**Project Report - Milestone 2**

**INST 737 - Introduction to Data Science**

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

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

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

# 1. Milestone 1 Recap:

Milestone 1 of our project (**'MindGame Insights'**) involved finding our target dataset from OSF Gaming Habits and Psychological Well-Being Dataset (OSF, n.d.) ,which is a publicly available dataset accessible through the **Open Science Framework (OSF)** platform. This dataset was collected through a survey taken by over **13,464 participants** from around **109 countries** in the world.

We also conducted **Preliminary Data Cleaning/Preparation** of our data followed by **Exploratory Data Analysis**. We have picked Python as our scripting language written in the Jupyter Notebook. Also a lot of variables had random values in our dataset which we had to manually clean and we addressed most of it in this Milestone but we still had a portion which we decided to clean in Milestone 2.

# 2. Finalized Research Question:

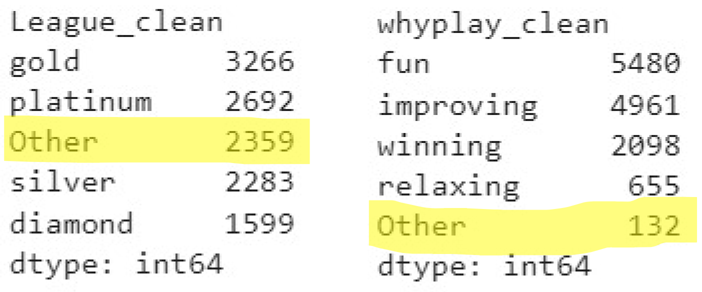
Based on the feedback that we got from the TA and Professor.Vanessa on our Research Question, we have further refined our research question, transitioning from a broad, generalized question to a more detailed and specific one.

*"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?"*

# 3. Milestone 2 Data Preparation Efforts:

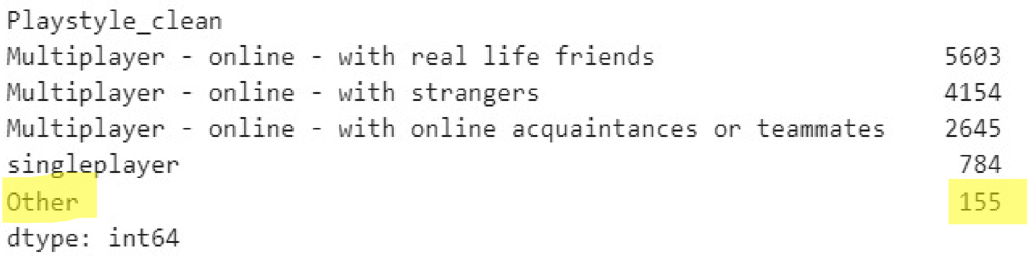
As mentioned earlier, the data which hasn't been cleaned in Milestone 1 has been labeled in the "***Others category"*** for all the variables and this has been further cleaned in Milestone 2.

Milestone 1 Cleaning results:



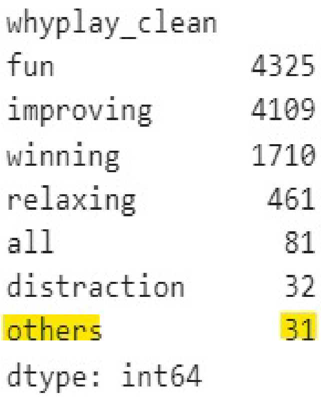
```
League_clean
gold          3266
platinum      2692
Other         2359
silver        2283
diamond       1599
dtype: int64
```

```
whyplay_clean
fun          5480
improving    4961
winning      2098
relaxing      655
Other         132
dtype: int64
```

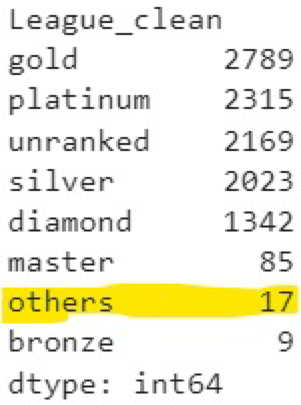


```
Playstyle_clean
Multiplayer - online - with real life friends                           5603
Multiplayer - online - with strangers                                   4154
Multiplayer - online - with online acquaintances or teammates           2645
singleplayer                                                             784
Other                                                                    155
dtype: int64
```

