Project Report - Milestone 3

INST 737 - Introduction to Data Science

Research Study: MindGame Insights: A Deep Dive into Gaming and Psychological Well-Being Relationships

(Areas of Research: Behavioral Data Science with a focus on Cyberpsychology)

(Team: Rajeevan Madabushi, Pranav Adiraju, Asmita Samanta)

1. Milestone 2 - Recap

In Milestone 2, we worked on different statistical models to test our hypothesis:

"What is the relationship between anxiety levels (GAD) on various factors, including Satisfaction with Life Total (SWL_T), Social Phobia Inventory Total (SPIN_T), Narcissism, Hours of play, Reasons to Play (whyplay), Work Status(work), and Play Style(playstyle) preferences?"

We started our exploration with the fundamental approaches of Univariate and multivariate Linear Regressions and then advanced to the Logistic Regression approach. We have also incorporated Lasso & Ridge Regression (Regularization Methods). Finally, we ended our research by experimenting with the Decision Trees by delving deeper into ensemble methods like Bagging and random Forests. Our research also involved a thorough comparative analysis of these diverse models by not only comparing the model performance but also highlighting specific nuances and each methodology that has been used on our dataset.

From our insights, we inferred that Logistic Regression was the best-performing model for predicting the ***GAD_T*** score. This further helped us understand our data and refine the strategies for the next part of our research.

For working on this milestone, we started with the same cleaned dataset (***https://drive.google.com/file/d/12j-TYs8fR4cu9z9sghUmK9nTAbdJN7T5/view?usp=sharing***) that was utilized for all model evaluations in Milestone 2. For Milestone 3, we have used both R and Python for scripting. This was decided because for some of the questions, running the code in R was very time consuming while running in Python was fast and more efficient. This could also be due to the complexity of our dataset which has 10k+ rows with a mix of numerical and categorical variables.

2. Question 1: SVM (State Vector Machines)

Data Pre-processing & Cleaning:

From our research question we treated ***GAD_T*** (Anxiety Levels) as a categorical variable, Although ***GAD_T*** can be treated as a numerical variable, we decided to take the classification approach rather than the regression model approach. In our code we have employed a one-vs-one approach for multi-class classification where each SVM model is trained for each pair of classes.

Coming to the steps, we have first imported us .csv into R and created a subset of the dataset with the required columns (***GAD_T, SPIN_T, SWL_T, Narcissism, Hours, whyplay_clean, work, Playstyle_clean)***.

The GAD_T was then converted into a factor as it is a categorical variable in our research hypothesis and the dataset was then split into an **80-20** training and testing split.

```
> #Data Pre-processing
> dataForSVM <- anxiety[c("GAD_T", "Hours", "SPIN_T", "SWL_T", "whyplay_clean", "Work", "Playstyle_clean", "Narcissism")]
> summary(dataForSVM)
     GAD_T            Hours            SPIN_T          SWL_T       whyplay_clean          Work           Playstyle_clean      Narcissism
 Min.   : 0.000   Min.   :  0.00   Min.   : 0.00   Min.   : 5.00   Length:10591       Length:10591       Length:10591       Min.   :1.000
 1st Qu.: 2.000   1st Qu.: 12.00   1st Qu.: 9.00   1st Qu.:14.00   Class :character   Class :character   Class :character   1st Qu.:1.000
 Median : 4.000   Median : 20.00   Median :17.00   Median :20.00   Mode  :character   Mode  :character   Mode  :character   Median :2.000
 Mean   : 5.174   Mean   : 21.36   Mean   :19.73   Mean   :19.84                                                            Mean   :2.021
 3rd Qu.: 8.000   3rd Qu.: 28.00   3rd Qu.:28.00   3rd Qu.:26.00                                                            3rd Qu.:3.000
 Max.   :21.000   Max.   :120.00   Max.   :68.00   Max.   :35.00                                                            Max.   :5.000
```

```
> #Categorizing the Data
> dataForSVM$GAD_T <- as.factor(dataForSVM$GAD_T)
>
> #Converting the data into string
> str(dataForSVM)
'data.frame':	10591 obs. of  8 variables:
 $ GAD_T          : Factor w/ 22 levels "0","1","2","3",..: 1 6 1 2 12 2 20 9 22 19 ...
 $ Hours          : num  25 5 9 7 25 10 25 20 40 10 ...
 $ SPIN_T         : num  3 31 39 29 17 11 47 25 45 30 ...
 $ SWL_T          : int  33 26 26 21 10 13 29 17 12 15 ...
 $ whyplay_clean  : chr  "fun" "fun" "improving" "fun" ...
 $ Work           : chr  "Employed" "Employed" "Employed" "Student at college / university" ...
 $ Playstyle_clean: chr  "Multiplayer - online - with real life friends" "Multiplayer - online - with strangers" "Multiplayer - online - wi
th real life friends" "Multiplayer - online - with strangers" ...
 $ Narcissism     : num  1 3 4 2 3 3 5 2 4 1 ...
```

```
> # Splitting the dataset
> dataForSVM_train <- dataForSVM[1:train_rows, ]
> dataForSVM_test <- dataForSVM[(train_rows + 1):total_rows,]
>
> #train & test summaries
> summary(dataForSVM_train)
     GAD_T          Hours            SPIN_T          SWL_T       whyplay_clean          Work           Playstyle_clean      Narcissism
 0      :1037   Min.   :  0.00   Min.   : 0.00   Min.   : 5.00   Length:8473        Length:8473        Length:8473        Min.   :1.00
 2      : 995   1st Qu.: 12.00   1st Qu.: 9.00   1st Qu.:14.00   Class :character   Class :character   Class :character   1st Qu.:1.00
 1      : 985   Median : 20.00   Median :17.00   Median :20.00   Mode  :character   Mode  :character   Mode  :character   Median :2.00
 3      : 942   Mean   : 21.31   Mean   :19.72   Mean   :19.86                                                            Mean   :2.02
 4      : 783   3rd Qu.: 28.00   3rd Qu.:28.00   3rd Qu.:26.00                                                            3rd Qu.:3.00
 5      : 628   Max.   :120.00   Max.   :68.00   Max.   :35.00                                                            Max.   :5.00
 (Other):3103
> summary(dataForSVM_test)
     GAD_T         Hours            SPIN_T          SWL_T       whyplay_clean          Work           Playstyle_clean      Narcissism
 2      :273   Min.   :  0.00   Min.   : 0.00   Min.   : 5.00   Length:2118        Length:2118        Length:2118        Min.   :1.000
 0      :248   1st Qu.: 12.00   1st Qu.: 9.00   1st Qu.:14.00   Class :character   Class :character   Class :character   1st Qu.:1.000
 3      :227   Median : 20.00   Median :17.00   Median :20.00   Mode  :character   Mode  :character   Mode  :character   Median :2.000
 1      :211   Mean   : 21.57   Mean   :19.77   Mean   :19.77                                                            Mean   :2.026
 4      :197   3rd Qu.: 28.00   3rd Qu.:28.00   3rd Qu.:26.00                                                            3rd Qu.:3.000
 5      :160   Max.   :112.00   Max.   :68.00   Max.   :35.00                                                            Max.   :5.000
 (Other):802
>
> #count of train & test values
> nrow(dataForSVM_train)
[1] 8473
> nrow(dataForSVM_test)
[1] 2118
```

Model Training & Evaluation:

Next, we imported the ***kernlab*** and ***caret*** libraries and created a function ***perform_svm*** which was used to train and evaluate our SVM model with the specified kernels. The function was also used to evaluate the training set performance on the test set.

The ***perform_svm*** function is also used to train models, make predictions and calculate the confusion matrix as well as the class-wise metrics.

The kernels that have been used to train our SVM Models were: ***Gaussian RBF ('rbfdot')***, ***Polynomial ('polydot')***, ***Euclidean Inner Product ('vanilladot')*** and ***Hyperbolic Tangent ('tanhdot')***. We then computed overall accuracy, confusion matrix and class-wise metrics for each kernel as follows:

Model Results:

Kernel 1 - Gaussian RBF ('rbfdot'):

Firstly, we used a Gaussian Radial Basis Kernel Function to train our model, having a total of **8420** support vectors. The training error in our dataset was ***79.22%*** (approx.)

Confusion Matrix:

Next, we generated the confusion matrix showing the measures of predicted classes to the actual classes where each row in the matrix represents an actual class and each column represents a predicted class.

Here is the confusion matrix for Gaussian RBF:

predictions	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	131	106	104	66	51	31	30	18	10	7	6	3	4	3	3	1	1	1	1	0	0	2
1	35	29	40	42	20	23	12	9	11	6	3	3	2	3	1	2	2	1	1	2	0	0
2	35	28	55	55	45	39	35	35	19	14	10	9	7	2	2	4	2	1	0	0	0	2
3	30	33	54	44	48	34	30	33	37	22	18	14	11	20	4	6	2	1	3	4	1	6
4	9	10	6	9	16	13	8	10	4	8	6	5	3	3	5	3	6	1	2	0	0	1
5	2	1	4	4	5	6	3	5	6	4	6	3	3	3	4	3	3	1	2	0	0	0
6	4	2	5	3	7	8	9	5	9	6	6	6	3	2	2	4	2	4	2	5	2	2
7	0	1	3	1	4	2	5	4	2	3	2	3	1	0	0	0	3	3	0	1	0	0
8	1	0	0	3	0	2	2	2	3	3	2	2	2	1	2	2	2	0	2	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1	1	0	0	0	1	2	1	2	1	1	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0
12	1	0	2	0	1	1	0	0	2	2	1	1	2	2	0	3	0	0	0	1	0	1
13	0	1	0	0	0	1	2	0	4	2	3	0	2	3	6	1	2	1	1	1	0	3
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	1	1

Accuracy & Metrics:

We then computed the overall statistics and metrics by class to further understand the effectiveness of our model where the overall accuracy of the SVM Model (Gaussian RBF) Kernel came up approximately to ***14.35%*** which indicated a limited predictive performance and the balanced accuracy score for Class 0 was close to ***64.43%***

```
Overall Statistics

               Accuracy : 0.1435
                 95% CI : (0.1289, 0.1592)
    No Information Rate : 0.1289
    P-Value [Acc > NIR] : 0.02525

                  Kappa : 0.0436

 Mcnemar's Test P-Value : NA
```

Statistics by Class:

	Class: 0	Class: 1	Class: 2	Class: 3	Class: 4	Class: 5	Class: 6	Class: 7	Class: 8	Class: 9	Class: 10	Class: 11
Sensitivity	0.52823	0.13744	0.20147	0.19383	0.081218	0.037500	0.066176	0.032787	0.027778	0.00000	0.000000	0.0200000
Specificity	0.76043	0.88568	0.81355	0.78265	0.941697	0.968335	0.955096	0.982966	0.985572	1.00000	0.994645	0.9990329
Pos Pred Value	0.22625	0.11741	0.13784	0.09670	0.125000	0.088235	0.091837	0.105263	0.093750	NaN	0.000000	0.3333333
Neg Pred Value	0.92398	0.90273	0.87318	0.88996	0.909045	0.924878	0.937129	0.943269	0.949664	0.96364	0.969625	0.9768322
Prevalence	0.11709	0.09962	0.12890	0.10718	0.093012	0.075543	0.064212	0.057602	0.050992	0.03636	0.030217	0.0236072
Detection Rate	0.06185	0.01369	0.02597	0.02077	0.007554	0.002833	0.004249	0.001889	0.001416	0.00000	0.000000	0.0004721
Detection Prevalence	0.27337	0.11662	0.18839	0.21483	0.060434	0.032106	0.046270	0.017941	0.015109	0.00000	0.005194	0.0014164
Balanced Accuracy	0.64433	0.51156	0.50751	0.48824	0.511458	0.502918	0.510636	0.507876	0.506675	0.50000	0.497322	0.5095164

	Class: 12	Class: 13	Class: 14	Class: 15	Class: 16	Class: 17	Class: 18	Class: 19	Class: 20	Class: 21
Sensitivity	0.0476190	0.065217	0.0000000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.0500000
Specificity	0.9913295	0.985521	0.9995208	1.00000	1.00000	1.000000	1.00000	1.000000	1.000000	0.9976168
Pos Pred Value	0.1000000	0.090909	0.0000000	NaN	NaN	NaN	NaN	NaN	NaN	0.1666667
Neg Pred Value	0.9809342	0.979376	0.9853566	0.98536	0.98725	0.992918	0.99339	0.992918	0.998111	0.9910038
Prevalence	0.0198300	0.021719	0.0146364	0.01464	0.01275	0.007082	0.00661	0.007082	0.001889	0.0094429
Detection Rate	0.0009443	0.001416	0.0000000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.0004721
Detection Prevalence	0.0094429	0.015581	0.0004721	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.0028329
Balanced Accuracy	0.5194743	0.525369	0.4997604	0.50000	0.50000	0.500000	0.50000	0.500000	0.500000	0.5238084

Kernel 2 - Polynomial (‘polydot’) Kernel:

Next, we used a Polynomial (‘polydot’) Kernel Function to train our model, having a total of **8372** support vectors. The training error in our dataset was ***84.39%*** (approx.)

Confusion Matrix:

Next, we generated the confusion matrix showing the measures of predicted classes to the actual classes where each row in the matrix represents an actual class and each column represents a predicted class.

