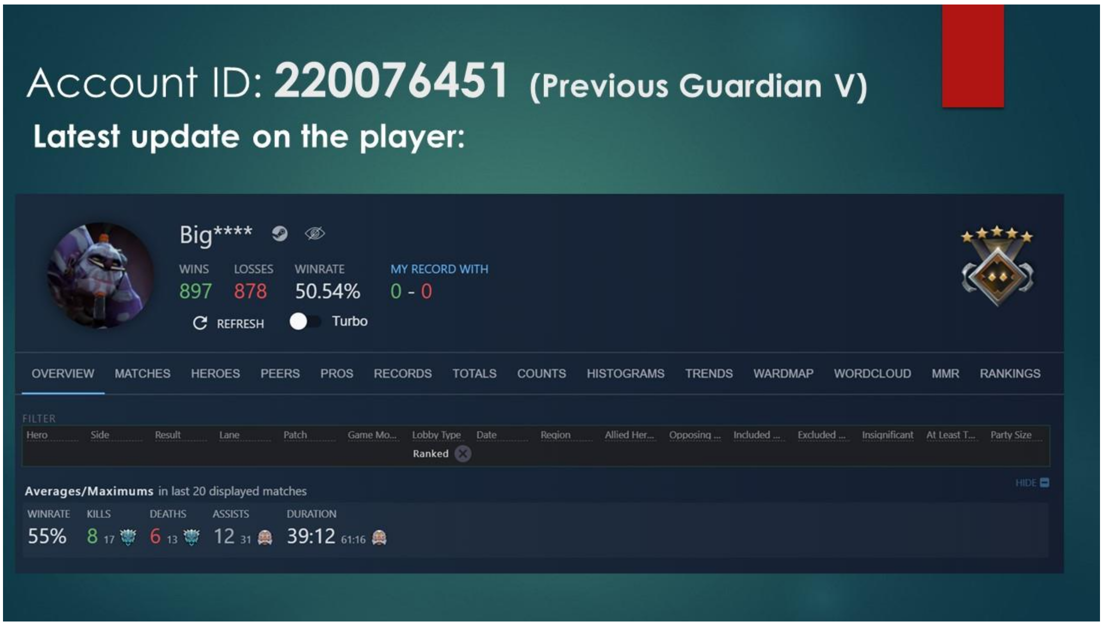


Source: OpenDota (2021)

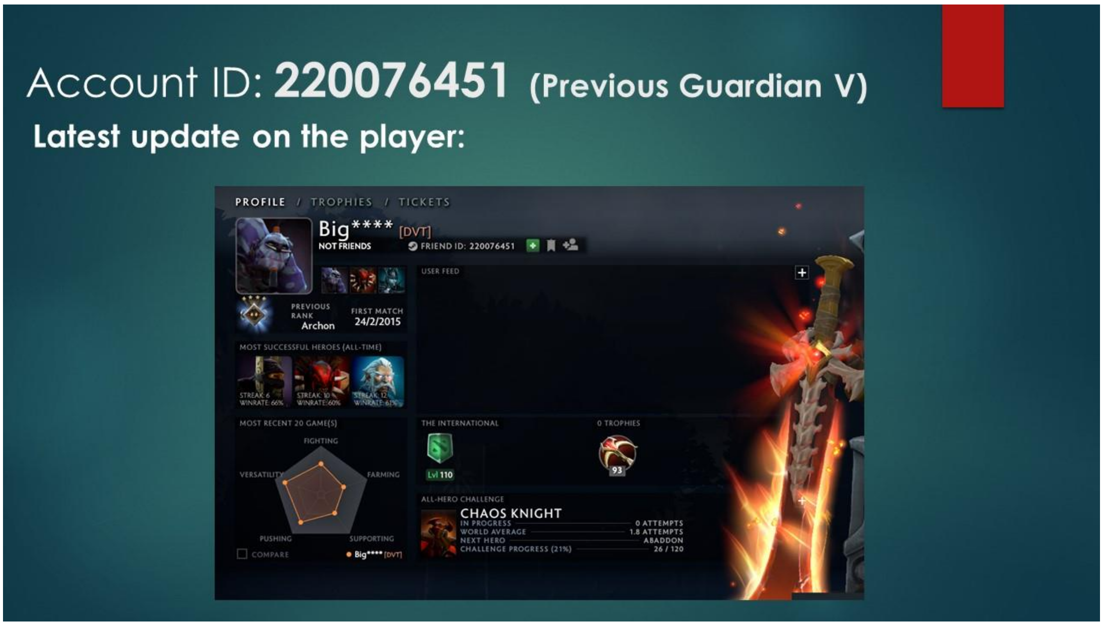


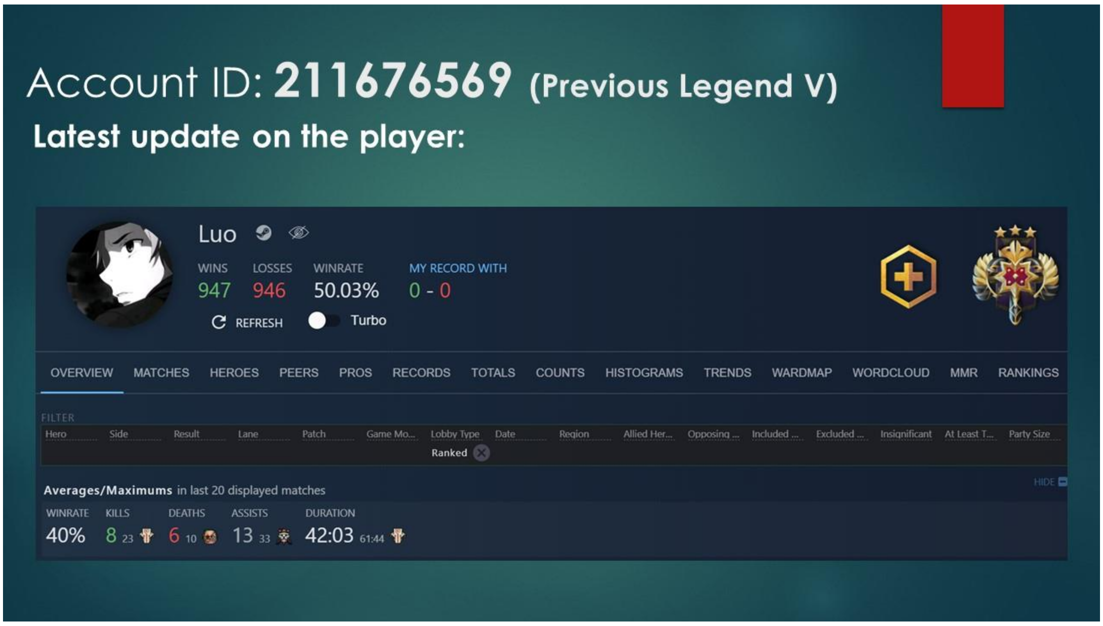
Source: OpenDota (2021)



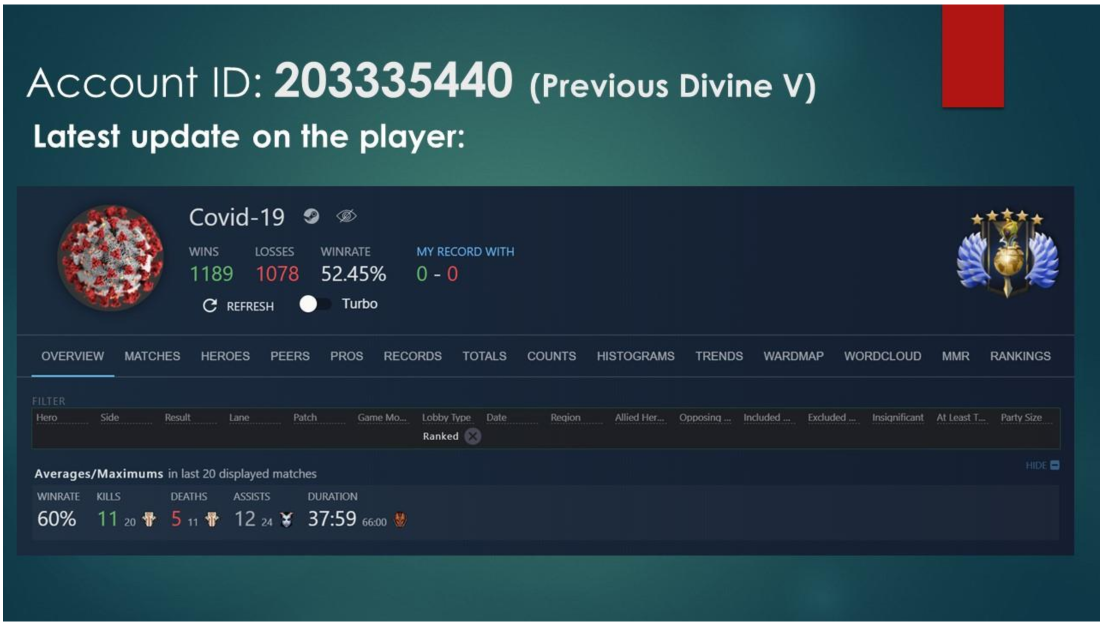


Source: OpenDota (2021)