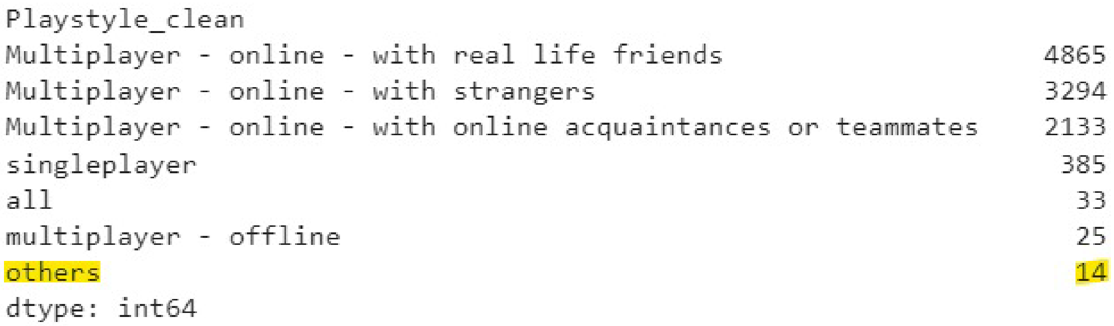
Milestone 2 Cleaning results:



```
whyplay_clean
fun            4325
improving      4109
winning        1710
relaxing        461
all              81
distraction      32
others           31
dtype: int64
```



```
League_clean
gold        2789
platinum    2315
unranked    2169
silver      2023
diamond     1342
master        85
others        17
bronze         9
dtype: int64
```



```
Playstyle_clean
Multiplayer - online - with real life friends                    4865
Multiplayer - online - with strangers                            3294
Multiplayer - online - with online acquaintances or teammates    2133
singleplayer                                                      385
all                                                                33
multiplayer - offline                                              25
others                                                             14
dtype: int64
```

As shown in the above images, we removed the ***"Others"*** category from every variable and further cleaned other columns similarly and created a cleaned dataset which we would be utilizing for Milestone 2 code.

**Link to cleaned dataset:**

***https://drive.google.com/file/d/12j-TYs8fR4cu9z9sghUmK9nTAbdJN7T5/view?usp=sharing***

# 4. Question 1:

## A. Linear Regression:

### a. Dependent & Independent Variables:

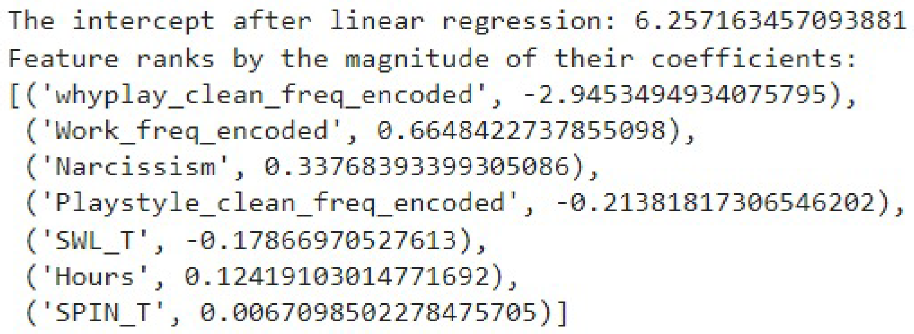
**The Dependent Variables:** Generalized Anxiety Disorder ***(GAD_T)***

**The Independent Variables:** "Narcissism", "Social Phobia Inventory ***(SPIN_T)***", "Hours", "Satisfaction With Life ***(SWL_T)***", "Reasons to Play ***(whyplay_clean)***", "Status of Work ***(Work)***", "Style of Play ***(Playstyle_clean)***"

## b. Data Preparation Efforts Prior to Linear Regression:

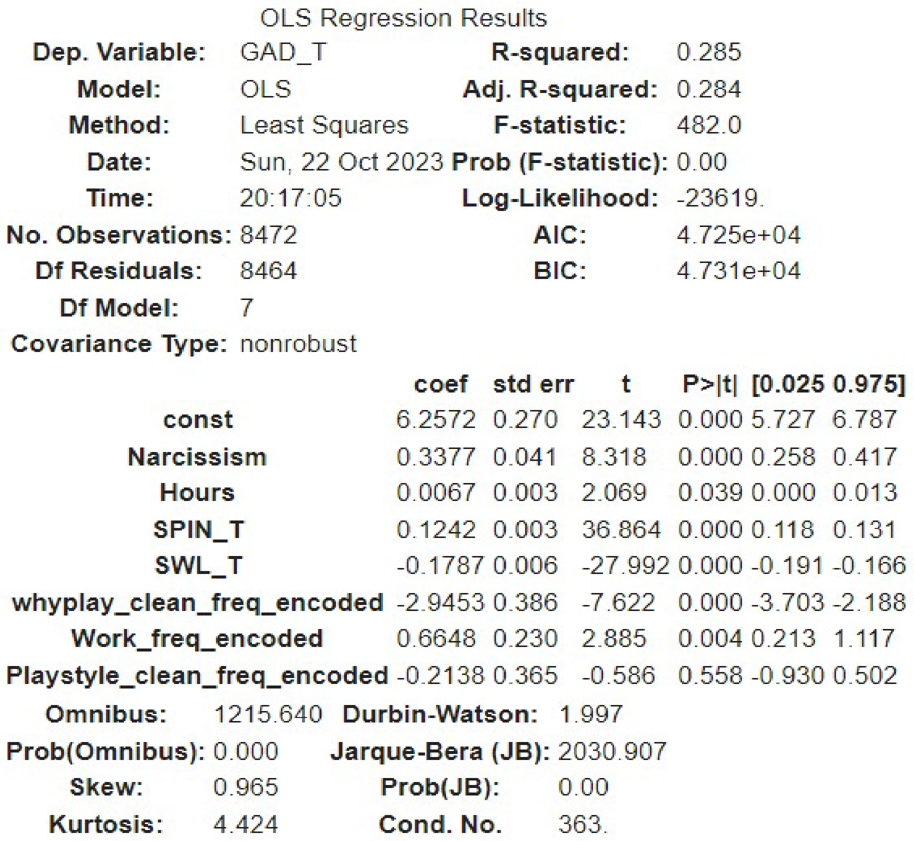
In preparation for our Linear Regression, we transformed our Categorical Variables using ***"Frequency Encoding"*** and updated the column names accordingly.

