Bot detection using Behavioral Analysis in MMORPG

MSc Research Project
MSc Data Analytics

Vino Wilson
Student ID: x18124801

School of Computing
National College of Ireland

Supervisor: Prof. Jorge Basilio

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vino Wilson
Student ID:	x18124801
Programme:	MSc Data Analytics
Year:	2019-20
Module:	MSc Research Project
Supervisor:	Prof. Jorge Basilio
Submission Due Date:	23/04/2019
Project Title:	Bot detection using Behavioral Analysis in MMORPG
Word Count:	4862
Page Count:	12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	26th May 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Bot detection using Behavioral Analysis in MMORPG

Vino Wilson
x18124801

Abstract

Gaming is one of the popular industries among all the others, especially online and MMORPG games and people around the globe are spending money on different games. Every game has its own features and gameplay and to increase the user-experience has been a challenging factor for the companies as Bots among the humans have populated in most of the games. For this purpose, the paper focuses on the behavioral of the player to detect and differentiate between human and bot players. Feature selection methods was used by which features were selected as per the rank importance for training the data. Algorithms such as Random Forest, Naïve Bayes, Ensemble technique and Generalized Liner Model were used to fit the data. Random forest indicated the best performance with an accuracy of 96 percent.

Keyword: Feature Selection, Classification algorithm, Machine Learning, MMORPG

1 Introduction

Gaming sector is one of the largest and growing industry. The global gaming market have a net worth of 152 billions dollar including all the platform with 68.5 billion part directly from Mobile games sector. Millions of people across the globe are attracted to Massively Multiplayer Online Role-Playing Games and Batchelors 2018. Adding millions of participants one can imagine the amount of players creating their own way of being popular over the internet for the game or making some money over winning. As there is increase in number of players online games have offered monthly/yearly subscription which offers extra benefits and points giving users the better experience than the normal users. So, it is a challenging part for the developers of the game to meet the nearby expectation of the user.

As there is increase in humans in gaming sector there is also increase in bots which has been a difficult part for the companies to deal with it. Giant company such as Blizzard Entertainment have already started releasing the games which include anti-cheat software. This software is Valve-Anti Cheat which detects whether it is bot or human by checking if there is any code executed by the user and automatically bans the user from entering the game. But there is a disadvantage that is the software only looks for the exact code which is been entered into their system and if there is any changes will not be detected by the software. (McCracken, 2018)

There is great increase in bot activity used by the users in tactic way. Bots are been deployed in games to gain experience counts, in-game money (which is gold) and to increase level this method is called FARMING. Humans tend to play 10-12 hours a day which will gain half the experience count as compared to bots which plays 24 hours the whole year and gain double the experience against humans which directly makes user experience less and more higher level player spawns in time. Bots are increasing on day to day note in games and which have been reported by many gamers and to detect these bots there are many software which are developed to restrict the entry in game by the developers but there is always a new way to breach this anti-cheat engines.

2 Research Question

1. Will using multiple models help to increase the detection rate?
2. To what point the bot detection false positive rate can be reduced?

3 Literature Review

Bot is a gaming term which is used to define a character that is controlled by the computer code system (Techopedia, 2019). In other words, bots are Non-player characters (NPCs) which are controlled by the program created by third-party. They involve in fights and other activity which are generally done by the humans and against other bots as well.

(Lee, et al., 2016) states that bots are deployed by the players which intent to gain more experience and rating as compared to the rest of the players which is nowadays widely used in MMORPG games which is a disadvantage to the original players.

Cheat codes were created during the time of making video games. The developers themselves have created the cheat codes for the testing purpose which gives whole access to unlimited count of all the features to see the mechanism of working in order to make the difficulty of the game accordingly. There are multiple ways of cheating. Aimbots is a type of cheating where enemies can direct take a headshot, or the body part shot which take high damage this is mostly done in war games which this option is available in some games or one uses cheat software. Cheat software also contains many other level exploiting techniques where a player can see through the walls where the opponents are spawned. The other one more way is by gaining more experience in terms of points or in game cash which allows player to buy or enter into royal features which a normal player can't access. (Wendel, 2012).