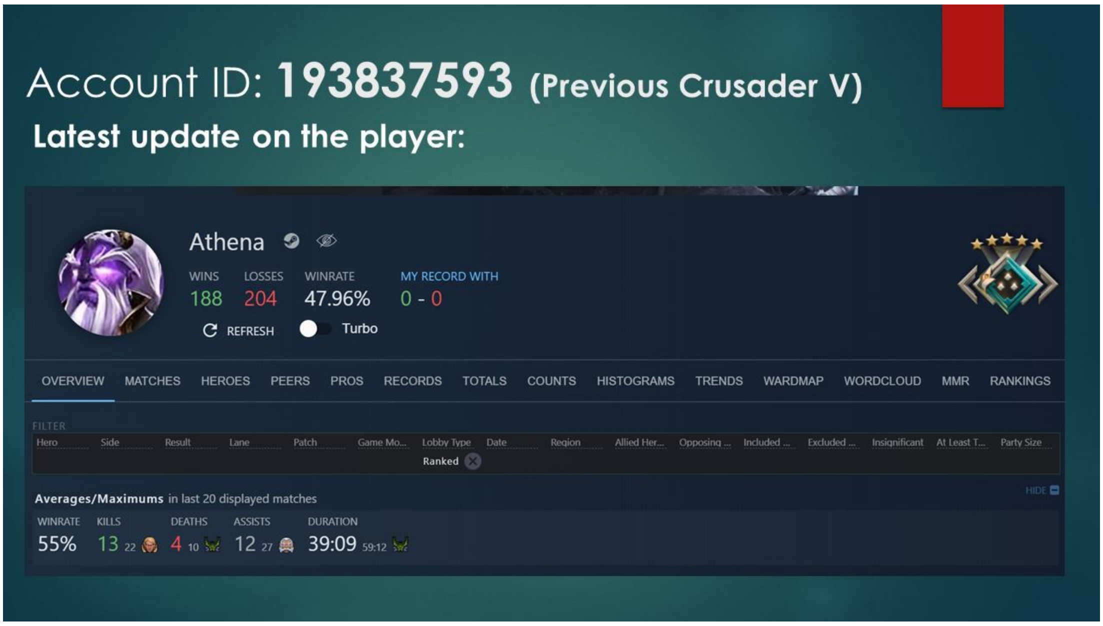




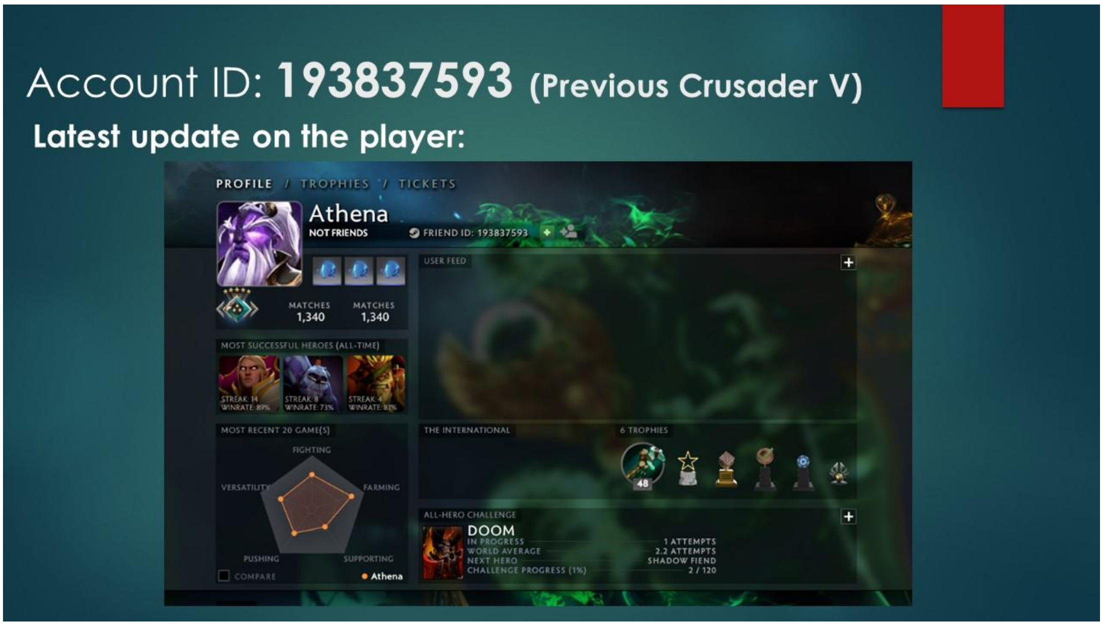
Source: OpenDota (2021)

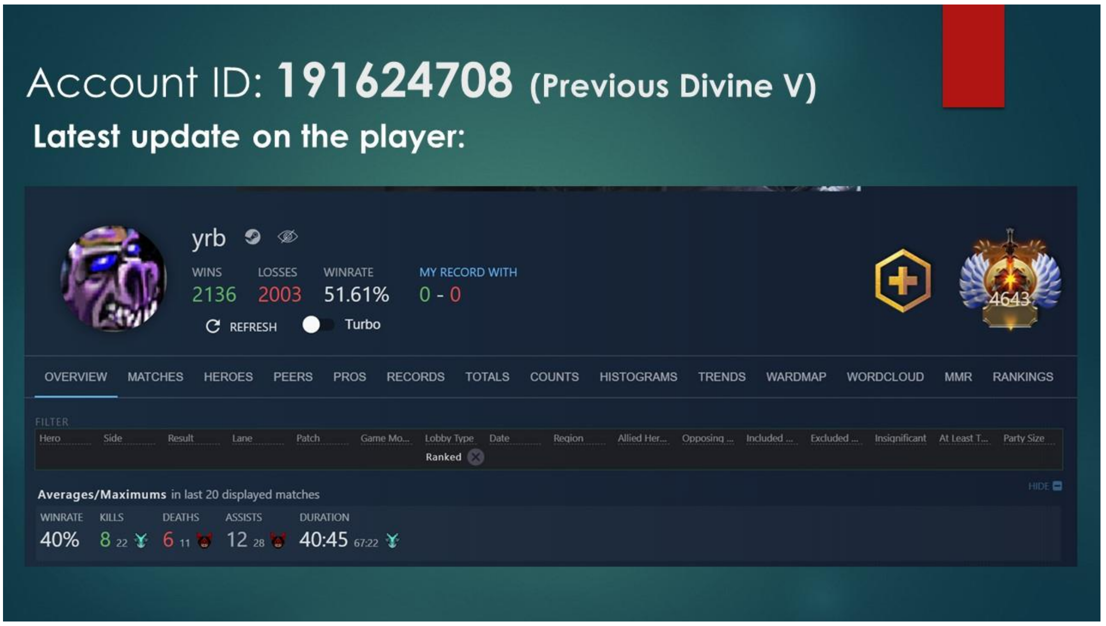


Source: OpenDota (2021)

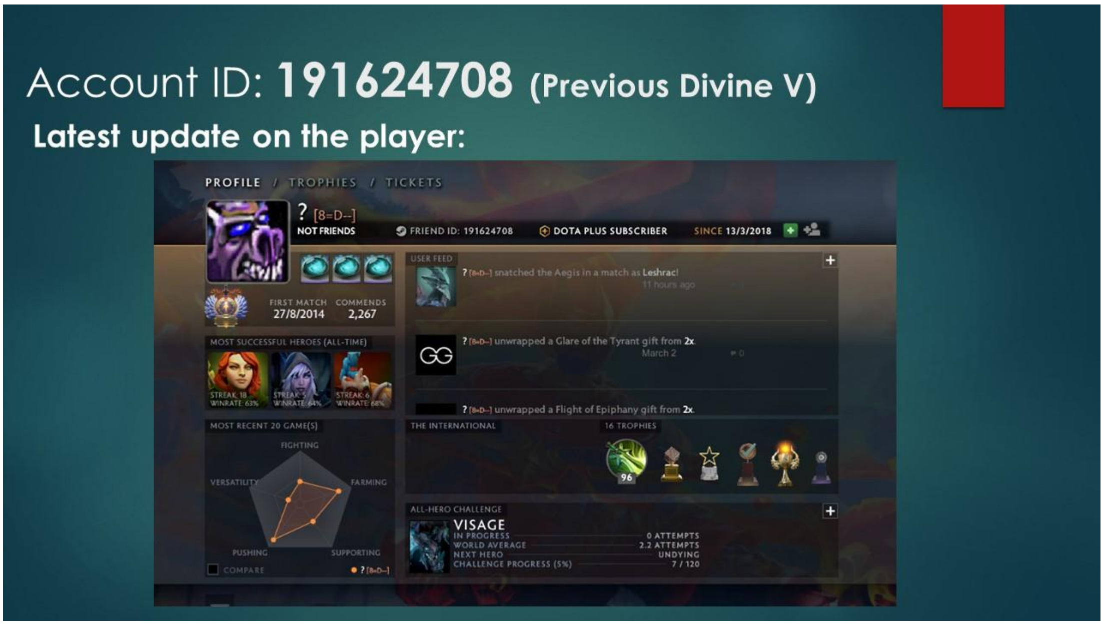


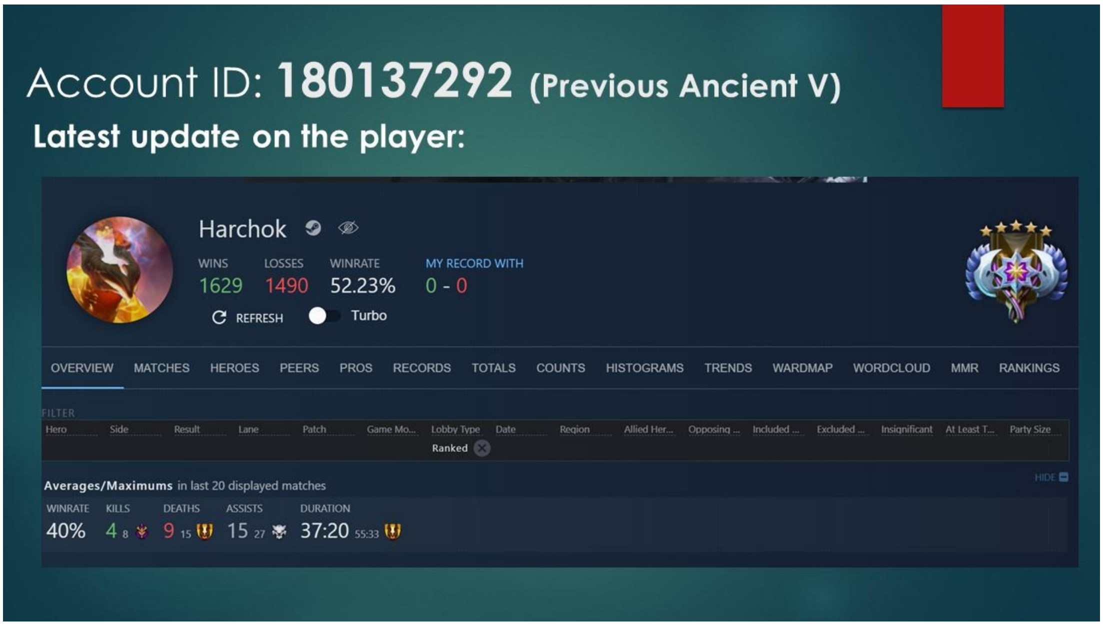
Source: OpenDota (2021)