## c. Intercept & Coefficient:



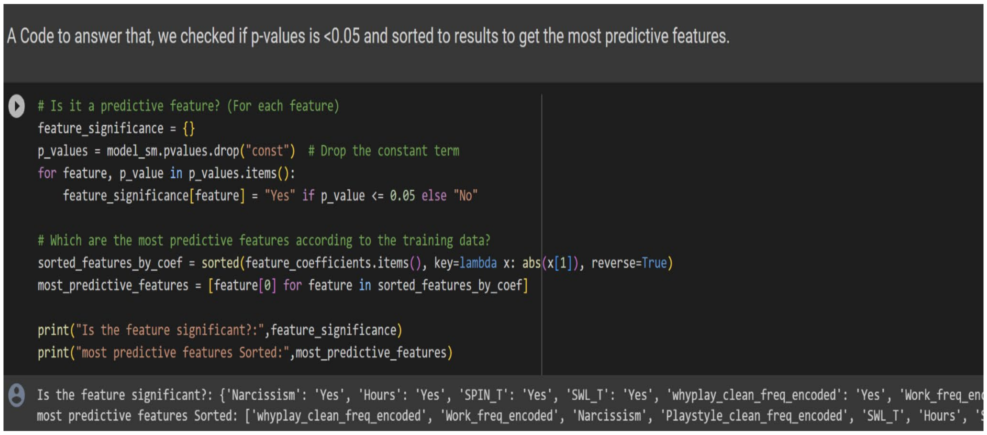
```
The intercept after linear regression: 6.257163457093881
Feature ranks by the magnitude of their coefficients:
[('whyplay_clean_freq_encoded', -2.9453494934075795),
 ('Work_freq_encoded', 0.6648422737855098),
 ('Narcissism', 0.33768393399305086),
 ('Playstyle_clean_freq_encoded', -0.21381817306546202),
 ('SWL_T', -0.17866970527613),
 ('Hours', 0.12419103014771692),
 ('SPIN_T', 0.0067098502278475705)]
```

## d. Predictive feature:



```
                              OLS Regression Results
     Dep. Variable:  GAD_T                  R-squared:        0.285
             Model:  OLS                    Adj. R-squared:   0.284
            Method:  Least Squares          F-statistic:      482.0
              Date:  Sun, 22 Oct 2023 Prob (F-statistic):    0.00
              Time:  20:17:05               Log-Likelihood:   -23619.
 No. Observations:   8472                   AIC:              4.725e+04
      Df Residuals:  8464                   BIC:              4.731e+04
          Df Model:  7
  Covariance Type:   nonrobust
                                    coef    std err  t        P>|t|  [0.025  0.975]
                      const         6.2572  0.270    23.143   0.000  5.727   6.787
                 Narcissism         0.3377  0.041    8.318    0.000  0.258   0.417
                      Hours         0.0067  0.003    2.069    0.039  0.000   0.013
                     SPIN_T         0.1242  0.003    36.864   0.000  0.118   0.131
                      SWL_T         -0.1787 0.006    -27.992  0.000  -0.191  -0.166
 whyplay_clean_freq_encoded         -2.9453 0.386    -7.622   0.000  -3.703  -2.188
          Work_freq_encoded         0.6648  0.230    2.885    0.004  0.213   1.117
Playstyle_clean_freq_encoded        -0.2138 0.365    -0.586   0.558  -0.930  0.502
       Omnibus:     1215.640  Durbin-Watson:     1.997
 Prob(Omnibus):     0.000     Jarque-Bera (JB):  2030.907
          Skew:     0.965     Prob(JB):          0.00
      Kurtosis:     4.424     Cond. No.          363.
```

