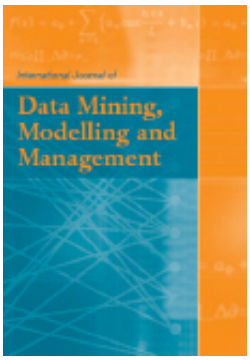


International Journal of Data Mining, Modelling and Management

ISSN online: 1759-1171 - ISSN print: 1759-1163
https://www.inderscience.com/ijdmmm

A deep-learning approach to game bot identification via behavioural features analysis in complex massively-cooperative environments

Alfredo Cuzzocrea, Fabio Martinelli, Francesco Mercaldo

DOI: 10.1504/IJDMMM.2023.10055201

Article History:

Received:	28 September 2020
Last revised:	06 May 2021
Accepted:	11 June 2021
Published online:	04 April 2023

A deep-learning approach to game bot identification via behavioural features analysis in complex massively-cooperative environments

Alfredo Cuzzocrea*

iDEA Lab,
University of Calabria,
Rende, Italy
and
LORIA,
Nancy, France
Email: alfredo.cuzzocrea@unical.it
*Corresponding author

Fabio Martinelli and Francesco Mercaldo

Institute for Informatics and Telematics,
National Research Council of Italy (CNR),
Pisa, Italy
Email: fabio.martinelli@iit.cnr.it
Email: francesco.mercaldo@iit.cnr.it

Abstract: In the so-called *massively multiplayer online role-playing games* (MMORPGs), malicious players have the possibility of obtaining some kind of gains from competitions, via easy victories achieved thanks to the introduction of game bots in the games. In order to maintain fairness among players, it is important to detect the presence of game bots during video games so that they can be expelled from the games. This paper describes an approach to distinguish human players from game bots based on behavioural analysis. This implemented via supervised *machine learning* (ML) and *deep learning* (DL) algorithms. In order to detect game bots, considered algorithms are first trained with labelled features and then used to classify unseen-before features. In this paper, the performance of our game bots detection approach is experimentally obtained. The dataset we use for training and classification is extracted from logs generated during online video games matches of a real-life MMORPG.

Keywords: game bot detection; complex massively-cooperative environments; machine learning; deep learning; massively multiplayer online role-playing games; MMORPGs.

Reference to this paper should be made as follows: Cuzzocrea, A., Martinelli, F. and Mercaldo, F. (2023) 'A deep-learning approach to game bot identification via behavioural features analysis in complex massively-cooperative environments', *Int. J. Data Mining, Modelling and Management*, Vol. 15, No. 1, pp.1–29.

Biographical notes: Alfredo Cuzzocrea is a Professor of Computer Engineering at University of Calabria, Italy, where he also covers the role of Head of the Big Data Engineering and Analytics Lab. He also holds the

Excellence Chair in Computer Engineering at LORIA, University of Lorraine, Nancy, France. His current research interests are: big data, database systems, data mining, data warehousing, and knowledge discovery. He is author or co-author of more than 680 papers in international conferences (including *CIKM*, *MDM*, *EDBT*, *SSDBM*, *PAKDD*, and *DOLAP*), international journals (including *JCSS*, *IS*, *FGCS*, *INS*, and *JMLR*) and international books. He is recognised in prestigious international research rankings worldwide.

Fabio Martinelli is a Researcher Director at the IIT-CNR, Pisa, Italy, where he is the scientific coordinator of the security group. His main research interests involve security and privacy in distributed and mobile systems and foundations of security and trust. He serves as PC-chair/organiser in several international conferences/workshops. He is the co-initiator of the International Workshop series on Formal Aspects in Security and Trust (FAST). He chairs the WG on security and trust management (STM) of the European Research Consortium in Informatics and Mathematics (ERCIM). He usually manages R&D projects on information and communication security.

Francesco Mercaldo received the Master's degree in Computer Engineering from the University of Sannio, Benevento, Italy, with a thesis in software testing, and PhD degree with a dissertation on malware analysis using machine learning techniques, in 2015. He is currently working as a researcher with the University of Molise, Campobasso, Italy. His research interests include software testing, verification, and validation, with the emphasis on the application of empirical methods.

1 Introduction

Video games rapidly evolved in recent years from single player systems to distributed systems in which a community of players shared a single video game. This has given rise to team game organisations, tournaments and competitions of various types. This evolution is characterised by different states of progress, which are briefly described below. Until 1990, video games were essentially single systems used by a single player. In the 90s video games began to be distributed through internet (Adams, 2014) and played by community of players also from different countries (Seay et al., 2004; Wellman and Gulia, 1999). In the late 90s, *massively multiplayer online role-playing games* (MMORPGs) began to spread (Quandt and Kröger, 2013). These are role-playing video games with a large number of players sharing the games via network infrastructure. MMORPGs are the basis of a vast video game market with a large number of heterogeneous customers in terms of nationality, age, type of work but who share the same interest in the same types of video games (Yee, 2008; Griffiths et al., 2004; Taylor, 2009). Video game providers therefore develop more and more innovative systems to arouse increasing interest in their customers. For example, since the 2000s video game providers have vastly expanded the platforms that can be used by customers for playing, which can range from personal computers to various types of mobile devices to dedicated systems that can expand the type of interaction with the player. However, recently, automatic players have been introduced into the communities of human players. Automatic players, or game bots, are software programs which, using artificial intelligence principles, play the video game instead of human players (Yampolskiy and

Govindaraju, 2008; Fernández-Ares et al., 2017; Cocar et al., 2017; Kim et al., 2005) thus attacking the whole game infrastructure. These types of attacks lead to various types of problems for video game providers. First of all, the performance of game bots is generally much higher than that of human players in terms of speed, accuracy and duration (Aziz et al., 2017; Wang et al., 2017). Some video games are giving away cybernetic money which can sometimes be converted into real money (Paulson and Weber, 2006; Esmaeili and Woods, 2016). Game bots can win more than human players who in these cases can feel not only frustrated but really angry. In these cases, human players can leave the gaming communities causing significant market losses. Secondly, in such ways attackers gain popularity in the player community with very little efforts.

Another possibility, as described in Chen et al. (2004), may be that attackers, for some kind of economic advantage, may wish to acquire confidential information about other players using game bots. In conclusion, game bots can be a serious problem for video game producers because they can cause several problems in the video game market (Kang et al., 2016). For this reason, new approaches to detect the existence of game bots in the player communities are constantly being developed. Moreover, it must be considered that the development of game bots must follows the continuous complexity increase of video games.

Consequently, the techniques to detect existing game bots become more and more sophisticated. Therefore, recent years have seen the development of game bots detection approaches ranging from the implementation of a series of Turing tests as described in Hingston (2009), to the analysis of network traffic as described in Lo and Chen (2008), to the scanning of bots. Common problems with these approaches are that they generally interfere with video games and can easily be avoided by game bots.

In this paper, a game bot detection approach which is based on the behaviour of video game players is worked out. Our starting hypothesis is that human players and game bots behave differently during the game.

For this reason, we defined behavioural features that can discriminate human players from game bots. The separation is achieved through classifiers that we build using *machine learning* (ML) (Bernardi et al., 2017a, 2017b; Aha and Kibler, 1991) and *deep learning* (DL) (Ioannidou et al., 2017; Rav et al., 2017; Deng and Yu, 2014; LeCun et al., 2015; Bengio et al., 2009; Collobert and Weston, 2008; Dahl et al., 2012 Deng et al., 2014) principles.

The effectiveness of the set of proposed behavioural features and of the proposed classifiers are verified using a real video game called *Ainon: The Tower of Eternity* (https://en.aion.gameforge.com/website/). With the datasets obtained by the video game we assess the discriminative power of the proposed approach and compare it with classifiers normally used for this purpose. We chose Ainon because it is a very popular MMORPG (https://www.badosoft.com/knowledgebase/top-10-most-played-mmorpgs.php), because it is a free fantasy game but above all because the game producer made publicly available its dataset. The dataset was obtained by collecting and labelling the operations carried out by both human and robotic players during the game.

This paper is organised in the following way. Section 2 describes some preliminary information about ML and DL while Section 3 describes the used data and the behavioural features reporting some descriptive statistics. Section 4 describes the developed classification approach while in Section 5 we report the experimental results. In Section 6, we discuss significant work done previously in this area. Section 7 reports

and discusses some threats of the described approach, while Section 8 describes concluding remarks and future work.

2 Background: classification algorithms

In this Section, we provide some preliminary information on the classification algorithms we use in this work for evaluating the discrimination between human players and game bots using our behavioural features.

The observations are sequences of features obtained from a running video game. Our algorithms are trained using observations labelled as belonging to the 'human player' class if the video game is played by a human being. Conversely, they are labelled 'game bots' if the video game is played by a bot.

The supervised classification algorithms estimate the category to which an unseen before input observation belongs, based on a training phase that uses observations whose category of belonging is known.

Our classification system is based on ML and DL algorithms.

2.1 ML algorithms

ML algorithms are able to learn their behaviour directly from observations without being explicitly programmed (Mitchell, 1997) using artificial intelligence approaches.

ML algorithms can be the supervised or unsupervised ones. The supervised ones work upon a series of observations provided to the algorithm consisting of data and their belonging classes, called training set. The ML algorithm determines an input-output function based on the training set.

The nonlinear input-output function built by the ML algorithm after the training phase is then used to estimate the class of unseen before data.

In unsupervised ML algorithms, instead, input data is completely unlabeled. The algorithms build the input output function by uncovering the patterns hidden in the data.

The unsupervised ML algorithms cannot derive the correctness of the model itself because the training data is not labelled but they can obtain the statistical density of the data. In our case the unsupervised drives could be used to summarise the main characteristics of the data.