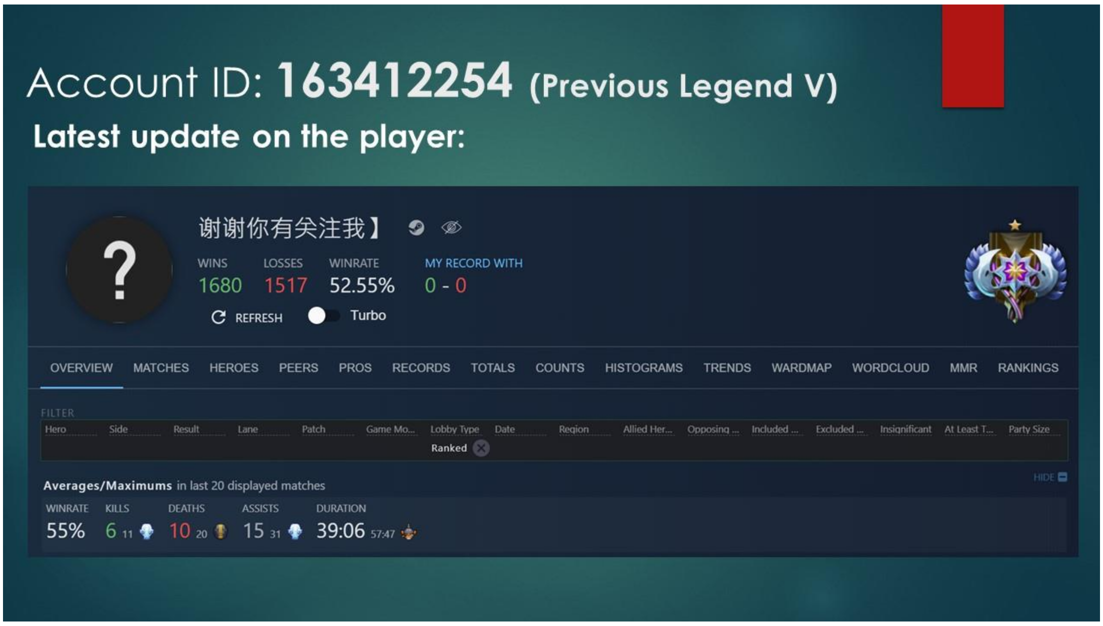


Source: OpenDota (2021)

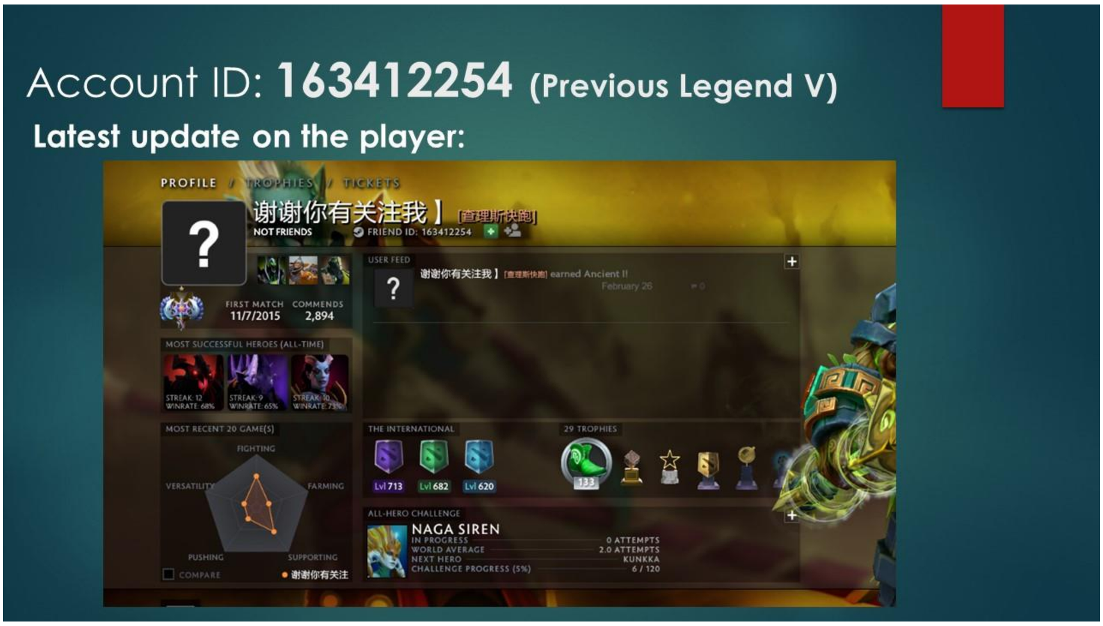


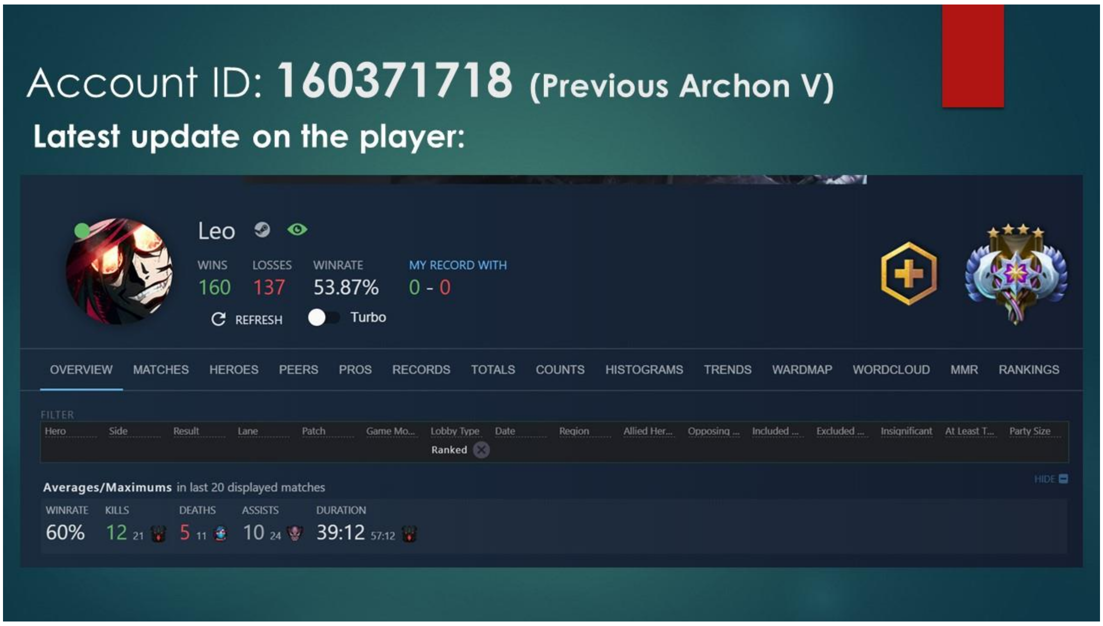


Source: OpenDota (2021)

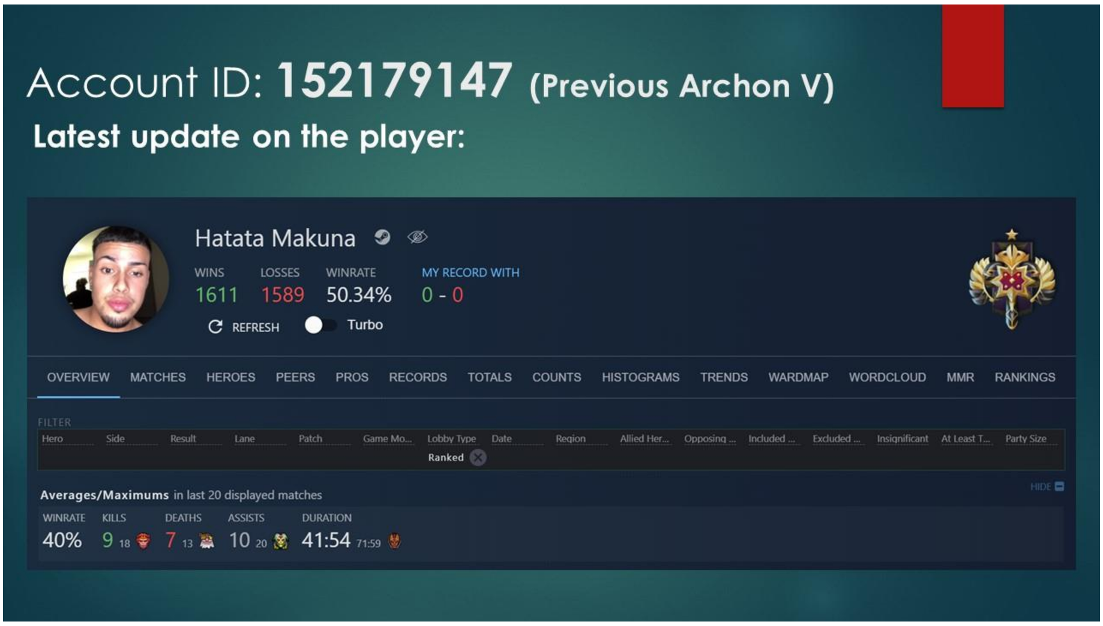


Source: OpenDota (2021)