## e. Most predictive features (According to Training Data):



A Code to answer that, we checked if p-values is <0.05 and sorted to results to get the most predictive features.

```
# Is it a predictive feature? (For each feature)
feature_significance = {}
p_values = model_sm.pvalues.drop("const")  # Drop the constant term
for feature, p_value in p_values.items():
    feature_significance[feature] = "Yes" if p_value <= 0.05 else "No"

# Which are the most predictive features according to the training data?
sorted_features_by_coef = sorted(feature_coefficients.items(), key=lambda x: abs(x[1]), reverse=True)
most_predictive_features = [feature[0] for feature in sorted_features_by_coef]

print("Is the feature significant?:",feature_significance)
print("most predictive features Sorted:",most_predictive_features)
```

```
Is the feature significant?: {'Narcissism': 'Yes', 'Hours': 'Yes', 'SPIN_T': 'Yes', 'SWL_T': 'Yes', 'whyplay_clean_freq_encoded': 'Yes', 'Work_freq_enc
most predictive features Sorted: ['whyplay_clean_freq_encoded', 'Work_freq_encoded', 'Narcissism', 'Playstyle_clean_freq_encoded', 'SWL_T', 'Hours', 'S
```

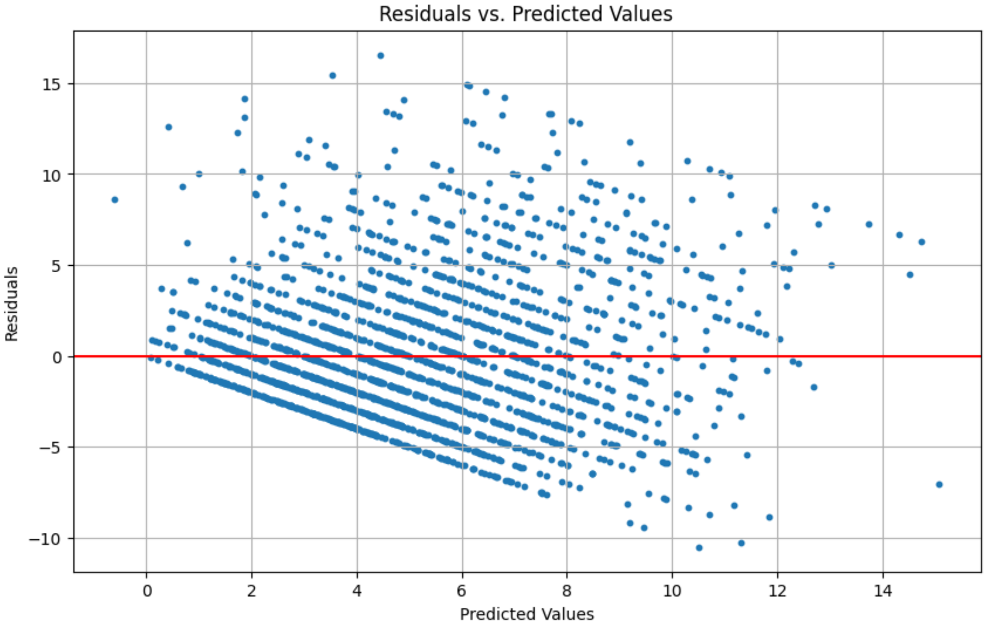
<u>Output</u>

**Is the feature significant?:** {'Narcissism': 'Yes', 'Hours': 'Yes', 'SPIN_T': 'Yes', 'SWL_T': 'Yes', 'whyplay_clean_freq_encoded': 'Yes', 'Work_freq_encoded': 'Yes', 'Playstyle_clean_freq_encoded': 'No'}

**Most Predictive Features Sorted:** ['whyplay_clean_freq_encoded', 'Work_freq_encoded', 'Narcissism', 'Playstyle_clean_freq_encoded', 'SWL_T', 'Hours', 'SPIN_T']

## f. Residuals:

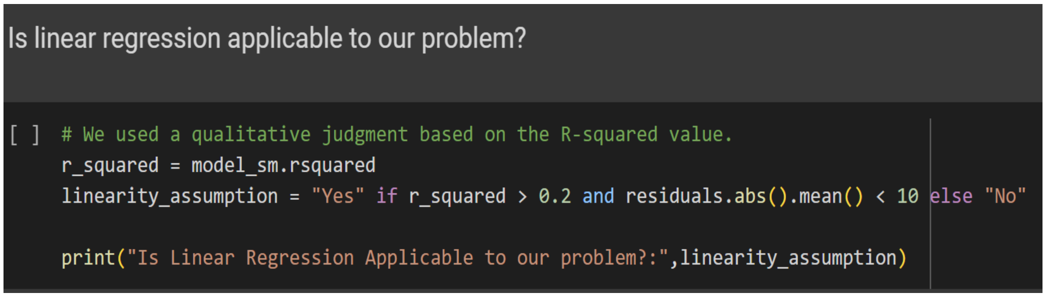
We calculated the Residuals by subtracting the prediction values from the actual values and plotted the below mentioned graphs.