There are many approaches proposed for bot detection in game. These approaches can be classified into server-side client-side and network side. Client-side servers are used by most of the gaming companies by the disadvantage is that it can decrease the system performance by increase in bot developed. Network side creates problem such as lag in game due to overload and high network traffic which effects the gaming experience of the user. As there is increase in bot on client side, data mining techniques are used in server-side method which detects whether the user is bot or human and directly bans the bot on client side without executing any kind of code or program. (Kang, et al., 2016)

Based on the analysis check carried out by Action analysis it is noticed that bots are more as compared to humans in the game. This approach shows accuracy rate high but the negative part is that it does not show or confuses in between the actual players who play for a long time and the bots which are tend to play for a longer time. (Kang, etal., 2016).

The other analysis was on social activity based which uses different features based on the social network to compare between humans and bots. The results are analyzed based on the graphs generated by social networks. It is observed that bots and humans use social network links for different purpose, bots may use it for information or for exchange of currency as humans use this network for multiple purpose. The drawback to this approach is that it can only detect bots in group or multiple party squad and not in solo mode. (Kang,et al., 2016). Another approach was proposed which is based on similarity analysis which finds the routine of the humans and bots. Bots tend to have same routine all day in terms of in game activity like taking similar steps in pattern or same action during any event, where else actual players have different approach or strategy in every game. The drawback to this approach is that it should have enough data collected having same behavior to differentiate. (Kang, et al., 2016).

(Pao, et al., 2010) proposed an approach which used trajectory analysis defining the movements and patterns carried by the bots and is indicated as their signature moves. For dimension reduction Isomap is used which is then followed by classification algorithms. The models produce higher accuracy but the drawback to this approach is that it only focuses on the movement of character which can be easily reduced in the programmed bots.

(Chung, et al., 2013) proposed a model which was based on the behavioural analysis to detect bots. This was based on three different actions which are Battle, Collect and Move. The users were then divided into groups based on the similar actions and a model was created was each of the following groups. The model showed high accuracy rate of detection but the drawback to this model if there is increase in number of players that will make new groups and have to increase models as well. (Suznjevic, et al., 2011) presented a model which uses combination of previously used model with user behaviour to get more details about the network model with higher accuracy. This model can be implemented in couple of MMORPG game, but changes have to be made to Markov chain values as per the game.

(Park, et al., 2019) presented a technique for bot detection in game with Long short -Term Memory (LSTM) using leveraging analysis. It mainly focused on the financial status or activities of the player and stated that a bot would not be down in terms of financial pattern as it is always on-line and would be easy to detect. The earlier approaches gave higher accuracy due to topological network as compared to this model but in terms of efficiency it better as deployment and cost resources are required less due to Neural Networks. So, to increase the accuracy rate of this mode, two or more model should be combined. To get better results than the previous approaches applied, we propose to use behavioural characteristic to detect bots in games.

4 Methodology

The data has to be extracted from a MMORPG company which was a difficult task as the companies does not publish dataset as it contains details of the player. The players and company sign and agreement before the start of the game stating the policy where the data will not be published outside the company. Some companies give APIs where it is an disadvantage, because the data extracted from API does not have the quality. The dataset used in the research is taken from HCR Labs by signing an NDA form.

4.1 Data Acquisition

The dataset had 4 sheets out of which attributes of network measure were removes as it was less important feature and contribution of this attribute towards the result gave less accuracy. In this research for every model different feature selection method were used. As both input and output were known Supervised learning method was used and classification algorithm was used as the variable to be predicted was in form of binary. The algorithm was then implemented, and the results were compared based on the accuracy rate and the best was chosen from the one of them.

The dataset on MMORPG game called Aion was used for analysis. The dataset contains 47,739 observations and 7702 were marked as 'bot'. The observations contain the log of 88 days details from 9th April to 5th July 2010 overall. The dataset is of integer and numerical type with 43 variables.

All the missing values were removed, the dataset network measure was not included. As the data was redundant and the contribution of this dataset towards the overall accuracy and prediction rate was been less. It had 113 variables which was likely impossible to compute.