Here is the confusion matrix for the Polynomial (polydot) Kernel:

predictions	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	133	117	118	77	60	36	27	19	17	10	5	5	4	5	3	1	0	2	1	0	0	1
1	39	23	32	28	20	18	20	8	6	4	3	2	0	1	0	1	2	0	0	0	0	1
2	58	57	84	97	92	76	69	69	45	37	30	26	23	20	8	12	12	1	5	5	1	8
3	16	13	37	23	23	27	19	25	32	21	19	11	9	15	11	9	10	10	6	6	2	3
4	2	1	2	2	2	3	1	1	8	4	6	6	6	4	8	8	3	2	2	3	1	6
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Accuracy & Metrics:

We then computed the overall statistics and metrics by class to further understand the effectiveness of our model where the overall accuracy of the SVM Model (Polynomial) Kernel came up approximately to ***12.56%*** which indicated a limited predictive performance and the balanced accuracy score for Class 0 was close to ***63.23%***

```
Overall Statistics

               Accuracy : 0.1256
                 95% CI : (0.1118, 0.1405)
    No Information Rate : 0.1289
    P-Value [Acc > NIR] : 0.6845

                  Kappa : 0.0094

 Mcnemar's Test P-Value : NA
```

Statistics by Class:

	Class: 0	Class: 1	Class: 2	Class: 3	Class: 4	Class: 5	Class: 6	Class: 7	Class: 8	Class: 9	Class: 10	Class: 11
Sensitivity	0.5363	0.10900	0.30769	0.10132	0.0101523	0.0000000	0.00000	0.0000	0.00000	0.00000	0.0156250	0.00000
Specificity	0.7283	0.90299	0.59295	0.82866	0.9588756	0.9994893	1.00000	1.0000	1.00000	1.00000	0.9985394	1.00000
Pos Pred Value	0.2075	0.11058	0.10060	0.06628	0.0246914	0.0000000	NaN	NaN	NaN	NaN	0.2500000	NaN
Neg Pred Value	0.9221	0.90157	0.85269	0.88481	0.9042710	0.9244214	0.93579	0.9424	0.94901	0.96364	0.9701987	0.97639
Prevalence	0.1171	0.09962	0.12890	0.10718	0.0930123	0.0755430	0.06421	0.0576	0.05099	0.03636	0.0302172	0.02361
Detection Rate	0.0628	0.01086	0.03966	0.01086	0.0009443	0.0000000	0.00000	0.0000	0.00000	0.00000	0.0004721	0.00000
Detection Prevalence	0.3026	0.09821	0.39424	0.16383	0.0382436	0.0004721	0.00000	0.0000	0.00000	0.00000	0.0018886	0.00000
Balanced Accuracy	0.6323	0.50600	0.45032	0.46499	0.4845139	0.4997446	0.50000	0.5000	0.50000	0.50000	0.5070822	0.50000

	Class: 12	Class: 13	Class: 14	Class: 15	Class: 16	Class: 17	Class: 18	Class: 19	Class: 20	Class: 21
Sensitivity	0.00000	0.0000000	0.00000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.000000
Specificity	1.00000	0.9995174	1.00000	1.00000	1.00000	1.000000	1.00000	1.000000	1.000000	1.000000
Pos Pred Value	NaN	0.0000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Neg Pred Value	0.98017	0.9782711	0.98536	0.98536	0.98725	0.992918	0.99339	0.992918	0.998111	0.990557
Prevalence	0.01983	0.0217186	0.01464	0.01464	0.01275	0.007082	0.00661	0.007082	0.001889	0.009443
Detection Rate	0.00000	0.0000000	0.00000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.000000
Detection Prevalence	0.00000	0.0004721	0.00000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.000000
Balanced Accuracy	0.50000	0.4997587	0.50000	0.50000	0.50000	0.500000	0.50000	0.500000	0.500000	0.500000

Kernel 3- Euclidean Inner Product ('vanilladot') Kernel:

Third, we used a Euclidean Inner Product ('vanilladot') Kernel Function to train our model, having a total of **8372** support vectors. The training error in our dataset was ***84.42%*** (approx.)

Confusion Matrix:

Next, we generated the confusion matrix showing the measures of predicted classes to the actual classes where each row in the matrix represents an actual class and each column represents a predicted class.

Here is the confusion matrix for the Euclidean (vanilladot) Kernel:

predictions	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	133	117	118	77	60	36	27	19	17	10	5	5	4	5	3	1	0	2	1	0	0	1
1	39	24	32	28	20	18	20	8	6	4	3	2	0	1	0	1	2	0	0	0	0	1
2	58	56	84	96	92	76	69	68	44	37	29	26	23	20	8	12	12	1	5	5	1	8
3	16	13	37	24	23	27	19	26	33	21	20	11	9	15	11	9	10	10	6	6	2	3
4	2	1	2	2	2	3	1	1	8	4	6	6	6	4	8	8	3	2	2	3	1	6
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Accuracy & Metrics:

We then computed the overall statistics and metrics by class to further understand the effectiveness of our model where the overall accuracy of the SVM Model (Euclidean Inner Product) Kernel came up approximately to ***12.65%*** which indicated a limited predictive performance and the balanced accuracy score for Class 0 was close to ***63.23%***

Statistics by Class:

	Class: 0	Class: 1	Class: 2	Class: 3	Class: 4	Class: 5	Class: 6	Class: 7	Class: 8	Class: 9	Class: 10	Class: 11
Sensitivity	0.5363	0.11374	0.30769	0.10573	0.0101523	0.0000000	0.00000	0.0000	0.00000	0.00000	0.0156250	0.00000
Specificity	0.7283	0.90299	0.59566	0.82708	0.9588756	0.9994893	1.00000	1.0000	1.00000	1.00000	0.9985394	1.00000
Pos Pred Value	0.2075	0.11483	0.10120	0.06838	0.0246914	0.0000000	NaN	NaN	NaN	NaN	0.2500000	NaN
Neg Pred Value	0.9221	0.90204	0.85326	0.88512	0.9042710	0.9244214	0.93579	0.9424	0.94901	0.96364	0.9701987	0.97639
Prevalence	0.1171	0.09962	0.12890	0.10718	0.0930123	0.0755430	0.06421	0.0576	0.05099	0.03636	0.0302172	0.02361
Detection Rate	0.0628	0.01133	0.03966	0.01133	0.0009443	0.0000000	0.00000	0.0000	0.00000	0.00000	0.0004721	0.00000
Detection Prevalence	0.3026	0.09868	0.39188	0.16572	0.0382436	0.0004721	0.00000	0.0000	0.00000	0.00000	0.0018886	0.00000
Balanced Accuracy	0.6323	0.50837	0.45168	0.46640	0.4845139	0.4997446	0.50000	0.5000	0.50000	0.50000	0.5070822	0.50000

	Class: 12	Class: 13	Class: 14	Class: 15	Class: 16	Class: 17	Class: 18	Class: 19	Class: 20	Class: 21
Sensitivity	0.00000	0.0000000	0.00000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.000000
Specificity	1.00000	0.9995174	1.00000	1.00000	1.00000	1.000000	1.00000	1.000000	1.000000	1.000000
Pos Pred Value	NaN	0.0000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Neg Pred Value	0.98017	0.9782711	0.98536	0.98536	0.98725	0.992918	0.99339	0.992918	0.998111	0.990557
Prevalence	0.01983	0.0217186	0.01464	0.01464	0.01275	0.007082	0.00661	0.007082	0.001889	0.009443
Detection Rate	0.00000	0.0000000	0.00000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.000000
Detection Prevalence	0.00000	0.0004721	0.00000	0.00000	0.00000	0.000000	0.00000	0.000000	0.000000	0.000000
Balanced Accuracy	0.50000	0.4997587	0.50000	0.50000	0.50000	0.500000	0.50000	0.500000	0.500000	0.500000

Kernel 4 -Hyperbolic Tangent ('tanhdot') Kernel:

Finally, we used a Hyperbolic Tangent ('tanhdot') Kernel Function to train our model, having a total of **8310** support vectors. The training error in our dataset was ***89.22%*** (approx.)

Confusion Matrix:

Next, we generated the confusion matrix showing the measures of predicted classes to the actual classes where each row in the matrix represents an actual class and each column represents a predicted class.

Here is the confusion matrix for the Hyperbolic Tangent (tanhdot) Kernel:

predictions	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	115	90	105	64	63	49	37	32	27	14	14	7	4	7	3	2	5	1	0	1	1	3
1	23	24	28	22	28	11	16	11	8	5	2	1	3	0	0	1	2	0	0	0	0	0
2	29	22	48	44	26	22	22	22	13	14	10	10	7	4	2	3	4	1	0	2	0	2
3	30	23	24	24	19	18	12	15	13	11	2	3	2	3	2	0	0	0	0	1	0	1
4	4	8	3	7	4	3	6	6	5	2	6	4	0	2	3	2	0	1	1	0	1	1
5	2	0	2	1	4	2	2	1	2	1	0	1	1	3	0	0	0	0	0	0	0	0
6	1	0	0	2	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
7	29	26	40	41	30	36	22	16	15	8	12	10	11	14	8	8	7	6	5	5	1	5
8	3	1	7	4	7	7	6	6	9	7	7	5	5	5	5	10	4	2	1	1	1	4
9	3	3	2	4	1	2	1	0	3	2	2	0	2	0	2	0	1	0	1	1	0	1
10	9	12	11	11	8	6	11	8	7	9	7	8	2	5	2	2	2	3	3	3	0	1
11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	1	0	2	0	1	0	0	2	1	0	0	4	0	0	1	2	0	0	0
13	0	0	3	0	5	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0
14	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0
15	0	1	0	0	1	0	0	1	2	0	0	0	1	1	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	1	0	1	0	1	0	2	1	2	0	0	3	2	0	2	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1

Accuracy & Metrics:

We then computed the overall statistics and metrics by class to further understand the effectiveness of our model where the overall accuracy of the SVM Model (Euclidean Inner Product) Kernel came up approximately to ***11.90%*** which indicated a limited predictive performance and the balanced accuracy score for Class 0 was close to ***59.04%***

```
Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4  Class: 5 Class: 6 Class: 7 Class: 8  Class: 9 Class: 10 Class: 11
Sensitivity            0.4637  0.11374  0.17582  0.10573 0.020305 0.0125000 0.000000 0.131148 0.083333 0.0259740  0.109375 0.0000000
Specificity            0.7171  0.91557  0.85962  0.90534 0.966163 0.9897855 0.995964 0.830160 0.951244 0.9857913  0.940117 0.9995164
Pos Pred Value         0.1786  0.12973  0.15635  0.11823 0.057971 0.0909091 0.000000 0.045070 0.084112 0.0645161  0.053846 0.0000000
Neg Pred Value         0.9098  0.90326  0.87576  0.89399 0.905808 0.9246183 0.935545 0.939875 0.950771 0.9640632  0.971328 0.9763817
Prevalence             0.1171  0.09962  0.12890  0.10718 0.093012 0.0755430 0.064212 0.057602 0.050992 0.0363551  0.030217 0.0236072
Detection Rate         0.0543  0.01133  0.02266  0.01133 0.001889 0.0009443 0.000000 0.007554 0.004249 0.0009443  0.003305 0.0000000
Detection Prevalence   0.3041  0.08735  0.14495  0.09585 0.032578 0.0103872 0.003777 0.167611 0.050519 0.0146364  0.061379 0.0004721
Balanced Accuracy      0.5904  0.51466  0.51772  0.50553 0.493234 0.5011427 0.497982 0.480654 0.517289 0.5058827  0.524746 0.4997582


                     Class: 12 Class: 13 Class: 14 Class: 15 Class: 16 Class: 17 Class: 18 Class: 19 Class: 20 Class: 21
Sensitivity            0.00000  0.000000  0.000000  0.000000   0.00000  0.000000   0.00000  0.000000  0.000000 0.0500000
Specificity            0.99326  0.994208  0.998563  0.996646   1.00000  0.992392   1.00000  1.000000  1.000000 0.9985701
Pos Pred Value         0.00000  0.000000  0.000000  0.000000       NaN  0.000000       NaN       NaN       NaN 0.2500000
Neg Pred Value         0.98004  0.978158  0.985343  0.985315   0.98725  0.992864   0.99339  0.992918  0.998111 0.9910123
Prevalence             0.01983  0.021719  0.014636  0.014636   0.01275  0.007082   0.00661  0.007082  0.001889 0.0094429
Detection Rate         0.00000  0.000000  0.000000  0.000000   0.00000  0.000000   0.00000  0.000000  0.000000 0.0004721
Detection Prevalence   0.00661  0.005666  0.001416  0.003305   0.00000  0.007554   0.00000  0.000000  0.000000 0.0018886
Balanced Accuracy      0.49663  0.497104  0.499281  0.498323   0.50000  0.496196   0.50000  0.500000  0.500000 0.5242850
```

Comparative Analysis of Kernels:

Kernel Type	Support Vectors	Training Error	Overall Accuracy
Gaussian	8420	***79.22%***	***14.35%***
Polynomial	8372	***84.39%***	***12.56%***
Euclidean	8372	***84.42%***	***12.65%***
Hyperbolic Tangent	8310	***89.22%***	***11.90%***

3. **Question 2: Neural Networks**

Data Preprocessing:

