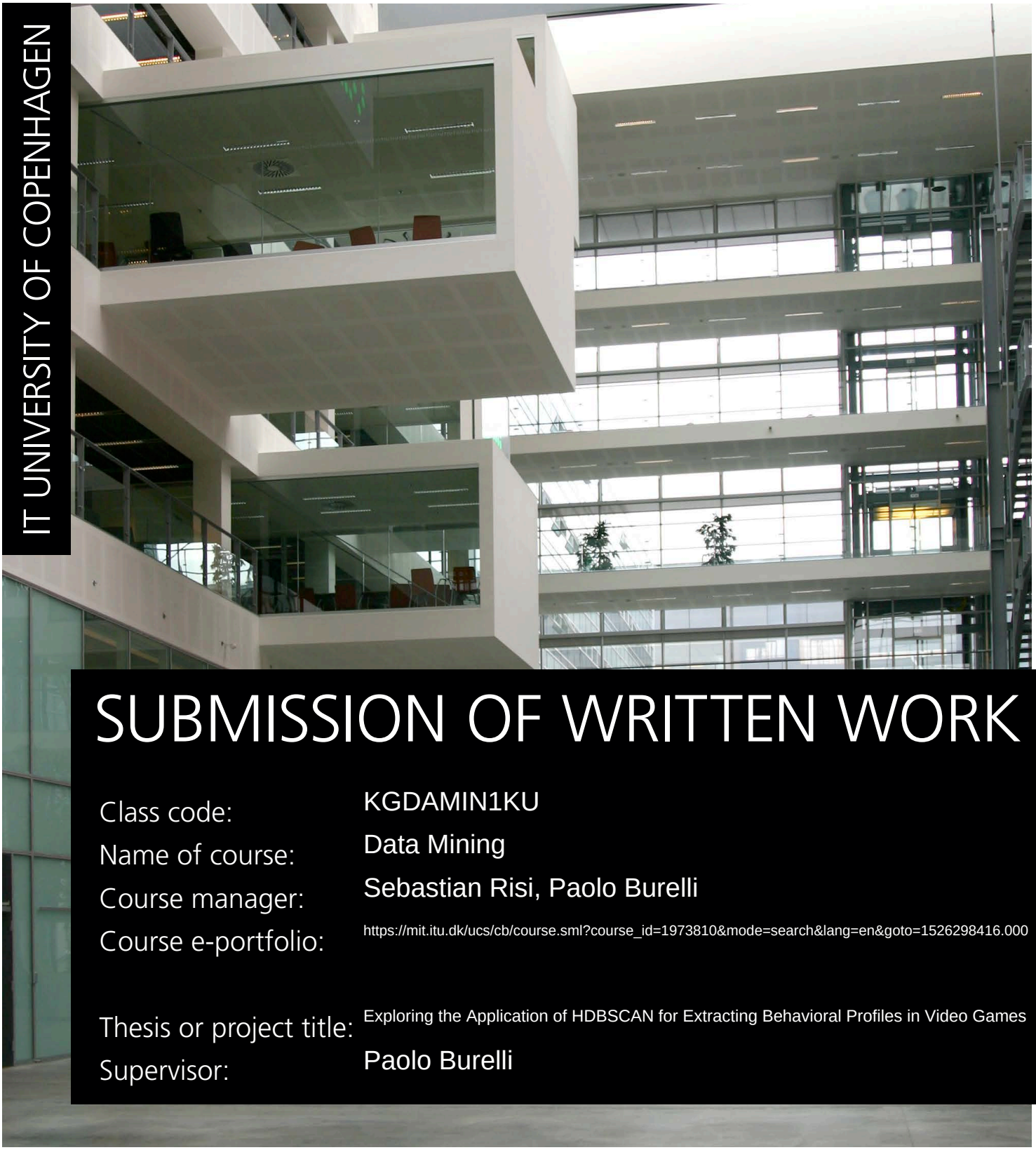
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code:	KGDAMIN1KU
Name of course:	Data Mining
Course manager:	Sebastian Risi, Paolo Burelli
Course e-portfolio:	https://mit.itu.dk/ucs/cb/course.sml?course_id=1973810&mode=search&lang=en&goto=1526298416.000
Thesis or project title:	Exploring the Application of HDBSCAN for Extracting Behavioral Profiles in Video Games
Supervisor:	Paolo Burelli

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Luka Locniskar	14/09/1992	lulo@itu.dk
2.	Martin Sorensen	03/02/1992	mkso@itu.dk
3.	Andreas Feldstedt	01/07/1994	anfe@itu.dk
4.	Jonas Tingmose	14/10/1983	joti@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk

Exploring the Application of HDBSCAN for Extracting Behavioral Profiles in Video Games

Luka Locniskar
Email: lulo@itu.dk

Martin Sørensen
Email: mkso@itu.dk

Andreas Feldstedt
Email: anfe@itu.dk

Jonas Tingmose
Email: joti@itu.dk

ABSTRACT

In the present paper we explore how the clustering method known as *Hierarchical Density-Based Spatial Clustering of Applications with Noise* (HDBSCAN) may be applied for extracting behavioral profiles from behavioral telemetry in video games. Behavioral profiles have been argued to be vital for several stakeholders in a game development context, and this naturally leaves an interest to accurately and satisfactorily establish said profiles. Hence, in this paper we will attempt to apply an HDBSCAN to a dataset consisting of certain behavioral data from the video game "PlayerUnkown's Battleground" (aka. PUBG). Our results indicates that it is likely that behavioral profiles may extractable from behavioral telemetry, but it may be a complication and computational procedure to do so. The algorithm appears to weight certain dimensions, which in turn may be all-defining towards the end result, However, we believe this result to be highly correlated to the dataset applied, alongside how the dataset was analyzed. Further work is required to more accurately apply the HDBSCAN, and direct measures for comparing the algorithms is required to deduct a more conclusive result.

I. INTRODUCTION AND BACKGROUND

As it has been voiced for by e.g. Drachen et al. [7][6][6][4], Bauckhage et al. [8], Yannakakis and Togelius [9], El-Nasr et al. [10][11], and Sifa et al. [12], the introduction of data telemetry in video games has lead a significant increase in the availability of, among others, quantitative behavioral data about players. The logging of data could easily exceed terabytes of data, with several million data objects along several dimensions [7][11]. In essence, this leaves the game developers with an interesting potential to tap crucial knowledge about their players, but the vastness of data may additionally complicates the extraction of actionable and usable information [8]. As such, knowledge extraction in the context of games is similar to any other kind of big data mining endeavor [9], as the algorithms applicability to large dataset is faced with several potential problems and considerations, which is usually associated with data mining algorithms. These include including (among others): scalability, dealings with noise and different attribute types, arbitrarily shapes etc. [9][13].