## g. Significance of Linear Regression to our Research Questions:

We calculated it using ***"R-Squared Values"*** and ***"Residual Mean".***

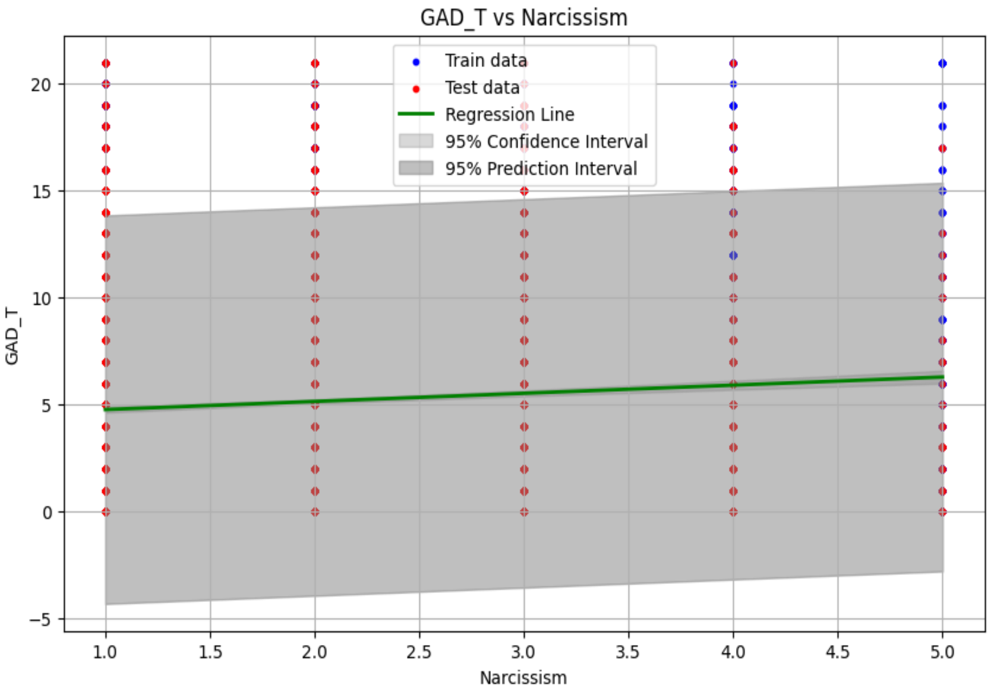
The answer to this is in Table 1 at the bottom of linear regression analysis**.**