In this segment of the research, we navigated through the complexities of our dataset, which mainly consisted of categorical variables. A key aspect of our study was the target variable ***GAD_T.*** This variable is extremely versatile, serving both as a numerical and categorical measurement. ***GAD_T*** represents a score that can be further categorized to determine the presence or absence of anxiety in individuals. We first imported the dataset. Then we determined the categorical and numerical variables. Then we used the MinMaxScaler to adjust the dataset to a **0-1** scale to standardize the range of continuous initial variables, improving the comparability and performance of neural network models. Then we performed One-Hot Encoding for our categorical data. Then we displayed the transformation from its original state to its normalized and encoded format. These are our results of this step:

```
Before Normalization:              Age        GAD_T         Hours    Narcissism        SPIN_T  \
count  10591.000000  10591.00000  10591.000000  10591.000000  10591.000000
mean      20.828911      5.17411     21.360023      2.021433     19.730998
std        3.154145      4.66851     13.257550      1.057378     13.368407
min       18.000000      0.00000      0.000000      1.000000      0.000000
25%       18.000000      2.00000     12.000000      1.000000      9.000000
50%       20.000000      4.00000     20.000000      2.000000     17.000000
75%       22.000000      8.00000     28.000000      3.000000     28.000000
max       56.000000     21.00000    120.000000      5.000000     68.000000

              SWL_T
count  10591.000000
mean      19.838825
std        7.181354
min        5.000000
25%       14.000000
50%       20.000000
75%       26.000000
max       35.000000
After Normalization:              GAD_T         Hours        SPIN_T         SWL_T  \
count  10591.000000  10591.000000  10591.000000  10591.000000
mean       0.246386      0.178000      0.290162      0.494628
std        0.222310      0.110480      0.196594      0.239378
min        0.000000      0.000000      0.000000      0.000000
25%        0.095238      0.100000      0.132353      0.300000
50%        0.190476      0.166667      0.250000      0.500000
75%        0.380952      0.233333      0.411765      0.700000
max        1.000000      1.000000      1.000000      1.000000
```

Then we split the dataset to test and train **(80, 20 splits)** and created the X_train, X_test, y_train, and y_test and this is the result from that:

```
((8472, 35), (2119, 35))
```

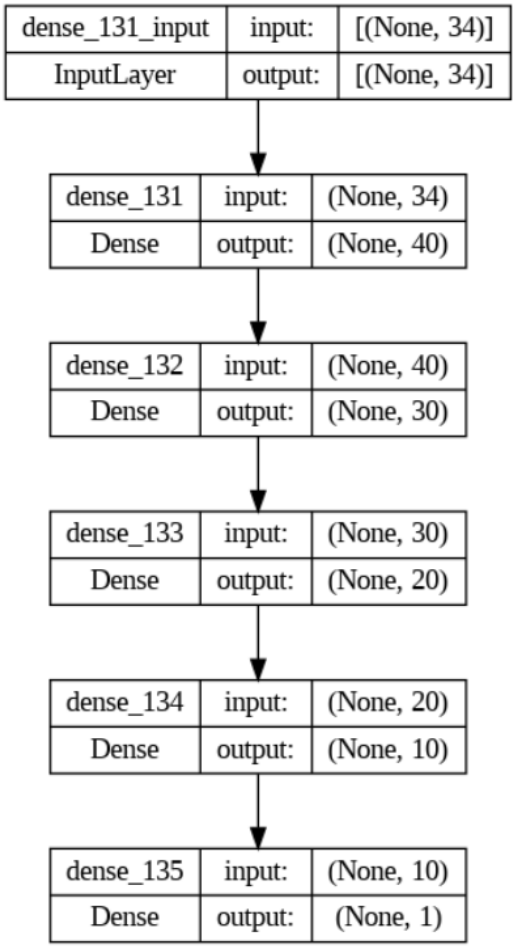
```
[58] # Preparing the data for neural network models
     X_train = train_data.drop('GAD_T', axis=1)
     y_train = train_data['GAD_T']
     X_test = test_data.drop('GAD_T', axis=1)
     y_test = test_data['GAD_T']
```

Building and evaluating the model:

As explained earlier about our target variable, we first did a neural network regression model with all the scores **(0-21)** of ***GAD_T*** against our other features.
We wrote a function to build, train, and evaluate the neural network model and then created different layer combinations we wanted to test for our model with layer and node details and different activations in different variables. Then we looped through these both and called the above-defined function to build, train, and evaluate the model and stored the results in a new variable.

Then we plotted the neural network architecture of the final model that ran in the loop, and this is the result we got from that.

```python
# Assuming 'model' is your Keras model
plot_model(model_reg, to_file='model_reg_plot.png', show_shapes=True, show_layer_names=True)
```



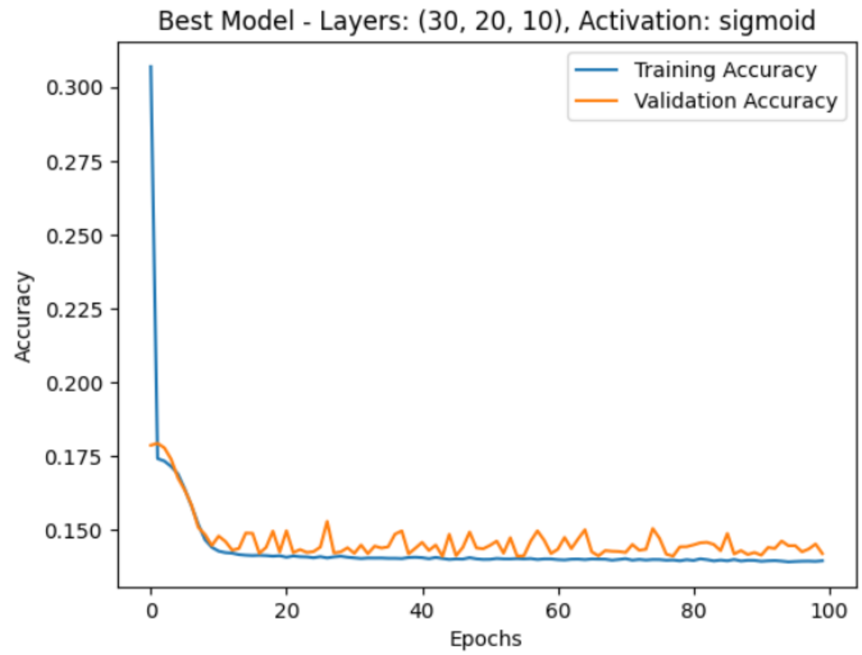
Then we printed out the results for all the different combinations of different layers, nodes, and activations which can be found in our code file. This is a small snippet of the results that we printed out.

```
results_reg_nn
```

```
{((10,),
  'relu'): {'loss': [0.06763207912445068,
   0.04185984397888184,
   0.03746723756194115,
   0.036340437829494476,
   0.0357864573597908,
   0.035379815846681595,
   0.035160861909389496,
   0.03506303206086159,
   0.0349060520529747,
   0.03486179932951927,
   0.034709978848695755,
   0.034604039043188095,
   0.0345996618270874,
   0.034503109753131866,
   0.03446292504668236,
   0.03434351459145546,
   0.03439820557832718,
   0.03435637801885605,
   0.03432324528694153,
   0.03415769338607788,
   0.034300852566957474,
   0.034225933253765106,
   0.03406434878706932,
   0.034058745950460434,
   0.034002020955085754,
   0.03400377184152603,
   0.03401808813214302,
   0.03397216647863388,
   0.03400079160928726,
   0.03396153822541237,
   0.03390432149171829,
   0.03401123359799385,
   0.033840008080005646,
   0.03387155383825302,
```

Then we stored the best configuration in a new variable based on the best MAE. And plotted the results showing the training and validation accuracy changes for different epochs.

```python
# Plotting the training and validation metrics of the best model
plt.plot(best_history_reg_nn['mae'], label='Training Accuracy')
plt.plot(best_history_reg_nn['val_mae'], label='Validation Accuracy')
plt.title(f"Best Model - Layers: {best_config_rg_nn[0]}, Activation: {best_config_rg_nn[1]}")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



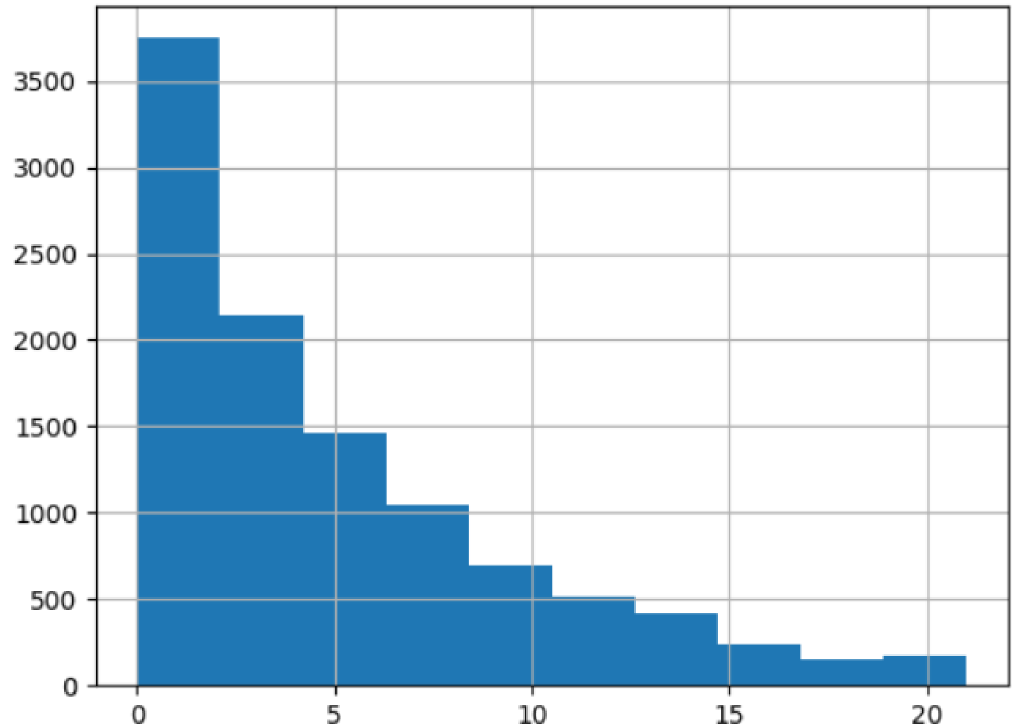
And then we sorted out the results and printed out the top 5 configurations with the highest accuracies based on the 'mae' and these are the results for that:

```
Top 5 Configurations:
Rank 1: Configuration ((30, 20, 10), 'sigmoid'), Best Validation Accuracy: 0.1793
Rank 2: Configuration ((10,), 'relu'), Best Validation Accuracy: 0.1781
Rank 3: Configuration ((10,), 'sigmoid'), Best Validation Accuracy: 0.1759
Rank 4: Configuration ((40, 30, 20, 10), 'sigmoid'), Best Validation Accuracy: 0.1744
Rank 5: Configuration ((10, 10), 'sigmoid'), Best Validation Accuracy: 0.1743
```

From the above results, the best configuration was with **3 hidden layers** with **30 nodes in the first hidden layer, 20 in the second, and 10 in the third,** and using sigmoid activation with an **accuracy score** of ***17.93%***

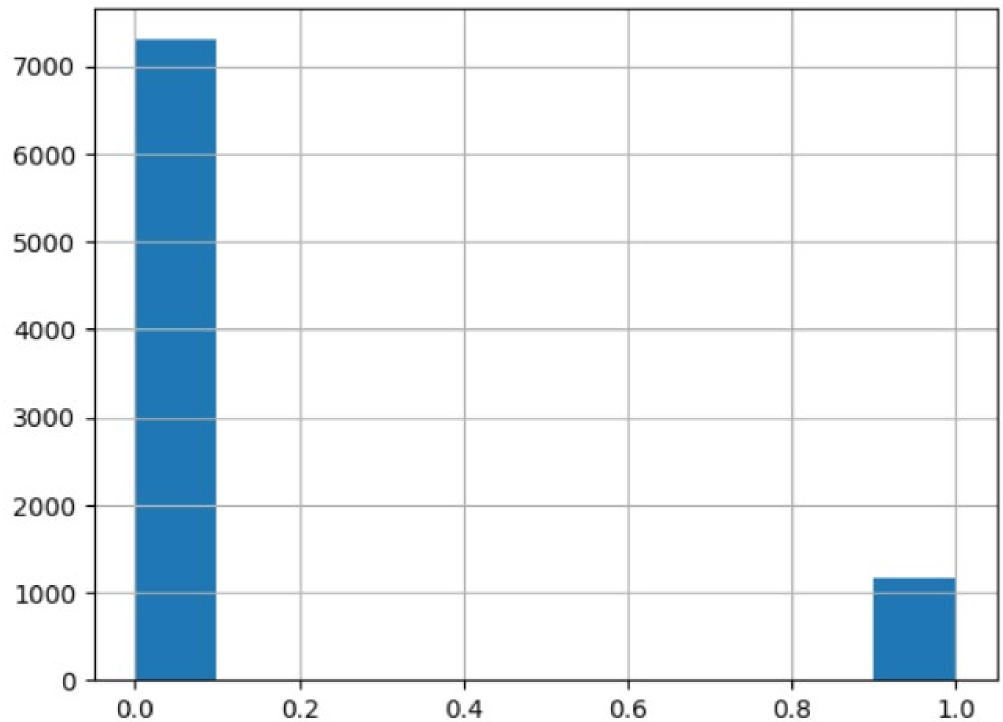
We wanted to see why we got such a low accuracy score and we found out that the distribution of our dependent variable in different classes is extremely skewed as you can see from the below plot that we generated to see the class balance.

<Axes: >



Then we explored the impact of categorizing the ***GAD_T*** variable into two distinct groups: individuals with anxiety and those without, based on the median score of ***GAD_T***. By doing this, we are aiming to see if this approach would improve our neural network model to have better accuracy using the neural network classification model.

So, we first converted the ***GAD_T*** to two categories **(0, 1)**. 0 representing no anxiety and 1 representing anxiety. And we plotted the distribution of this using a histogram and this is the result of that:



Then we changed the function that we used previously to build, train, and evaluate the neural network model to the neural network classification model and followed the same steps as before with minor changes such as MAE and MSE in regression to accuracy and correlation for classification and also made a few changes to activations and ran the models and stored the results similarly in a new variable. And displayed the results. And here is a snippet of the results that we got from this.