Although that within the field of *game analytics* - that being "the application of analytics to game development and research" [11, p. 5] - the act of applying the concepts of data mining to game telemetry (referred to as *game data mining*) may aid several different stakeholders[5][10]. Nevertheless, El-Nasr et al. [11] argues that "the introduction of analytics in game development has, to a significant extent, been driven by the need to gain better knowledge about the users the players"[11, p. 4]. One particular field of interest with respect herewith is the extraction of behavioral profiles based on game behavioral telemetry, i.e. "telemetry data about how people play games" [5, p. 206]. The concept of establishing insightful and beneficial behavioral profiles based on pattern discovery in the behavioral data, and is currently considered a key challenge of game analytics [12]. As argued for previously, the behavioral profiles may benefit several stakeholders within the game development team, and aid in e.g. ensuring optimal play experience, game design, discovering potential players and game testers, monetization, player retention, determining unwanted play etc. [5][8][12].

As we shall be uncovering the the upcoming sections, previous attempts in extracting behavioral profiles using unsupervised clustering methods have revealed several considerations with respect to which clustering methods needs to be applied and how. However, the advancements in the field of data mining is still ever growing, and thus newer and more optimal algorithms are being developed [13][18].

Given this acknowledgement, we will in the present paper attempt to examine whether a certain clustering method - known as Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)[18] - may be satisfactorily applied for establishing behavioral profiles from player telemetry. As will be argued for in the related works, this decision is based on a recognized of a lack of applying alternative clustering methods with respect to profiling. Hence, we will initialize in the coming by presenting an overview of the related work within the field of establishing behavioral profiles using cluster analysis. Following this, we will present a brief outline of the general concepts that make up cluster analysis, alongside an outline for the underlying idea of HDBSCAN. Based on this, we will present our methodological approach for applying HDBSCAN to a real-life dataset, consisting of certain behavioral data from the immensely popular "PlayersUnknown's Battleground" (PUBG)[24]. Lastly, the results will be presented, discussed and concluded.

II. RELATED WORK

As it was claimed during the introduction, it is crucial for game developers to know their players in order to gain

actionable insight on, for example, how the development of the game must take it's course or the game must be monetized[7][4][8]. One aspect of this relies on the extraction of behavioral profiles from game telemetry. Let it be noted here that behavioral profiles may - among others - be referred to as player models, player types, player profiles, and player classification ([10][4][22][9][14]), with a very similar or identical meaning. However, we shall for the present paper be using *behavioral profiles* as an all encompassing term that revolves around "identifying groups for players with similar behaviors and identify the most important behavioral features in terms of the underlying patterns in the dataset" [15, p. 93].

Having acknowledged this, it becomes evident that despite "user profiling via data mining is a relatively new endeavor in game development and research"[7, p. 2], a fair amount of research has been conducted within the given context. To review all research would be out of scope, and thus we refer to [12] for a relatively recent comprehensive collection of work in said field of interest.

Nevertheless, Drachen et al. [4] attempted to apply different clustering techniques to behavioral data in an attempt to develop behavioral profiles of players in a popular online shooter game. It was concluded that different sorts of clustering algorithms may yield "different insights into the underlying behavior"[4, p. 1] of the players, and thus care must be taken towards selecting appropriate algorithms. In another paper, Drachen et al.[3] similarly attempted to extract player models from behavioral data gathered from an adventure game by applying self-organizing maps (SOM). They suggest that patterns of similar play style can be detected by said technique and thus player models extracted. However, they additionally voice that care must be taken towards selecting features, as they are defining towards the end-result; they advice that features should be central to the playing characteristics of the core game mechanics. Pirker et al. [15] also attempts to conduct profiling, however, with an increased focus on progression within the game. The authors shows that Archetypical Analysis (AA) for player profiling along with progression analysis provides a frame for how players may be grouped w.r.t. their progress in the game. Drachen et al. [7] attempts to compare different methods for player clustering via behavioral telemetry; methods included k-Means, Non-negative factorization (NMF), Principle Component Analysis (PCA), and AA via Simple Volume Maximization (SIVM). The authors argues that although all algorithms provides categorizations of players, the interpretability (and thus usefulness) varies between them, with AA being the most useful. Similarly, Thurau and Drachen [23] compares k-Means, c-Means, NMF, PCA and AA, and ultimately advocates for the usage of AA over other clustering methods, as AA seemingly provides a more interpretable result. Bauckhage et al. [8] argues for a general lack of knowledge of how to satisfactorily apply clustering techniques on behavior data. Hence, the authors provides a comprehensive review of several clustering algorithms, and examples are shown on how these may be applied to behavioral data. Ramirez-Cano et al. [16] proposes the usage of a meta-clustering approach for player classification, in which several levels of handling the data is being applied. They argue their approach solves some of the problems associated with clustering player behaviors. A particular note they make is that future work should be attempt to replace the k-Means algorithm with more advanced algorithms that may handle densely clustered area, such as Density Based Spatial Clustering of Applications with Noise (DBSCAN). Francillette et al. [17] attempts to cluster players as a mean for increasing the game experience by providing optimal matches between the players. Based on the players recorded actions, a DBSCAN clusters the players to increase in-game experiences. Their reasoning for utilizing the DBSCAN is due to it being "a robust clustering algorithm and it has the advantage that it need not to give the number of cluster as input parameter." [17, p. 233]; a crucial acknowledgement when exploring a dataset for patterns.

An interesting aspect to consider is the rather scares usage of certain algorithms for clustering player behaviors, and moreover the lack of comparison between different algorithms. The reviewed literature presented here appears to highly rely on certain clustering methods, with little to no acknowledgement to other potential applicable methods, despite the rather large array of other algorithms available - see e.g. Han et al. [13], Yannakakis & Togelius [9], and Sifa et al. [12]. Although the more uncommonly algorithms are mentioned in the context of establishing behavioral profiles - for example by Bauckhage et al. [8], Sifa et al. [12], Yannakakis & Togelius [9], and [7] - it appears that they are being neglected in favor of comparing or utilizing other and more common algorithms (e.g. k-Means/c-means and AA). Hence, this provides a natural interest to discover how other clustering algorithms performs with respect to extracting behavioral profile. In this acknowledgement, we have taken a particular interest for discovering the applicability and usability of the algorithm Hierarchical Density Based Spatial Clustering of Applications with Noise (HDBSCAN) [18]. HDBSCAN has been chosen as - to the extent of our knowledge and research - no utilization of the algorithm in the context of behavioral profiles exists. Given some of the evident potential advantages of HDBSCAN over, for example, the very popular k-Means algorithm (see e.g. [18][20][19][21]), it shows potential for extracting usable and interpretable behavioral profiles. Therefore, for the remaining of this paper we shall have an increased focus on the application of said algorithm.

III. BASICS OF CLUSTER ANALYSIS AND HDBSCAN

Before examining the underlying concepts that make HDBSCAN, we shall initialize this section with a brief overview of clustering as a general concept.