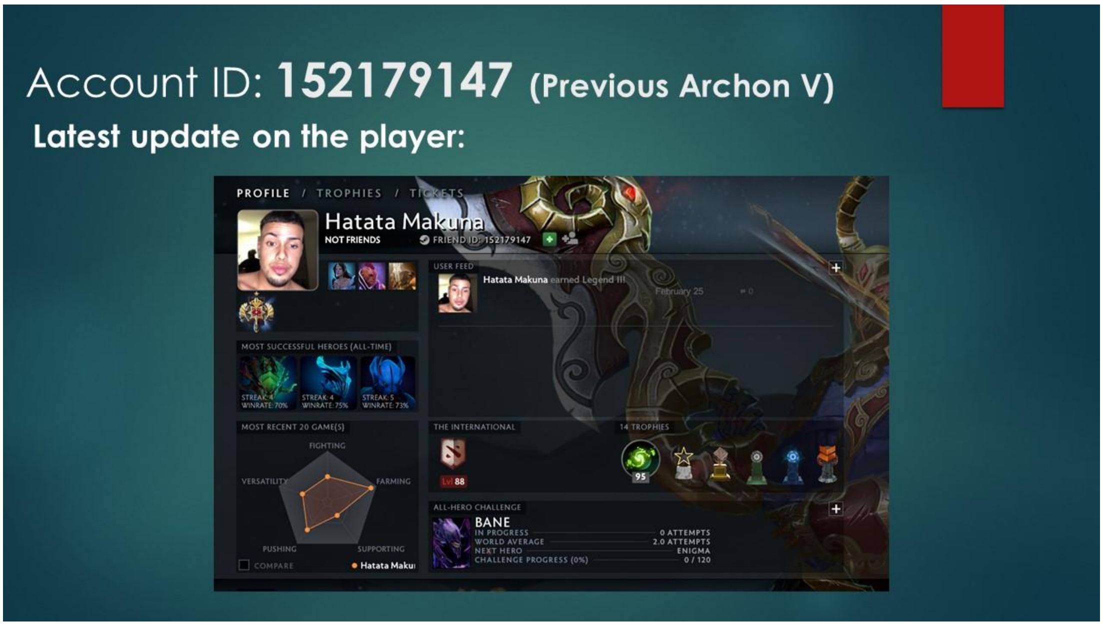


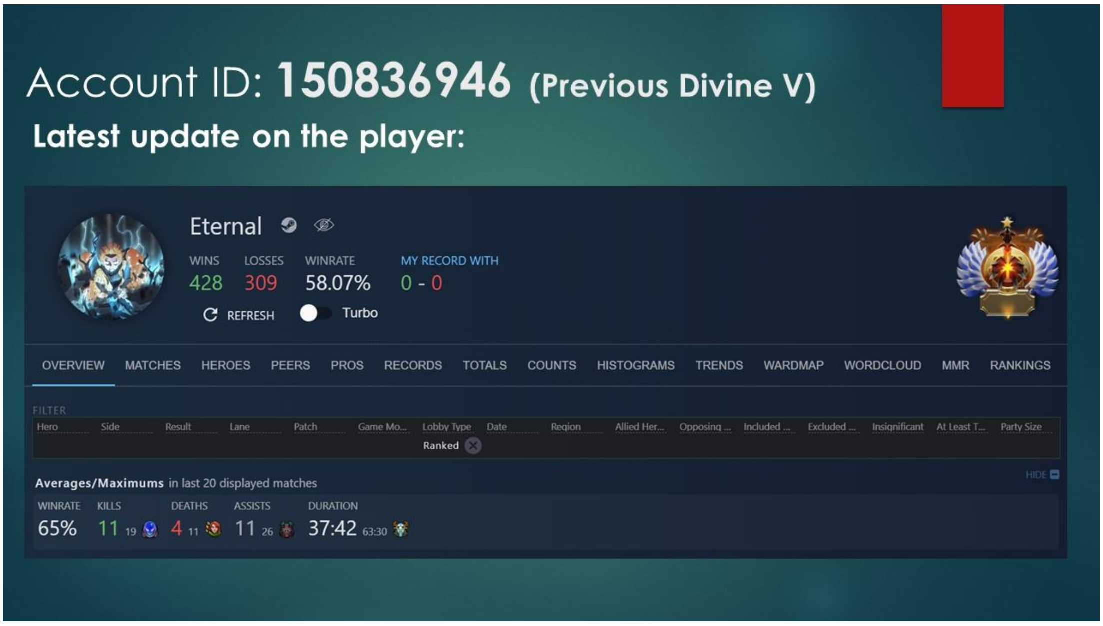


Source: OpenDota (2021)

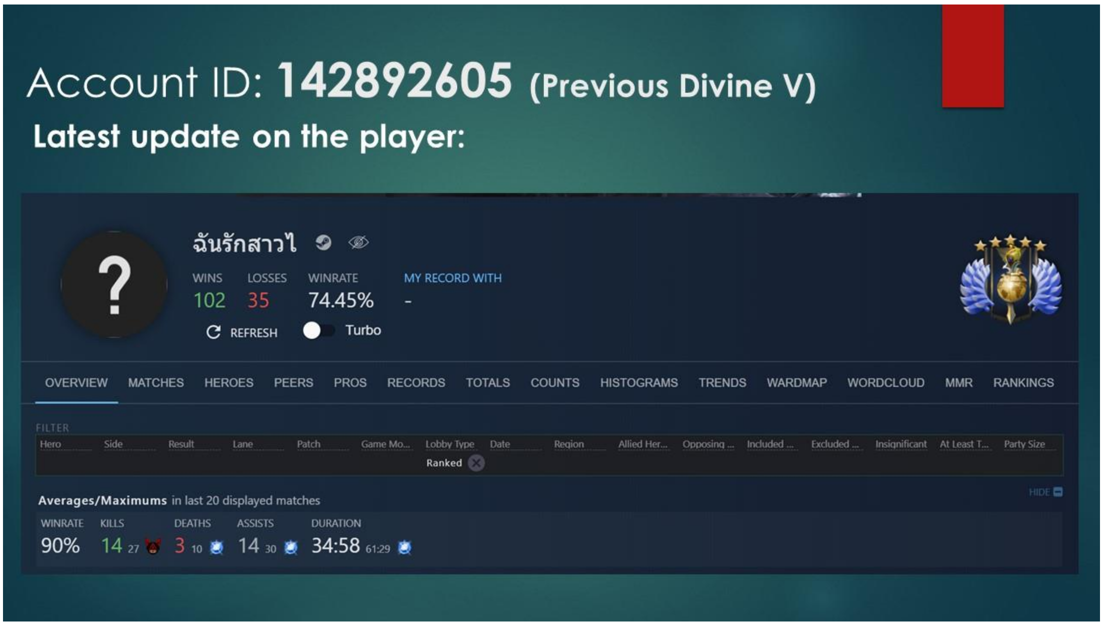


Source: OpenDota (2021)

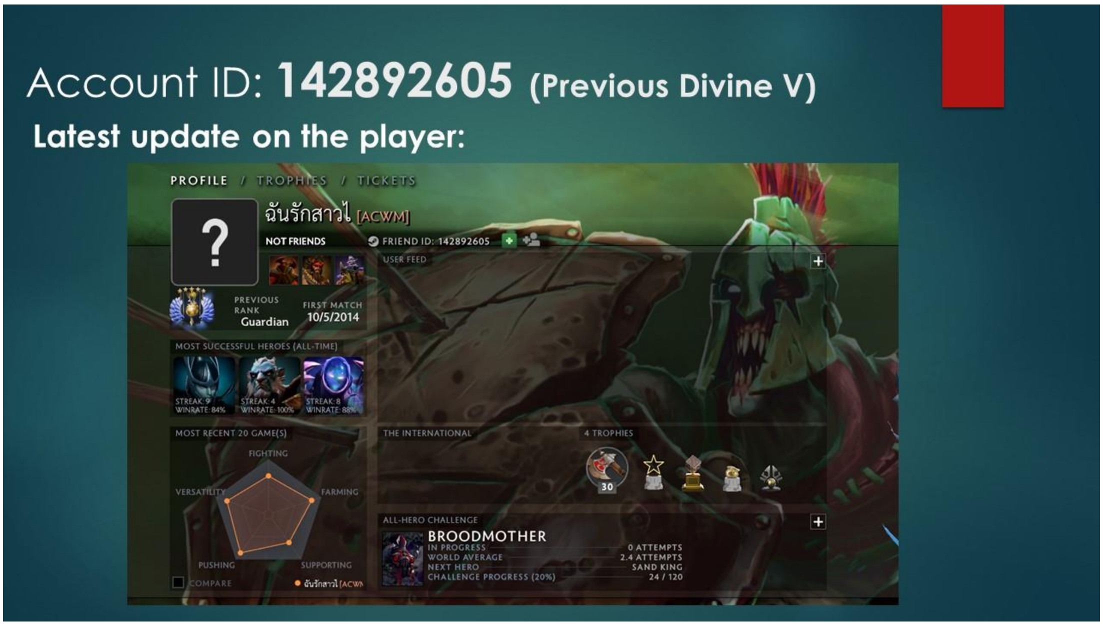


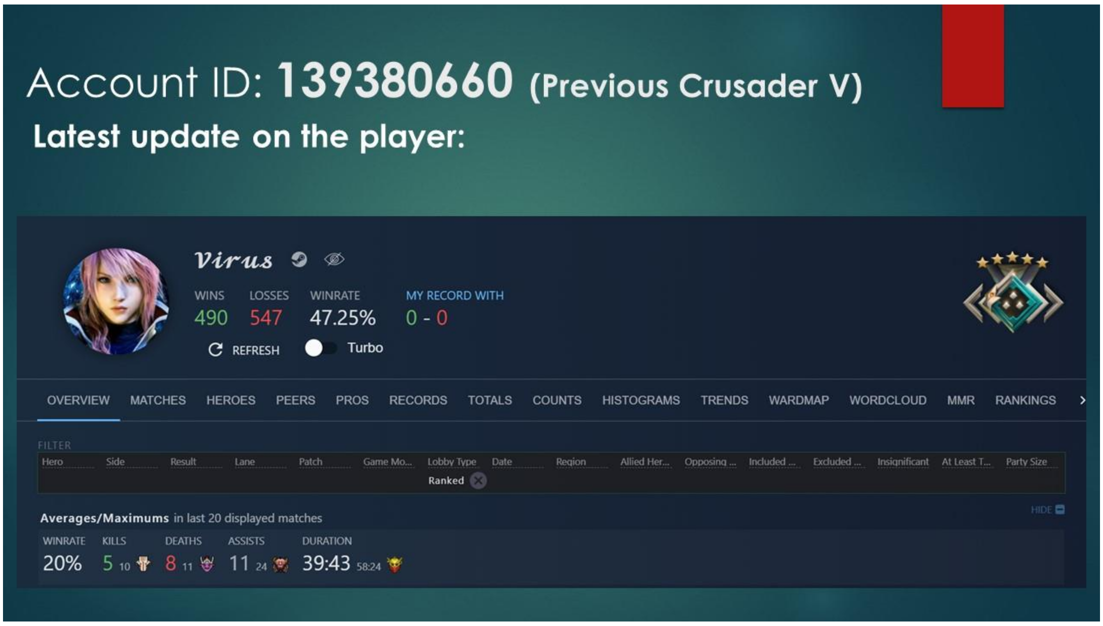


Source: OpenDota (2021)