	Actor	A_Acc	Login_day_count	Logout_day_count	Playtime	playtime_per_day	avg_money	Login_count	ip_count	Max_level
1	1047	6482393	46	42	764520	18202.857	26576.5613	97	27	51
2	1049	6275719	16	16	48300	3018.750	902.5117	32	13	47
3	1120	6596993	4	4	37867	9466.750	60.9084	8	6	19
4	1164	6670686	9	9	34592	3843.556	127755.7357	9	6	50
5	1184	4220231	11	11	117686	10698.727	7589.3723	37	8	40
6	1257	4458955	26	26	113372	4360.462	455.3869	66	14	42

	collect_max_count	Sit_ratio	Sit_count	sit_count_per_day	Exp_get_ratio	Exp_get_count	exp_get_count_per_day
1	6	1.0430	1012	24.0952	15.5210	15060	358.5714
2	0	3.5570	141	8.8125	13.6226	540	33.7500
3	3	4.9882	190	47.5000	15.6734	597	149.2500
4	0	0.8581	30	3.3333	17.8776	625	69.4444
5	4	4.2667	215	19.5455	18.0194	908	82.5455
6	690	1.7386	279	10.7308	20.1845	3239	124.5769

	Item_get_ratio	Item_get_count	item_get_count_per_day	Money_get_ratio	Money_get_count	money_get_count_per_day
1	6.2847	6098	145.1905	10.2010	9898	235.66667
2	6.9122	274	17.1250	7.5429	299	18.68750
3	13.5994	518	129.5000	9.7926	373	93.25000
4	6.8650	240	26.6667	13.5584	474	52.66667
5	8.0572	406	36.9091	9.2876	468	42.54545
6	18.4957	2968	114.1538	5.1972	834	32.07692

	Abyss_get_ratio	Abyss_get_count	abyss_get_count_per_day	Exp_repair_count	Exp_repair_count_per_day	Use_portal_count
1	11.0636	10735	255.5952	66	1.5714	2
2	2.8507	113	7.0625	3	0.1875	2
3	0.0000	0	0.0000	15	3.7500	0
4	8.0950	283	31.4444	0	0.0000	0
5	1.2502	63	5.7273	3	0.2727	0
6	1.4458	232	8.9231	10	0.3846	0

	Use_portal_count_per_day	Killed_bypc_count	Killed_bypc_count_per_day	Killed_bynpc_count	Killed_bynpc_count_per_day
1	0.0476	690	16.4286	172	4.0952
2	0.1250	30	1.8750	16	1.0000
3	0.0000	0	0.0000	20	5.0000
4	0.0000	26	2.8889	6	0.6667
5	0.0000	6	0.5455	17	1.5455
6	0.0000	15	0.5769	8	0.3077

	Teleport_count	Teleport_count_per_day	Reborn_count	Reborn_count_per_day	Social_diversity	Avg_PartyTime	GuildAct_count
1	1051	25.0238	2	0.0476	0.6407	6760.206	1
2	118	7.3750	0	0.0000	1.3499	4792.692	2
3	37	9.2500	0	0.0000	0.6931	5341.333	0
4	61	6.7778	0	0.0000	0.9345	4584.714	0
5	54	4.9091	0	0.0000	0.9736	6350.500	0
6	131	5.0385	0	0.0000	1.3656	3977.500	0