Rather simplified, *cluster analysis* (or more commonly referred to simply as *clustering*) is an unsupervised learning technique and its core in "the process of partitioning a set of data objects (or observations) into subsets."[13, p. 444], where each of said subsets are considered clusters. For each

cluster there exist a high similarity between objects within the cluster, and a high dissimilarity between each respective cluster. Hence, the overall concept of clustering relies on reducing the dimensionality of the dataset such that meaningful and actionable results can be extracted from otherwise more abstract and higher dimensional data [5]. As recognitized by clustering being an unsupervised method, clustering is based on the concept of learning by observation, rather than learning by example [13]. As the class label for each class (i.e. each cluster) is unknown, cluster analysis assess the dissimilarities and similarities of the attribute values describing each objects (often based on a distance measurement), from which point the clusters are formed [13]. This entities cluster analysis to be considered exploratory, as the clusters needs to be discovered, rather than be determined a priori, as is the case with another common (and similar) data mining concept known as classification [13]. However, we will not be covering the similarities and dissimilarities between the two data mining concepts.

There exists multiple clustering techniques for how the clusters are detected, and are primarily "dissimilar in the way they define what a cluster is and how they form it" [9, p. 78]. The clustering algorithms can, however, arguably be split into four main categories: (1) partitioning-, (2) hierarchical-, (3) density-based-, and (4) grid-based methods [13]. Although we shall not cover specifics of each family of clustering methods, let it be noted that the "clustering algorithm and its corresponding parameters, such as which distance function to use or the number of clusters to expect, depends on the aims of the study and the data available" [9, p. 78] and hence care must be taken towards selecting appropriate algorithms for one's study.

Moreover, clustering algorithms in general have several requirements that need to be considered [13]. To cover of these requirements is out of scope for the present paper, but the more prominent requirements include scalability, adaptability to different attribute types, capability of dealing with arbitrary shapes, noise handling, clustering of high dimensional data, interpretable and usability [13, p. 446-448].

A. OUTLINING HDBSCAN

As we argued for in the section on related works, HDBSCAN could provide an interesting alternative for extracting behavioral profiles from behavioral telemetry. However, as claimed by Bauckhage et al. [8], there exists pros and cons of any algorithm relative to any other algorithm with respect to any given study, but one natural advantage of density-based clusterings is the of detecting clusters of arbitrary shape, which e.g. partitioning and hierarchical methods struggle with [13]. Moreover, as it was stated for by e.g. Yannakakis & Togelius [9], Bauckhage et al. [8], and [7], knowledge on how to satisfactorily applying the clustering algorithms and its associated parameters is paramount towards the usefulness and interpretability of the result. If a clustering technique is inaccurately applied, the results will be biased, and in the worst case be misleading and ultimately useless [8]. Therefore, in the following an outline for the general underlying ocncept of HDBSCAN will be provided.

HDBSCAN is a clustering method in the region of density-based clustering (as hinted by the name of the clustering method). The method is an extension and improvement over the previously mentioned and well-established DBSCAN[18][21][19]. Given here is that HDBSCAN discovers clusters based on the general idea of density-based clustering - that is, using the notion of density. Clusters will be formulated either based on the density of the neighborhood objects, or with respect to a density function[13]. To simplify the understanding of HDBSCAN, we shall start by examining DBSCAN. In the case of DBSCAN, clusters are formulated based on the *density* of an object, as given by the number of objects close to the object in question. Hence, DBSCAN "finds core objects, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters." [13, p. 471]. In essence, the algorithm is based on two user-defined parameters, that is, the E-*neighborhood* (where radius E thresholds the range the algorithm checks), and *MinPts* (the minimum density threshold for what is considered a dense region)[13][18]. The algorithm detects a cluster by searching through each object in a given dataset (starting at a random point), and for the object determine if the E-neighborhood for the object contains the minimum MinPts objects to form a cluster. If so, it is considered a *core object*; a cluster will be formed for the object, and all objects in the E-neighborhood is added to a candidate set. If its not a core object, it is considered a *noise object*. The algorithm iteratively assigns objects in the candidate set to the created cluster, given they do not belong to another cluster. The algorithm does so by labeling each object as "*unvisted*" until the object has been visited by the algorithm, and will thus subsequently be labeled as "*visited*". When completed, the algorithm will again randomly choose an unvisited object, and run the process once more. This continues until all objects have been visited by the algorithm [13].

This forms the basis of DBSCAN, and given that HDBSCAN is based on DBSCAN, it provides the basis of this algorithm, too. However, HDBSCAN hold several benefits over DBSCAN, and thus naturally differs in is clustering approach. The original DBSCAN suffers from the fact that it does "only provide a "flat" (i.e. non-hierarchical) labeling of the data objects, based on a global density threshold [MinPts, e.d.]" [18, p. 163]. In turn, DBSCAN will often not be capable of correctly characterize data sets containing clusters with varying and very different densities and/or nested clusters[18]. To go into complete details about the actualities of HDBSCAN is out of scope - we may refer to Campello et al. [18] and McInnes et al. [21] for this - however, HDBSCAN relies on the same basic concepts of detecting clusters, but does so in a hierarchical fashion. HDBSCAN extends DBSCAN using the concept of creating hierarchical clusters, and then extracting a flat clustering based on how "stable" the clusters is considered [21]. The intuitive explanation of the algorithm by McInnes et al. [21] explains that the HDBSCAN can be

considered as a 5 stage process consisting of: (1) transforming the space based on the density of the objects to extract a mutual reachability metric, and based on this metric (2) build a minimum spanning tree of the distance weighted graph. Then, (3) using the minimal spanning tree, the algorithm needs to form a cluster hierarchy of the connected components in the tree, and (4) this cluster hierarchy is condensed down, such that each cluster is constructed from a larger amount of data. Lastly, (5) the clusters are extracted based on the notion of stability which may be formed from the condensed cluster hierarchy. The stability essentially determines when a cluster is selected; the clusters of highest stability is chosen. The stability is simply speaking a result of whether the persistence of leaf nodes from a parent node is considered more "stable" compared to the parent. If so, the leaf notes are selected as clusters; otherwise, its parent node. The crucial aspect here is that once a decision for selecting a parent node has been made, none of the descending child nodes may be selected [21].

IV. METHODS

In this section, we will be describing the applied methods utilized in our endeavor to construct behavioral profiles from behavioral telemetry. It should come as no surprise that the behavioral profiles will ultimately be provided as a result of an HDBSCAN. However, in this section, we will additionally be explaining factors such as our dataset and the preprocessing of this, attribute and feature selections, dimensionality reduction, cluster evaluation, cluster method comparison etc.