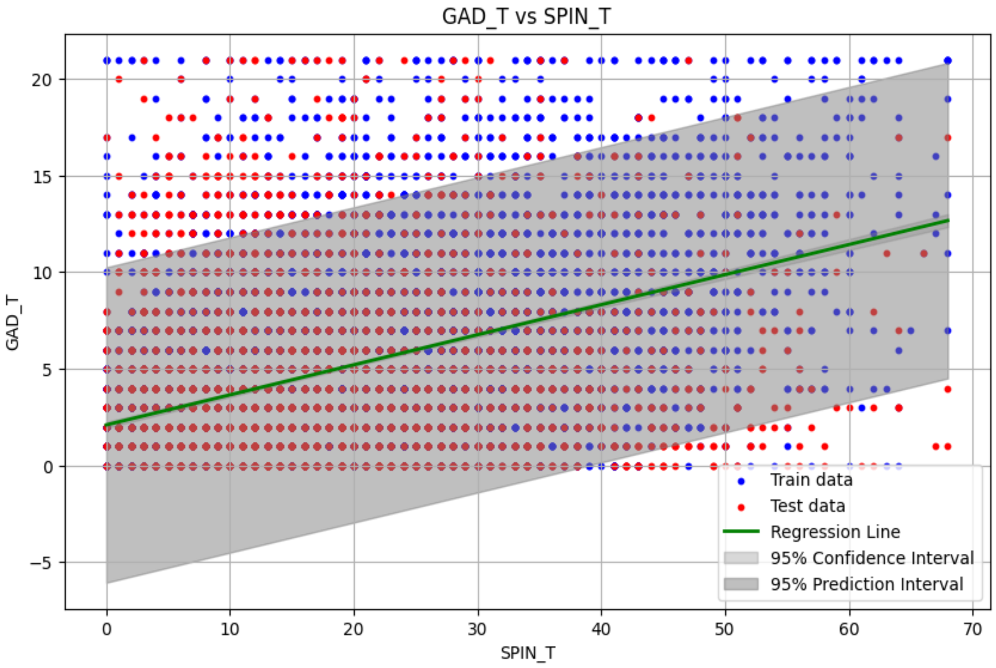


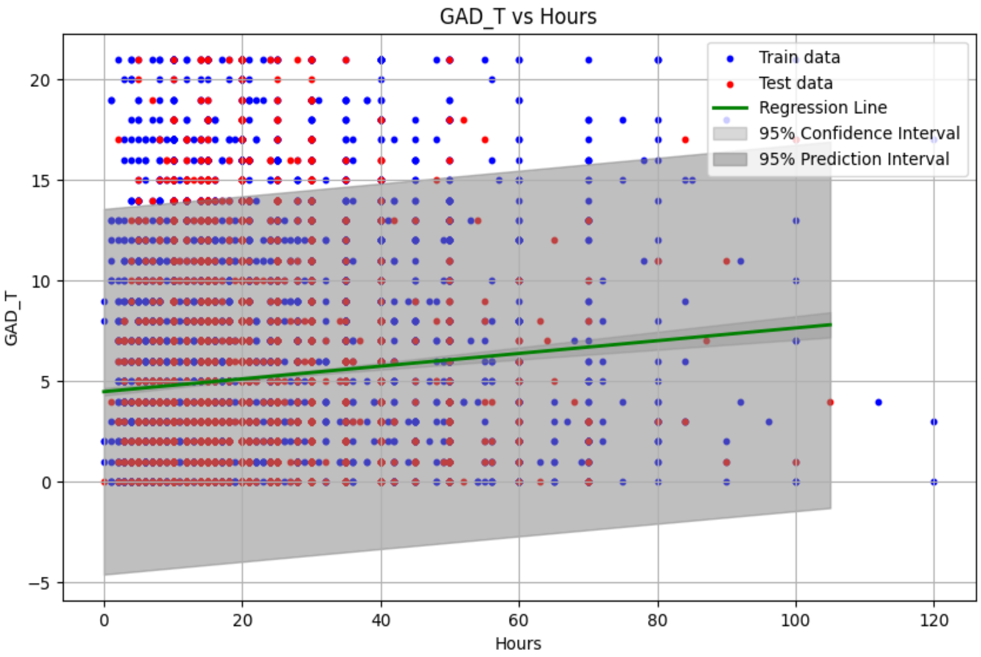
Is linear regression applicable to our problem?

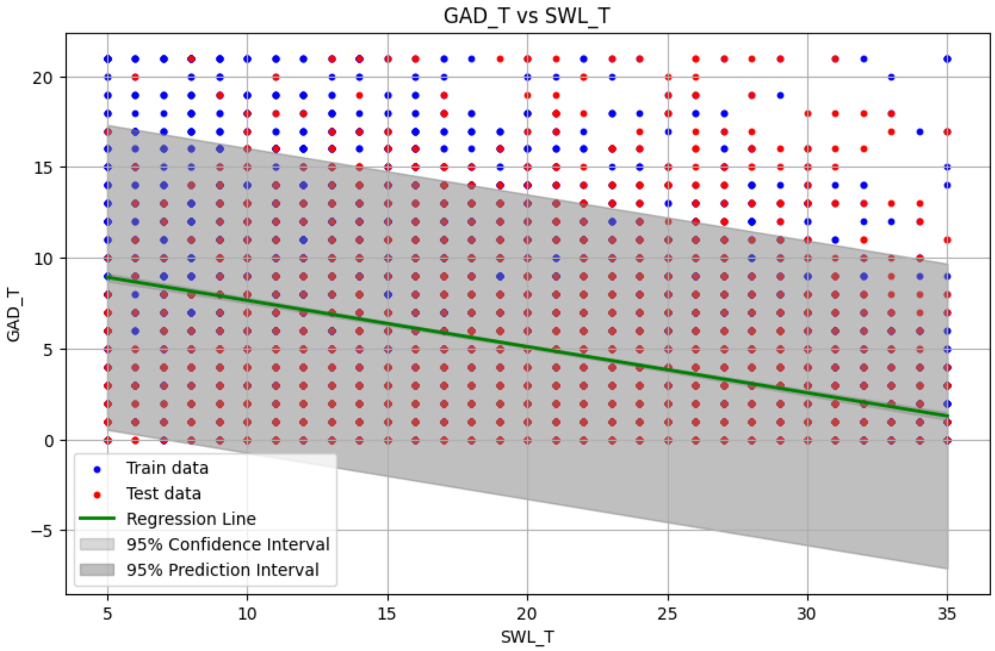
```
# We used a qualitative judgment based on the R-squared value.
r_squared = model_sm.rsquared
linearity_assumption = "Yes" if r_squared > 0.2 and residuals.abs().mean() < 10 else "No"

print("Is Linear Regression Applicable to our problem?:",linearity_assumption)
```