```
{((10,), 'relu'): (0.8669183577159038,
  array([[1759,   46],
         [ 236,   78]]),
  0.5285775502196223),
 ((20,), 'relu'): (0.8636149126946673,
  array([[1755,   50],
         [ 239,   75]]),
  0.5026444226163517),
 ((10, 10), 'relu'): (0.8664464369985843,
  array([[1761,   44],
         [ 239,   75]]),
  0.5086555225069296),
 ((20, 10), 'relu'): (0.8570080226521944,
  array([[1740,   65],
         [ 238,   76]]),
  0.46629597113176174),
 ((20, 20), 'relu'): (0.8522888154789995,
  array([[1724,   81],
         [ 232,   82]]),
  0.4303738545892036),
 ((30, 20, 10), 'relu'): (0.8381311939594148,
  array([[1682,  123],
         [ 220,   94]]),
  0.38906739492211834),
 ((40, 30, 20, 10), 'relu'): (0.8334119867862199,
  array([[1690,  115],
         [ 238,   76]]),
  0.29454821714020474),
 ((10,), 'logistic'): (0.8673902784332232,
  array([[1774,   31],
         [ 250,   64]]),
  0.5420874089625065),
```

We sorted these results based on accuracy and printed out the best configuration and evaluation metrics of that and here are the results for that.

```
Best Configuration: ((40, 30, 20, 10), 'logistic')
Best Accuracy: 0.8716375648890986
Best Correlation: 0.5325210659339328
Confusion Matrix:
[[1767   38]
 [ 234   80]]
```

We got the best accuracy for the configuration with **four hidden layers** and **40 nodes** in the **first hidden layer** followed by **30** in the **second** and **20** in the **third** and **10** in the **fourth layer** using **logistic activation** which gave us an **accuracy of 87.16%**.

Neural Network final summary:

From the above analysis, we could say that neural networks would not be really applicable to make predictions since we got such low scores if we use our dataset as it is without categorizing it.

4. Question 3: Clustering

Data Preprocessing:

In this section, we first set the Google Collab environment to run R codes. Then started with the installation of necessary packages and loading of these packages into the R workspace in Google environment. Below are the packages that were installed and its importance going forward to view clustering results -

- **factoextra:** To visualize clusters obtained
- **NbClust:** To evaluate the number of clusters
- **cluster:** To perform clustering
- **readr:** To read the uploaded clean dataset csv in Google Collab Workspace
- **dplyr:** To filter, select, and transform datasets before clustering
- **ggplot2:** To visualize the clustering results using scatter plot
- **Rtsne:** To reduce a dataset with many variables into two components so that visualization can be done in 2D
- **fpc:** To evaluate the quality of clusters methods like finding out the silhouette width for each point
- **dbscan:** To identify clusters from a dataset without specifying the no. of clusters. Used for DBSCAN clustering
- **dendextend:** To customize the dendrogram
- **caret:** To evaluate performance of different classifiers across cross-validation methods
- **Mlbench:** To do comparative analysis of different classifiers using artificial and real-world benchmark data
- **lattice:** To understand the structure of high-dimensional data before and after clustering

We then moved with loading the cleaned dataset and selecting only the variables that concern our research question - ***"GAD_T", "Narcissism", "SPIN_T", "Hours", "SWL_T", "whyplay_clean", "Work", "Playstyle_clean"***. Lastly, all the character columns amongst the chosen variables in our dataset are changed to factors so that statistical models can treat them appropriately.

```
Rows: 10591 Columns: 21
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (15): Birthplace, Degree, GADT_Cat, Game, Gender, League_clean, Narcissi...
dbl  (6): Age, GAD_T, Hours, Narcissism, SPIN_T, SWL_T

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Model Training & Evaluation:

Since we have a mix of categorical and numerical variables, we went ahead with not scraping them off but including them and performing three clustering - **PAM, Hierarchical** and **DBSCAN.**

Since we are tackling mixed data , we worked with **Gower Distance** over Euclidean Distance to build a distance matrix that is going to be used for clustering for quantifying the similarity or dissimilarity between each pair of the observations in the dataset.

Model Results:

PAM Clustering

First, we start with creating the Gower distance object, then a summary of the object reveals an overview of the distances calculated.

```
56079345 dissimilarities, summarized :
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.3036  0.3861  0.3850  0.4761  0.9133
Metric :  mixed ;  Types = I, I, I, I, I, N, N, N
Number of objects : 10591
```

Then the conversion of the Gower object to a matrix so that it's easier to access the individual elements and perform matrix operations. Once we got similar pairs, it helped validate clustering by seeing if they belonged to the same or different clusters.

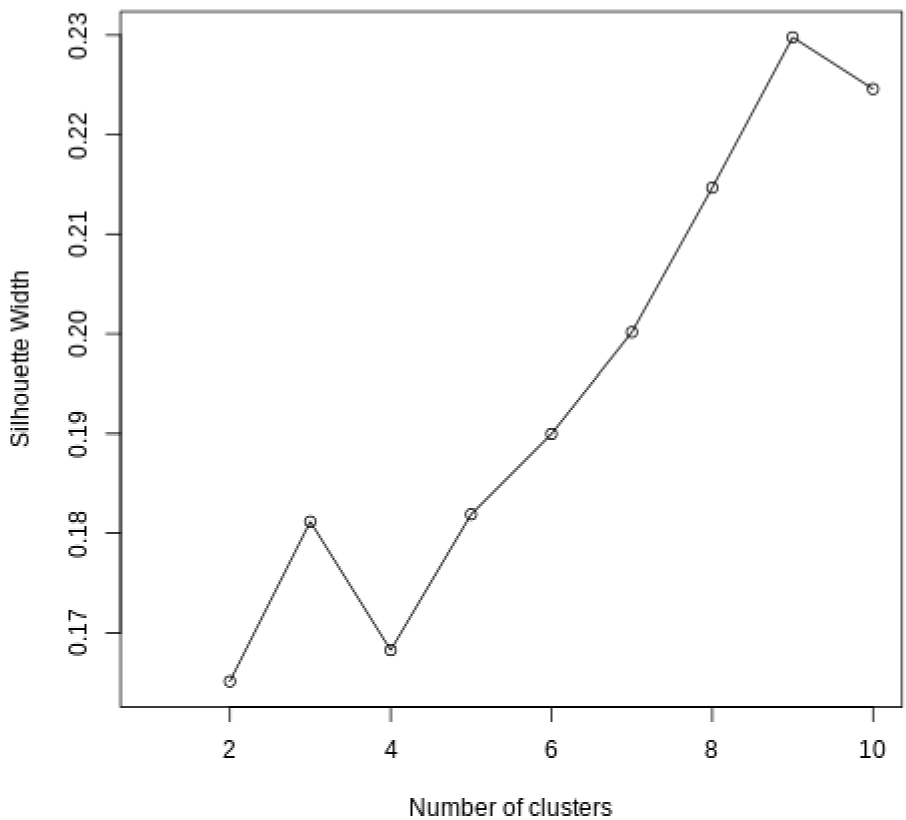
```
    GAD_T Narcissism SPIN_T Hours SWL_T whyplay_clean Work         Playstyle_clean
    <dbl>      <dbl>  <dbl> <dbl> <dbl> <fct>         <fct>        <fct>
  1     1          2     19    22    24 fun           Student at … Multiplayer - …
  2     1          2     19    21    24 fun           Student at … Multiplayer - …
```

For dissimilar pairs it helped to see if there are potential outliers and how diverse is the dataset.

```
    GAD_T Narcissism SPIN_T Hours SWL_T whyplay_clean Work         Playstyle_clean
    <dbl>      <dbl>  <dbl> <dbl> <dbl> <fct>         <fct>        <fct>
  1    21          5     68    40     5 winning       Student at … Multiplayer - …
  2     3          1      0    84    35 improving     Employed     Multiplayer - …
```

A. Optimal number of clusters

The optimal no. of clusters is identified by applying the gower matrix to the PAM algorithm for each value of k (no. of clusters) and then calculating the corresponding average silhouette width. This loop runs till k=10. The average silhouette width will help us understand how similar an object is in its cluster compared to other clusters. We then plotted the silhouette width.



From the graph, we observed that the optimal number of clusters is 9 since the graph observes a sudden dip in the width after a long continuous increasing trend. Such a high value of optimal no. of clusters indicates that the data is quite complex with several subtypes or variations and noisy.

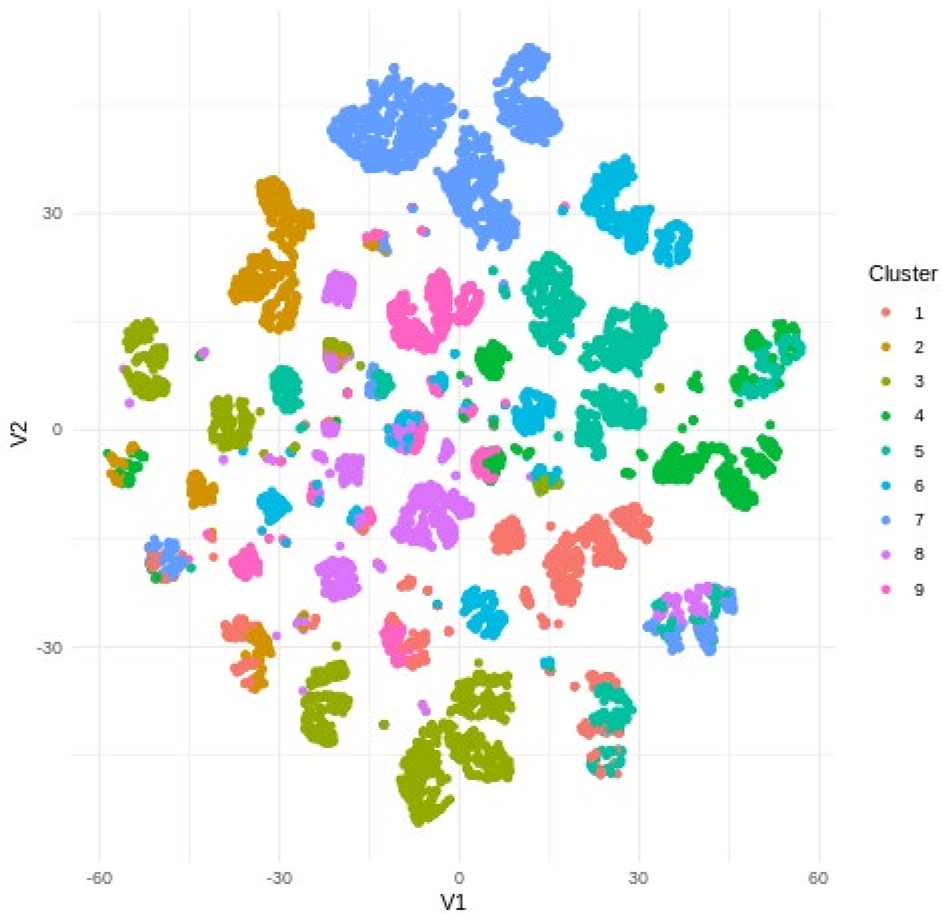
B. Elements per cluster

Below is the distribution of elements across 9 clusters -

1	2	3	4	5	6	7	8	9
1085	851	1501	888	1554	1093	1679	1084	856

C. Structural Plots

Our dataset is a high-dimensional data and hence we had to reduce it to two dimensions using t-SNE method so that we can view the scatter plot for a 9 cluster PAM clustering. It can be seen that there are some dense clusters and some clusters that are spread out. The latter shows the diversity and the outliers while the former shows similarity between those points. There are a lot of overlapping areas as well. Overall, the higher no. of clusters is showing the complexity of our dataset.



D. Interpretation of clusters

There are some key observations that we made by looking at the statistics of each cluster. (Note- Brackets have values of the measure that is being discussed in each point)

- ***GAD_T (Generalized Anxiety Disorder Test Score):*** Cluster 8 with highest mean (6.982) could represent individuals with highest levels of anxiety compared to cluster 1 with lowest mean (3.197).
- ***Narcissism:*** Cluster 8 with highest mean (2.321) contains individuals with highest narcissistic traits than those in cluster 9 (1.612).
- ***SPIN_T (Social Phobia Inventory Test Score):*** Clusters 2 and 3 with highest SPIN_T (68) have individuals with the most severe instances of social phobia,

while clusters 1,2 and 9 include individuals with the least or no indications of social phobia (0). Interestingly, Cluster 2 and 9 have the highest diversity because of the most as well as least social phobia individuals.