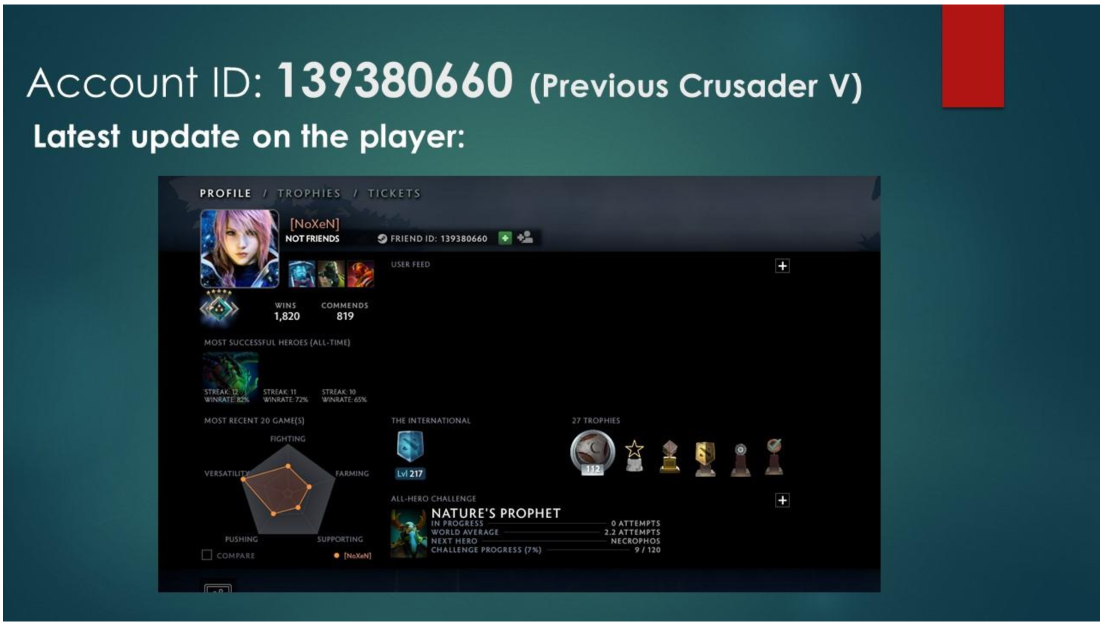


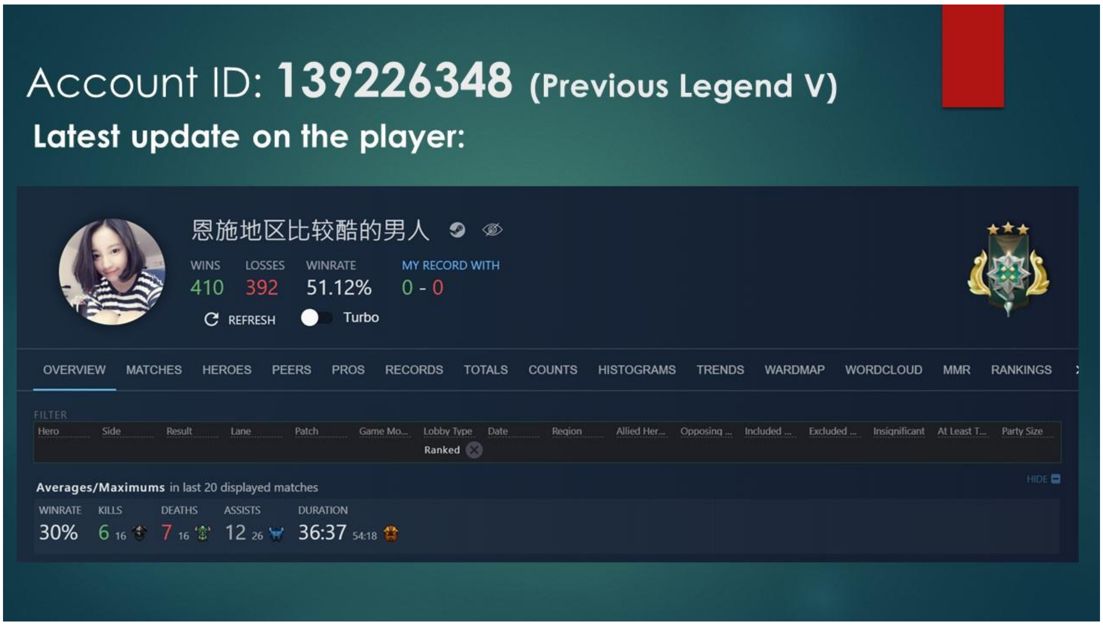
Source: OpenDota (2021)



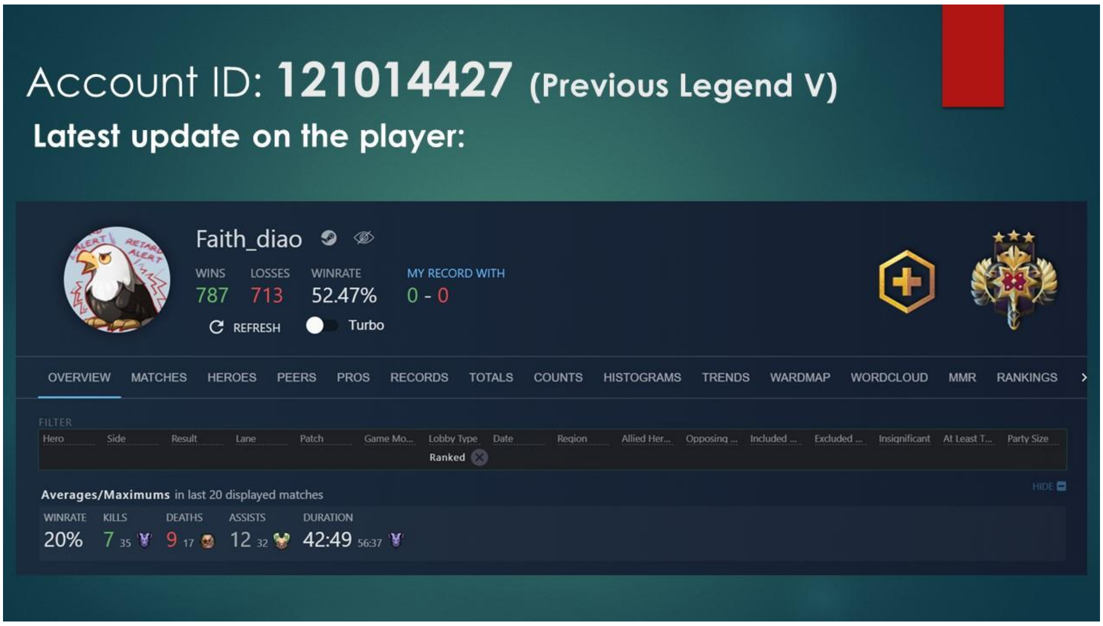


Source: OpenDota (2021)

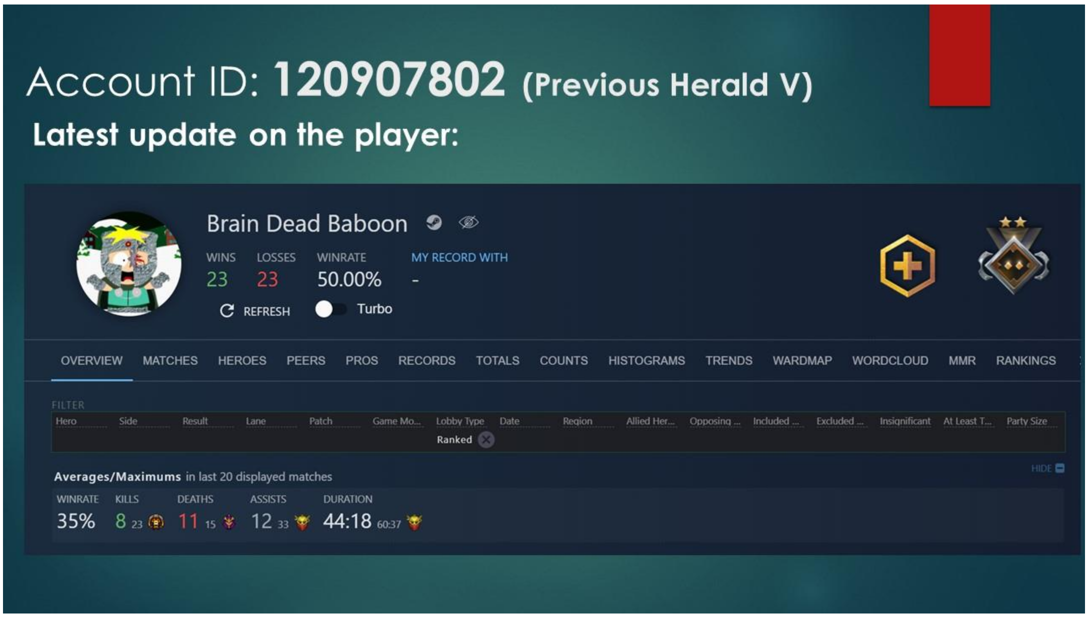




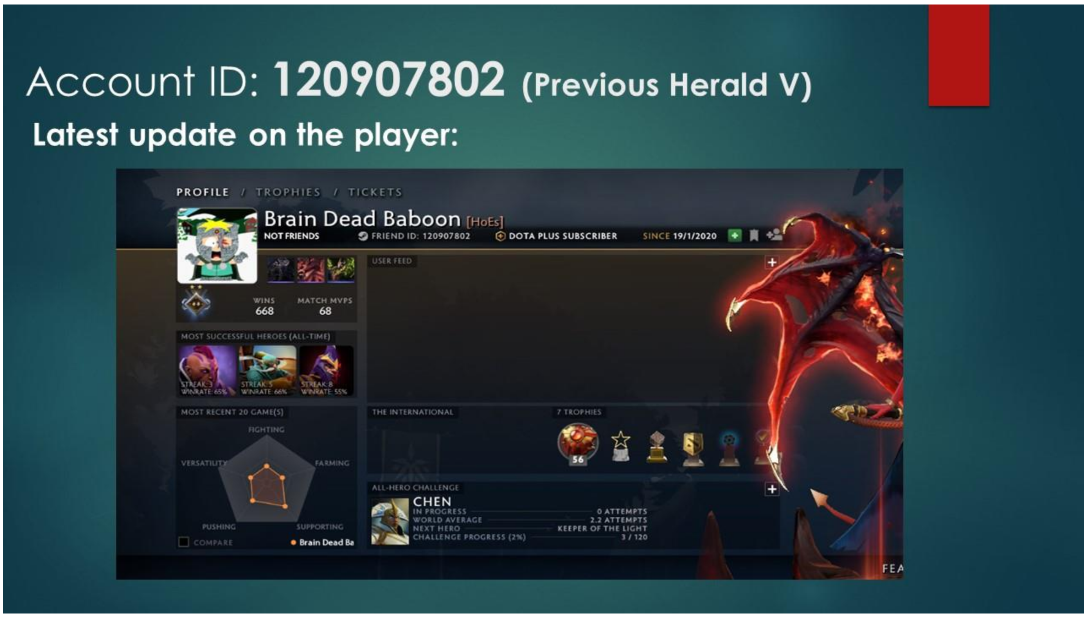
Source: OpenDota (2021)

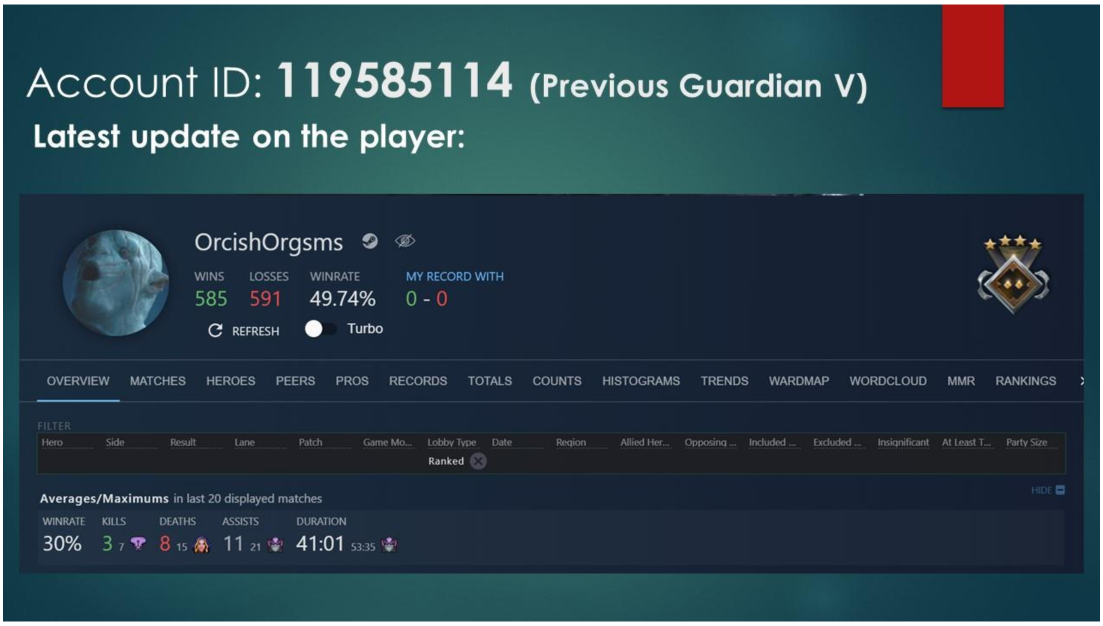


Source: OpenDota (2021)