A. DATASET AND PREPROCESSING

The data used for generating behavioral profiles was provided from Kaggle's 'PUBG Match and Death Statistics' dataset submitted by Kelvin Pei, which contains over 65 mio death entries for players[24]. This dataset contains two subsets, labeled 'aggregates' and 'deaths'. The aggregates contains information about the match type that was played, and overall performance qualifiers in that match. It has one row for each player participating in the match. The deaths file contains information about a player that had been killed, and the circumstances that player was killed under. With 65 million rows and 12 columns, the death data files collectively filled 7.2 gigabytes[1]. The aggregate data contains data on each player per match, causing it to contain more rows than the death data. Because of this, and since it had 14 columns, the file size of these were 9.8GB. The size of the aggregate in particular was inflated due to large amounts of unnecessary duplicate data. The file was created to contain both, player information, such and kill counts and distances traveled, and match information, such as the game mode and how many teams participated. Each row contained four columns of match data, which were entirely identical for all players participating in the same match, this could have been condensed into a singular match id column, had the match data been stored as a separate table.

In essence, this dataset was chosen as it provided a substantial amount of gathered behavioral telemetry from a very popular video game, essentially providing the ground for us to work on with respect to the endeavor we wished to undertake. Due to the immense size of the dataset and its telemetry based gathering, we expected the dataset to contain the difficulties usually associated with data mining undertakings (such as containing noise, outliers, missing values, and general other abnormalities). Hence, we believed it to provide a good representation of a real-life data mining undertaking.

ATTRIBUTE AND METRICS

Once the import of the data had been optimized, the following metrics were formulated, upon which the clustering was done.

Metrics	
Metric name	**Description**
Travel Ratio	The rate of traveling by foot compared to traveling by vehicle. A value of 1.0 described only traveling by foot.
Kill-Knockdown Ratio	The rate of kills compared to knockdowns. A value of 1.0 described killing as many players as was knocked down.
Kill Distance	The average distance between the player and all victims he had killed throughout the match.

TABLE I
THE METRICS CREATED TO DESCRIBE CORE PLAYER BEHAVIOR.

Aside from these metrics, there were also created a method of describing a player's favorite weapon numerically. The initial thought was to assign each weapon type with a value, but the value each weapon was assigned would have a large impact on which weapons it was associated with. If the value of the 'Shotgun' weapon class was set to 1, and the 'Sidearm' weapon class was assigned the value 2, then shotguns would be associated with sidearms more than they would be associated with any weapons of higher values. Since any possible assigned values would indicate an undesired relation between weapons, it was instead decided add each weapon class as another dimension. This added 14 dimensions to the data; a player would have a 1 in this value if their most frequent method of killing players was the same as the category, otherwise they would have a 0.

The following table displays the full range of dimensions (attributes and metrics) utilized in the data exploration:

GENERAL PREPROCESSING

Preprocessing the large amount of data proved to be considerably difficult. Originally, the intention was to join the aggregates to the death data by player- and match id in a one-to-many relationship, to get information about which kills a player had performed in each match. However, with a 32GB RAM consumer desktop, this process was staggeringly slow. After several attempts it became apparent that there was no way to process this amount of data in a reasonable amount of time with the given intention of joining it. Because of this

[1]Further abbreviated GB.

Dimensions	
Dimension name	**Description**
Walk Distance	The distance walked by the player
Drive Distance	The distance driven by the player
Travel Ratio	The rate of traveling by foot compared to traveling by vehicle. A value of 1.0 described only traveling by foot.
Kill Count	The amount of kills performed by the player
Knockdown Count	The amount of knockdowns performed by the player
Assist Count	The amount of kills this played assisted in
Kill-Knockdown Ratio	The rate of kills compared to knockdowns. A value of 1.0 described killing as many players as was knocked down.
Kill Distance	The average distance between the player and all victims he had killed throughout the match.
Survive Time	The amount of time the player survived in seconds.
Damage	The amount of damage dealt by the player.
Killed from	The distance of which the killing blow on this player was dealt from.
Team placement	The rank of the team at the end of the match. A lower value indicates a better rank.
Most Used Weapon	This was 14 dimensions used to each describing a weapon class. The classes were: Sniper Rifle, Carbine Assault Riflem, LMG, SMG, Shotgun, Pistol and Sidearm, Melee, Crossbow, Throwable, Vehicle, Environment Zone, Other, and Down and Out. The dimension contained 1 if the player had obtained most kills with weapons from this class, and 0 if they had not.

TABLE II
SUMMARIZES THE FULL RANGE OF DIMENSIONS UTILIZED IN EXTRACTING THE BEHAVIORAL PROFILES

issue, the following optimization steps were employed: (1) First, lookups on joining were heavily reduced. Previously the application iterated through players, and joined each player to the kills that he/she had performed in one match. This meant that the entire death dataset had to be looked up once for every player. The way to optimize this behavior was to instead iterate through matches, and join players from the match data. This reduced to amount of lookups in the full dataset to only once per match, and since a match contained up to one hundred players, this decreased the time spent on looking up data to around 1/100th of the previous time. The next step (2) was to create an incremental system to allow for working with the data prior to having the entire file processed. The application was set up to generate a .csv file after every thousand matches processed, which was around 100.000 rows. This allowed not only for earlier testing of the clustering algorithms, but also allowed for scalability between speed and accuracy.
Nevertheless, even with these optimizations, a decision was made to reduce the quantity of data processed to 1/5th of the total, in order to be able to run the algorithms in a more timely manner.

To remove the outliers present in the dataset, outlier detection was employed (based on the interquartile ranges (IQR) of the dataset). The outliers were detected using median-based outliers, with outliers being defined as data outside the inner and outer fences on both lower and higher ends of the data. Due to the superflous amount of data present in the set, any rows containing outlier values were simply removed.

After calculating kill distance for the data, and normalizing from 0 to 1 based on the maximum and minimum values in reach respective column, a confusing pattern was discovered: the median kill distance was remarkably low compared with the average. Initially, this was thought to be caused by outliers, but after several attempts cleaning data through outlier detection filtering using different fences, it was discovered that these supposed outliers made up a significant amount of the kill data. Players using weapons commonly associated with long-range capabilities, were often overshadowed several magnitudes by kills performed with grenades and punches. This confusing trend had, however, been noticed prior to this paper. Another Kaggle user[2] also attempted to analyze the dataset provided from [24]. He noticed the same abnormalities, while attempting to analyze the best sniper spots in the game. He concluded that the irregularities in kill distances were caused by a lack of positional data from either killer or victim. In these instances, the missing coordinates were replaced by the default value x,y = [0, 0], the top left corner of the map [25]. Since the top left corner of the map was not reachable by players in the 'Miramar' game map, and was in the middle of the ocean in the 'Erangel' game map, it was considered safe to remove all kills where either the killer or victim had a position of x,y = [0, 0], based on the assumption that these positions were caused by missing values. After this correction, the maximum distance of a kill was reduced to approximately 1/9th of the previous one, and the top range kills were no longer performed with grenades or punches, but rather sniper rifles and carbines.