	GuildJoin_count	Type
1	0	Human
2	0	Human
3	0	Human
4	0	Human
5	0	Human
6	0	Human

Dimension and summary of the dataset:

```
[1] 49739    43
> summary(all_features_data)
     Actor             A_Acc           Login_day_count Logout_day_count    Playtime        playtime_per_day
 Min.   :  1047   Min.   :       0   Min.   : 0.00   Min.   : 1.00    Min.   :  10802   Min.   :   136.7
 1st Qu.:272154   1st Qu.: 6878566   1st Qu.: 8.00   1st Qu.: 8.00    1st Qu.:  36657   1st Qu.:  3941.2
 Median :400583   Median : 8290550   Median :22.00   Median :22.00    Median : 148651   Median :  8935.4
 Mean   :344941   Mean   : 8483737   Mean   :30.86   Mean   :30.67    Mean   : 576831   Mean   : 16363.6
 3rd Qu.:442201   3rd Qu.:10433334   3rd Qu.:49.00   3rd Qu.:49.00    3rd Qu.: 575728   3rd Qu.: 18016.7
 Max.   :472898   Max.   :11369388   Max.   :88.00   Max.   :88.00    Max.   :7334033   Max.   :160935.0
   avg_money          Login_count          ip_count         Max_level     collect_max_count   Sit_ratio
 Min.   :      0.0   Min.   :    0.0   Min.   :  0.0   Min.   : 1.0   Min.   :    0.0   Min.   : 0.0000
 1st Qu.:    811.6   1st Qu.:   16.0   1st Qu.:  3.0   1st Qu.:23.0   1st Qu.:    0.0   1st Qu.: 0.4688
 Median :   4536.2   Median :   48.0   Median :  7.0   Median :36.0   Median :    2.0   Median : 1.3127
 Mean   :  19138.7   Mean   :  107.2   Mean   : 11.8   Mean   :34.8   Mean   :  248.2   Mean   : 2.2898
 3rd Qu.:  17177.6   3rd Qu.:  124.0   3rd Qu.: 16.0   3rd Qu.:50.0   3rd Qu.:   72.0   3rd Qu.: 2.8573
 Max.   :2524605.2   Max.   :14865.0   Max.   :205.0   Max.   :55.0   Max.   :13529.0   Max.   :45.3627
   Sit_count       sit_count_per_day  Exp_get_ratio   Exp_get_count     exp_get_count_per_day Item_get_ratio
 Min.   :     0   Min.   :   0.000   Min.   : 0.00   Min.   :     0   Min.   :    0.0       Min.   : 0.000
 1st Qu.:    35   1st Qu.:   3.125   1st Qu.:11.92   1st Qu.:   468   1st Qu.:   51.6       1st Qu.: 8.587
 Median :   188   Median :  11.929   Median :15.97   Median :  2188   Median :  140.0       Median :12.810
 Mean   :  1962   Mean   :  57.606   Mean   :15.62   Mean   : 13561   Mean   :  423.6       Mean   :13.031
 3rd Qu.:   908   3rd Qu.:  30.785   3rd Qu.:19.08   3rd Qu.: 10400   3rd Qu.:  306.7       3rd Qu.:16.607
 Max.   :219363   Max.   :6011.500   Max.   :79.41   Max.   :562944   Max.   :22019.7       Max.   :48.961
 Item_get_count     item_get_count_per_day Money_get_ratio  Money_get_count   money_get_count_per_day Abyss_get_ratio
 Min.   :      0   Min.   :    0.0         Min.   : 0.000   Min.   :     0   Min.   :    0.00       Min.   : 0.0000
 1st Qu.:    408   1st Qu.:   41.4         1st Qu.: 2.315   1st Qu.:   141   1st Qu.:   12.60       1st Qu.: 0.0000
 Median :   1848   Median :  108.1         Median : 6.338   Median :   629   Median :   48.26       Median : 0.1987
 Mean   :  13540   Mean   :  421.0         Mean   : 6.382   Mean   :  4389   Mean   :  136.98       Mean   : 2.5390
 3rd Qu.:   8743   3rd Qu.:  247.9         3rd Qu.: 9.160   3rd Qu.:  3200   3rd Qu.:  115.39       3rd Qu.: 4.3279
 Max.   :1076296   Max.   :13798.7         Max.   :43.461   Max.   :332142   Max.   :14819.12       Max.   :21.7240
 Abyss_get_count   abyss_get_count_per_day Exp_repair_count   Exp_repair_count_per_day Use_portal_count
 Min.   :     0   Min.   :   0.000        Min.   :   0.00   Min.   :  0.0000         Min.   : 0.0000
 1st Qu.:     0   1st Qu.:   0.000        1st Qu.:   0.00   1st Qu.:  0.0000         1st Qu.: 0.0000
 Median :    29   Median :   1.308        Median :   5.00   Median :  0.3220         Median : 0.0000
 Mean   :  2337   Mean   :  43.254        Mean   :  20.33   Mean   :  0.8091         Mean   : 0.9879
 3rd Qu.:  1094   3rd Qu.:  39.306        3rd Qu.:  21.00   3rd Qu.:  1.0000         3rd Qu.: 0.0000
 Max.   :132445   Max.   :4282.000        Max.   :1464.00   Max.   :146.0000         Max.   :91.0000
 Use_portal_count_per_day Killed_bypc_count Killed_bypc_count_per_day Killed_bynpc_count Killed_bynpc_count_per_day
 Min.   :0.00000          Min.   :   0.00   Min.   :  0.0000          Min.   :   0.00   Min.   :  0.000
 1st Qu.:0.00000          1st Qu.:   0.00   1st Qu.:  0.0000          1st Qu.:   3.00   1st Qu.:  0.320
 Median :0.00000          Median :   3.00   Median :  0.1538          Median :  19.00   Median :  1.385
 Mean   :0.02382          Mean   :  95.89   Mean   :  1.9654          Mean   :  94.06   Mean   :  2.585
 3rd Qu.:0.00000          3rd Qu.:  45.00   3rd Qu.:  1.4304          3rd Qu.:  95.00   3rd Qu.:  3.414
 Max.   :3.80000          Max.   :7388.00   Max.   :174.4500          Max.   :4552.00   Max.   :101.667
 Teleport_count     Teleport_count_per_day  Reborn_count       Reborn_count_per_day Social_diversity Avg_PartyTime
 Min.   :    0.0   Min.   :  0.000         Min.   :   0.000   Min.   : 0.00000     Min.   :0.0000   Min.   :     0
 1st Qu.:   26.0   1st Qu.:  3.056         1st Qu.:   0.000   1st Qu.: 0.00000     1st Qu.:0.4101   1st Qu.:   653
 Median :  117.0   Median :  7.700         Median :   0.000   Median : 0.00000     Median :0.8148   Median :  3456
 Mean   :  453.4   Mean   : 11.946         Mean   :   4.297   Mean   : 0.07567     Mean   :0.7288   Mean   :  4388
 3rd Qu.:  489.0   3rd Qu.: 16.015         3rd Qu.:   1.000   3rd Qu.: 0.02615     3rd Qu.:1.0822   3rd Qu.:  5774
 Max.   :14336.0   Max.   :268.000         Max.   :1869.000   Max.   :23.18180     Max.   :1.8095   Max.   :476725
 GuildAct_count     GuildJoin_count     Type
 Min.   : 0.0000   Min.   :0.000    Bot   : 6250
 1st Qu.: 0.0000   1st Qu.:0.000    Human:43489
 Median : 0.0000   Median :0.000
 Mean   : 0.7467   Mean   :0.279
 3rd Qu.: 1.0000   3rd Qu.:1.000
```