In this paper, we use different types of supervised ML algorithms to classify between human players and game bots. The ML algorithms used in this paper are divided in *decision tree* (DT) algorithms, *Bayesian networks* (BN) type algorithms, *multinomial logistic regression* (MLR) classifiers, regression classifiers (RC), *rules* (R) and *lazy* (L) classifiers.

DTs encode the training information in a tree in which information on the data is put in the branches and the decisions are put in the leaves of the tree. The following different DT algorithms are used in this paper, namely the J48 DT described in Mihăescu et al. (2015), the decision stump described in Iba and Iba (1992), *Hoeffding tree* of Hoeglinger and Pears (2007), *random forest* of Breiman (2001), *random tree* described in Zhao and Zhang (2008), REP tree of Nadiammai and Hemalatha (2013) and logistic model tree (LMT) described in Landwehr et al. (2005).

The class of algorithms called *BN* are based on a graph in which the nodes represent the random variables and the arcs represent probability dependencies among them. The

BN graph does not contain cycles (Friedman et al., 1997). Several algorithms belonging to this category are used in this work, namely *BayesNet* (Lee and Shimoji, 1993), *naïve Bayes* (John and Langley, 1995), *naïve Bayes multinomial* (McCallum and Nigam, 1998) and *naïve Bayes multinomial text* (Kibriya et al., 2004).

The logistic regression algorithm is a two-class classifier and hence admits two possible output values. MLR is its multiclass extension and therefore its output values can be greater than two. MLR estimates the probability of a random variable starting from a set of independent variables. In this work we used two algorithms belonging to the MLR namely *logistic* and MLP (Buhrke and LoCicero, 1992).

In the regression-based classifiers RC (Freedman, 2009) the dependent variable is categorical. The probability of a binary response based on several features is usually estimated by means of a binary logistic model. Moreover, the *Adaboost ML* classification algorithms (Kégl, 2013) discover the relationships between the studied variables by analysing a large amount of data.

In particular, the *JRrip* algorithm (Cohen, 1995), which belongs to the *Adaboost ML* class, has been used in this paper.

The ML algorithms used in this study are listed in Table 1, where each name is specified with a brief description of the algorithms.

2.2 DL algorithms

DL is a branch of Artificial Intelligence that refers to artificial neural networks. It is a type of hierarchical learning that is part of a wider family of ML methods (Deng and Yu, 2014) that learn how to solve problems directly from data, as opposed to the algorithms that are built expressly for the execution of specific tasks.

DL architectures have been successfully applied in many fields, for example in solving computer vision problems, in automatic recognition of spoken language, in natural language processing, in audio recognition and in bioinformatics (Ioannidou et al., 2017; Rav et al., 2017).

DL algorithms constitute a class of ML algorithms that use various levels of cascaded nonlinear units to perform feature extraction and transformation tasks. The input of each level uses the output of the previous level. The algorithms can be both supervised and unsupervised and applications include pattern analysis and classification. DL algorithms learn multiple levels of representation that correspond to different levels of abstraction; creating a hierarchy of concepts.

In this paper, two types of DL algorithm are used, namely the *multilayer perceptron* (MLP) and the *convolutional neural network* (CNN). Their structures are composed by a number of layers formed by a number of neurons. A neuron is a computing element which adds all of its inputs and then uses an activation function, which is nonlinear (Gardner and Dorling, 1998).

We chose these algorithms due to the following reason. These algorithms are a kind of DL algorithms, and it has been already proved they are suitable to support the knowledge discovery extraction from complex environments like the one we investigate, with accuracy well-beyond classical data mining and ML approaches, similarly to other research experiences (e.g., Hung et al., 2017).

Normally MLP consists of three layers trained by means of a backpropagation technique (Ruck et al., 1990). CNN are a family of neural networks widely used in the field of computer vision and, more generally, with data that have spatial relationships.

Their graphical representation consists of a graph with a series of levels where, starting from a complex input, information is gradually extracted from the input data, or features, which are increasingly representative in the context of the problem under consideration.

CNNs therefore follow a tiered architecture, typically non-cyclical. The most important levels are the convolution layers, from which it takes its name (Kim, 2017).

A CNN can have many dozens of layers, each of which learns to detect the different features of an image. Images are analysed at different resolutions, and the output of each convolved image is used as input for the next layer. The great advantage of CNNs is that they can automatically optimise the features for image recognition without being defined by the researchers (Krizhevsky et al., 2012).

Table 1 The ML algorithms used in our research

Method	*Algorithm*	*Algorithm description*
DT	J48	A DT based on the attribute values of the training dataset is created in order to classify a new item. The set of items discriminating the various instances is extracted so when a new item is encountered it can be classified.
	Decision Stump	A DT model with one internal root node immediately connected to the terminal nodes. The prediction is allowed by a decision stump basing on the value of just one input feature.
	Hoeffding tree	An incremental, anytime DT induction algorithm learning from massive data streams. Basing on the assumption that the distribution generating examples does not change over time, it often uses a small sample to choose an optimal splitting attribute.
	Random forest	It operates by constructing, at training time, a multitude of DTs and outputting the class that is the mode (the most frequent value appearing in a set of data) of the classes of the individual trees.
	Random tree	It constructs a tree containing some randomly chosen attributes at each node. No pruning is performed.
	Rep tree	It builds a DT using information gain/variance and reduces errors using reduced-error pruning. The values are ordered for numeric attributes once and missing values are recovered with by splitting the corresponding instances into pieces.
	LMT	LMTs are built as classification trees with logistic regression functions at the leaves. This algorithm can deal with missing values, multi-class and binary target variables, nominal and numerical.
BN	BayesNet	Bayes network learning using various search algorithms and quality measures to learn the probability density functions of individual pattern classes from a collection of learning samples.
	Naïve Bayes	It uses numeric estimator precision values selected basing on the analysis of the training data.

Table 1 The ML algorithms used in our research (continued)

Method	*Algorithm*	*Algorithm description*
BN	Naïve Bayes Multinomial	It is based on a multinomial event model where samples (described as feature vectors) represent the frequencies with which certain events have been generated. The multinomial vector describes the probability that each event occurs.
	Naive Bayes multinomial text	It is similar to naïve Bayes multinomial but it operates directly and exclusively on text.
MLR	Logistic	It builds a MLR model with a ridge estimator.
	MLP	It is a feedforward artificial neural network model mapping sets of input data onto a set of outputs. It consists of multiple layers of nodes related in a directed graph where each layer is fully connected to the next one. Except for the input nodes, each node is a neuron with a nonlinear activation function.
		A supervised learning technique (called backpropagation) is used for training the network.
RC	AdaBoostM1	It is usually used in conjunction with some other learning algorithms with the aim to improve their performance. The output of these other learning algorithms is combined into a weighted sum representing the final output of the boosted classifier.
R	JRip	It analyses classes basing on their increasing size and an initial set of rules for each class is generated using incremental reduced error. For the first class it analyses all the examples of a particular judgement in the training data and finds a set of rules able to cover all the class members. These operations are repeated for the next class until all the classes have been covered.
L	IBk	It is an application of the K-nearest neighbour's algorithm where a parameter specifies the number of nearest neighbours allowed to classify a test instance and the outcome is determined by majority vote.

3 The features model and the dataset

In this paper, we experimentally prove that, by using a series of behavioural features, it is possible to distinguish human players from game bots. Experimental evidence is provided through the analysis of a video game very popular in the MMORPG community and described in Kang et al. (2016). Behavioural features can be divided into the following classes: *player information* (PI), *player actions* (PA), *group activities* (GA), *diversity of social interactions* (SID) and *network measures* (NM). The feature classes we have considered in this work, along with their final features, are listed in Table 2 for each class reported above.

It is important to highlight why these behavioural features have been selected. According to our experimental evaluation of classification settings and parameters on our target dataset, these features have exposed the best domain knowledge capture and classification accuracy among all the available combinations of features that can be derived from the dataset. To this end, classical feature testing and extraction approaches have been applied.

Let's start with the PA features. In general, the features belonging to the PA class represent the different behaviour performed by human players or game bots which should be very different. For example, consider the feature called PA_1 in Table 2. It denotes the sitting rate of the players. Normally human players sit less frequently than bot players so this number should differentiate between the two players. Consider now the feature PA_5. It denotes the amount of points earned by killing enemies. These points could be used to buy various types of item or to set the level of the player. In particular, in the Aion game the amount of kill points increases the rank of the player. In the Aion game, the rank of the player leads to more power which increases the easiness to kill enemies. Now, it is obvious that game bot can easily acquire more power than human player because they can play continuously since the bots is not needed to eat or to sleep. Let's consider for example the feature class we called 'GA', in particular GA_1 and GA_2. These two features represent the amount of activities related to social relationships among players. The first feature for instance is the average time spent by party play which is an association among players for solving difficult steps in the game. The way that human players spend their time during a party play is different from that spent by a bot because they are supported by a different type of socialisation. This difference is highlighted by the features defined under the GA class. According to the social diversity issue between human players and game bots, the entropy of party play is another effective feature for representing the diversity between the two types of players. In fact, game bots normally execute single actions while human players normally execute multiple tasks.

Table 2 The features involved in the study with the correspondent category

Category	*Features*
Player information	Login frequency (PI_1), playtime (PI_2), game money (PI_3), number of IP address (PI_4)
Player actions	Sitting (PA_1), earning experience points (PA_2), obtaining items (PA_3), earning game money (PA_4), earning player kill points (PA_5), harvesting items (PA_6), resurrecting (PA_7), restoring experience points (PA_8), being killed by a non-player and/or player character (PA_9), using portals (PA_{10})
Group activities	The average duration of party play (GA_1), number of guild activities (GA_2)
Social interaction diversity	Entropy of party play (SID_1)
Network measures	Degree centrality (NM_1), betweenness centrality (NM_2), closeness (NM_3), eigenvector centrality (NM_4), eccentricity (NM_5), authority (NM_6), hub (NM_7), PageRank (NM_8), clustering coefficient (NM_9)

The network interactions between the players are described in the features described in Table 3. For this purpose, a graph is used in which the nodes represent the players and the arcs their interactions. For example, an arc between two nodes could represent some type of transfer between two players.