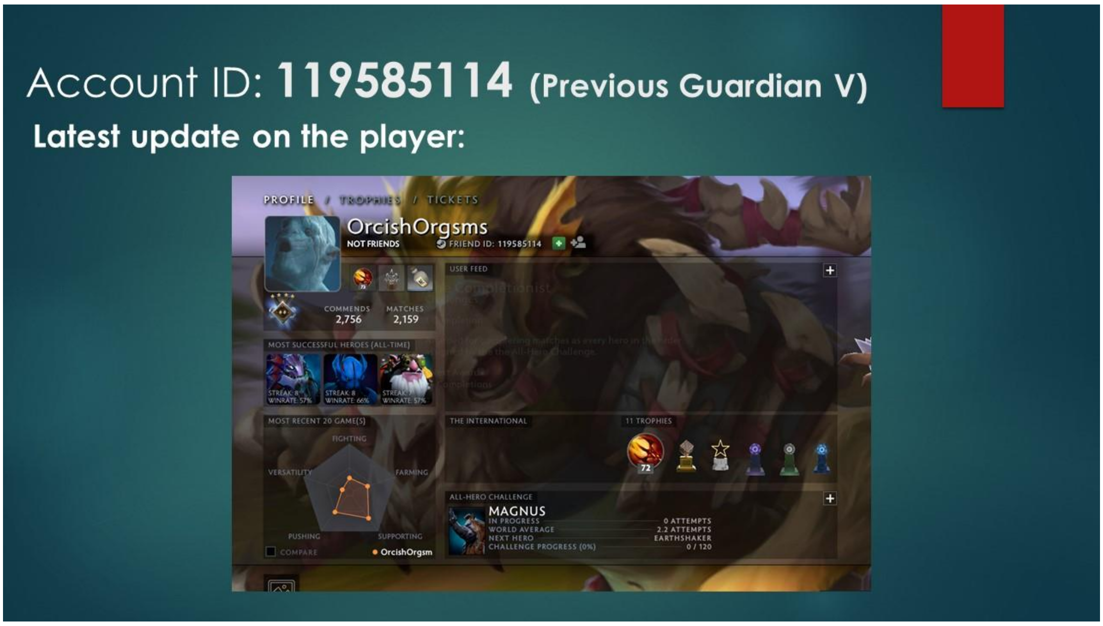


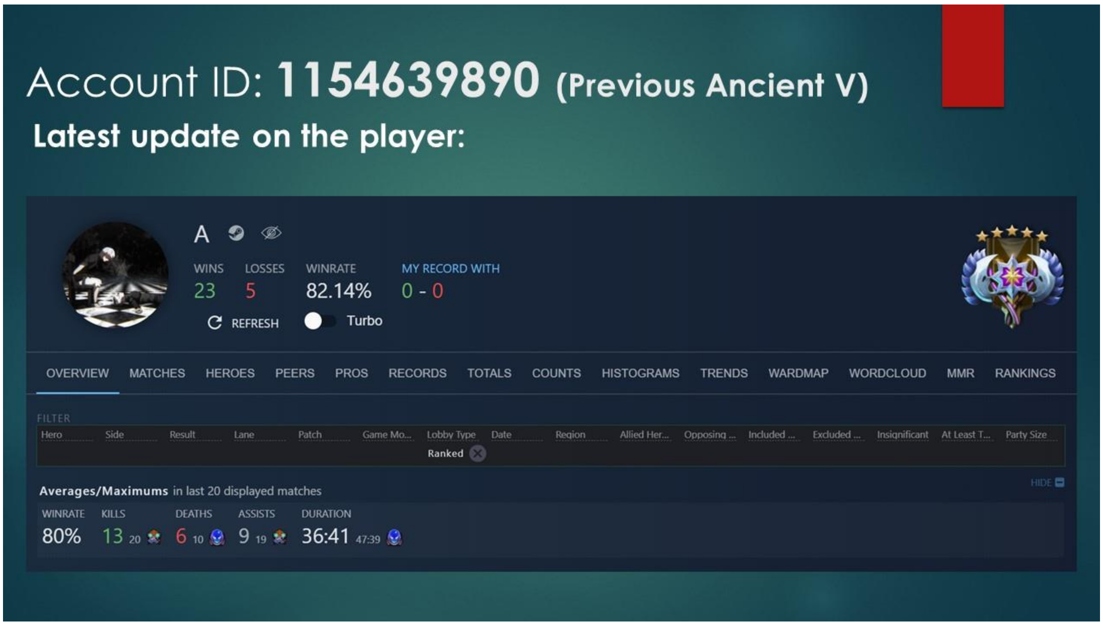
Source: OpenDota (2021)



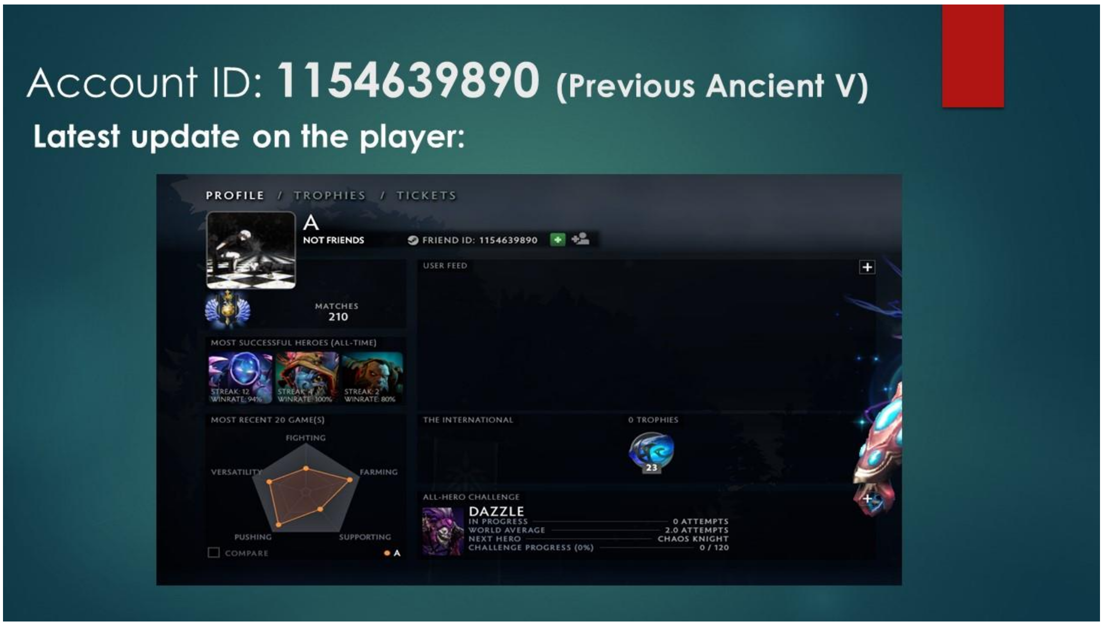


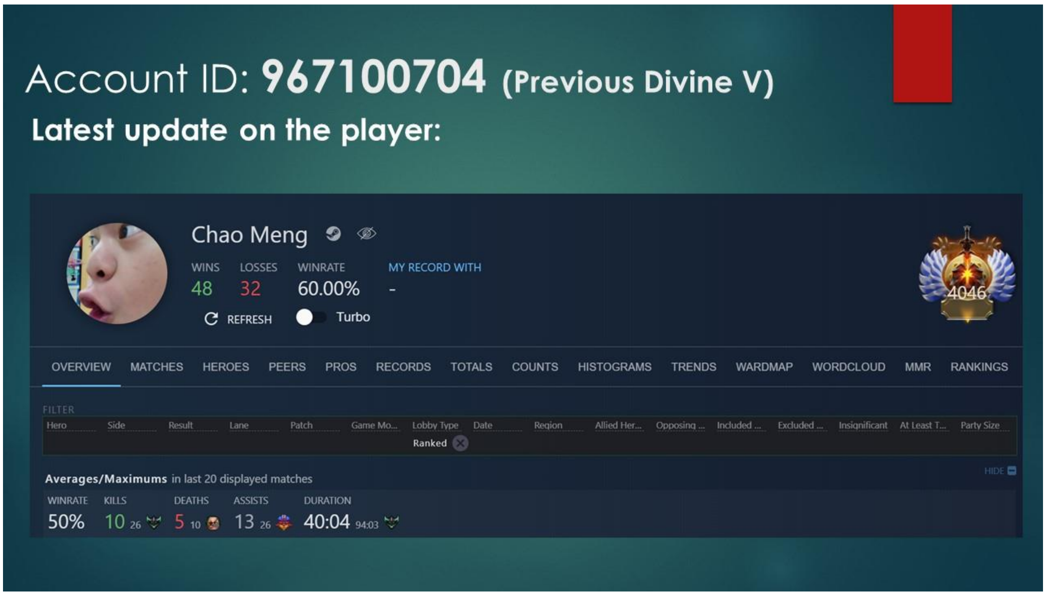
Source: OpenDota (2021)