5 Modelling Implementation

The motive behind this implementation was to predict between two labels that is Human and Bot which is been achieved by using classification algorithm.

- **Naïve Bayes:**

As the dataset is larger with 43 variables, this algorithm is best for the big dataset. It is easy to understand and can be built easily as the algorithm is not sensitive to feature that are irrelevant. It is often used in deploying real time system as it can handle real discrete data.

1. Caret (Classification and Regression training) package is installed as it is the most powerful package in R which is used to split the data, variable importance and for feature selection.

2. ‘rpart’ was used to establish a relation between the variables and was also used for training the data. To visualize data plot() function was used.

```
c_data <- train(Type~., data = all_data, method = "rpart")
```

3. The important varibales are plotted using varImp().

```
> imp_data <- varImp(c_data)
> imp_data
rpart variable importance

  only 20 most important variables shown (out of 42)

                         Overall
playtime_per_day         100.000
item_get_count_per_day    95.991
exp_get_count_per_day     91.055
Item_get_ratio            70.937
sit_count_per_day         62.900
Reborn_count_per_day       2.019
Item_get_count             1.994
Reborn_count               1.933
collect_max_count          0.000
Teleport_count_per_day     0.000
Money_get_ratio            0.000
Killed_bypc_count          0.000
Abyss_get_count            0.000
Killed_bypc_count_per_day  0.000
Abyss_get_ratio            0.000
Sit_count                  0.000
Playtime                   0.000
Actor                      0.000
ip_count                   0.000
abyss_get_count_per_day    0.000
```