The dataset used to train and test the algorithm proposed in this paper has been made publicly available by the videogame company and consists of the log files generated by running the Ainon video game continuously for 88 days by 49,739 players (https://sites.google.com/a/hksecurity.net/ocslab/Datasets/game-bot-detection). The game company has identified 7702 players as game bots. Since all the entire dataset was available, we do not consider sampling approach to collect data. On the other hand, if some of the applied ML/DL techniques were requiring sampling, then we used the kind

of sampling (e.g., uniform, stratified, etc.) that the respective technique is considering the best in order to magnify the pattern discovery effect.

The identification of the game bots was obtained first by monitoring and was then checked and verified manually. The dataset provided by the game company was then extended with our features that were labelled as generated by human players or game bots according to what is indicated by the company. Furthermore, all personal identity information has been removed from the entire data set for privacy reasons.

Table 3 The features belonging to the NM category with their description

NM category feature	*Description*
Degree centrality	This feature represents the centrality focused on the degree. The more edges an actor has, the more important it is.
Betweenness centrality	It counts the number of shortest paths between two nodes on which a given actor resides.
Closeness centrality	An actor is considered important if it is relatively close to all other actors. Closeness is based on the inverse of the distance of each actor to every other actor in the network.
Eigenvector centrality	Indicates that a given node has a relationship with other valuable nodes. A high eigenvector value for an actor means that a node has several neighbours with high eigenvector values.
Eccentricity	The eccentricity of node v is calculated by computing the shortest path between node v and all other nodes in the graph; then the longest shortest path is chosen.
Authority	Exhibits a node pointed to by many good hubs.
Hub	Exhibits a node that points to many good authorities.
PageRank	Assigns a numerical weight to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of 'measuring' its relative importance within the set.
Clustering coefficient	It quantifies how close neighbours are to being a clique: a clique is a subset of all of the edges connecting pairs of vertices of an undirected graph.

3.1 Descriptive statistics

A common way to describe a random variable is through its statistical moments such as the central trend and dispersion indices. The most used central trend indices are the average, the median, the mode while the most used dispersion indices are the quartiles, the variance and the mean square deviation. Statistical distributions can be briefly described by other indices such as skewness and kurtosis of the distribution. The skewness index serves to distinguish functions of symmetric distributions such as the Gaussian function from functions that are displaced with respect to the average value. The kurtosis index measures the thickness of the tails or the degree of flattening of the distribution. The range of a distribution defines the range of variability of the variables and is given by the difference between the maximum and minimum value of the variable. Furthermore, the distribution of a variable can be described using graphic notations such as box-plot and violin-plot which are a way to graphically represent groups of numerical data through their quartiles. For example, box-plots are a non-parametric notation in the

sense that they are provided without referring to the underlying distribution. In other words, box-plots graphically represent groups of numeric data through their quartiles.

Figures 1, 2, 3, 4 and 5 show the box plots for a subset of features belonging to each category considered by the proposed method. Specifically, in Figures 1, 2, 3 and 4 we report the box plots of a subset of features of the class human players and game bots respectively. To keep discussion focused we only show a subset of the box-plots related to interesting distributions.

In particular, the box plot shown in Figure 1 represents the PL_2 feature, that is, the game time. We see that using this feature the difference between human players and game bots is well separated as human players have a small inter-quartile range when compared to game bots. In this case the median values fall out of the inter-quartile range as the game bots are able to play longer and without interruption than the human counterpart.

Figure 1 The box plot relating to the game bot and human distributions for the playtime feature belonging to the category PI (see online version for colours)

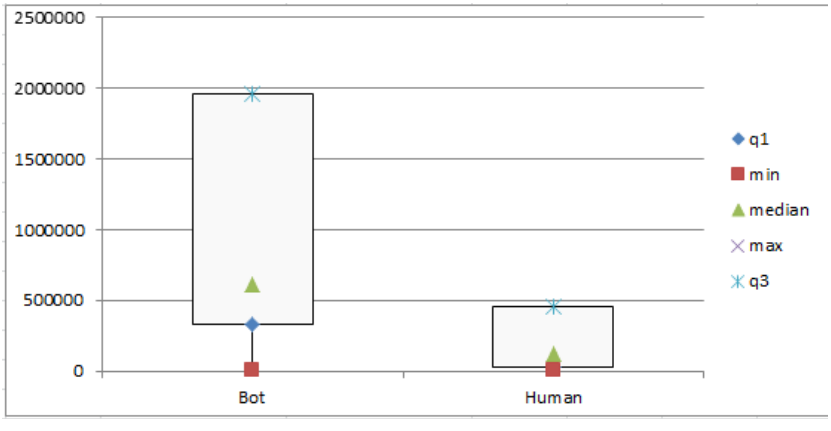
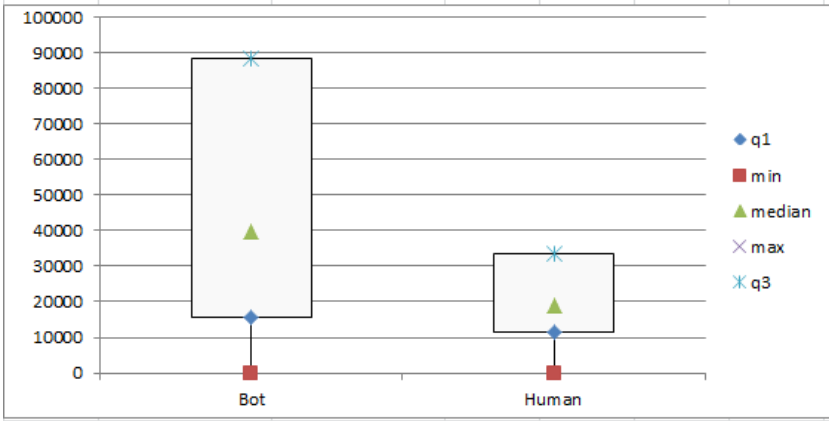


Figure 2 The box plot relating to the game bot and human distributions for the earning experience points feature, belonging to the category PA (see online version for colours)



The box-plot related to the PA_2 feature is shown in Figure 2. The PA_2 feature describes the ability of a player to acquire points for purchasing power. Here too there is a clear distinction between human players and game bots, due to the fact that game bots are able to acquire more points than human players. If we look at Figure 2 it is evident that the median of the distribution of bots is greater than the third quartile of the distribution of human players.

The box plot of the GA_1 feature is shown in Figure 3. In this case, unlike the previous cases, the box plot for human players is greater than that of game bots. This is justified by the behaviour of the players belonging to the two categories. While human players have an inherent tendency to team up with others to solve difficult problems, game bots only attempt to gain points for increasing power.

Figure 3 The box plot relating to the game bot and human distributions for the playtime feature belonging to the category GA (see online version for colours)

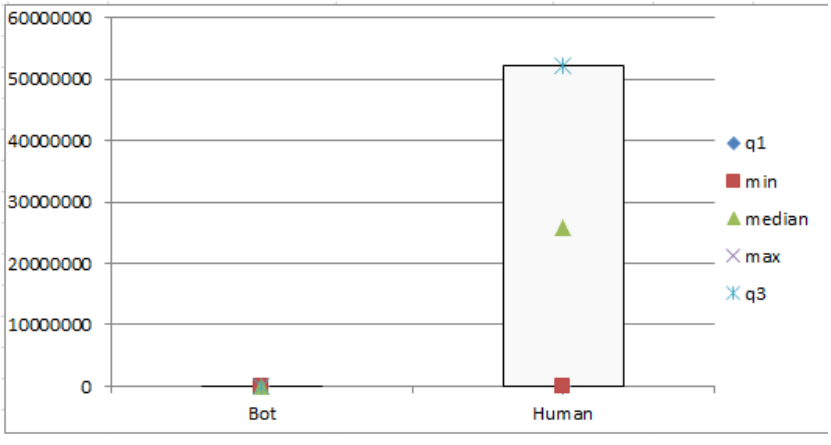
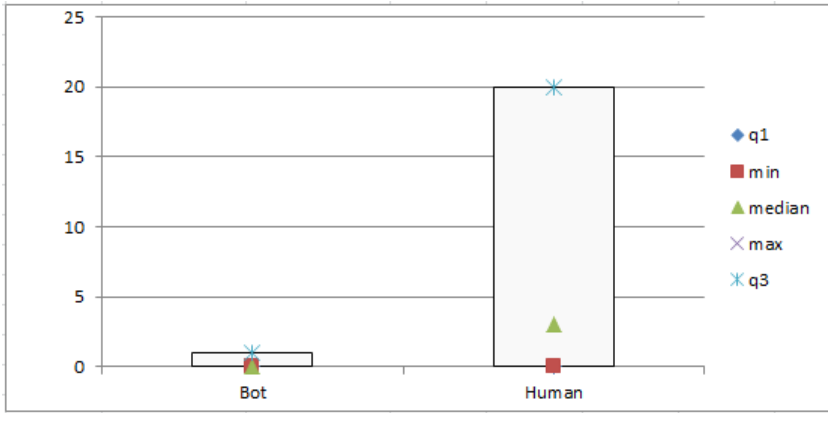


Figure 4 The box plot relating to the game bot and human distributions for the degree centrality feature belonging to the category NM (see online version for colours)



Another example is represented by Figure 4, where the box plots related to the NM_1 feature are shown, which shows how the box plots of human players are greater than the game bots. This feature is also due to the fact that human players tend to have a greater relationship with other players than game bots. Finally, in the example shown in Figure 5, the game plots relating to the NM_2 feature are shown. As in the previous examples we see that in human players the feature has a wider inter-quartile range than game bots, due to the fact that human players unlike game bots look for the shortest path when they have to reach a goal.