- ***Hours (Time Spent Playing Games):*** Cluster 3 includes the most hardcore gamers (120) who spend a significant amount of time playing games, while cluster 1 includes casual gamers or those who spend the least or no time playing games (0).
- ***SWL_T (Satisfaction With Life Test Score):*** All clusters have individuals who are fully satisfied with their lives (35), but clusters 2,3,6 and 7 also include individuals who are least satisfied with their lives (5). Hence, in general, there are a lot of gamers who are satisfied with life compared to those who aren't'.
- ***'whyplay_clean' (Reasons for Playing):*** Cluster 3 contains highest no. of individuals who play games with the intent to improve skills or achieve mastery (1434 mentions), whereas cluster 2 has lowest no. of individuals who play games for distraction (2 mentions). Hence, playing games for mastering is the most popular reason and distraction is the least.
- ***Work (Employment Status):*** Cluster 1 is primarily composed of individuals who are employed (1039), suggesting older adults with jobs, while cluster 7 is predominantly made up of university students (1583), suggesting a younger demographic. Two extreme categories of gamers, predominantly high in number, indicate diversity in the dataset.
- ***Playstyle_clean (Gaming Style):*** Cluster 5 consists of social gamers who play online with real-life friends (1532), indicating a preference for social gaming experiences, whereas cluster 6 seems to include gamers who prefer playing alone (70). Two extreme categories of gamers, predominantly high in number, indicate diversity in the dataset.

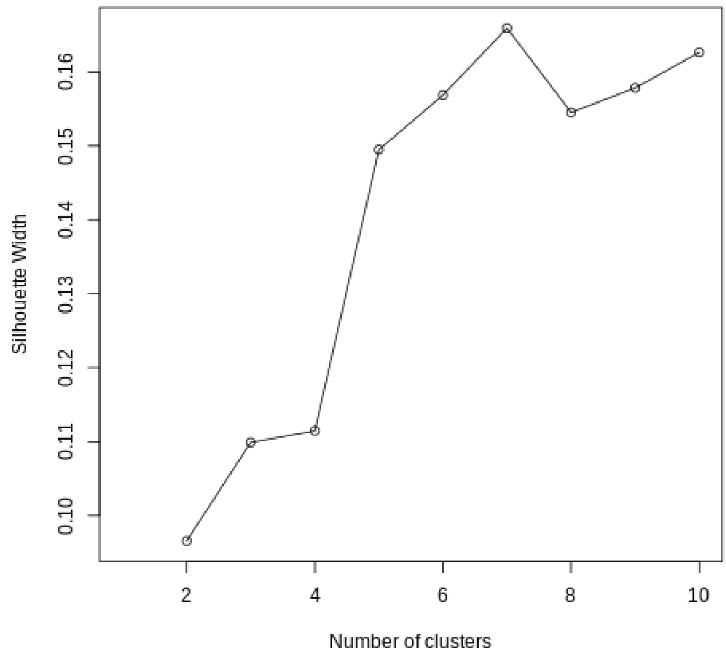
Below is the output for cluster 1. Similar outputs are received for each of the clusters 2 to 9.

```
[[1]]
     GAD_T          Narcissism        SPIN_T          Hours       
 Min.   : 0.000   Min.   :1.000   Min.   : 0.00   Min.   :  0.00  
 1st Qu.: 1.000   1st Qu.:1.000   1st Qu.: 5.00   1st Qu.: 10.00  
 Median : 2.000   Median :2.000   Median :10.00   Median : 15.00  
 Mean   : 3.197   Mean   :1.988   Mean   :13.24   Mean   : 18.17  
 3rd Qu.: 4.000   3rd Qu.:3.000   3rd Qu.:18.00   3rd Qu.: 24.00  
 Max.   :21.000   Max.   :5.000   Max.   :62.00   Max.   :100.00  
     SWL_T           whyplay_clean                              Work     
 Min.   : 5.00   all        : 13   Employed                       :1039  
 1st Qu.:18.00   distraction:  0   Student at college / university:   0  
 Median :24.00   fun        :724   Student at school              :   0  
 Mean   :22.73   improving  :156   Unemployed / between jobs      :  46  
 3rd Qu.:28.00   relaxing   : 53                                         
 Max.   :35.00   winning    :139                                         
                                                Playstyle_clean
 all                                                    :  5   
 multiplayer - offline                                  :  8   
 Multiplayer - online - with online acquaintances or teammates:120   
 Multiplayer - online - with real life friends          :776   
 Multiplayer - online - with strangers                  :125   
 singleplayer                                           : 51   
    cluster 
 Min.   :1  
 1st Qu.:1  
 Median :1  
 Mean   :1  
 3rd Qu.:1  
 Max.   :1  
```

Hierarchical Clustering

A. Optimal number of clusters

First, we start with creating the Gower distance matrix. The optimal no. of clusters is identified by applying the gower matrix to the Hierarchical clustering algorithm. For each value of k (no. of clusters) from 2 to 10, the dendrogram produced by Hierarchical clustering algorithm is cut into k clusters. The silhouette width is calculated given the cluster cuts and gower matrix. Then average silhouette width for each k will help us understand how similar an object is in its cluster compared to other clusters. We then plotted the silhouette width.



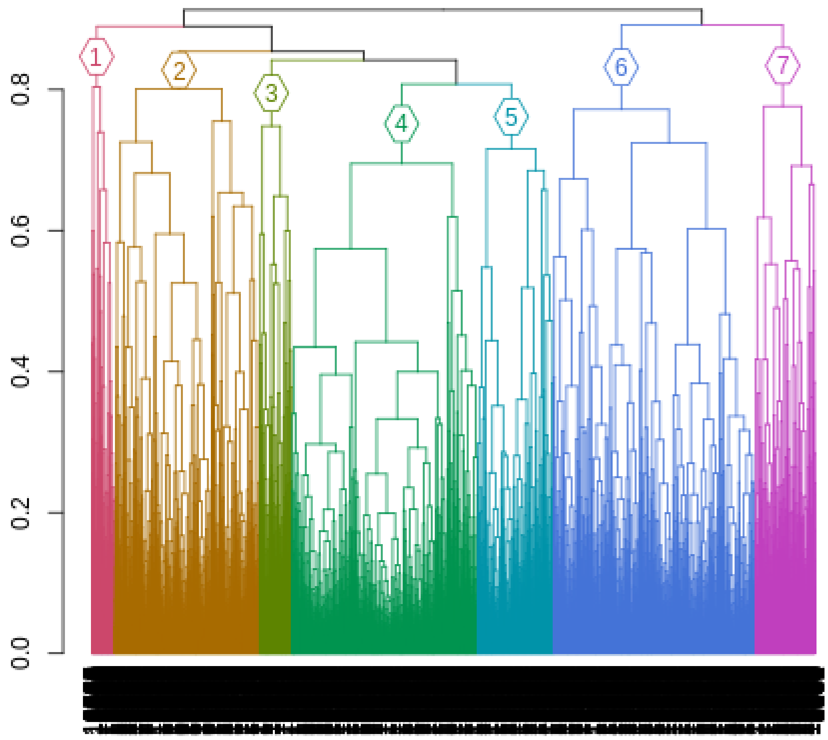
B. Elements per cluster

Below is the distribution of elements across 9 clusters -

1	2	3	4	5	6	7
2118	2950	874	2719	1121	474	335

C. Structural Plots

The dendrogram is used to see the clusters produced in hierarchical clustering. Clusters numbered 1, 2, and 7 merge at higher points on the scale, suggesting that they are quite distinct from each other and from the rest of the data. Cluster 4 is the most distinct, merging last and at the highest point. Width of the colored areas show areas that are denser than others and vice versa.



D. Interpretation of clusters

There are some key observations that we made by looking at the statistics of each cluster. (Note- Brackets have values of the measure that is being discussed in each point)

- ***GAD_T (Generalized Anxiety Disorder Test Score):*** Cluster 7 with highest mean (10.24) could represent individuals with highest levels of anxiety compared to cluster 1 with lowest mean (4.328).
- ***Narcissism:*** Cluster 7 with highest mean (2.236) contains individuals with highest narcissistic traits than those in cluster 3 (1.928).
- ***SPIN_T (Social Phobia Inventory Test Score):*** Clusters 7,5 and 2 with highest SPIN_T (68) have individuals with the most severe instances of social phobia, while all clusters include individuals with the least or no indications of social phobia (0). Interestingly, Cluster 7,5 and 2 has the highest diversity because of the most as well as least social phobia individuals.
- ***Hours (Time Spent Playing Games):*** Cluster 1 and 3 includes the most hardcore gamers (120) who spend a significant amount of time playing games, while cluster 1,2 and 4 includes casual gamers or those who spend the least or no time playing games (0).

- ***SWL_T (Satisfaction With Life Test Score):*** All clusters have individuals who are fully satisfied with their lives (35) and who are least satisfied with their lives (5). Hence, in general, there are a lot of gamers who are satisfied with life and those who aren't'.
- ***'whyplay_clean' (Reasons for Playing):*** Cluster 4 contains the highest no. of individuals who play games for fun (1362 mentions), whereas cluster 4 and 5 have lowest no. of individuals who play games for distraction, relaxation (2 mentions). Hence, playing games for fun is the most popular reason and distraction, relaxation is the least.
- ***Work (Employment Status):*** Cluster 2 is primarily composed of individuals who are university students (2945), while cluster 7 is predominantly made up of employed people (1941), suggesting an older demographic. Two extreme categories of gamers, high in number, indicate diversity in the dataset.
- ***Playstyle_clean (Gaming Style):*** Cluster 7 consists of social gamers who play online with real-life friends (1639), indicating a preference for social gaming experiences, whereas cluster 6 seems to include gamers who prefer playing with strangers (1454). Two extreme categories of gamers, predominantly high in number, indicate diversity in the dataset.

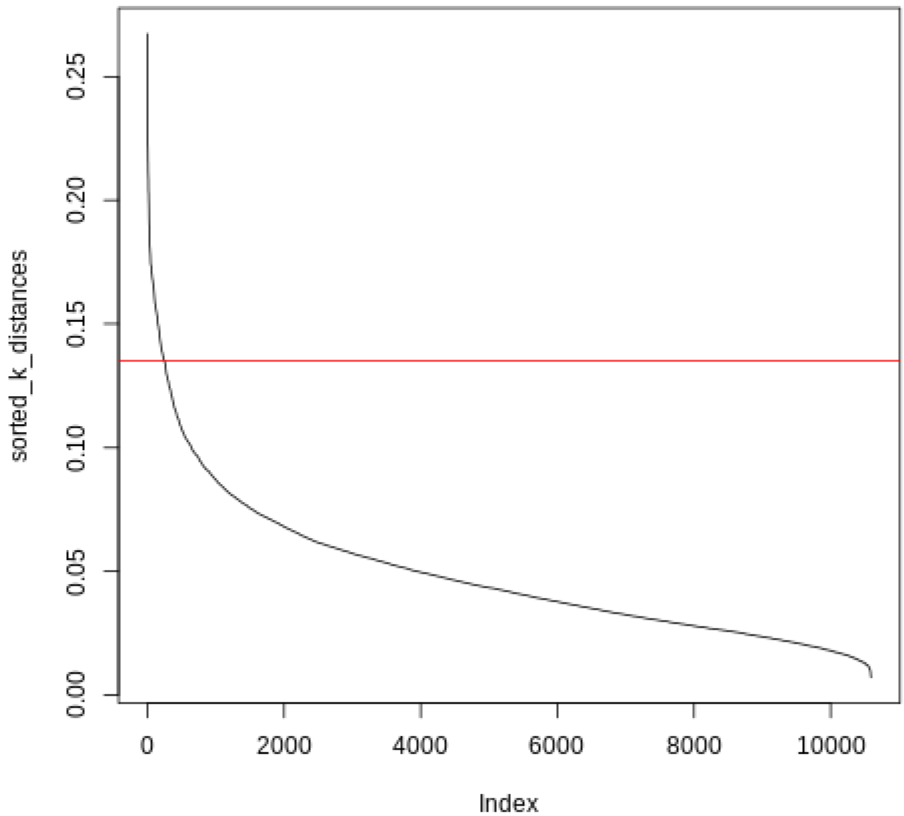
  Below is the output for cluster 1. Similar outputs are received for each of the clusters 2 to 7.

```
[[1]]
     GAD_T          Narcissism        SPIN_T          Hours       
 Min.   : 0.000   Min.   :1.000   Min.   : 0.00   Min.   :  0.00  
 1st Qu.: 1.000   1st Qu.:1.000   1st Qu.: 7.00   1st Qu.: 10.00  
 Median : 3.000   Median :2.000   Median :14.00   Median : 20.00  
 Mean   : 4.328   Mean   :2.005   Mean   :16.77   Mean   : 19.66  
 3rd Qu.: 6.000   3rd Qu.:3.000   3rd Qu.:23.00   3rd Qu.: 25.00  
 Max.   :21.000   Max.   :5.000   Max.   :64.00   Max.   :120.00  
     SWL_T          whyplay_clean                                 Work     
 Min.   : 5.00   all        : 24   Employed                          :1941  
 1st Qu.:15.00   distraction:  1   Student at college / university: 128  
 Median :21.00   fun        :864   Student at school              :  46  
 Mean   :21.05   improving  :742   Unemployed / between jobs      :   3  
 3rd Qu.:27.00   relaxing   :182                                          
 Max.   :35.00   winning    :305                                          
                                                       Playstyle_clean
 all                                                           : 23   
 multiplayer - offline                                         : 19   
 Multiplayer - online - with online acquaintances or teammates:438   
 Multiplayer - online - with real life friends                :881   
 Multiplayer - online - with strangers                        :755   
 singleplayer                                                  :  2   
    cluster 
 Min.   :1  
 1st Qu.:1  
 Median :1  
 Mean   :1  
 3rd Qu.:1  
 Max.   :1  
```

DBSCAN Clustering

A. Optimal number of clusters

First, we start with setting the seed at 123 so that results of the processes are reproducible. Then, using the Gower distance, k-nearest neighbor distances for each point in the dataset is calculated. k=5 indicates that the distance to the 5th nearest neighbor is calculated for each point. Once these distances are sorted, they are plotted against the point. Then “elbow point” of the plot helps to determine the no. of clusters. From the below picture, the elbow was spotted at 0.135 (marked by a red line).