After running several initial iterations of clustering on the data yet another interesting pattern was discovered. Even with the highest possible minimum sample requirement that could be used without running out of memory, a dense and low-radius cluster persisted in the data. After some analysis of this cluster, it was found that this consisted of the players with a kill count of 0. Since several of the metrics - including most frequent weapon, kill-knockdown ratio, and average kill distance - all relied on the player having one or more kills, these values were identical for all players with a kill count of 0, hence resulting in a very dense cluster. While it could be concluded that not having any kills was a tendency of a behavioral pattern, this was seen more as a result of the choice of metrics. Since the metrics placed a large weight on kills (with 16 of the 27 dimensions relying on at least one kill to have been performed), it was decided to remove all players without kills.

The decision was also made to only look at matches with 4-player teams. This was due to the fact that some features, like knockdowns, were only available in matches with several players per team.

[2]Named 'Mithrillion'

B. APPLICATION OF HDBSCAN AND K-MEANS

For comparison reasons, it was decided that both HDBSCAN and k-Means clustering were to be applied to the dataset. These two algorithms contained different parameters to apply, and these needed to be set in a way that allowed the two algorithms to be comparable.

There were several adjustable parameters in the HDBSCAN algorithm, and without full insight into the inner workings of the algorithm, it could be difficult to deduce what these did. According to McInnes et al. [21] (authors of the scikit HDBSCAN library), it was recommended only altering the 'minimum_cluster_size' parameter, as this would solve most problems that could be encountered while clustering using the algorithm. This parameter changed the minimum required amount of samples that a cluster needed to contain, clusters with less than samples than required were considered noise. Because of the scalability of data, the amount of samples being worked on changed frequently, and this called for a fraction-based minimum, instead of the implemented amount-based solution. This value was created and referred to as the 'cluster significance'. The significance was set to 1%, meaning that each cluster should at minimum contain 1% of all samples.

k-Means required a parameter describing the desired amount of starting centroids, which in turn determined the amount of outputted clusters. To make it comparable with HDBSCAN, the choice was made to set the k-Means clustering algorithm to produce as many clusters as the HDBSCAN had.

After this, the weights of each dimension in the dataset had to be chosen. Initially, the clustering was attempted with every dimension weighted equally. The output of this was, however, that a large weight was placed on the dimensions describing most used weapons, due to the fact that 14 of the 27 columns described this choice. This meant that a player having a favorite weapon value of Assault Rifle would have 14 dimensions in common with other people primarily using assault rifles. Because of that it was decided to reduce the weight of each column describing most used weapons to 0.4.

C. CLUSTER EVALUATION AND LABELING

Before any conclusions could be made on the actual content of the clusters, the clustering needed to be evaluated.

Initially, it was intended to make use of the silhouette coefficient in order to find the denseness and separation of clusters, which would indicate if clusters were well separated or arbitrary. However, the silhouette coefficient applies a distance matrix in order to find the mean distances between points, and this distance matrix consists of one entry for each combination of points which exists in the dataset, and therefore grows exponentially. Because there were approx. 6 million rows and 27 columns in the data after adding metrics and cleaning, this distance matrix was larger than what could possibly be handled by the 32GB RAM desktop used. Therefore the Calinski-Harabasz index was applied, as this method of evaluating clusters had lower RAM requirement. The index described the similar properties as the silhouette coefficient, within-cluster dispersion compared to inter-cluster dispersion. Unlike the silhouette coefficient, this index does not provide a score between -1 and 1, but rather a score between 0 and infinity, where higher scores are a sign of more dense and separated clusters. While a silhouette coefficient can be used to say something about the clustering by itself, the Calinski-Harabasz index could only be interpreted when it was compared with other clustering methods. Both the silhouette coefficient and the Calinski-Harabasz index favor density based clustering methods, as the density of a cluster is an important factor in the calculation of these scores[1]. It was important to consider this potential bias when comparing the index for HDBSCAN and k-Means, as the HDBSCAN algorithm is density-based.

Despite our efforts to generate the clusters, having them evaluated and visualized, it was still difficult to interpret what they actually meant. To give an insight into what the clusters contained, an average cluster player was established. This was simply an average of all dimensions within a single cluster, which meant to give an insight into the typical player who belonged to a specific cluster.

Aside from this, an ID3 classification tree was also constructed from the k-Means clustering in order to give insight into the parameters that defined which cluster a player was placed into. The classes of the tree was equivalent to the k-Means cluster labels, and the training data was the first half of the data, while the second half was test data. The prediction accuracy of the tree was 99,7%.

D. VISUALIZATION

In order to visualize the data, it naturally needed to be reduced from 27 dimensions. This called for a dimensionality reduction algorithm. After receiving results that were difficult to read from employing Principal Component Analysis (PCA), it was decided to also employ t-Distributed Stochastic Neighbor Embedding (t-SNE) to generate a two-dimensional map. This is an unsupervised learning method built on the symmetrizing SNE algorithm, and utilizes the students t-test to calculate similarity between points in order to avoid a crowding problem[2].

Aside from the choice of algorithms used when reducing dimensions, it also had to be decided if the algorithm should be applied to the data pre- or post clustering. Reducing the dimensionality before clustering would lose information, as it simply is not possible to store 27 dimensions worth of information in 2 dimensions. However, some clustering algorithms suffer from what is referred to as the curse of dimensionality[13], where the higher amounts of dimensions in the data will cause the clustering to be less reliable, which could potentially be a good trade-off. Performing dimensionality reduction after clustering would purely aid in visualizing the data. Since HDBSCAN boasts working with 50 to 100 dimensions[21], it should not require dimensionality reduction before clustering the current amount of data. Because of this, it was decided to employ t-SNE post-clustering, only to aid in the visualization. This could bias the results of the k-Means clustering algorithm in a negative manner.

V. RESULTS

In this section we will present the generated results based on the approach explained in the previous section.

See Table 3 - 6 for the average player profiles. Abbreviations TR, KKR, K-Downs, K-Dist, and Dmg stands for Travel Ratio, Kill-Knockdown Ratio, Knockdowns, Kill Distance, and Damage, respectively. The 'Weapon' column describes the weapon killed with most frequently, and the percentage stands for the percentage of players in the cluster killing most with this weapon.

The Calinski-Harabasz scores were calculated to 9683 for HDBSCAN and 92291 for k-Means. See figure 2 - 5 for visualizations of data.

Each average player was given a label intended to sum up the behavior the represented. In HDBSCAN the following behavioral profiles were discovered and labeled:

- **Elites** were the best performing of the players, they survived long, preferred travelling by vehicle, used exclusively assault rifles, got most kills, and placed in top positions.
- **Shotgun Dummies** These players used solely shotguns, travelled mainly by foot, and placed among the bottom ranks.
- **AR Dummies** These players used solely assault rifles, travelled mainly by foot, and placed among the bottom ranks.
- **SMG Dummies** These players used mainly SMG's, travelled mainly by foot, and placed among the bottom ranks.