4 The proposed classification approach

The approach described by us in this paper achieves the classification of video game players between humans and bots through a sequence of operations, listed below and then described in detail:

1 initially the dataset is pre-processed to normalise the data, clean up the whole from spurious data and label

2 the supervised classification algorithm is trained before classification

3 the best features are selected.

The following subsection discusses each step in more details.

Figure 5 The box plot relating to the game bot and human distributions for the betweenness centrality feature belonging to the category NM (see online version for colours)

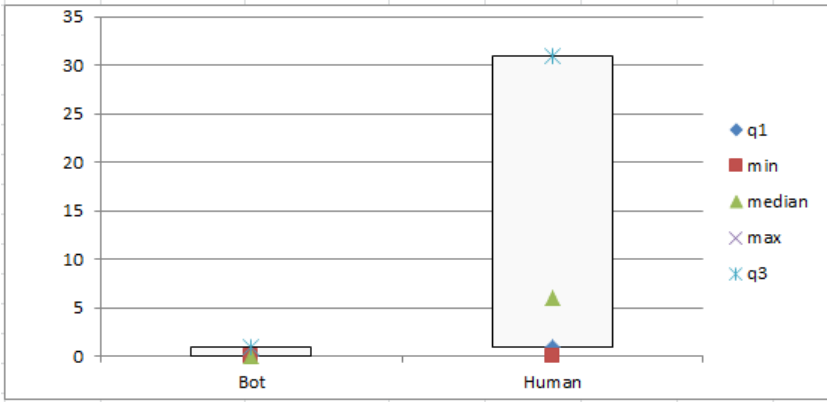
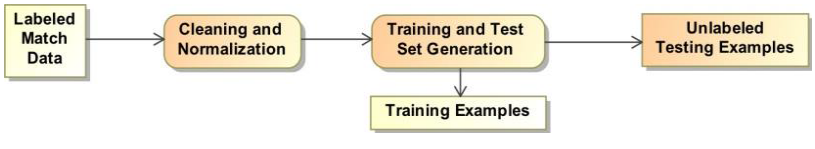


Figure 6 The pre-processing step (see online version for colours)



4.1 Pre-processing

As illustrated in Figure 6, data pre-processing consists of cleaning the raw data by eliminating incomplete acquisition sessions and normalising the data. From the cleaned up data the training set and the test set are obtained. All data are obviously labelled as relating to human players or game bots.

4.2 Classification

The classifications carried out in this paper are based on the classification algorithms described above, i.e., those belonging to the ML class (Canfora et al., 2013, 2015) and those to the DL class (Deng and Yu, 2014). All the algorithms belonging to these two classes are used, namely those described in Table 1 for the ML class and the MLP and CNN algorithms for the DL class. The DL algorithms are related to one layer and two layers. Operationally both the ML and DL algorithms can be described in Figure 7. The classification is applied to the features described above and belonging to the PI, PA, GA, SID and NM categories. In Section 5 we report the obtained results.

Figure 7 The classification step (see online version for colours)

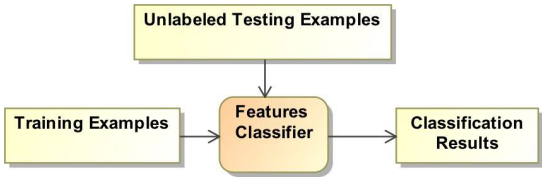
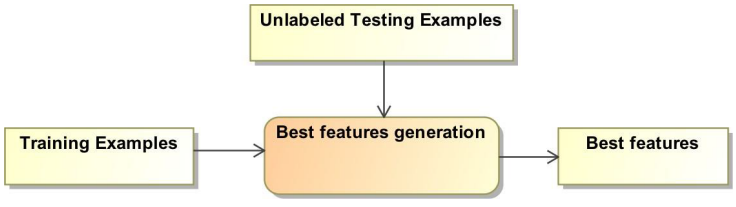


Figure 8 The feature selection process (see online version for colours)



4.3 Feature selection

In the context of the processing or classification of raw data, the extraction of features from raw data is necessary to reduce the quantity of data by eliminating redundant information. Redundant information is obtained by identifying the most important aspects of the data itself. Of all the features that can be defined and extracted from the data, some better identify the most important aspects of the data and give better results than others. In this paper we have used three feature selection schemes, namely the *BestFirst*, the *greedy stepwise* and the *ranker*. The first approach identifies the features that lead to the best results giving the learning technique and the metric used by the classification algorithm. The second approach selects the best features using an iterative *greedy* heuristic. The third feature selection approach ranks the effectiveness of each feature and

chooses the top ones for classification. To obtain the best results in each test, we have always used the best features based on an initial search, which can be schematised in Figure 8.

5 The evaluation and analysis

In the previous Section 2, in particular in Table 1 we have reported the ML and DL category algorithms used in this paper while in Section 3; in particular in Table 2 and Table 3, the features that describe respectively the behaviour of the players and the use of the network are listed. In Section 3 we also describe the used dataset.

We now describe the experimental tests carried out with the described algorithms and dataset. First we specify the accuracy measures used, then we report the results obtained.

5.1 Evaluation setting

The classifications were made using *Weka* (https://www.cs.waikato.ac.nz/ml/weka/), which is a software tool widely used to make classifications with ML algorithms. For the classifications with the DL algorithms we used the open-source *Deeplearning4j* (https://deeplearning4j.org/) library written in Java and distributed under the *Apache License 2.0* license. Classifications are evaluated using the following metrics: *precision*, *recall*, *F-measure* and *ROC area*. *Precision* and *Recall* measurements are calculated with the following equations:

$$Precision = \frac{t_p}{t_p + f_p}$$

$$Recall = \frac{t_p}{t_p + f_n}$$

where t_p is the number of true positives and f_n is the number of false negatives.

In other words, *Precision* is the proportion of correct positive classifications actually correct. *Recall* is the ratio between the number of correct classifications and the total number of results. In other words, *Recall* is the proportion of correct positive classification identified correctly. The *F-measure* is the weighted average of the *Precision* and *Recall* measurements:

$$F\text{-}Measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The last metric that we used in this paper is the *ROC area*. The ROC curve, also called receiver operating characteristic curve, represents the curve of true positives versus false positive at different classification thresholds. The *ROC area* is the area under the ROC curve, and measure the performance of classification across all possible classification thresholds. The validation of the classification algorithms was carried out according to the principles of *k*-fold cross-validation, which we summarise as follows. The set of original data are cleaned up and normalised. Then data is randomly divided into *k* subsets (k = 10). An iteration is performed on the *k* groups. At each iteration step, the current

group is used for testing the classification algorithm while all the other ($k - 1$) groups are used for its training. The final result is the average of the k classifications.

Calling M the raw data traces, B the game bot label and H the human player label, the dataset is formed by labelled traces (M, l). From raw data traces M, we derive the features vector $F \in Ry$, where y is the feature vector dimension. For instance, PI features has a $y = 4$ dimension, the PA features has a $y = 9$ dimension, GA features has a dimension $y = 2$, SID features ha a dimension $y = 1$ and NM features has a dimension equal to $y = 9$.

In compact form trainings and classifications can be described as follows:

1 build a training set $T \subset D$

2 build a testing set $T' = D \div T$

3 perform training using the training set T

4 perform classification using the testing set T'.

From the original dataset, two distinct datasets are obtained, one for training and the other for classification. Only 10% of the classification dataset is used for testing, while 90% of the training dataset is used, thus giving higher priority to training.

The testing and training phases were performed on a desktop with *Intel Core I5* processor and 4 GB of RAM memory. The operating system used is *64-bit Linux Mint*.

5.2 Classification results

The results of the ML supervised classification with PA, GA, SID and NM features are shown in Tables 4 and 5.

The results with PI features are the following. Precision is 0.764 and 0.95 with the *naïve Bayes multinomial text* and J48 algorithms respectively. The recall is 0.773 and 0.952 with the naïve Bayes multinomial and LMT algorithms respectively. With the PI features, the algorithm that provides the best results of precision is therefore the J48. Let's see for completeness all the results obtained with J48: precision and recall of 0.95 and 0.951 respectively. The F-measure is equal to 0.949 and the ROC curve is equal to 0.856.

The results with the PA features are the following. The *naïve Bayes multinomial* algorithm gives precision and recall of 0.858 and 0.851 respectively. The values of precision and recall obtained with *random forest* algorithm are of 0.954 and 0.955 respectively.

With PA features, the algorithm that provides the best results from precision is therefore *random forest*. All the results obtained with the *random forest* algorithm are: precision and recall of 0.954 and 0.955, F-measure of 0.953 and the ROC curve of 0.95.

The training phase of the *naive Bayes* algorithm is the fastest among all the others algorithms. However, *naive Bayes* algorithm provides the worst results, represented by a maximum F-measure of 0.92. The *naive Bayes* can therefore be used when not much precision is needed but a fast algorithm is needed.

The results obtained with the GA features are the following. The precision and recall are equal to 0.759 and 0.553 with the *naïve Bayes multinomial* algorithm, and equal to 0.928 and 0.973 with the LMT algorithm.