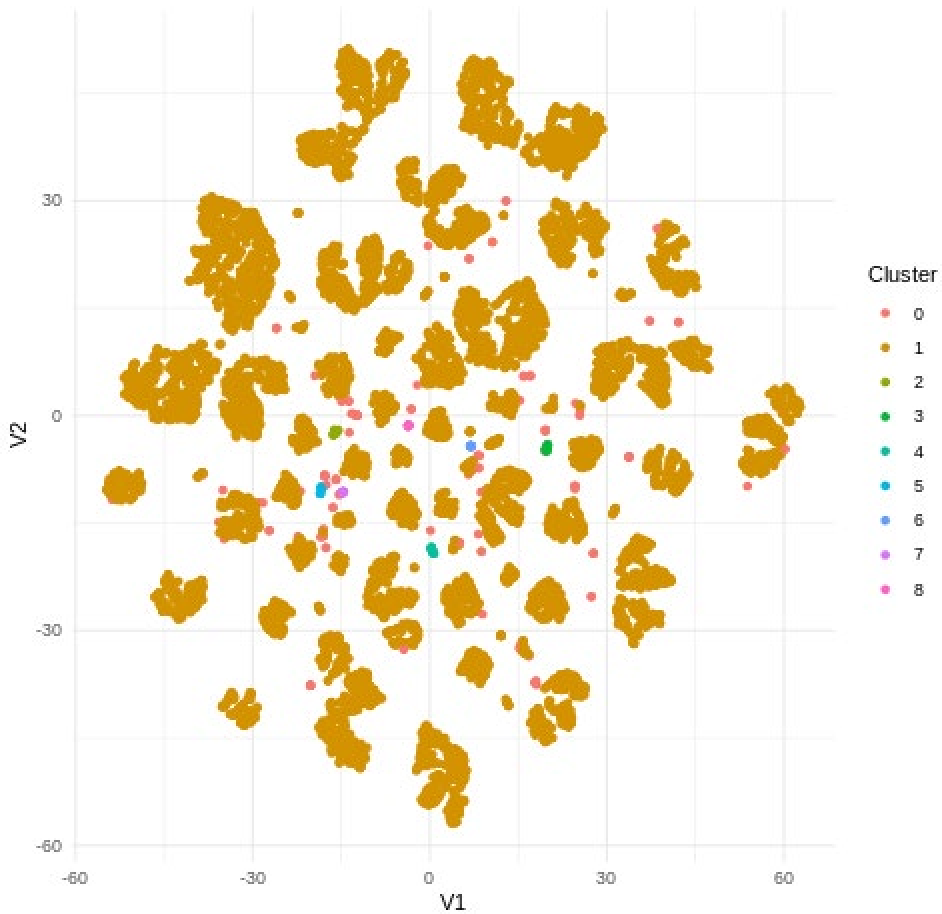
B. Elements per cluster

The below of 0.135 results in 9 clusters of below distribution -

0	1	2	3	4	5	6	7	8
116	10386	15	17	13	14	8	9	13

C. Structural Plots

Our dataset is a high-dimensional data and hence we had to reduce it to two dimensions using t-SNE method so that we can view the scatter plot for a 9 cluster DBSCAN clustering. It can be seen that cluster 1 is very dense as well as spread out showing diversity as well as the outliers while showing similarity between those clusters. There are a lot of overlapping areas as well. Overall, the higher no. of clusters is showing the complexity of our dataset.



D. Interpretation of clusters

There are some key observations that we made by looking at the statistics of each cluster. (Note- Brackets have values of the measure that is being discussed in each point)

- ***GAD_T (Generalized Anxiety Disorder Test Score):*** Cluster 6 with highest mean (9.429) could represent individuals with highest levels of anxiety compared to cluster 8 with lowest mean (3.889).
- Narcissism: Cluster 1 with highest mean (2.319) contains individuals with highest narcissistic traits than those in cluster 3 (1.4).
- ***SPIN_T (Social Phobia Inventory Test Score):*** Clusters 1 and 2 with highest SPIN_T (68) have individuals with the most severe instances of social phobia, while 1,2 and 9 clusters include individuals with the least or no indications of social phobia (0). Interestingly, Cluster 1 and 2 has the highest diversity because of the most as well as least social phobia individuals.
- ***Hours (Time Spent Playing Games):*** Cluster 1 and 2 includes the most hardcore gamers (120) who spend a significant amount of time playing games, while cluster 2 includes casual gamers or those who spend the least or no time playing games (0). Cluster 2 is quite diverse with hardcore as well as casual gamers.

- ***SWL_T (Satisfaction With Life Test Score):*** Clusters 1 and 2 have individuals who are fully satisfied with their lives (35) and clusters 1,2,3 and 6 who are least satisfied with their lives (5). Hence, in general, there are a lot of gamers who are satisfied with life and those who aren't in clusters 1 and 2.
- ***'whyplay_clean' (Reasons for Playing):*** Cluster 2 contains the highest no. of individuals who play games for fun (4266 mentions), whereas cluster 2 has lowest no. of individuals who play games for any reason (2 mentions). Hence, playing games for fun is the most popular reason and people with no motivation are the least.
- ***Work (Employment Status):*** Cluster 2 is primarily composed of individuals who are university students (5654), while cluster 2 is predominantly made up of employed people (2062), suggesting an older demographic. Two extreme categories of gamers, predominantly high in number, indicate diversity in the dataset, especially in cluster 2.
- ***Playstyle_clean (Gaming Style):*** Cluster 2 consists of social gamers who play online with real-life friends (4797), indicating a preference for social gaming experiences, whereas cluster 2 seems to include gamers who prefer playing with strangers (3199). Two extreme categories of gamers, predominantly high in number, indicate diversity in the dataset, especially in cluster 2.

Below is the output for cluster 1. Similar outputs are received for each of the clusters 2 to 9.

```
    GAD_T          Narcissism       SPIN_T          Hours       
Min.   : 0.000   Min.   :1.000   Min.   : 0.00   Min.   :  2.00  
1st Qu.: 3.000   1st Qu.:1.000   1st Qu.: 9.00   1st Qu.: 15.75  
Median : 7.000   Median :2.000   Median :21.00   Median : 25.00  
Mean   : 8.776   Mean   :2.319   Mean   :24.57   Mean   : 30.28  
3rd Qu.:14.000   3rd Qu.:3.000   3rd Qu.:36.25   3rd Qu.: 40.00  
Max.   :21.000   Max.   :5.000   Max.   :68.00   Max.   :120.00  
    SWL_T          whyplay_clean                              Work   
Min.   : 5.00   all        :21    Employed                      :25  
1st Qu.:10.00   distraction:26    Student at college / university:22  
Median :17.00   fun        :12    Student at school             :26  
Mean   :16.82   improving  :15    Unemployed / between jobs     :43  
3rd Qu.:21.25   relaxing   :27                                      
Max.   :35.00   winning    :15                                      
                                                   Playstyle_clean
all                                                        :22    
multiplayer - offline                                      :13    
Multiplayer - online - with online acquaintances or teammates:22    
Multiplayer - online - with real life friends              :18    
Multiplayer - online - with strangers                      :21    
singleplayer                                               :20    
   cluster 
Min.   :0  
1st Qu.:0  
Median :0  
Mean   :0  
3rd Qu.:0  
Max.   :0  
```

5. Question 4: Comparative Analysis

In this section, we first set the Google Collab environment to run python codes and import the necessary libraries.
We then moved with loading the cleaned dataset and selecting only the variables that concern our research question - ***"GAD_T", "Narcissism", "SPIN_T", "Hours", "SWL_T", "whyplay_clean", "Work", "Playstyle_clean".*** Lastly, all the character columns are encoded using Label Encoder to numerical form for them to be used in the machine learning models - Random Forests, Decision Trees, SVMs and Neural Networks. Lastly, we define the target variable and the features.

Model Training & Evaluation:

We initialized four different classifications models that have to be used in Comparative Analysis - **Random Forest, Neural network, DecisionTree, and SVM.**
Then we define two functions - 1. A function that will perform cross-validation using the specified model, features, target variable and one of the three cross-validation strategies 2. A function that will perform bootstrapping cross validation using random sampling and replacement as there is no in-built ML model function that can perform it.

Then we initialize the remaining two cross-validation strategies using in-built Python functions KFold and RepeatedKFold. Lastly, we perform the four different models across three cross validation strategies (**standard K-Fold, repeated K-Fold, and bootstrapping**)

Model Results:

The results from the evaluated models across cross-validation strategies were viewed in terms of statistics like standard deviation, mean etc.

```
Summary Statistics for Model Performances:
           rf_cv  rf_repeatedcv    rf_boot    nnet_cv  nnet_repeatedcv  \
count  10.000000      30.000000  10.000000  10.000000        30.000000
mean    0.127466       0.129386   0.125858   0.140685         0.140717
std     0.008450       0.009828   0.003984   0.009935         0.011228
min     0.112370       0.110482   0.118707   0.122757         0.121813
25%     0.123466       0.123702   0.125215   0.136449         0.132200
50%     0.128895       0.129367   0.126430   0.142587         0.138679
75%     0.132169       0.134561   0.127431   0.147038         0.148017
max     0.141643       0.152030   0.132437   0.153919         0.167139

       nnet_boot   rpart_cv  rpart_repeatedcv  rpart_boot  svmRadial_cv  \
count  10.000000  10.000000         30.000000   10.000000     10.000000
mean    0.136756   0.113869          0.112675    0.106350      0.147484
std     0.004637   0.010174          0.009117    0.006878      0.010366
min     0.129863   0.101039          0.100000    0.096682      0.132200
25%     0.133581   0.105996          0.105052    0.100830      0.140699
50%     0.135584   0.110482          0.111426    0.105406      0.145352
75%     0.139588   0.124646          0.116856    0.111270      0.155807
max     0.145309   0.127358          0.135977    0.117563      0.162417

       svmRadial_repeatedcv  svmRadial_boot
count             30.000000       10.000000
mean               0.146791        0.143993
std                0.009538        0.004347
min                0.131256        0.137872
25%                0.140699        0.139731
50%                0.145892        0.145166
75%                0.151086        0.147669
max                0.166195        0.148741
```

Below is the summary that was extracted from the above result.

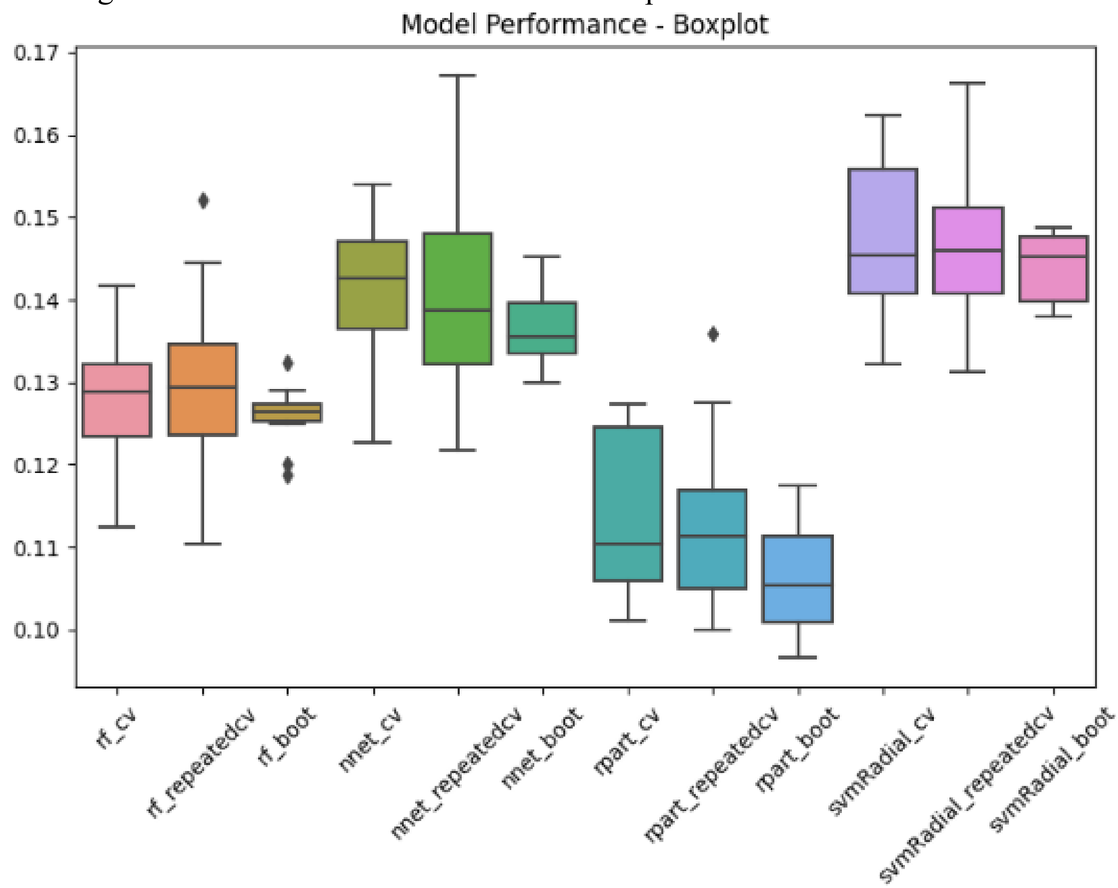
Metric	Best Model Performance	Worst Model Performance
Mean	SVM (Standard K-Fold): 14.75%	Decision Tree (Bootstrapping): 10.64%
Standard Deviation	Random Forest (Bootstrapping): 0.40%	Neural Network (Repeated K-Fold): 1.12%
Minimum	Random Forest (Bootstrapping): 11.87%	Decision Tree (Bootstrapping): 9.67%
25th Percentile	Random Forest (Bootstrapping): 12.52%	Decision Tree (Bootstrapping): 10.08%
Median (50%)	SVM (Standard K-Fold): 14.54%	Decision Tree (Bootstrapping): 10.54%
75th Percentile	SVM (Repeated K-Fold): 15.11%	Decision Tree (Standard K-Fold): 12.46%
Maximum	Neural Network (Repeated K-Fold): 16.71%	Decision Tree (Bootstrapping): 11.76%

SVM with Standard K-Fold cross-validation is the best overall model and Decision Tree with Bootstrapping method appears to be the least effective overall, given their scores across the considered metrics.