In k-Means the following behavioral profiles were discovered and labeled:

- **Snipers** were the best performing of the players, they survived long, preferred travelling by vehicle, got most kills, killed from long range, and placed in top positions.
- **Travelers** were players who spent their time on-the-move, they traversed more distance in vehicles than by foot, but generally placed in the lower half of the teams.
- **Stragglers** were people who only survived for the first third of the match, they mostly went by foot, and placed in the lower half of the teams.
- **Veterans** were performers beaten only by the Snipers, they preferred traveling by foot, and placed among the top quarter of the teams.

VI. DISCUSSION

Obtaining sensible behavioral profiles from the behavioral telemetry was substantially difficult for a variety of reasons which shall be commented on here.

Difficulties of Extracting Behavioral Profiles

The behavioral profiles extracted from the dataset seemed to vary largely between HDBSCAN with and k-Means. While HDBSCAN behavioral profiles were separated largely by the weapon type used by the player, k-Means profiles only contained players using mostly assault rifles.

As several dimensions of the data logically correlated, players who followed one parameter were often considered similar to all other players following the same parameter. For example, players who had a high distance traveled by both foot and car often naturally had a long survival time, and in turn those players reached a higher rank. Similarly, players with large amounts of kills also had large amounts of damage, which also followed hand-in-hand with long survive times. Survive time seemed to be the core behavior of players, with every other dimension seemingly being affected by it to a certain extent.

PUBG is a game that makes use of randomization. Throughout the game, players would find random weapons in different locations, which is reflected in the data. Players who survived for a low amount of time tended to use many different weapons, where players with a higher rank tended to use mainly assault rifles. This could either mean that the player group that did better tended to use assault rifles, but it could also mean that players who died early did not have time to find their preferred weapon. This issue could have been circumvented if other data was available, such as when a player chose to switch one weapon for another, but with the current data, this issue made it difficult to separate player decisions from necessity caused by what randomly generated content was available.

Complications due to the Dataset and Hardware limitations

The hardware available also presented an issue during the processing of data. Due to the large amounts of data, along with the computational requirements of the HDBSCAN algorithm, the memory usage made it very troublesome to complete a hierarchical clustering of the desired amount of data, even after only using 1/5th of the original 65 mio row dataset. With the hardware available to us, it was only possible to cluster approx. 180 thousand of the intended 6 mio rows with the HDBSCAN algorithm. Because the results from clustering needed to be accurately compared, the k-Means was limited to this same amount of rows.

Another major issue was the visualization of the clusters in the context of HDBSCAN. As the computational complexity involved in performing HDBSCAN over e.g. k-Means is substantial, we never managed to complete a visualization of the HDBSCAN in time. As such, no visualization other than PCA was performed for HDBSCAN.

Complications due to errors

The Calinski-Harabasz index seemed to indicate that clusters created by the k-Means algorithm were more densely packed and separated from others, than the ones created by HDBSCAN. This was despite the fact that the index had a bias towards density based values. Because of the large difference between the two values, it could only be assumed that the index had been applied wrong, and that these scores were not reliable. This error was likely caused by the noise samples of HDBSCAN being included as a cluster, which would naturally cause a much lower Calinski-Harabasz.

Cluster	Nickname	Walked	Rode	TR	Kills	K-Downs	Assists	KKR	K-Dist	Survive	Dmg	K-From	Place	Weapon
1	Elite	2352	3291	0.42	1.64	1.45	0.27	0.52	2095	1454	199	2845	7	AR
2	Shotgun Dummy	312	0	1.00	1.33	1.33	0.09	0.50	558	288	139	1285	22	SG
3	AR Dummy	453	0	1.00	1.41	1.39	0.11	0.50	1137	370	153	1582	20	AR
4	SMG Dummy	336	0	1.00	1.29	1.27	0.11	0.50	1013	305	135	1317	22	SMG

TABLE III
AVERAGE PLAYERS (HDBSCAN, 1%)

Cluster	Nickname	Walked	Rode	TR	Kills	K-Downs	Assists	KKR	K-Dist	Survive	Dmg	K-From	Place	Weapon
1	Sniper	2613	4271	0.40	2.75	2.17	0.84	0.60	6278	1665	333	6260	4	AR,Carbine
2	Straggler	570	14	0.99	1.58	1.50	0.23	0.53	1564	424	176	2400	19	AR,SG
3	Veteran	3248	667	0.86	2.53	2.00	0.71	0.60	5291	1527	297	6450	6	AR,SMG
4	Traveler	1470	2679	0.38	1.80	1.64	0.39	0.55	2937	973	212	5396	13	AR,SMG

TABLE IV
AVERAGE PLAYERS (K-MEANS, K = 4)

Cluster	Nickname	Walked	Rode	TR	Kills	K-Ds	Assists	KKR	K-Dist	Survive	Dmg	K-From	Place	Weapon
1	Melee	923	829	0.82	1.56	1.04	0.20	0.65	780	587	179	3114	29	Melee
2	AR	1960	1796	0.69	2.20	1.75	0.54	0.59	3720	1147	256	4872	10	AR
3	Shotgun	1050	761	0.84	1.69	0.99	0.23	0.70	1006	686	188	3519	27	Shotgun
4	AR	1902	1661	0.70	2.35	0.67	0.27	0.84	3549	1143	263	4999	21	AR
5	SMG	1356	1048	0.79	1.78	0.99	0.28	0.71	1660	851	202	4098	23	SMG
6	Sniper	1969	2034	0.65	2.14	1.11	0.42	0.72	5568	1166	251	4629	16	Carbine, Snipe
7	Torturer	1993	1930	0.67	2.51	2.18	0.57	0.53	5234	1169	296	5079	12	Down & Out
8	Pistol	855	559	0.88	1.43	0.80	0.18	0.72	1045	568	161	3079	32	Pistols

TABLE V
AVERAGE PLAYERS (K-MEANS, K = 8)

Cluster	Nickname	Walked	Rode	TR	Kills	K-Ds	Assists	KKR	K-Dist	Survive	Dmg	K-From	Place	Weapon
1	Elite	3095	2778	0.55	6.81	5.85	1.22	0.53	7485	1713	796	4634	3	AR, Carb, Snipe
2	Brawler	1120	52	0.97	1.87	1.83	0.32	0.50	2351	681	217	3522	13	AR, SMG, SG
3	Travellers	1392	2539	0.38	1.84	1.77	0.39	0.51	2715	932	217	4991	13	AR, SMG, SG
4	Drive-By	2488	4454	0.37	2.25	1.92	0.76	0.53	5894	1627	279	6325	5	SMG; Carb
5	Straggler	948	74	0.96	1.15	0.03	0.32	0.99	2396	585	111	3735	17	AR, SMG, SG
6	Target Dummy	351	4	1.00	1.49	1.53	0.18	0.49	1171	314	166	1829	22	AR, SMG, SG
7	Assassin	2518	3411	0.45	1.34	0.05	0.63	0.99	5528	1510	151	6724	6	AR, SMG, Carb
8	Veteran	3433	698	0.85	2.35	1.85	0.74	0.57	5506	1599	277	6339	5	AR, SMG, Carb

TABLE VI
AVERAGE PLAYERS (K-MEANS, K = 8, UNWEIGHTED)

VII. CONCLUSION

In conclusion, it appears that HDBSCAN did a poor job at extracting behavioral profiles. Although we managed to label certain detected behaviors, it appears that these 'profiles' is based on too few and highly weighted dimensions. Hence, the labels provided is considered misleading. This essentially means that the clustering of the behavioral data using HDBSCAN was unsatisfactory.