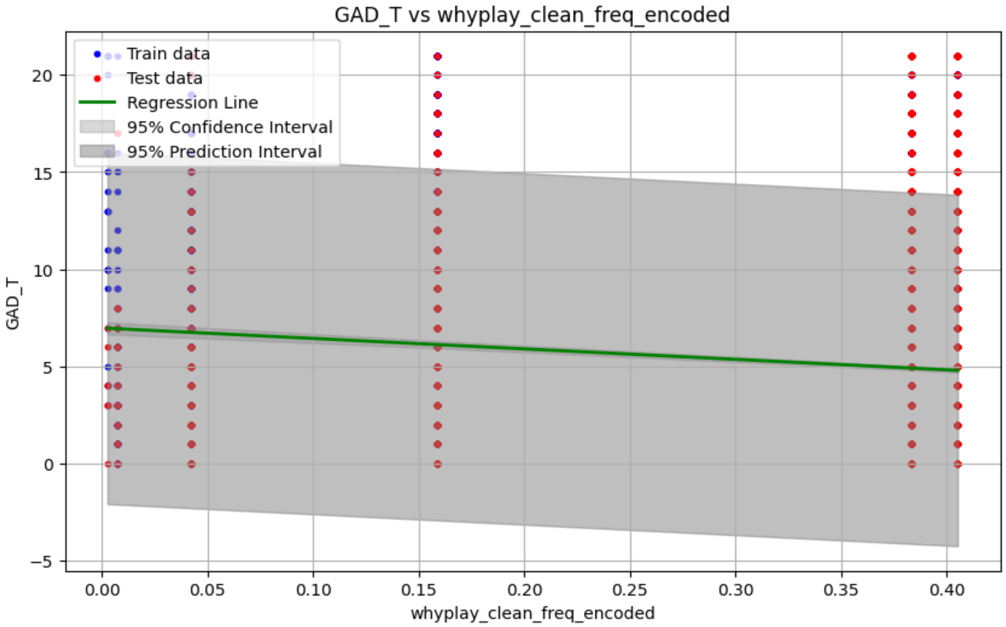
## h. Confidence & Prediction Bands (Using Trained Model against Test):