Table 4 Classification results: precision, recall, F-measure and ROC area for the first seven algorithms on all features of all groups

Algorithm	*Feature group*	*Precision*	*Recall*	*F-measure*	*ROC area*	*Time*
J48	PI	0.95	0.951	0.949	0.856	1.97
	PA	0.948	0.950	0.947	0.862	7.01
	GA	0.858	0.881	0.843	0.764	0.38
	SID	0.836	0.875	0.821	0.698	0.37
	NM	0.917	0.923	0.917	0.810	24.75
Decision stump	PI	0.942	0.944	0.941	0.823	0.21
	PA	0.934	0.936	0.935	0.830	0.58
	GA	0.764	0.874	0.816	0.721	0.05
	SID	0.764	0.874	0.816	0.690	0.06
	NM	0.871	0.884	0.875	0.673	0.99
Hoeffding tree	PI	0.941	0.944	0.941	0.872	0.3
	PA	0.942	0.944	0.942	0.872	0.94
	GA	0.854	0.880	0.839	0.783	0.17
	SID	0.826	0.874	0.823	0.710	0.32
	NM	0.892	0.903	0.891	0.769	2.68
Random forest	PI	0.952	0.953	0.950	0.895	37.8
	PA	0.954	0.955	0.953	0.95	57.23
	GA	0.820	0.855	0.833	0.766	17.71
	SID	0.805	0.863	0.822	0.688	8.31
	NM	0.923	0.928	0.923	0.858	42.42
Random tree	PI	0.917	0.916	0.916	0.814	0.64
	GA	0.820	0.854	0.833	0.758	0.42
	SID	0.805	0.865	0.822	0.682	0.14
	NM	0.886	0.887	0.887	0.751	0.69
MLP	PI	0.943	0.946	0.943	0.880	52.6
	PA	0.950	0.952	0.950	0.894	246.46
	GA	0.886	0.986	0.933	0.770	24.19
	SID	0.873	0.887	0.877	0.807	19.57
	NM	0.918	0.909	0.921	0.871	524.03
BayesNet	PI	0.939	0.942	0.940	0.876	0.72
	PA	0.941	0.942	0.941	0.896	1.8
	GA	0.897	0.899	0.898	0.840	0.71
	SID	0.882	0.893	0.885	0.820	0.19
	NM	0.875	0.708	0.756	0.830	4.45

Table 5 Classification results: precision, recall, F-measure and ROC area for the remaining algorithms on all feature groups

Algorithm	*Feature group*	*Precision*	*Recall*	*F-measure*	*ROC area*	*Time*
REP tree	PI	0.946	0.948	0.945	0.880	0.93
	PA	0.948	0.950	0.948	0.888	3.36
	GA	0.858	0.882	0.845	0.784	0.47
	SID	0.829	0.874	0.823	0.717	0.11
	NM	0.917	0.923	0.917	0.845	5.5
Naïve Bayes multinomial	PI	0.815	0.773	0.792	0.590	0.06
	PA	0.858	0.851	0.855	0.685	0.07
	GA	0.749	0.553	0.628	0.416	0.06
	SID	0.812	0.539	0.615	0.534	0.01
	NM	0.854	0.564	0.635	0.656	0.19
Logistic	PI	0.943	0.945	0.942	0.879	2.03
	PA	0.945	0.947	0.943	0.880	5.05
	GA	0.884	0.878	0.825	0.768	1.47
	SID	0.826	0.873	0.826	0.801	0.65
	NM	0.854	0.564	0.635	0.656	0.19
Ada Boost M1	PI	0.942	0.944	0.941	0.877	2.1
	PA	0.938	0.941	0.938	0.888	0.07
	GA	0.885	0.894	0.888	0.814	1.46
	SID	0.872	0.890	0.868	0.805	1.49
	NM	0.903	0.912	0.903	0.841	8.8
JRip	PI	0.949	0.951	0.947	0.839	33.57
	PA	0.950	0.952	0.950	0.848	62.61
	GA	0.899	0.908	0.899	0.721	26.51
	SID	0.886	0.900	0.885	0.669	29.6
	NM	0.939	0.982	0.960	0.771	133,42
LMT	PI	0.951	0.952	0.949	0.887	103.4
	PA	0.951	0.952	0.950	0.898	245.38
	GA	0.928	0.973	0.950	0.838	37.47
	SID	0.886	0.900	0.885	0.669	29.6
	NM	0.882	0.892	0.879	0.789	425.46
IBk	PI	0.927	0.927	0.927	0.834	0.01
	PA	0.927	0.927	0.927	0.830	0.01
	GA	0.888	0.886	0.887	0.748	0.01
	SID	0.865	0.863	0.864	0.696	0.01
	NM	0.895	0.895	0.895	0.762	0.01

Considering the GA features we can see that the worst performances are obtained with the naive Bayes algorithm and the best with the LMT algorithm. The values of precision and recall with the LMT algorithm are 0.928 and 0.973 respectively, while the F-measure is of 0.950 and the ROC area of 0.838.

Now consider the SID features. In this case the *naïve Bayes multinomial text* and *decision stump* algorithms provide a precision value of 0.764 while the *JRip* algorithm of 0.886. As for recall, we see that the *naïve Bayes multinomial* algorithm provides a value of 0.539 while the *JRip* algorithm provides a value of 0.900.

Considering again the SID features, we see that the best precision is obtained with the *JRip* algorithm and is equal to 0.886.

Now let's consider the NM features. The *naïve Bayes multinomial text* and *naïve Bayes* algorithms provide precision and recall values of 0.764 and 0.534 respectively, while the *JRip* algorithm provides precision and recall values of 0.939 and 0.982.

Let's now pass to the DL algorithms used in this paper that is Dl4jMlp classifier and neural networks. The precision values obtained with all the features are generally higher with neural networks than with the *Dl4jMlp classifier* algorithm. The only case in which this result is reversed is with the SID features. In fact, with SID features the *Dl4jMlp classifier* algorithm gives a precision value of 0.832 while the other algorithm stops at 0.764. Considering the computation time required to train the two algorithms, we see that *neural networks* always require lower time than the other, requiring only 12.66 seconds while training the *Dl4jMlp classifier* algorithm requires 185.59 (example relating to PI features).

A fundamental initial step of ML is feature definition, which is needed to define the dataset to be used for training and classification. It is performed by looking at the best performing set of features which can be defined on the raw dataset. The set of best features is obtained by searching the space of attributes defined on the row dataset for the best classification results.

Identifying the best features is important for many reasons. First, the best feature reduce overfitting because they reduce redundancy among the features, and it must be noted that less redundancy means less noise. It is important also because the best features allow to obtain the best possible results and it is finally important to assure that the training times are small.

Table 6 Precision, recall, F-measure and ROC area for classifying the feature categories, computed with two DL different classification algorithms

Category	*Algorithm*	*Precision*	*Recall*	*F-measure*	*ROC area*	*Time*
PI	Dl4jMlp classifier	0.930	0.933	0.931	0.852	185.59
	Neural network	0.942	0,.945	0.941	0.882	12.66
PA	Dl4jMlp classifier	0.924	0.928	0.925	0.833	185.54
	Neural network	0.947	0.949	0.946	0.893	10.15
GA	Dl4jMlp classifier	0.825	0.864	0.836	0.675	214.91
	Neural network	0.890	0.875	0.817	0.771	9.68
SID	Dl4jMlp classifier	0.832	0.871	0.839	0.776	221.42
	Neural network	0.764	0.874	0.816	0.803	5.23
NM	Dl4jMlp classifier	0.878	0.880	0.879	0.805	222.31
	Neural network	0.912	0.919	0.909	0.862	43.9

In this paper the space of attributes is searched with three heuristic approaches that, even if sub-optimal, are quite popular in the ML community, namely *BestFirst*, *greedy stepwise* and *ranker*.

Table 7 Precision, recall, F-measure and roc area for classifying the best feature selected (PI and PA groups), computed with 16 different classification algorithms

Algorithm	*Feature group*	*Precision*	*Recall*	*F-measure*	*ROC area*	*Time*
J48	PI	0.954	0.984	0.969	0.824	0.36
	PA	0.958	0.986	0.972	0.870	0.48
Decision Stump	PI	0.954	0.984	0.969	0.823	0.05
	PA	0.957	0.971	0.964	0.830	0.15
Hoeffding tree	PI	0.953	0.985	0.968	0.845	0.41
	PA	0.957	0.985	0.971	0.882	0.25
Random forest	PI	0.943	0.944	0.943	0.831	15.06
	PA	0.960	0.986	0.973	0.890	56.67
Random tree	PI	0.943	0.944	0.943	0.774	0.37
	PA	0.954	0.954	0.954	0.818	0.98
REP tree	PI	0.954	0.984	0.969	0.837	0.33
	PA	0.959	0.985	0.972	0.889	0.74
BayesNet	PI	0.954	0.984	0.969	0.849	0.23
	PA	0.957	0.976	0.966	0.881	0.32
Naïve Bayes	PI	0.956	0.976	0.966	0.847	0.1
	PA	0.955	0.973	0.964	0.871	0.1
Naïve Bayes multinomial	PI	0.874	1.000	0.933	0.500	0.06
	PA	0.907	0.851	0.878	0.499	0.01
Naïve Bayes multinomial text	PI	0.874	1.000	0.933	0.500	0.07
	PA	0.874	1.000	0.933	0.500	0.01
Logistic	PI	0.953	0.984	0.969	0.844	0.75
	PA	0.947	0.988	0.967	0.851	0.89
MLP	PI	0.953	0.984	0.968	0.846	11.67
	PA	0.954	0.984	0.969	0.861	24.45
Adaboost M1	PI	0.954	0.984	0.969	0.850	0.59
	PA	0.957	0.971	0.964	0.865	1.14
JRip	PI	0.954	0.984	0.969	0.825	1.84
	PA	0.954	0.986	0.970	0.826	10.57
IBk	PI	0.943	0.944	0.943	0.776	0.01
	PA	0.949	0.951	0.950	0.799	0.03
LMT	PI	0.954	0.984	0.969	0.845	8.38
	PA	0.954	0.987	0.970	0.865	24.63

The *BestFirst* approach navigates the space of attribute using a best first strategy, while the *greedy* approach uses a greedy heuristic. Finally, the *ranker* is the simpler approach, as it simply ranks the effectiveness of each feature, and selecting only the top in rank does not requires any search operation.