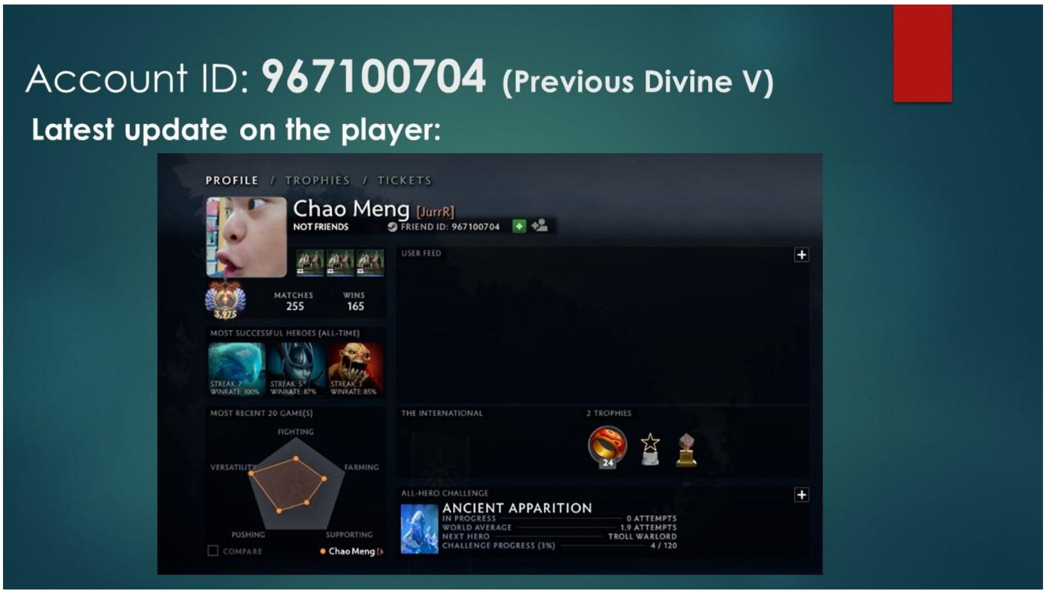


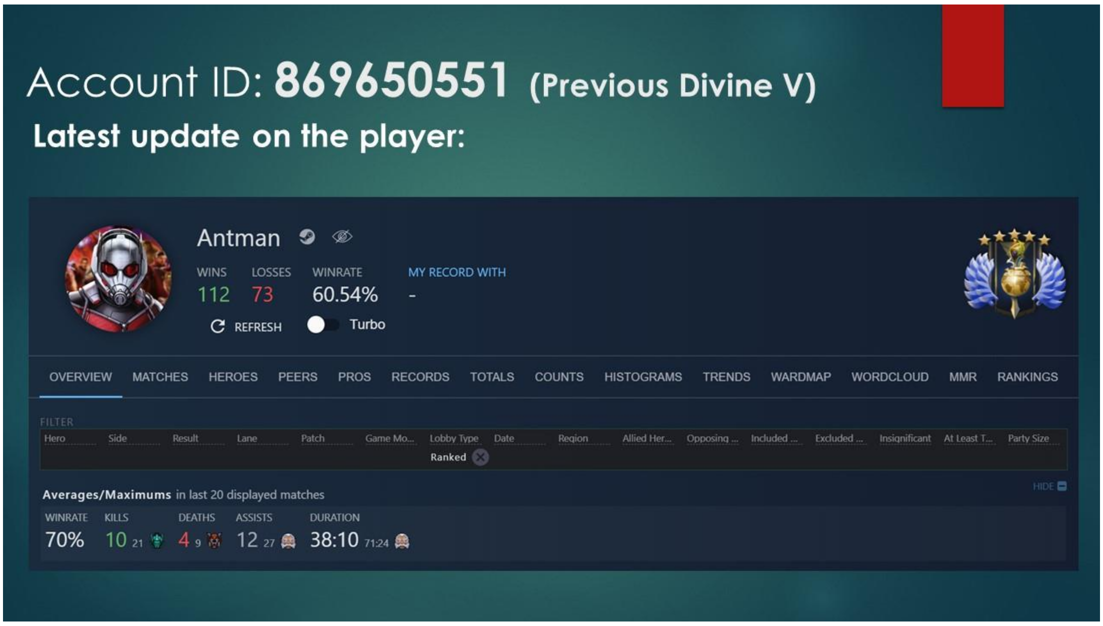
Source: OpenDota (2021)



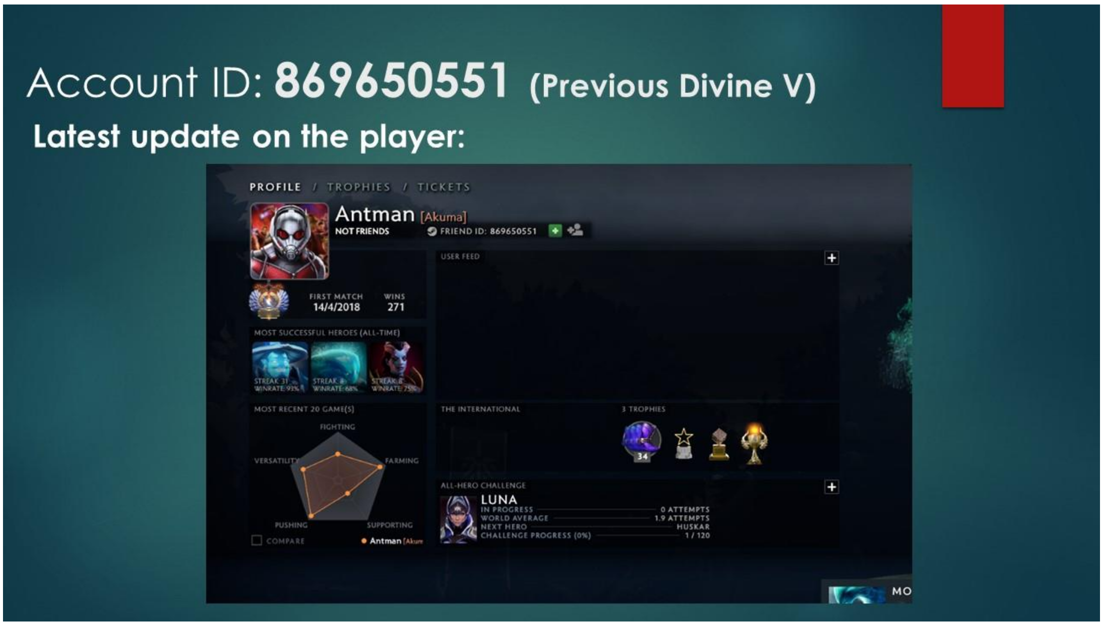


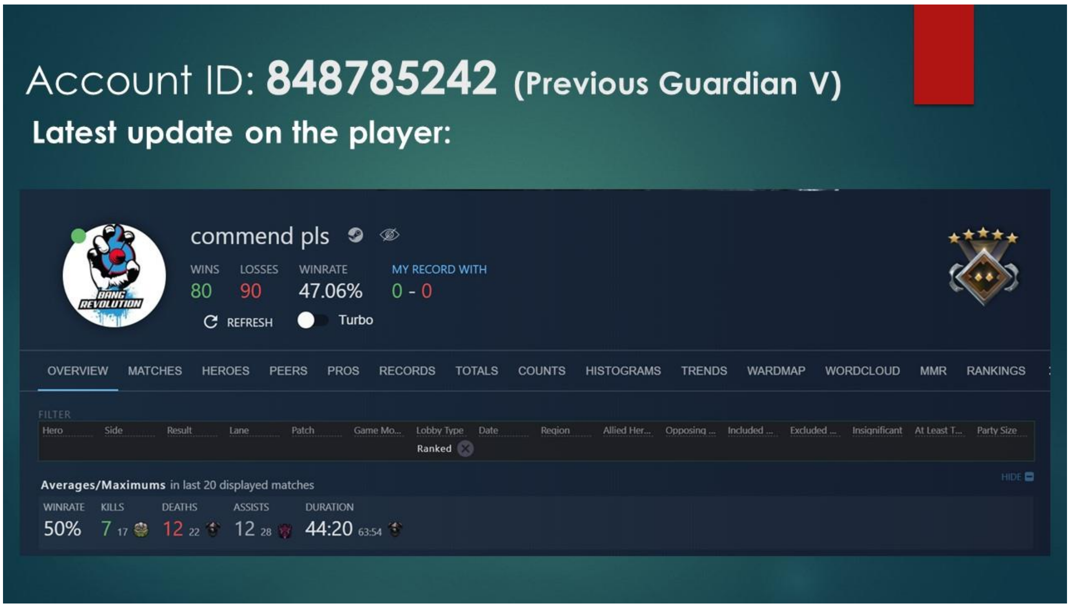
Source: OpenDota (2021)



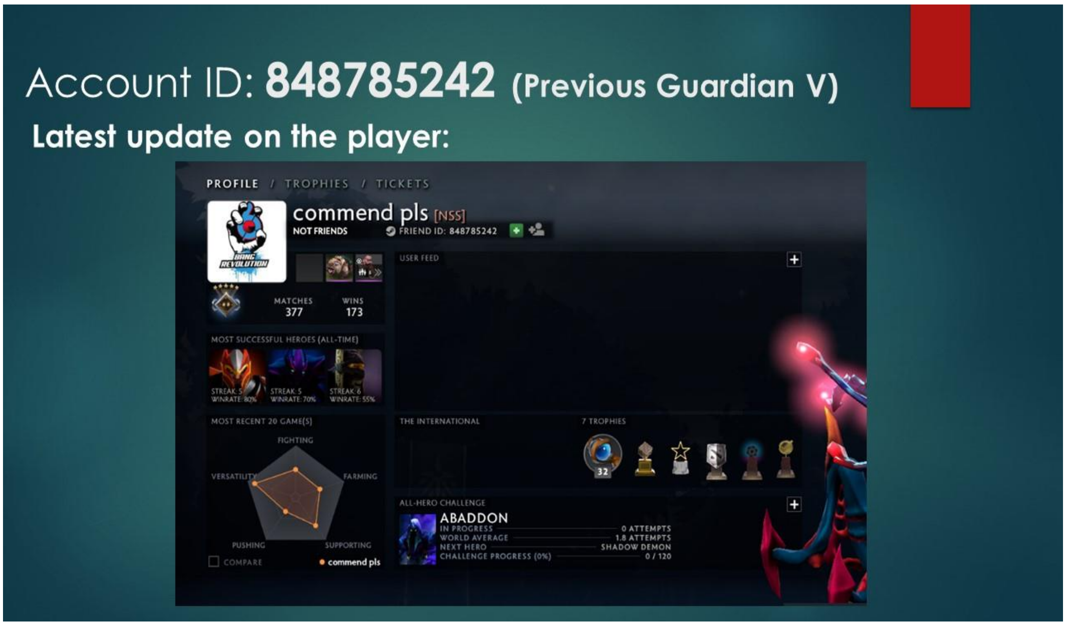


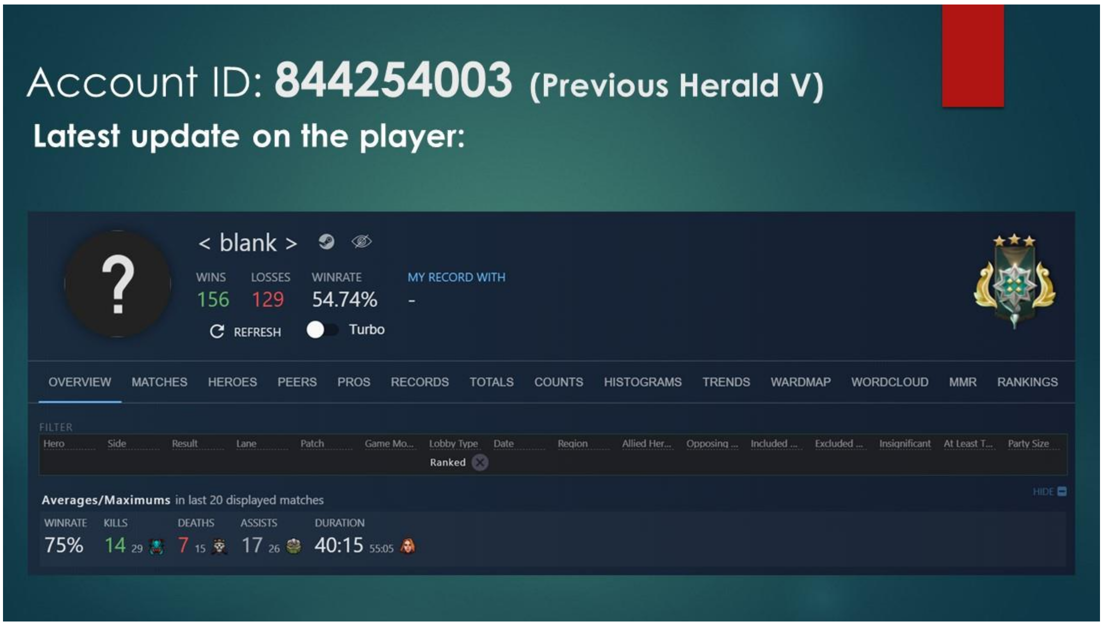
Source: OpenDota (2021)