Nevertheless, we additional believe this concluded result to be highly correlated to the applied dataset alongside how it was analyzed. As we did not have access to log other behavioral attributes other than what was provided by the dataset, we were limited to whatever was provided.

The limitation of analyzed data makes it difficult to conclude upon the overall tendencies in the data. It is not known if the extracted behavioral profiles, or the comparison between k-Means and HDBSCAN, would have had vastly different result if the entire dataset had been worked with. Because it was not known if the data was homogeneous within the set, it is not unlikely that some behavioral profiles only existed in the remaining 4/5ths of the data, which had never been analyzed. Randomization for the entirety of the dataset would have been a possible solution in order to sample the dataset, but this would have resulted in further complication of analyzing the dataset.

We believe it necessary to extend the work done here, such that the HDBSCAN is better applied to a behavioral dataset. In addition, a more direct comparison between the algorithms needs occur; likely based on a correct application of silhouette coefficient and/or Calinski-Harabasz index.

REFERENCES

[1] Calinski, T., & Harabasz, J. (1974). A Dendrite Method for Cluster Analysis. Communications in Statistics - Simulation and Computation, 3(1), 1-27.

[2] Maaten, L. V., & Hinton, G. (2008). Visualizing Data using t-SNE (Y. Bengio, Ed.). Journal of Machine Learning Research 9, 2579-2605. Retrieved May 13, 2018, from http://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

[3] Drachen, A., Canossa, A., & Yannakakis, G. N. (2009, September). Player modeling using self-organization in Tomb Raider: Underworld. In Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on (pp. 1-8). IEEE.

[4] Drachen, A., Sifa, R., Bauckhage, C., & Thurau, C. (2012, September). Guns, swords and data: Clustering of player behavior in computer games in the wild. In Computational Intelligence and Games (CIG), 2012 IEEE Conference on (pp. 163-170). IEEE.

[5] Drachen, A., Thurau, C., Togelius, J., Yannakakis, G. N., & Bauckhage, C. (2013). Game data mining. In Game analytics (pp. 205-253). Springer, London.

[6] Drachen, A., El-Nasr, M. S., & Canossa, A. (2013). Game Analytics - The Basics. In Game analytics (pp. 13-40). Springer, London.

[7] Drachen, A., Thurau, C., Sifa, R., & Bauckhage, C. (2014). A comparison of methods for player clustering via behavioral telemetry.

[8] Bauckhage, C., Drachen, A., & Sifa, R. (2015). Clustering game behavior data. IEEE Transactions on Computational Intelligence and AI in Games, 7(3), 266-278.

[9] Yannakakis, G. N., & Togelius, J. (2018). Artificial Intelligence and Games. Springer.

[10] El-Nasr, M. S., Drachen, A., & Canossa, A. (2013). Benefit of Game Analytics: Stakeholders, Context and Domains. London: Springer.

[11] El-Nasr, M. S., Drachen, A., & Canossa, A. (2013). Introduction. London: Springer.

[12] Sifa, R., Drachen, A., & Bauckhage, C. (2016). Profiling in Games: Understanding Behavior from Telemetry. Social Interaction in Virtual Worlds.

[13] Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[14] Schaekermann, M., Ribeiro, G., Wallner, G., Kriglstein, S., Johnson, D., Drachen, A., & Nacke, L. E. (2017, October). Curiously Motivated: Profiling Curiosity with Self-Reports and Behaviour Metrics in the Game Destiny. In Proceedings of the Annual Symposium on Computer-Human Interaction in Play (pp. 143-156). ACM.

[15] Pirker, J., Griesmayr, S., Drachen, A., & Sifa, R. (2016, September). How playstyles evolve: progression analysis and profiling in Just Cause 2. In International Conference on Entertainment Computing (pp. 90-101). Springer, Cham.

[16] Ramirez-Cano, D., Colton, S., & Baumgarten, R. (2010, April). Player classification using a meta-clustering approach. In Proceedings of the 3rd Annual International Conference Computer Games, Multimedia & Allied Technology (pp. 297-304).

[17] Francillette, Y., Abrouk, L., & Gouaich, A. (2013, July). A players clustering method to enhance the players' experience in multi-player games. In Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational & Serious Games (CGAMES), 2013 18th International Conference on (pp. 229-234). IEEE.

[18] Campello, R. J., Moulavi, D., & Sander, J. (2013, April). Density-based clustering based on hierarchical density estimates. In Pacific-Asia conference on knowledge discovery and data mining (pp. 160-172). Springer, Berlin, Heidelberg.

[19] McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. The Journal of Open Source Software, 2(11), 205.

[20] McInnes, L., & Healy, J. (2017, November). Accelerated Hierarchical Density Based Clustering. In Data Mining Workshops (ICDMW), 2017 IEEE International Conference on (pp. 33-42). IEEE.

[21] McInnes, L., Healy, J., & Astels, S. (2016). The hdbscan Clustering Library. Retrieved May 13, 2018, from http://hdbscan.readthedocs.io/en/latest/index.html

[22] Canossa, A. (2013). Meaning in gameplay: filtering variables, defining metrics, extracting features and creating models for gameplay analysis. In Game Analytics (pp. 255-283). Springer, London.

[23] Thurau, C., & Drachen, A. (2011). Introducing archetypal analysis for player classification in games. In 2nd International Workshop on Evaluating Player Experience in Games (epex 2011).