All the sets of features are reported in Table 2. Best features selection identifies the same set of features as we describe below. As for the *PI* category features, PI_2 features are those that lead to better discrimination between human players and game bots. As for *PA*, the features that lead to the best discrimination are PA_1, PA_2, PA_3 and PA_4. What we want to see now is whether reducing all the sets of features described in Table 2, that is the sets PI_1–PI_4 for *PI* features, PA_1–PA_{10} for *PA* features, and sets GA_1–GA_2, SID_1 and NM_1–NM_9 for all others, with only PI_2 set of category PI and PA_1–PA_4 of the PA category, better results are obtained.

For this purpose, we train and use different classifiers and compare the results, which are presented in Table 7.

All the algorithms we use to select the best features identify the same set of features as we describe below.

From Table 7 we can see that for the PI category the best precision value is 0.954, obtained with the following classifiers: J48, *decision stump*, *REP tree*, *BayesNet*, *AdaBoostMI* and *JRip*. From the same table we see that the best recall value is 1, obtained with the *naïve Bayes multinomial* and *naïve Bayes multinomial text* classifiers. If we compare the improvement of the results obtained with this reduced set of features compared to the results obtained with the complete set and described in Tables 4 and 5, we see that the precision has improved from 0.95 to 0.954 and the recall from 0.951 to 1.00.

Always from Table 7 we see then that for the PA category the best precision is 0.960 obtained with the *random forest* classifier while the best recall value is 1 obtained with the *naïve Bayes multinomial text* classifier. Comparing the improvement obtained before we see that the precision has increased from 0.954 to 0.958 and the recall from 0.955 to 1.

To complete the test phase we now use the DL classifiers that we have considered in this paper, that is *neural networks* and *convolutional networks* and we compare the results with those obtained with the ML classifiers that we have just described. The features used are the two subsets of PI and PA category features that have proven to be optimal from tests with ML classifiers. The results are shown in Table 8.

Table 8 Classification results: precision, recall, F-measure and ROC area obtained using the best feature categories, computed with two DL classification algorithms on a single and dual layer networks

Category	*Algorithm*	*Precision*	*Recall*	*F-measure*	*ROC area*	*Time*
PI_{best}	Dl4jMlp classifier	0.951	0.984	0.968	0.845	240.85
	Neural network	0.949	0.988	0.968	0.844	7.12
	Neural network (two-layer)	0.987	0.990	0.987	0.872	16.74
PA_{best}	Dl4jMlp classifier	0.945	0.988	0.966	0.848	229.13
	Neural network	0.954	0.984	0.969	0.853	8.05
	Neural network (two-layer)	0.992	0.994	0.996	0.889	15.34

Even with tests with the DL classifiers the two subsets of features lead to a performance increase. In particular, the Precision value obtained with the *Dl4jMlp classifier* algorithm obtains a precision value of 0.984 with only the PI2 feature while with the complete set of features it is 0.930. The recall is equal to 0.984 while with the whole set of features it is 0.945.

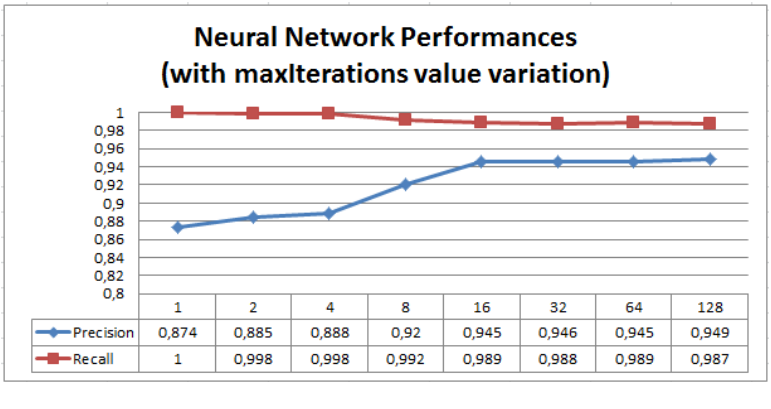
Using the *neural network* algorithm, a precision of 0.949 is obtained, while it is 0.942 with the whole set of features, and a recall of 0.988 while with the whole set of features it is of 0.945.

Now consider the optimal subset of features of the PA category, which we call PA_{best}. In this case the *Dl4jMlp classifier* algorithm obtains a precision of 0.945 while it was 0.924 as reported in Table 6. The recall with the *Dl4jMlp classifier* algorithm is 0.988 while it was 0.928.

Using the PA_{best} subset, the *neural networks* algorithm obtains a precision of 0.954 while it was 0.947 in Table 6. The recall is equal to 0.984, compared to 0.949 in Table 6.

In Table 8, we also report the results obtained using neural networks with two layers. The results shown in Table 8 show that the performances obtained with the neural network algorithm with two layers exceed those obtained with all the other algorithms, both all the ML category algorithms and the CNN type algorithm considered, i.e., *Dl4jMlp classifier*. In fact, we obtain, with the PI_{best} features, a precision of 0.987 and a recall of 0.990, while with the PA_{best} features we have a precision of 0.992 and a recall of 0.994. The calculation time of the neural network with two layers used is 16.74 and 15.34 seconds respectively with the features PI_{best} and PA_{best}. We also observe that the F-measure is 0.872 and 0.996 with the two features. We conclude that the best classification between human players and game bots is obtained with a neural network with two layers and with the features PI_{best} and PA_{best}.

Figure 9 Precision and recall trends for the *neural network* DL algorithm with one-layer when the *MaxIterations* parameter is ranging between 1 and 128 (see online version for colours)



	1	2	4	8	16	32	64	128
Precision	0,874	0,885	0,888	0,92	0,945	0,946	0,945	0,949
Recall	1	0,998	0,998	0,992	0,989	0,988	0,989	0,987

Having established that the best classifier is made up of neural networks with two layers, there is another level of optimisations that should be made, that is, the determination of the learning parameters of the neural network. Since network learning is carried out with the backpropagation algorithm, which can still be used with two hidden layers (if the number of layers were greater, the backpropagation algorithm would no longer be usable due to the vanishing gradient problem), the learning parameters, which are described in

Table 9, should be optimised. However, since the optimisation of these parameters is quite complex, we adopt an empirical approach which is summarised below. In Figure 9, we report the trend of the neural network with one layer as the *MaxIterations* parameter changes. From Figure 9 we see that the precision increases from 0.874 to 0.949 as the value of *MaxIterations* increases from 1 to 128 while recall decreases from 1 to 0.987. The same behaviour is obtained with the other parameters. Since the increase in precision is much more significant than the decrease in recall, we set the value of the *MaxIterations* parameter to 128, so in Table 8 the value of recall is 0.988.

Table 9 The algorithm parameters with the considered values in the neural network and Dl4jMlp classifier evaluations

Parameter	*Neural network*	*Dl4jMlp classifier*
batchSize	100	100
hiddenLayers	100	
hiddenLayersDropoutRate	0.5	
inputLayerDropoutRate	0.2	
inputWidth	0	
learningRate	0.0	
maxIterations	1,000	
numberOfEpochs		10
numDecimalPlaces		2
seed		0.2

Note: We considered the same parameter values for the classifications with one and two layers.

6 Related work

In recent years there has been a major increase in the MMORPG video game market. Video game manufacturers have therefore tried to improve video game performance so that they are increasingly attractive to increase the market.

Consequently, the number of game bots has continuously increased and therefore also the amount and complexity of algorithms to detect their presence, as reported in Kang et al. (2016).

The game bot detection algorithms described in Kang et al. (2016) are focused on the approaches they perform on the game server. These approaches analyse the log files on the game server using data mining algorithms, and are normally preferred to the approaches they run directly on the client because they allow to detect and block game bots without interfering with the players. They also do not require players to upload, update and run detection programs.

The approaches they perform on the game server are normally divided into the following groups: social activities, sequences, mobile path, frequency of action, similarity and gold farming. The approach described in this paper is based on the analysis of social interactions between videogame players and therefore can be considered belonging to the group of social activities (Varvello and Voelker, 2010).

The hypothesis behind these approaches is that the discriminating factor between human players and game bots is player behaviour.

Another example of a behavioural approach is described in Oh et al. (2013) where the game bots detection algorithm is based on the analysis of social networks as used by players. First of all, from the use of social networks during the game, features are extracted which are previously compared with those obtained from human players and game bots. The algorithm described in Kang et al. (2013) analyses the logs generated during a game session and compares them with those generated separately by human players and game bots. Behaviours are identified through the use of previously defined thresholds.

The article described in Kim et al. (2005) does not use social interactions but the type and frequency of windows events that are used as features. The detection of game bots takes place through learning algorithms.

The papers presented in Chen et al. (2008, 2009) describe algorithms that detect game bots by analysing the movements of avatars which are considered as features. The basis of these works is the hypothesis that human players and game bots use different trajectories of the movements of the avatars. In Chen et al. (2008), the authors proposed an algorithm to estimate the trajectories of the movements. The algorithm was validated using a different game than the one used in this paper (quake 2). The results of the validation were that more than 95% of the movements were actually identified on a 200 second game track.

From the network traffic generated during a game session, features can be extracted that can be compared with those produced during game sessions with human players. This approach has been followed also in the work described in Chen et al. (2006).