4. Another package which is ‘dplyr’ is installed which is a powerful package as caret but is used for manipulating the data when the data frames are in memory and out of memory.
5. The data is then split into ratio of 75:25 of which 75 percent is of training set by using createDatapartion() function.
6. As the algorithm is used for prediction and plotting. Another R package ‘e1071’ is installed. It is generally used when one used SVM (support Vector machine) algorithm.

```
split_data <- createDataPartition(all_data$Type, p=0.75, list = FALSE)
trainData <- all_data[split_data,]
testData <- all_data[-split_data,]
```

- **Random Forest**

This algorithm shows better accuracy rate and performance as compared to decision tree. As both can be used for regression purpose and classification, But, for this purpose, we are using Random forest for classification. Random forest is considered to be the best learner whereas, each single tree in random forest is considered to be weak learner. The limitation or drawback of this algorithm is that it is biased with features with larger number of classes.

1. Package name ‘boruta’ is installed. This name is taken from a demon who lived in pine forests (Dutta, 2016) in Slavic mythology from ancient. The package is used for selecting variables.
2. Package name ‘ranger’ is installed which act as a catalyst for Random forest to increase the speed of the implementation process.

3. Using package ‘boruta’ the data was trained and maxRun was kept to 11 due to lack of hardware resource as random forest uses high computational power.
4. The data was split into same ratio of 75:25.

```
b_data <- Boruta(Type~., all_features_data, doTrace = 1, maxRuns = 11)
names(b_data)
b_significant <- getSelectedAttributes(b_data, withTentative = TRUE)
b_significant


tent_fix <- TentativeRoughFix(b_data)
b_significant <- getSelectedAttributes(tent_fix)
b_significant
```

5. Variables are sorted and ordered and a mean importance of the variable is generated.

- **GLM (Generalized Linear Model)**

GLM is used to check the relationship between the response variable and the features available. It is also considered best for curve fitting. Therefore, it is used to project the relation between the selected feature and the target variable which is ‘Type’. There is multiple package installed while implementing GLM technique.

```
set.seed(123)
install.packages("mlbench")
install.packages("caret")
install.packages("lattice")
install.packages("ggplot2")
install.packages("dplyr")
install.packages("ROCR")
library(mlbench)
library(caret)
library(lattice)
library(ggplot2)
library(dplyr)
library(ROCR)
```

1. Mlbench – A framework which is used in distributed Machine Learning. It is mainly used to enhance or improve the robustness, transparency, reproducibility and to give good measures. (Github, 2018)
2. GGplot2 – It is generally used for creating graphs in the system.
After the variables are defined and mapped with graphical primitive, rest part is done by ggplot2. (Wickham, 2016)
3. Lattice – It is used for plotting multivariate data and for data visualization.
4. ROCR – It is basically used to get the ROC curve recall and precision. It is also used to display the relation between the specificity and sensitivity.
5. Correlation Matrix is generated using cor() function with a cutoff of 0.5.

```
highlyCorrelated <- findCorrelation(cor(all_Features), cutoff = 0.5)
highlyCorrelated
```

6 Evaluation

Through confusion matrix accuracy, sensitivity and specificity of each model is calculated and evaluated.

- **1.Naïve Bayes Matrix**

  The result below is confusion matrix which is generated through caret package.

```
Confusion Matrix and Statistics

          Reference
Prediction   Bot Human
     Bot    1023   142
     Human   539 10730

               Accuracy : 0.9452
                 95% CI : (0.9411, 0.9492)
    No Information Rate : 0.8744
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7202
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.65493
            Specificity : 0.98694
         Pos Pred Value : 0.87811
         Neg Pred Value : 0.95217
             Prevalence : 0.12562
         Detection Rate : 0.08227
   Detection Prevalence : 0.09369
      Balanced Accuracy : 0.82093

       'Positive' Class : Bot
```

  The result above shows the accuracy rate of 94.52 percent. Calculating the precision of this model by TP/TP+FP which is 0.681 and recall is calculated by TP/TP+FN which gives 0.872.