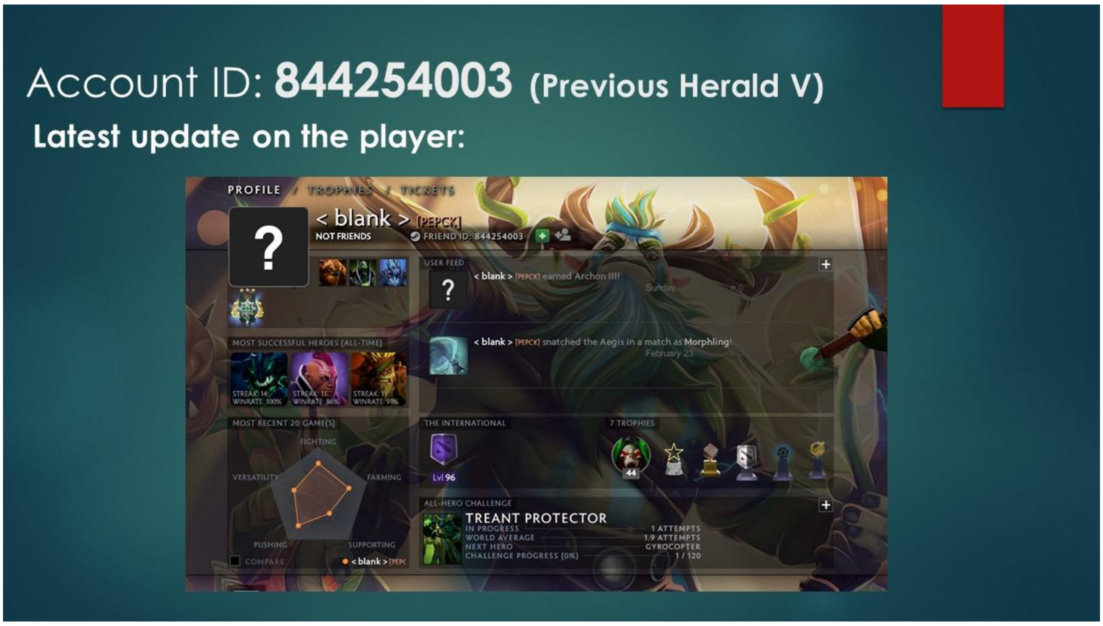


Source: OpenDota (2021)





Source: OpenDota (2021)



APPENDIX G PROFILE VERIFICATION FEEDBACK

Medal	Total Matches	Overall winrate	Winrate in 20 Match	Average Kills	Average Deaths	Average KDA Ratio	Average GPM	Average XPM	O	W	L	majority	smurf
Ancient V	3500	49.69	55	5.05	8.9	1.8	297.	401.05	0	1	0	0	0
Archon V	2182	49.22	55	7.05	6.15	2.7	449.55	599.3	0	1	0	0	0
Ancient V	4400	51.77	50	6.45	7.3	2.45	406.95	534.25	0	0	0	0	0
Divine V	727	65.89	60	10.85	5.85	3.55	604.25	720.5	1	0	1	1	1
Archon V	379	56.2	55	15.25	7.25	5.65	650.3	764.95	0	0	1	0	1
Guardian V	303	53.47	75	16.9	4.95	10.65	746.55	844.4	1	1	1	1	1
Crusader V	1370	49.49	50	7.9	8.6	2.05	437.85	601.45	0	0	0	0	0
Divine V	4656	50.3	65	10.6	5	6	612.3	714.85	1	1	0	1	1
Legend V	3922	49.31	45	8.35	8.15	2.1	463.4	636.55	0	0	0	0	0
Divine V	1571	53.34	80	14.4	3.25	9.15	634.1	765.75	1	1	1	1	1
Guardian V	1168	49.83	50	2.55	8.65	1.85	316.75	469.15	0	0	0	0	0
Herald V	45	48.89	65	8.3	7.55	2.7	429.45	573.6	0	1	0	0	0
Legend V	1483	53	30	9.7	9.1	3	486.2	630.15	0	1	0	0	0
Legend V	740	52.3	30	6.45	7.4	2.7	424.55	572.45	0	1	0	0	0
Crusader V	1036	47.3	20	5.3	8.15	1.2	336.2	469	0	1	0	0	0
Divine V	136	74.26	90	14.1	3.25	9.2	677.55	782.3	1	1	1	1	1
Divine V	737	58.07	65	11.25	3.65	8.2	736.05	719.7	1	1	1	1	1
Archon V	3149	50.33	55	6.85	6.65	2.8	489.45	607.9	0	1	0	0	0
Archon V	168	47.62	85	15.6	4.8	8	569.95	744	1	1	1	1	1
Legend V	3157	52.52	55	5.75	8.8	1.6	356.8	498.4	0	0	0	0	0
Ancient V	3115	52.23	45	3.95	8.95	2	301.9	468.7	0	1	0	0	0
Divine V	4098	51.68	85	15.4	4.55	5.8	675.35	777.55	1	1	1	1	1
Crusader V	392	47.96	65	14.65	3.75	7.75	576.45	774.6	1	1	0	1	1
Divine V	2254	52.35	35	10.15	8.65	2.35	542.55	678.2	0	1	0	0	0
Legend V	1881	50.08	25	6.55	7	2.35	452.3	577.85	0	1	0	0	0
Guardian V	1775	50.54	55	8.4	6.35	2.75	478.1	687.3	0	0	0	0	1
Divine V	1646	51.34	25	3.65	7.25	1.75	456.1	488.25	0	1	0	0	0
Archon V	1238	51.29	45	6.5	6.2	2.45	460.55	595.85	0	1	0	0	0
Crusader V	164	60.98	65	17.25	6.05	7.8	611.95	760.4	1	1	1	1	1
Guardian V	175	51.43	85	10.5	3.55	8	600.45	671.2	1	1	1	1	1
Archon V	3581	50.82	50	5.65	5.65	3.5	389	532.8	0	0	0	0	0
Guardian V	692	53.47	50	5.75	9.05	2.6	371.05	551.4	0	0	0	0	0
Divine V	252	64.68	90	11.45	2.45	9	749.5	817.75	1	1	1	1	1
Legend V	164	68.9	55	12.2	4.55	6.05	608.9	743.9	1	1	1	1	1
Divine V	376	60.9	60	15.5	6.3	4.65	659.6	760.45	1	0	1	1	1
Herald V	240	50.42	80	23.1	4.7	9.15	714.1	921.75	1	1	1	1	1
Guardian V	170	47.06	50	7.2	11.55	1.3	428.8	614.85	0	0	0	0	0
Divine V	185	60.54	70	11.65	4.25	5.65	644.55	740.8	1	1	1	1	1
Divine V	65	63.08	65	15.05	4.75	4.75	641.4	753.75	1	0	1	1	1
Ancient V	28	82.14	90	13.5	4.95	4.8	797.85	808.15	1	1	1	1	1

This table records all the players that were sent to the reviewers and their feedback compared to our research result. Column O refers to Ohaiyo's feedback, column W refers to Winter's feedback while column L refers to Mr. Loh's feedback. The rows that are highlighted in yellow are the rows with opposing majority and smurf values.

APPENDIX H 20 MATCHES OF PLAYER 2

HERO	RESULT	GAME MODE	DURATION	K	D	A
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft Unknown Skill	39:55 Radiant	4	7	3
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft Normal Skill	47:50 Radiant	15	13	6
Zeus 4 months ago	Won Match Ranked	All Draft Normal Skill	61:16 Radiant	16	11	31
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft Normal Skill	35:12 Radiant	7	3	7
Storm Spirit 4 months ago	Won Match Ranked	All Draft Normal Skill	33:52 Dire	8	6	15
Winter Wyver... 4 months ago	Won Match Ranked	All Draft Normal Skill	36:11 Dire	12	4	16
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft Normal Skill	36:17 Dire	3	9	12
Winter Wyver... 4 months ago	Won Match Ranked	All Draft Normal Skill	35:49 Dire	10	4	9
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft Normal Skill	45:29 Radiant	5	8	8
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft x2 Normal Skill	34:18 Radiant	0	6	4
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft Normal Skill	41:56 Radiant	5	9	13
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft Normal Skill	41:15 Radiant	10	8	17
Winter Wyver... 4 months ago	Lost Match Ranked	All Draft Normal Skill	33:41 Radiant	5	6	1
Winter Wyver... 4 months ago	Won Match Ranked	All Draft Normal Skill	39:52 Dire	10	5	19
Winter Wyver... 4 months ago	Won Match Ranked	All Draft Normal Skill	33:08 Radiant	7	1	16
Winter Wyver... 4 months ago	Won Match Ranked	All Draft Normal Skill	48:50 Radiant	9	7	13
Shadow Sham... 4 months ago	Won Match Ranked	All Draft Normal Skill	31:41 Dire	5	4	16
Winter Wyver... 4 months ago	Won Match Ranked	All Draft Normal Skill	32:53 Radiant	7	3	10
Winter Wyver... 4 months ago	Won Match Ranked	All Draft Normal Skill	36:24 Dire	13	7	5
Winter Wyver... 4 months ago	Won Match Ranked	All Draft Normal Skill	38:24 Radiant	17	6	11

Source: OpenDota (2021)