Another possibility is to consider the MMORPG game environment from the point of view of economic transactions (in virtual currency) and to detect anomalous exchanges. This approach is the basis of the work described in Kwon et al. (2017) which proposes a method to detect gold farming groups (GFG). The transactions are described in a graph from which features used for the identification of the GFG are derived. The main contribution of Kwon et al. (2017) is to propose recommendations for defence against CFGs without affecting the economic environment of the game.

The paper reported in Kim et al. (2017) describes a system that detects thefts to protect players from malicious users. The system of Kim et al. (2017) runs on the game server by analysing the log files. The measured accuracy is 88%.

The principle according to which player behaviour is analysed to discriminate between human players and game bots is described in many papers such as (Thawonmas et al., 2008; Kashifuji, 2008; Hilaire et al., 2010; Mishima et al., 2013; Kang et al., 2016). These works share the defect of using only one or two features extracted from the game environment rather than the players' behaviour, and therefore are dependent on the type of game itself (Chung et al., 2015).

Starting from this defect, in Chung et al. (2015) it is proposed to use a greater number of features, both dependent on the game and on the players.

Although the work proposed in Chung et al. (2015) is similar to that described in this paper, the performance in terms of precision and recall is lower than ours.

In this work, unlike Chung et al. (2015), we use a selection of features to identify those that are able to best discriminate human players from game bots. We also report the times required to detect a game bot by its behaviour.

In this paper we describe the results obtained with many ML and DL algorithms for the detection of game bots.

DL is widely used in many application fields such as voice recognition and natural language processing but does not detect game bots.

7 Threats to validity

The validity of the detection approach proposed in this paper could be attacked from several points of view. First of all, the measurements made are based on a labelled dataset, so the results depend on the correctness of the labelling that may be related to a human player or a game bot. We have tried to reduce these types of errors by using a dataset provided by the company that produced the Aion video game. To further increase the validity of the dataset, each label that turned out to be game bot was manually checked by a specialist.

Another possible element that could affect the validity of the method proposed in this paper is the generalisation of the approach, i.e., the fact that its application to datasets other than Ainon can give different results. This problem is reduced by the fact that we consider features that characterise the behaviour of the players and do not depend on the videogame considered.

8 Conclusions and future work

Current video games are shared by community of distributed players connected through the internet. During the game, they require multiple operations such as the organisation of groups of players, the activation of social interactions between them, acquisition of virtual money and other complex operations. This scenario of distributed resource sharing is very interesting to study from different points of view. As their popularity has increased, the number of malicious players who unfairly want to acquire points to outperform other players has also increased. Malicious players have introduced robotic players into the game that simulate the behaviour of human players and easily outperform human players. Robotic players, called game bots, in fact do not get tired and therefore they do not need to stop. Therefore, the need arises to develop algorithms to counter the spread of game bots for eliminating them from the game.

In this paper we have proposed a system that, by defining features that characterise player behaviour, is able to detect game bots by discriminating them from human players.

To verify the algorithm, we carried out extensive experimentation using many supervised ML and DL algorithms. The result is that the best classification algorithms are neural networks with two hidden layers. The Precision and Recall values obtained are 0.992 and 0.994 respectively. The time required to train the models is 15.34 seconds using a 2.3 GHz i5 processor with 4 GB of RAM. The features used that provide better results are two subsets of behavioural features.

In the future we will extend these results to detect robotic participants in some social networks using behavioural features.

We will also use process mining techniques (Bernardi et al., 2016) and formal methods (De Francesco et al., 2016) to distinguish robotic users from human users.

Finally, we will apply the proposed approach in various big data problems (for example, Cuzzocrea and Wang, 2007; Bonifati and Cuzzocrea, 2006; Cuzzocrea and Bertino, 2011; Cuzzocrea and Russo, 2009; Chatzimilioudis et al., 2013; Cuzzocrea, 2006; Cuzzocrea and Song, 2014; Campan et al., 2017).

Acknowledgments

This research has been conducted in the context of the Excellence Chair in Computer Engineering – Big Data Managements and Analytics at LORIA, Nancy, France. This work has been partially supported by H2020 EU-funded project NeCS and by the French PIA project 'Lorraine Université d'Excellence', reference ANR-15-IDEX-04-LUE. Authors are extremely grateful to Marta Cimitile and Mario Luca Bernardi for their contributions in early versions of this work.

References

Adams, E. (2014) *Fundamentals of Game Design*, Pearson Education, London, UK.

Aha, D. and Kibler, D. (1991) 'Instance-based learning algorithms', *Machine Learning*, Vol. 6, pp.37–66.

Aziz, A., Lai, W-L. and Manni, J. (2017) *System and Method for Bot Detection*, April 18, US Patent 9,628,498.

Bengio, Y. et al. (2009) 'Learning deep architectures for AI', *Foundations and Trends in Machine Learning*, Vol. 2, No. 1, pp.1–127.

Bernardi, M.L., Cimitile, M., Di Francescomarino, C. and Maggi, F.M. (2016) 'Do activity lifecycles affect the validity of a business rule in a business process?', *Inf. Syst.*, December, Vol. 62,No. C, pp.42–59.

Bernardi, M.L., Cimitile, M. and Mercaldo, F. (2017a) 'A time series classification approach to game bot detection', in *Proceeding of the 7th ACM International Conference on Web Intelligence, Mining and Semantics*, pp.512–519.

Bernardi, M.L., Cimitile, M., Distante, D. and Mercaldo, F. (2017b) 'Game bot detection in online role player game through behavioural features', in *Proceeding of the 12th International Conference on Software Technologies*.

Bonifati, A. and Cuzzocrea, A. (2006) 'Storing and retrieving xpath fragments in structured P2P networks', *Data Knowl. Eng.*, Vol. 59, No. 2, pp.247–269.

Breiman, L. (2001) 'Random forests', *Mach. Learn.*, October, Vol. 45, No. 1, pp.5–32.

Buhrke, E.R. and LoCicero, J.L. (1992) 'A learning algorithm for multi-layer perceptron networks with nondifferentiable nonlinearities', in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, June, Vol. 1, pp.944–949.

Campan, A., Cuzzocrea, A. and Truta, T.M. (2017) 'Fighting fake news spread in online social networks: Actual trends and future research directions', in *Proceedings of IEEE Big Data 2017*, pp.4453–4457.

Canfora, G., De Lorenzo, A., Medvet, E., Mercaldo, F. and Visaggio, C.A. (2015) 'Effectiveness of opcode ngrams for detection of multi-family android malware', *2015 10th International Conference on In Availability, Reliability and Security (ARES)*, IEEE, pp.333–340.

Canfora, G., Mercaldo, F. and Visaggio, C.A. (2013) 'A classifier of malicious android applications', in *2013 Eighth International Conference on Availability, Reliability and Security (ARES)*, IEEE, pp.607–614.

Chatzimilioudis, G., Cuzzocrea, A., Gunopulos, D. and Mamoulis, N. (2013) 'A novel distributed framework for optimizing query routing trees in wireless sensor networks via optimal operator placement', *J. Comput. Syst. Sci.*, Vol. 79, No. 3, pp.349–368.

Chen, K-T., Jiang, J-W., Huang, P., Chu, H-H., Lei, C-L. and Chen, W-C. (2006) 'Identifying MMORPG bots: a traffic analysis approach', *in Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACE '06*, ACM, New York, NY, USA.

Chen, K-T., Liao, A., Pao, H-K. and Chu, H-H. (2009) 'Game bot detection based on avatar trajectory', *Entertainment Computing-ICEC 2008*, pp.94–105.

Chen, K-T., Pao, H-K.K. and Chang, H-C. (2008) 'Game bot identification based on manifold learning', in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, ACM, pp. 21–26.

Chen, Y-C., Chen, P.S., Song, R. and Korba, L. (2004) 'Online gaming crime and security issue-cases and countermeasures from Taiwan', in *PST*, pp.131–136.

Chung, Y., Park, C.Y., Kim, N-R., Cho, H., Yoon, T.B., Lee, H. and Lee, J-H. (2015) 'A behavior analysis-based game bot detection approach considering various play styles', *CoRR*, abs/1509.02458.

Cocar, M., Harris, R. and Khmelevsky, Y. (2017) 'Utilizing minecraft bots to optimize game server performance and deployment', in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, IEEE, pp.1–5.

Cohen, W.W. (1995) 'Fast effective rule induction', in *Twelfth International Conference on Machine Learning*, Morgan Kaufmann, pp.115–123.

Collobert, R. and Weston, J. (2008) 'A unified architecture for natural language processing: deep neural networks with multitask learning', in *Proceedings of the 25th International Conference on Machine Learning*, ACM, pp.160–167.

Cuzzocrea, A. (2006) 'Combining multidimensional user models and knowledge representation and management techniques for making web services knowledge-aware', *Web Intelligence and Agent Systems*, Vol. 4, No. 3, pp.289–312.

Cuzzocrea, A. and Bertino, E. (2011) 'Privacy preserving OLAP over distributed XML data: a theoretically-sound secure-multiparty-computation approach', *J. Comput. Syst. Sci.*, Vol. 77, No. 6, pp.965–987.

Cuzzocrea, A. and Russo, V. (2009) 'Privacy preserving OLAP and OLAP security', in *Encyclopedia of Data Warehousing and Mining*, 2nd ed., Vol. 4, pp.1575–1581, Hershey, PA, USA.

Cuzzocrea, A. and Song, I-Y. (2014) 'Big graph analytics: the state of the art and future research agenda', in *Proceedings of ACM DOLAP 2014*, pp.99–101.

Cuzzocrea, A. and Wang, W. (2007) 'Approximate range-sum query answering on data cubes with probabilistic guarantees', *J. Intell. Inf. Syst.*, Vol. 28, No. 2, pp.161–197.