- **2. Generalized Linear Model**

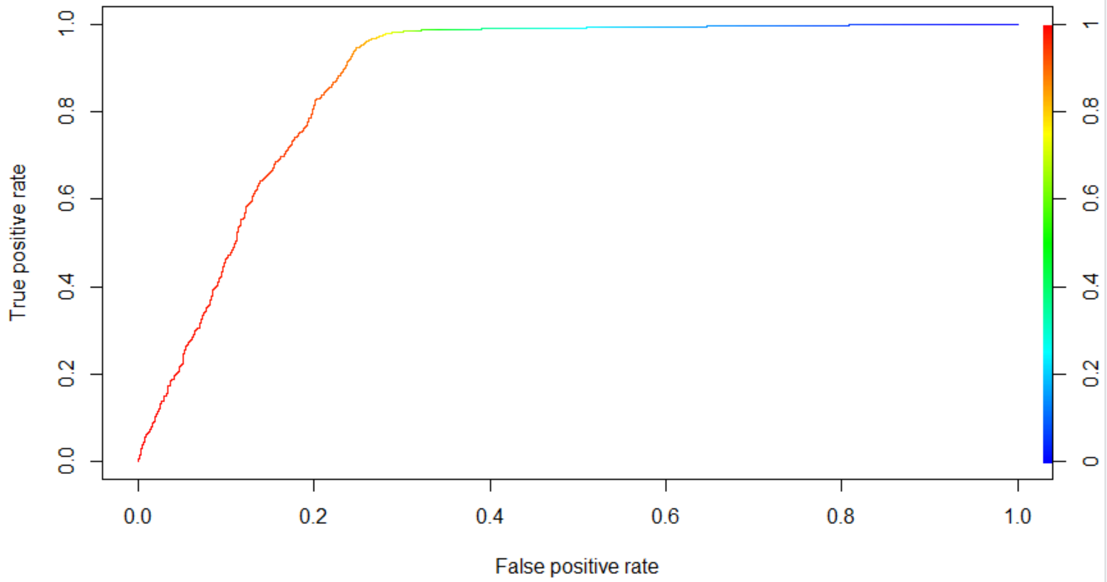
```
   FALSE  TRUE
 0  1035   539
 1   134 10726
> acc <- sum(diag(glm_table)/sum(glm_table))
> acc
[1] 0.9458742
```

  From the above the accuracy of the model is 94.58 percent whereas,

```
> prec(glm_table)
[1] 0.9521527
> recall(glm_table)
[1] 0.9876611
> |
```

  Here the Precision of GLM is 95.21 percent and Recall is 98.76 percent. Comparing this model with Naïve Bayes it does not show such good performance apart from the Recall rate.

  Below is the ROC curve for the above model



- **3.Random Forest**

```
Confusion Matrix and Statistics

          Reference
Prediction   Bot Human
     Bot    1130   432
     Human   109 10763

               Accuracy : 0.9565
                 95% CI : (0.9528, 0.96)
    No Information Rate : 0.9004
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7827
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.91203
            Specificity : 0.96141
         Pos Pred Value : 0.72343
         Neg Pred Value : 0.98997
             Prevalence : 0.09965
         Detection Rate : 0.09088
   Detection Prevalence : 0.12562
      Balanced Accuracy : 0.93672

       'Positive' Class : Bot
```

The above chart shows the accuracy rate of the model which is 95.99 percent. In this model, the precision rate is higher as compared to the recall rate which states that many numbers of positive examples have been missed which are false negative, but the predicted positive are exact.

Comparison Table

Model	Precision	Recall	F1-Score	Accuracy
Naïve Bayes	0.681	0.872	0.764	94.52
Generalized Linear Model	0.952	0.987	0.966	94.58
Random Forest	0.921	0.743	0.822	95.65

7 Discussion

Detailed explanation is mentioned discussing about the results of the model that is been used in the process and future work with limitations of the research.

This model is proposed to lower the damage that is causing the user experience and the game provider. From the behavioural observation, it shows that game bots perform same steps and actions which allow them to gain an unfair points against the actual user. They do not interact with the actual players and only transmit assets with each other. A discriminative model is been proposed after evaluating these behavioural features.The model gives accuracy of 95.99 which can be used to detect and ban the bots.