Now, we will look at the box plots results from the model performance across cross validation strategies.

Box plot:

Wider boxes suggest greater variability i.e., the model's performance is less consistent and more variable across the different iterations of the validation process and if the box is placed higher or lower than that decides the overall performance level.

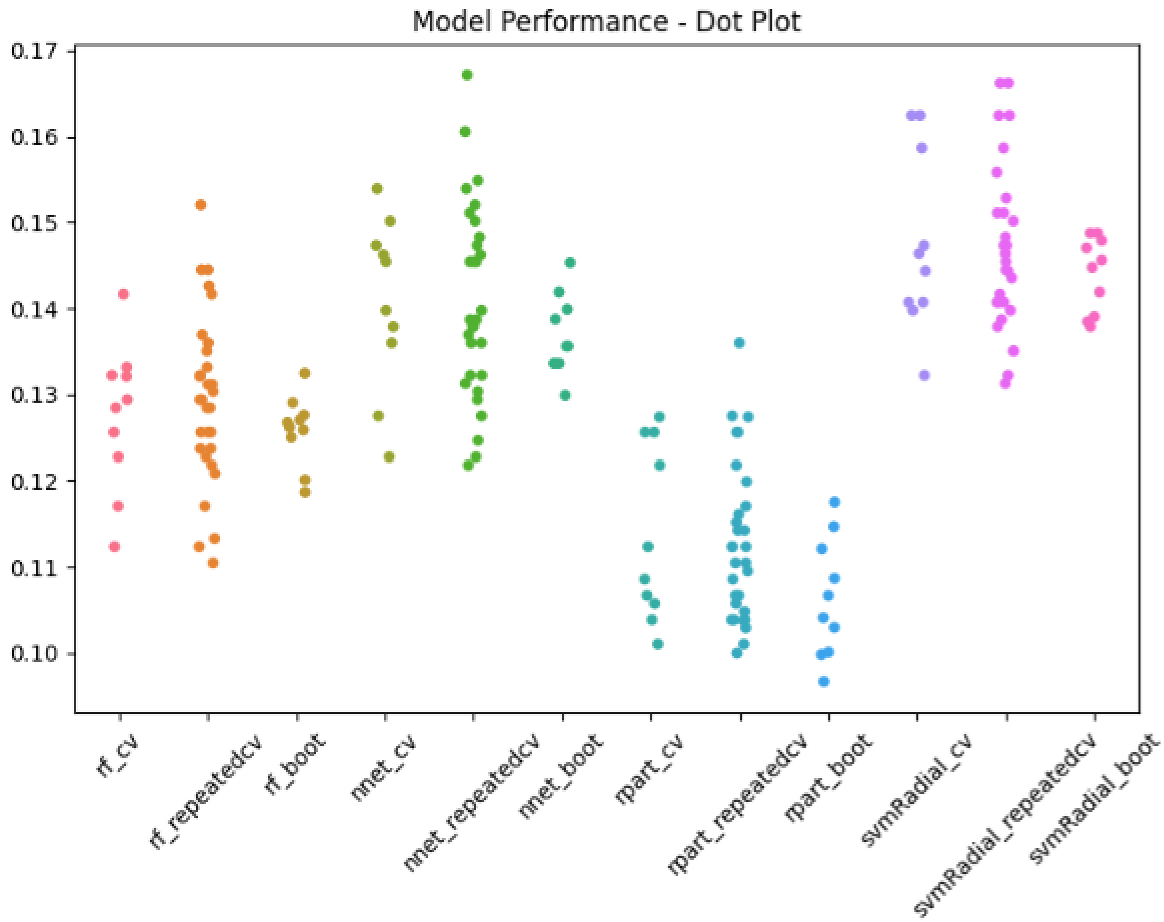


Decision Tree using bootstrap is the worst performing model due to it being placed at the lowest level. SVM with Standard K-Fold and that with repeated K-Fold cross-validation is placed almost at the same level providing best performance across models.

Decision tree with Standard K-Fold cross-validation is the widest hence its performance is less consistent and more variable while Random Forest with boosting has most consistent performance due its narrow width.

Dot Plot:

Wider spread suggests greater variability i.e., model's performance is less consistent and more variable across the different iterations of the validation process and also potential outliers if the points are not closely packed. If the group tends to have higher values, then it will have better performance compared to one with lower values.



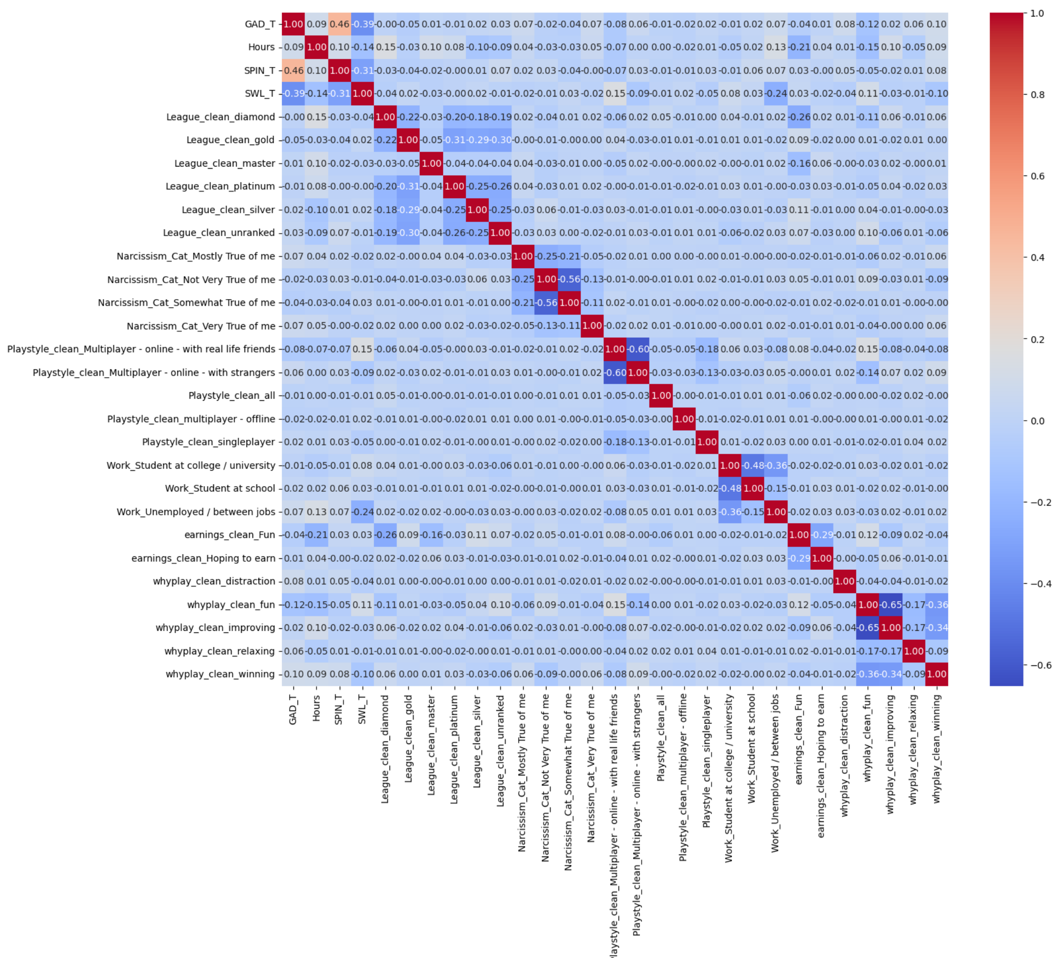
Decision Tree using bootstrap is the worst performing model due to it having lower values. SVM with Standard K-Fold and those with repeated K-Fold cross-validation have the highest values.

Decision tree with Standard K-Fold cross-validation has the widest spread hence its performance is less consistent and more variable while Random Forest with boosting has most consistent performance due its narrow width.

6. Question 5: Feature Selection

For feature selection with filters:

We first loaded the data and then encoded our categorical data and created the X_train, X_test, y_train, and y_test, and then calculated the correlations and created a heatmap for the correlations. Here is the result of that:



Then we ran the SVM model with all the features and calculated the metrics. And we got the following MSE:

```
Mean Squared Error: 16.06
Root Mean Squared Error: 4.01
```

Then from the correlation heatmap, we found out that all the classes related to playstyle had less correlation and we removed all the classes related to playstyle from our features.

```
# Update the model by removing a less significant feature
# For example, removing 'Playstyle_clean_all'
X_train_updated = X_train.drop(['Playstyle_clean_all', 'Playstyle_clean_Multiplayer - online - with real life friends', 'Playstyle_clean_Multiplayer - online -
X_test_updated = X_test.drop(['Playstyle_clean_all', 'Playstyle_clean_Multiplayer - online - with real life friends', 'Playstyle_clean_Multiplayer - online - w
```

And then we ran the SVM model with the updated dataset. And these are the results we got from that:

```
Mean Squared Error: 16.05
Root Mean Squared Error: 4.01
```

The MSE value **before removing any of the features** was **16.06** and **after removing** the **feature "playstyle"** it was **16.05**. Even though there is a very slight **decrease** in the **MSE**, it still represents how a model performance can be increased by removing any significant features from our feature matrix. As mentioned in the lecture it also improves the performance of the model.

Then for the next feature selection technique,

Applying Regularization(L1) (an embedded technique) With Neural Networks:

First, we followed the same code that we have for a neural network model that we created earlier. And then adding some changes to the code by adding an L1 regularization strength of 0.01. And printed the same results as we did in neural networks.

The results **after regularization**:

```
Top 5 Configurations:
Rank 1: Configuration ((10,), 'sigmoid'), Best Validation Accuracy: 0.1928
Rank 2: Configuration ((20, 20), 'sigmoid'), Best Validation Accuracy: 0.1874
Rank 3: Configuration ((20,), 'sigmoid'), Best Validation Accuracy: 0.1849
Rank 4: Configuration ((20, 20), 'relu'), Best Validation Accuracy: 0.1847
Rank 5: Configuration ((30, 20, 10), 'sigmoid'), Best Validation Accuracy: 0.1842
```

The results **before regularization**:

```
Top 5 Configurations:
Rank 1: Configuration ((30, 20, 10), 'sigmoid'), Best Validation Accuracy: 0.1793
Rank 2: Configuration ((10,), 'relu'), Best Validation Accuracy: 0.1781
Rank 3: Configuration ((10,), 'sigmoid'), Best Validation Accuracy: 0.1759
Rank 4: Configuration ((40, 30, 20, 10), 'sigmoid'), Best Validation Accuracy: 0.1744
Rank 5: Configuration ((10, 10), 'sigmoid'), Best Validation Accuracy: 0.1743
```

From the results, we can conclude that regularization improved the accuracy by a little bit.

For the third feature selection technique, we did SFS with linear regression:

We first loaded the dataset, then defined our base model with just no features, and then created a final model that had all the features. Then we used the stepwise algorithm in the forward direction and performed the Sequential Feature Selection (SFS). Then printed out the shortlisted vars and the summary of the step model.