Dahl, G.E., Yu, D., Deng, L. and Acero, A. (2012) 'Context-dependent pretrained deep neural networks for large-vocabulary speech recognition', *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 1, pp.30–42.

De Francesco, N., Lettieri, G., Santone, A. and Vaglini, G. (2016) 'Heuristic search for equivalence checking', *Software and System Modeling*, Vol. 15, No. 2, pp.513–530.

Deng, L. and Yu, D. (2014) *Deep Learning: Methods and Applications*, Now Publishers Inc., Hanover, MA, USA.

Deng, L., Yu, D. et al. (2014) 'Deep learning: methods and applications', *Foundations and Trends in Signal Processing*, Vol. 7, Nos. 3–4, pp.197–387.

Esmaeili, H. and Woods, P.C. (2016) 'Calm down buddy! it's just a game: behavioral patterns observed among teamwork MMO participants in wargaming's world of tanks', in *22nd International Conference on Virtual System & Multimedia, VSMM 2016*, Kuala Lumpur, Malaysia, 17–21 October, pp.1–11.

Fernández-Ares, A., Mora, A.M., Garcıa-Sánchez, P., Castillo, P.A. and Merelo, J.J. (2017) 'Analysing the influence of the fitness function on genetically programmed bots for a real-time strategy game', *Entertainment Computing*, Vol. 18, pp.15–29.

Freedman, D.A. (2009) *Statistical Models: Theory and Practice*, Cambridge University Press, Cambridge, UK.

Friedman, N., Geiger, D. and Goldszmidt, M. (1997) 'Bayesian network classifiers', *Machine Learning*, Vol. 29, No. 2, pp.131–163.

Gardner, M.W. and Dorling, S.R. (1998) 'Artificial neural networks (the multilayer perceptron) a review of applications in the atmospheric sciences', *Atmospheric Environment*, Vol. 32, No. 14, pp.2627–2636.

Griffiths, M.D., Davies, M.N.O. and Chappell, D. (2004) 'Online computer gaming: a comparison of adolescent and adult gamers', *Journal of Adolescence*, Vol. 27, No. 1, pp.87–96.

Hilaire, S., Kim, H-C. and Kim, C-K. (2010) 'How to deal with bot scum in MMORPGS?', *2010 IEEE International Workshop Technical Committee on In Communications Quality and Reliability (CQR)*, IEEE, pp.1–6.

Hingston, P. (2009) 'A turing test for computer game bots', *IEEE Transactions on Computational Intelligence and AI in Games*, September, Vol. 1, No. 3, pp.169–186.

Hoeglinger, S. and Pears, R. (2007) 'Use of hoeffding trees in concept based data stream mining', in *2007 Third International Conference on Information and Automation for Sustainability*, December, pp.57–62.

Hung, C-Y. et al. (2017) 'Comparing deep neural network and other machine learning algorithms for stroke prediction in a large-scale population-based electronic medical claims database', *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2017*.

Iba, W. and Langley, P. (1992) 'Induction of one-level decision trees', in *Proceedings of the Ninth International Workshop on Machine Learning, ML '92*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp.233–240.

Ioannidou, A., Chatzilari, E., Nikolopoulos, S. and Kompatsiaris, I. (2017) 'Deep learning advances in computer vision with 3d data: a survey', *ACM Comput. Surv.*, April, Vol. 50, No. 2, pp.20:1–20:38.

John, G.H. and Langley, P. (1995) 'Estimating continuous distributions in Bayesian classifiers', in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, UAI'95*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp.338–345.

Kang, A.R., Jeong, S.H., Mohaisen, A. and Kim, H.K. (2016) 'Multimodal game bot detection using user behavioral characteristics', *SpringerPlus*, Vol. 5, No. 1, p.523.

Kang, A.R., Woo, J., Park, J. and Kim, H.K. (2013) 'Online game bot detection based on party-play log analysis, *Computers & Mathematics with Applications*, Vol. 65, No. 9, pp.1384–1395.

Kashifuji, Y. (2008) 'Detection of MMORPG bots based on behavior analysis', *ACE*, Vol. 2008, p.4.

Kégl, B. (2013) 'The return of adaboost.mh: multi-class hamming trees', *CoRR*, abs/1312.6086.

Kibriya, A.M., Frank, E., Pfahringer, B. and Holmes, G. (2004) 'Multinomial naive Bayes for text categorization revisited', in *Proceedings of the 17th Australian Joint Conference on Advances in Artificial Intelligence, AI'04*, Springer-Verlag, Berlin, Heidelberg, pp.488–499.

Kim, H., Hong, S. and Kim, J. (2005) 'Detection of auto programs for MMORPGS', in *Australasian Joint Conference on Artificial Intelligence*, Springer, pp.1281–1284.

Kim, H., Yang, S. and Kim, H.K. (2017) 'Crime scene re-investigation: a postmortem analysis of game account stealers' behaviors', *CoRR*, abs/1705.00242.

Kim, P. (2017) 'Convolutional neural network', in *MATLAB Deep Learning*, pp.121–147, Springer, Berlin, Germany.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) 'Imagenet classification with deep convolutional neural networks', in *Advances in Neural Information Processing Systems*, pp.1097–1105.

Kwon, H., Mohaisen, A., Woo, J., Kim, Y., Lee, E. and Kim, H.K. (2017) 'Crime scene reconstruction: online gold farming network analysis', *IEEE Trans. Information Forensics and Security*, Vol. 12, No. 3, pp.544–556.

Landwehr, N., Hall, M. and Frank, E. (2005) 'Logistic model trees', *Machine Learning*, Vol. 59, No. 1, pp.161–205.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, Vol. 521, No. 7553, pp.436–444.

Lee, S. and Shimoji, S. (1993) 'Bayesnet: Bayesian classification network based on biased random competition using Gaussian kernels', in *IEEE International Conference on Neural Networks*, Vol. 3, pp.1354–1359.

Lo, N.W. and Chen, S-H. (2008) 'A study of anti-robot agent mechanisms and process on online games', in *2008 IEEE International Conference on Intelligence and Security Informatics*, June, pp.203–205.

McCallum, A. and Nigam, K. (1998) 'A comparison of event models for naive bayes text classification', *AAAI Workshop on Learning for Text Categorization*, pp.41–48.

Mihăescu, M.C., Popescu, P.S. and Burdescu, D.D. (2015) 'J48 list ranker based on advanced classifier decision tree induction', *Int. J. Comput. Intell. Stud.*, November, Vol. 4, Nos. 3/4, pp.313–324.

Mishima, Y., Fukuda, K. and Esaki, H. (2013) 'An analysis of players and bots behaviors in MMORPG', in *IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, IEEE, pp.870–876.

Mitchell, T.M. (1997) *Machine Learning*, 1st ed., McGraw-Hill, Inc., New York, NY, USA.

Nadiammai, G.V. and Hemalatha, M. (2013) *Performance Analysis of Tree Based Classification Algorithms for Intrusion Detection System*, pp.82–89, Springer International Publishing, Cham.

Oh, J., Borbora, Z.H., Sharma, D. and Srivastava, J. (2013) 'Bot detection based on social interactions in MMORPGS', in *2013 International Conference on, Social Computing (SocialCom)*, IEEE, pp.536–543.

Paulson, R.A. and Weber, J.E. (2006) 'Cyberextortion: an overview of distributed denial of service attacks against online gaming companies', *Issues in Information Systems*, Vol. 7, No. 2, pp.52–56.

Quandt, T. and Kröger, S. (2013) *Multiplayer: The Social Aspects of Digital Gaming*, Vol. 3, Routledge, Routledge Studies in European Communication Research and Education, Taylor & Francis, London, UK.

Rav, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B. and Yang, G.Z. (2017) 'Deep learning for health informatics', *IEEE Journal of Biomedical and Health Informatics*, January, Vol. 21, No. 1, pp.4–21.

Ruck, D.W., Rogers, S.K. and Kabrisky, M. (1990) 'Feature selection using a multilayer perceptron', *Journal of Neural Network Computing*, Vol. 2, No. 2, pp.40–48.

Seay, A.F., Jerome, W.J., Lee, K.S. and Kraut, R.E. (2004) 'Project massive: a study of online gaming communities', in *CHI'04 Extended Abstracts on Human Factors in Computing Systems*, ACM pp.1421–1424.

Taylor, T.L. (2009) *Play between Worlds: Exploring Online Game Culture*, MIT Press, Cambridge, MA, USA.

Thawonmas, R., Kashifuji, Y. and Chen, K-T. (2008) 'Detection of MMORPG bots based on behavior analysis', in *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACM, pp.91–94.

Varvello, M. and Voelker, G.M. (2010) 'Second life: a social network of humans and bots', in *Proceedings of the 20th International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV '10*, ACM, New York, NY, USA, pp.9–14.

Wang, Y., Zhang, Y., Hei, X., Ji, W. and Ma, W. (2017) 'Game strategies for distributed denial of service defense in the cloud of things', *Journal of Communications and Information Networks*, Vol. 1, No. 4, pp.143–155.

Wellman, B. and Gulia, M. (1999) 'Virtual communities as communities', *Communities in Cyberspace*, pp.167–194, Taylor & Francis, London, UK.

Yampolskiy, R.V. and Govindaraju, V. (2008) 'Embedded noninteractive continuous bot detection', *Computers in Entertainment (CIE)*, Vol. 5, No. 4, p.7.

Yee, N. (2008) 'Maps of digital desires: exploring the topography of gender and play in online games', *Beyond Barbie and Mortal Kombat: New Perspectives on Gender and Gaming*, pp.83–96, Cambridge, MA, USA.

Zhao, Y. and Zhang, Y. (2008) 'Comparison of decision tree methods for finding active objects', *Advances in Space Research*, Vol. 41, No. 12, pp.1955–1959.