[24] Pei, K. (2018, January 12). PUBG Match Deaths and Statistics — Kaggle. Retrieved May 13, 2018, from https://www.kaggle.com/skihikingkevin/pubg-match-deaths

[25] Mithrillion. (n.d.). Kill Distance Analysis — Kaggle. Retrieved May 13, 2018, from https://www.kaggle.com/mithrillion/kill-distance-analysis

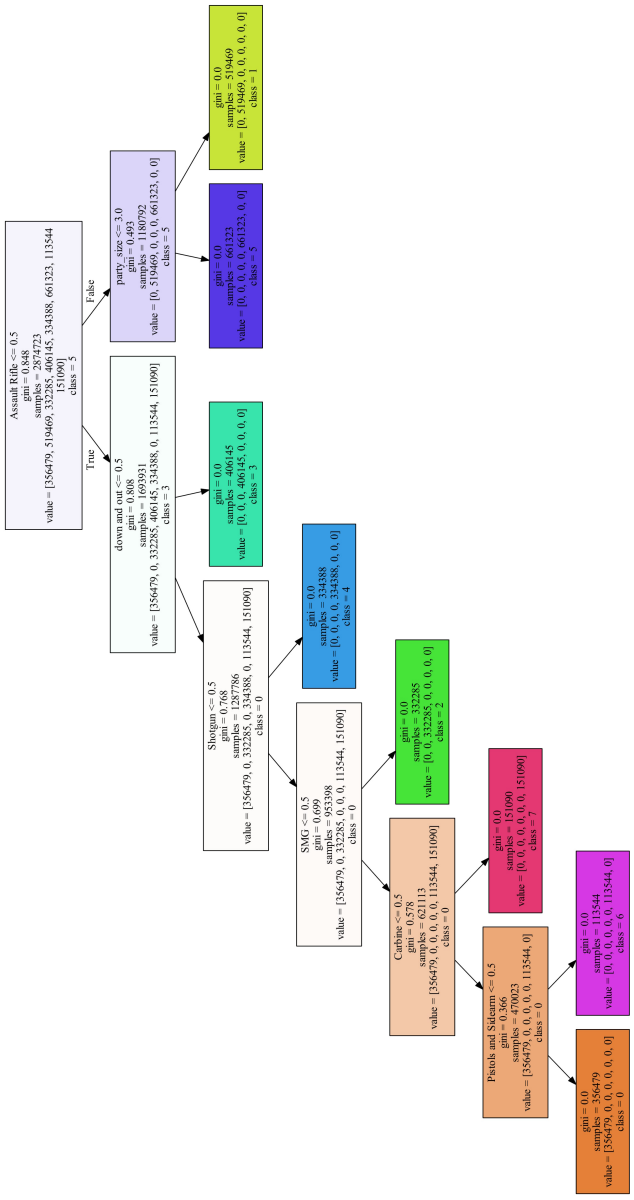


Fig. 1. ID3 prediction of k-Means Clustering; k=8

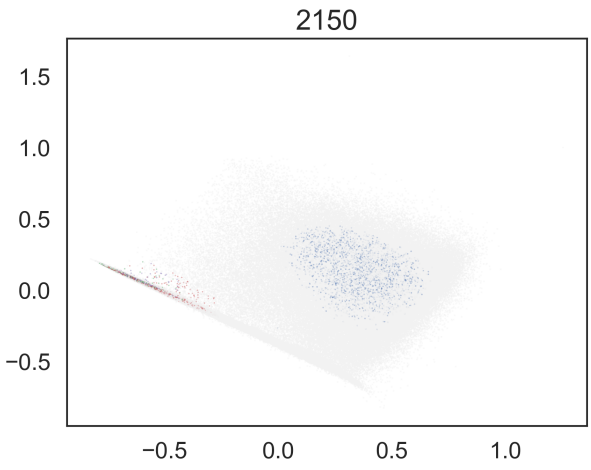


Fig. 2. PCA visualization of HDBSCAN

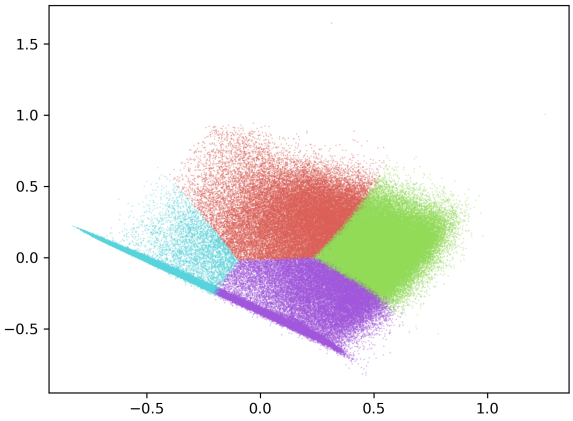


Fig. 3. PCA visualization of k-Means Clustering; k=4

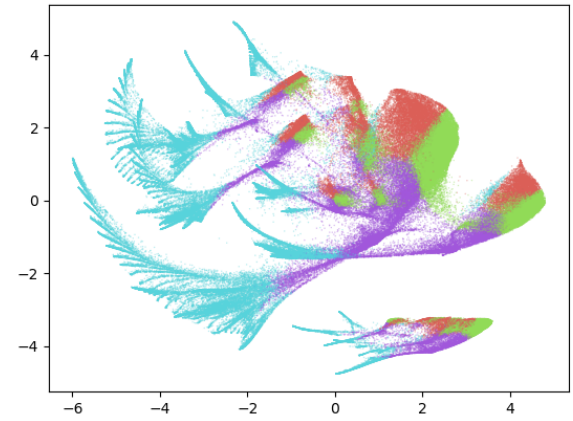


Fig. 4. t-SNE visualization of k-Means Clustering; k=4

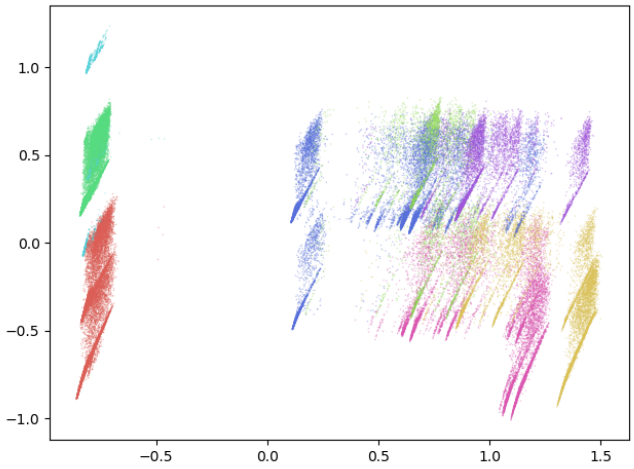


Fig. 5. PCA visualization of k-Means Clustering; k=8

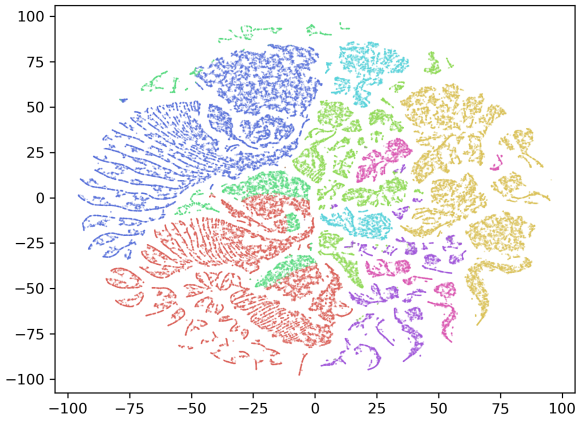


Fig. 6. t-SNE visualization of k-Means Clustering; k=8