This is the result of the shortlisted vars:

```
 [1] "(Intercept)"                          "SPIN_T"
 [3] "SWL_T"                                "whyplay_cleandistraction"
 [5] "whyplay_cleanfun"                     "whyplay_cleanimproving"
 [7] "whyplay_cleanrelaxing"                "whyplay_cleanwinning"
 [9] "Narcissism_CatMostly True of me"      "Narcissism_CatNot Very True of me"
[11] "Narcissism_CatSomewhat True of me"    "Narcissism_CatVery True of me"
[13] "League_cleandiamond"                  "League_cleangold"
[15] "League_cleanmaster"                   "League_cleanplatinum"
[17] "League_cleansilver"                   "League_cleanunranked"
[19] "earnings_cleanFun"                    "earnings_cleanHoping to earn"
[21] "WorkStudent at college / university" "WorkStudent at school"
[23] "WorkUnemployed / between jobs"        "Hours"
```

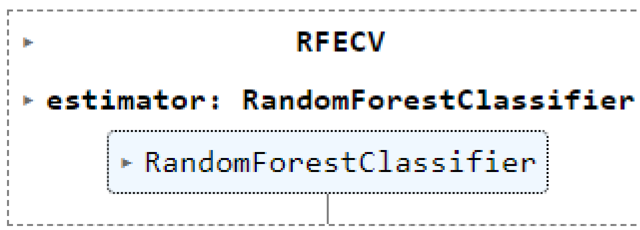
This is the summary of the step model:

```
lm(formula = GAD_T ~ SPIN_T + SWL_T + whyplay_clean + Narcissism_Cat +
    League_clean + earnings_clean + Work + Hours, data = dataforSFS)

Residuals:
     Min       1Q   Median       3Q      Max
-11.3310  -2.6256  -0.6518   1.8955  20.5710

Coefficients:
                                     Estimate Std. Error t value Pr(>|t|)
(Intercept)                          9.863388   1.402301   7.034 2.14e-12 ***
SPIN_T                               0.125444   0.003043  41.224  < 2e-16 ***
SWL_T                               -0.176996   0.005790 -30.572  < 2e-16 ***
whyplay_cleandistraction             3.731554   0.865384   4.312 1.63e-05 ***
whyplay_cleanfun                    -0.710103   0.448490  -1.583  0.11338
whyplay_cleanimproving              -0.304068   0.448629  -0.678  0.49793
whyplay_cleanrelaxing                0.642749   0.481473   1.335  0.18192
whyplay_cleanwinning                 0.043807   0.454999   0.096  0.92330
Narcissism_CatMostly True of me      0.666192   0.159269   4.183 2.90e-05 ***
Narcissism_CatNot Very True of me   -0.264509   0.111332  -2.376  0.01753 *
Narcissism_CatSomewhat True of me   -0.192584   0.114002  -1.689  0.09119 .
Narcissism_CatVery True of me        1.746655   0.266548   6.553 5.91e-11 ***
League_cleandiamond                 -3.245781   1.313055  -2.472  0.01345 *
League_cleangold                    -2.995952   1.309847  -2.287  0.02220 *
League_cleanmaster                  -2.845912   1.379940  -2.062  0.03920 *
League_cleanplatinum                -2.955779   1.310444  -2.256  0.02412 *
League_cleansilver                  -2.490981   1.310401  -1.901  0.05734 .
League_cleanunranked                -2.680360   1.310277  -2.046  0.04082 *
earnings_cleanFun                   -0.703626   0.150867  -4.664 3.14e-06 ***
earnings_cleanHoping to earn        -0.494799   0.443069  -1.117  0.26413
WorkStudent at college / university  0.275380   0.100839   2.731  0.00633 **
WorkStudent at school                0.173412   0.127854   1.356  0.17502
WorkUnemployed / between jobs       -0.035553   0.153667  -0.231  0.81704
Hours                                0.004823   0.003056   1.578  0.11454
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.92 on 10567 degrees of freedom
Multiple R-squared:  0.2965,    Adjusted R-squared:  0.2949
F-statistic: 193.6 on 23 and 10567 DF,  p-value: < 2.2e-16
```

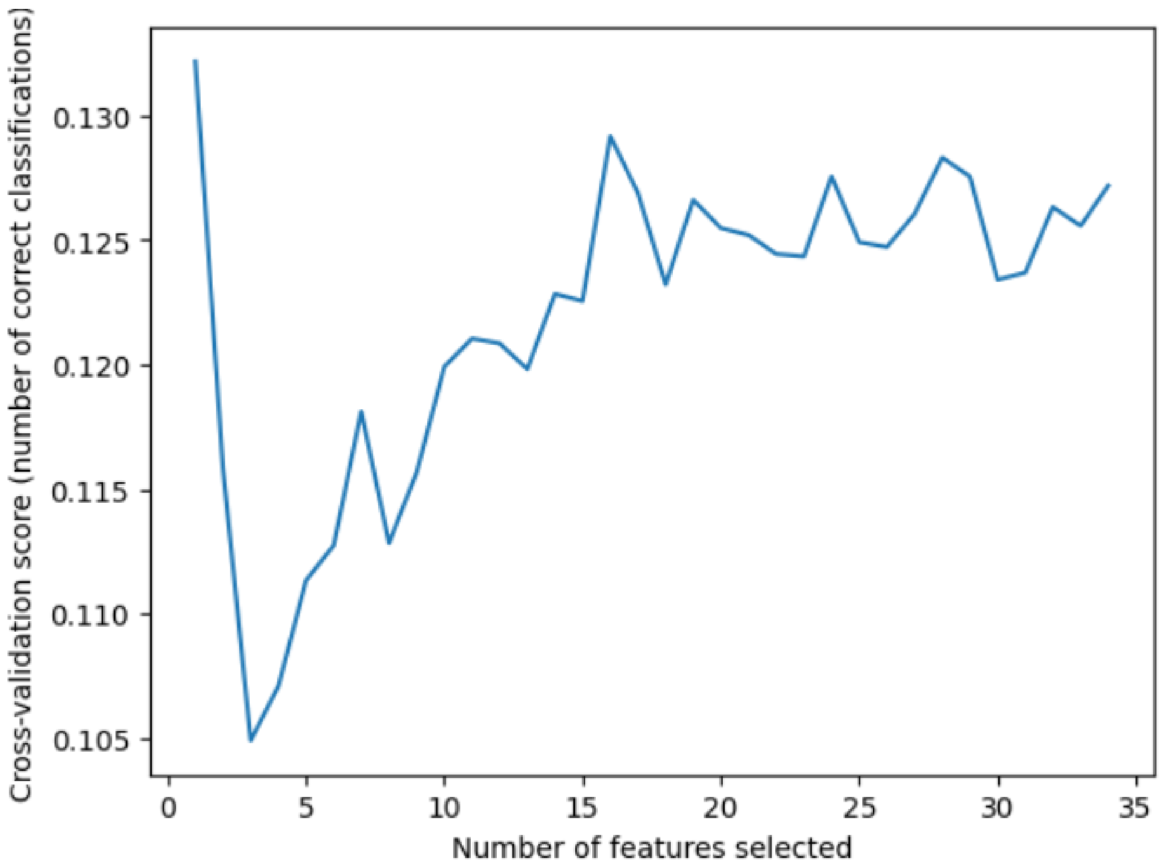
This summary clearly shows the features with higher significance with the * on the side.

For the final feature Selection, we also wanted to the hybrid technique i.e., RFE.

After loading the data, encoding the categorical variables, and splitting the data into features and target matrices. We then created the random forest classifier and then created the RFE object, and then computed the cross-validation score.



Then we plotted the number of features versus the cross-validation scores. Here is the plot we got:



We also printed out the optimal number of features and this is the result:

```
Optimal number of features : 1
Index(['SPIN_T'], dtype='object')
```

7. Question 6: Ethical Issues

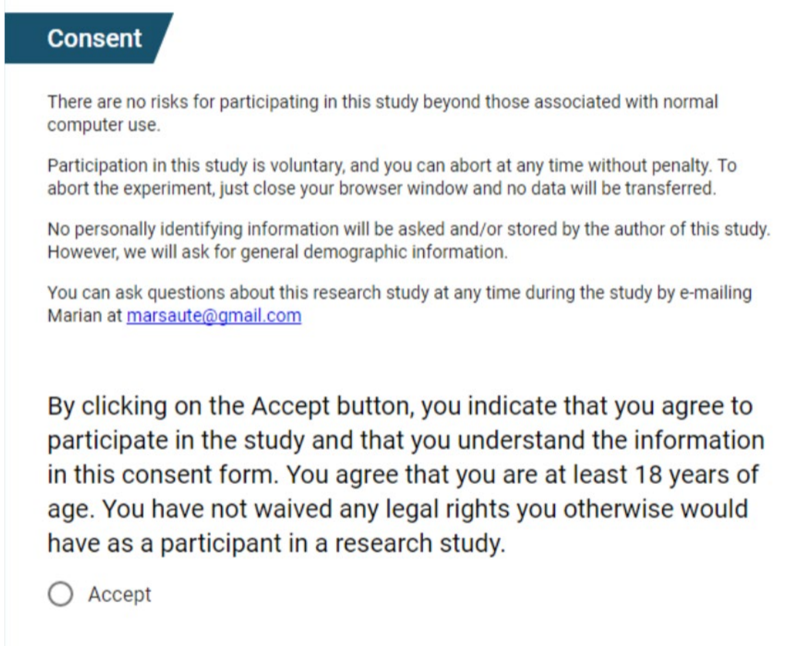
Our dataset originates from a comprehensive survey conducted in 2017, involving more than 13,000 participants. This makes it the largest dataset of its kind, openly accessible and bridging the gap between gaming habits, a range of socio-economic factors, and various psychological measures like anxiety, social phobia, life satisfaction, and narcissism. To understand the ethical considerations linked to this dataset, we looked into the methodology of its data collection and the initial data processing steps taken by the creators of the dataset.

How Data Was Collected?

The data was collected via a survey that was posted on different platforms such as Reddit, CrowdFlower, and TeamLiquid.net. The survey was aimed at the people who play video games, and it has important information about how the data will be used, what kind of data will be collected etc. at the beginning of the survey.

Permission Requested:

Informed Consent: The survey commenced with an informed consent query, a fundamental requirement in ethical research practices. This ensures that participants are thoroughly informed about the study's objectives, potential consequences, and how their data will be handled. The attached screenshot details the consent form presented to survey participants, confirming that the data was gathered with their explicit consent.


Consent

There are no risks for participating in this study beyond those associated with normal computer use.

Participation in this study is voluntary, and you can abort at any time without penalty. To abort the experiment, just close your browser window and no data will be transferred.

No personally identifying information will be asked and/or stored by the author of this study. However, we will ask for general demographic information.

You can ask questions about this research study at any time during the study by e-mailing Marian at marsaute@gmail.com

By clicking on the Accept button, you indicate that you agree to participate in the study and that you understand the information in this consent form. You agree that you are at least 18 years of age. You have not waived any legal rights you otherwise would have as a participant in a research study.

Accept

Withdrawal Rights: Participants have been informed of their right to withdraw from the study at any point without any consequences as you can see from the above screenshot.

Privacy (Pseudonym Technique):

Anonymization: Maintaining the privacy of participants is a top priority, and it's essential that the dataset is free of any personally identifiable information (PII). The use of pseudonyms or the anonymization of data is crucial for safeguarding participant identities. From the screenshot, it's evident that the survey didn't request direct PII such as names or dates of birth. However, it did gather details like age, gender, and countries of birth and residence. These were collected to understand the demographic profiles of gaming enthusiasts.
Data Security: The absence of personally identifiable information (PII) in our dataset reduces concerns about confidentiality. Based on our assessment, this data is publicly accessible and intended for research use.

Representativeness of the Data:

Demographic Diversity: With participants from 109 countries, the dataset appears globally diverse. However, the majority are from the USA, Germany, the UK, and Canada, which might skew the results towards these populations.
Gender Imbalance: The significant gender imbalance (12699 males, 713 females, 52 other) in the dataset could lead to biased conclusions, especially in aspects related to gender-specific experiences or perspectives in gaming.
So for most of our project throughout the semester, we did not consider such variables.

How were the Variables computed?

Selection of Variables:

In terms of relevance and sensitivity, the selection of variables such as anxiety, social phobia, and narcissism is a delicate matter. It is essential to provide a rationale for choosing these specific variables, especially in relation to the aims of the research. However, there was no explicit mention in the data collection process about why these particular measures were selected by the authors for the study.

Data Variable Manipulation:

The data from the survey was converted into a csv by recording the survey answers into different columns and some basic preprocessing to convert the survey answers to csv. And we used that csv for our project. The authors didn’t remove any of the data that was collected. They just organized the data into different columns for future analysis. Since the data is survey data and the survey had a few text fields the participants gave a lot of information that is not related to the question asked and we performed cleaning of this in our milestone 1. We tried to make sure we didn’t remove any of the data but rather just cleaned it. During the milestone 1 stage of cleaning, we spoke about how we handled the outliers and missing data clearly and you can find it in the milestone 1 report.

Data Cleaning and Preprocessing:

Handling Missing or Inaccurate Data:

Our project involved transforming the survey responses into a CSV format, which included some basic preprocessing which was done by the authors themselves. They retained all the collected data, simply organizing it into various columns for subsequent analysis.

In areas where survey responses were textual, some participants provided information irrelevant to the questions asked. In our initial milestone, we addressed this by cleaning the data, ensuring that no valuable information was discarded. Detailed information about how we managed outliers and missing data can be found in our Milestone 1 report.

Normalization and Standardization:

Although we did not explicitly use normalization or standardization techniques in our initial data preparation, we did normalization and standardization for individual models wherever required, to ensure data comparability and to prevent biases, especially when dealing with varied scales of measurement.

Data Transformation:

Aggregation and Segmentation:

In organizing the data, we aggregated or segmented it based on factors like country and age. This step was crucial for our analysis, allowing us to examine patterns and trends across different demographic groups. The decision on how to segment the data was driven by the need to derive meaningful insights while maintaining the integrity of the original responses.

Categorization of Open-Ended Responses:

Categorizing open-ended responses from the survey was a delicate task. We aimed to maintain transparency and avoid bias in this process. Our approach was to systematically categorize these responses in a way that accurately reflects the participants' input while aligning with the research objectives. This step was key in making unstructured data more analyzable and insightful.

8. Contributions

SVMS (code, report, and PPT) (100%) - Pranav A
Neural networks (code, report, and PPT) (100%) - Rajeevan M
Clustering (code, report, and PPT) (100%) - Asmita S
Feature Selection (code, report, and PPT) (100%) - Rajeevan M
Comparative Analysis (code, report, and PPT) - Asmita S
Extra Credit (Report) - Rajeevan M
Extra Credit (PPT) - Asmita S
PPT and Report Outline - Pranav A
Report Cleaning and Formatting - Pranav A
Presentation Collation: Pranav A

-THE END-