The motive behind the research is to find the best predictive model, for that the dataset has been trained with 3 different algorithms. Before splitting the data, feature selection method is used for every model trained keeping the ratio same for all. From the table, Generalized Liner Model have the best Precision and Recall rate but overall, the accuracy is lower than other two models. For accuracy Random Forest have given the best accuracy i.e. 95.99 percent, therefore we select this model as the best fit for the dataset.

False positive error is nothing but predicting an ID to be bot, but they are not actually a bot. This would be a challenge while bot detection. It is ok to predict a bot as human but not human as not. Therefore, decreasing the false positive number is the main thing in detecting the bots. The trained models gave higher precision rate where positively predicted are true. GLM shows higher precision rate but the accuracy rate is low.

8 Conclusion and Future Work

Online games have gathered from all the geographical location to play with each other or against, which allows them to gain in game currencies by defeating the enemies and the earned currency can be converted to real money as well.

Bot detection technique using behavioral analysis was accomplished using the dataset which consist of players information from AION game. This dataset was then used with different classification algorithm. From the models that have been used Random forest gave the best accuracy rate. We were successfully able to reduce the false positive prediction where it gave good results, but the accuracy was reduced.

Each day a new software is in the market for the bot detection method and this method is a continuous process. For every game that is different because each game have its own different design so the main motive or idea is to make a general detection method which will be useful for all the MMORPG games even if the design is different.

Further goal is to study the different style of the game bot, this can be done by reinforcement learning where bots can be deployed in every strategically different game to know the style of the game and the data can be collected where using these methods we can create a counter wall which can be used against the bot. More variations can be used with the techniques which would increase the accuracy rate and would also decrease the false positive.

9 Acknowledgement

Sincere thanks to My Mentor/Guide Professor. Jeorge Basillo who guided me helped, suggested me all through the way to the right path. In each meeting My Mentor without hesitation replied to all my queries with even good ideas. I would also thank my Friends and Family who all supported me all through this time.

References

1. Batchelor, J., 2018. Games Industry. [Online] Available at: https://www.gamesindustry.biz /articles /2018-12-18-global-games-market-value-roseto- usd134-9bn-in-2018

2. Chen, K.-T.et al., 2009. Identifying MMORPG Bots: A Traffic Analysis Approach. EURASIP Journal on Advances in Signal Processing, Volume 2009.

3. Chung, Y. et al., 2013. A Behavior Analysis-Based Game Bot Detection Approach Considering Various Play Styles. Etri Journal, 35(6).

4. Github, 2018. MLBench. [Online] Available at: https://mlbench.github.io/2018/09/07 /introducing-mlbench/ , A. R., . H. K. K., A. M. . S. H. J., 2016. SpringerPlus. [Online] Available at: https://springerplus.springeropen.com/articles/10.1186/s40064-016-2122-8rightslink

5. Kim, H. K. Hyukmin, K., 2011. Self-similarity based Bot Detection System in MMORPG. s.l., s.n.

6. Lee, E. et al., 2016. You are a Game Bot!: Uncovering Game Bots in MMORPGs via Self-similarity in the Wild. s.l., s.n. , J., 2018. Datacamp. [Online] Available at: https://www.datacamp.com/community/tutorials/logistic-regression-R

7. Pao, H.-K., Chen, K.-T. Chang, H.-C., 2010. Game Bot Detection via Avatar Trajectory Analysis. IEEE Transactions on Computational Intelligence and AI in Games, 2(3), pp. 162-175.

8. McCracken, G., 2018. Bot Detection in Online Games through Applied Machine Learning and Statistical Analysis of Mouse movements.

9. Techopedia, 2019. Techopedia. [Online] Available at: https://www.techopedia.com /definition /19278/farming

10. Suznjevic, M., Stupar, I. Matijasevic, M., 2011. MMORPG player behavior model based on player action categories. Ottawa, s.n.

11. Wickham, H., 2016. In: ggplot2: Elegant Graphics for Data Analysis. s.l.:Springer-Verlag New York.

12. Wendel, E., 2012. Cheating in Online Games.