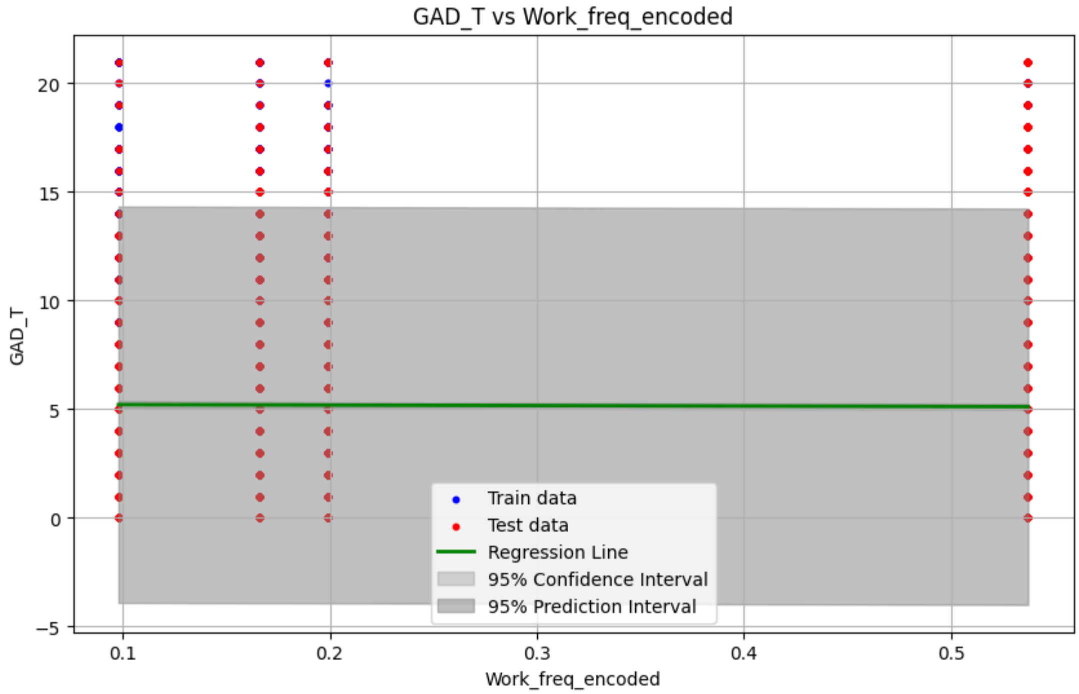


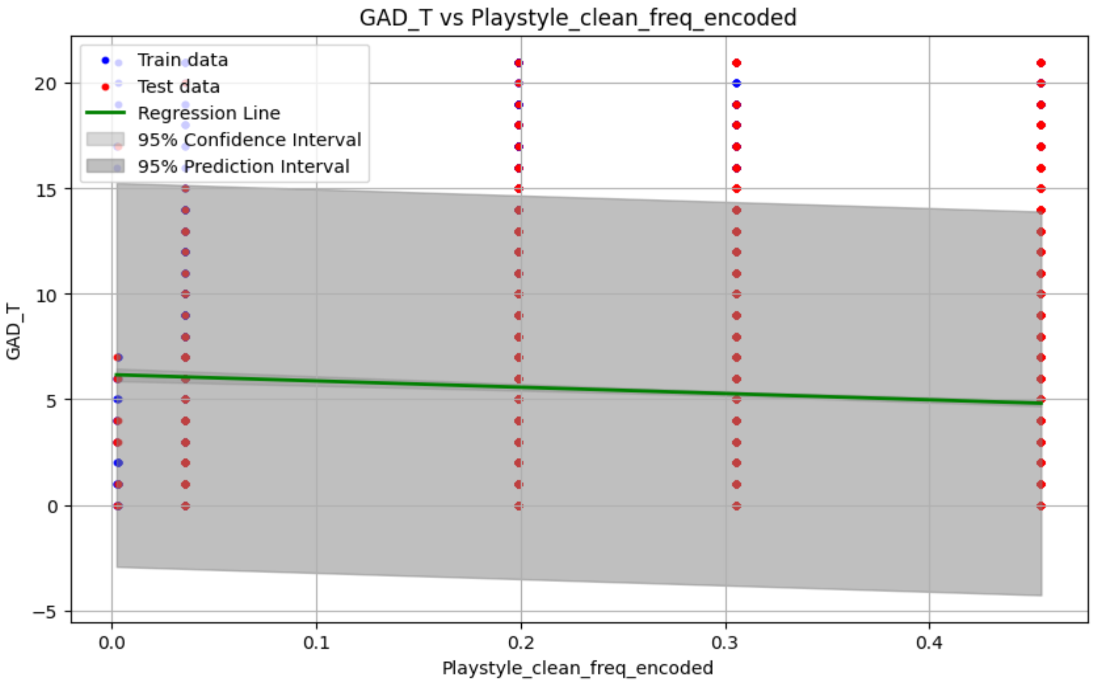

GAD_T vs SPIN_T
GAD_T
SPIN_T
−5
0
5
10
15
20
0
10
20
30
40
50
60
70
Train data
Test data
Regression Line
95% Confidence Interval
95% Prediction Interval


GAD_T vs Hours
GAD_T
Hours
−5
0
5
10
15
20
0
20
40
60
80
100
120
Train data
Test data
Regression Line
95% Confidence Interval
95% Prediction Interval


GAD_T vs SWL_T
GAD_T
SWL_T
20
15
10
5
0
−5
5
10
15
20
25
30
35
Train data
Test data
Regression Line
95% Confidence Interval
95% Prediction Interval


GAD_T vs whyplay_clean_freq_encoded
GAD_T
whyplay_clean_freq_encoded
20
15
10
5
0
−5
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Train data
Test data
Regression Line
95% Confidence Interval
95% Prediction Interval


GAD_T vs Work_freq_encoded


GAD_T vs Playstyle_clean_freq_encoded

### i. Prediction Accuracy using: Correlation between Predicted vs. Real values & Mean Square Error between the two:

| Feature Name | Correlation | MSE | R2 | Linear Regression Applicable? |
|---|---|---|---|---|
| 'Narcissism' | 0.0328252883418951 | 22.548577019344748 | -0.001961405181236664 | No |
| 'SPIN_T' | 0.4823824036824145 | 17.3087032239395 | 0.2308759621840034 | Yes |
| 'Hours' | 0.09612500873633444 | 22.307360590362098 | 0.008757211427204203 | No |
| 'SWL_T' | 0.4050639387511801 | 18.827819677657327 | 0.1633729860407539 | No |
| 'whyplay_clean_freq_encoded' | 0.11865233399365668 | 22.197061706731258 | 0.013658417580384663 | No |
| 'Work_freq_encoded' | 0.06189679913985132 | 22.490471348833033 | 0.0006205599345651125 | No |
| 'Playstyle_clean_freq_encoded' | 0.047618796779909434 | 22.47923184471596 | 0.0011199949689364441 | No |

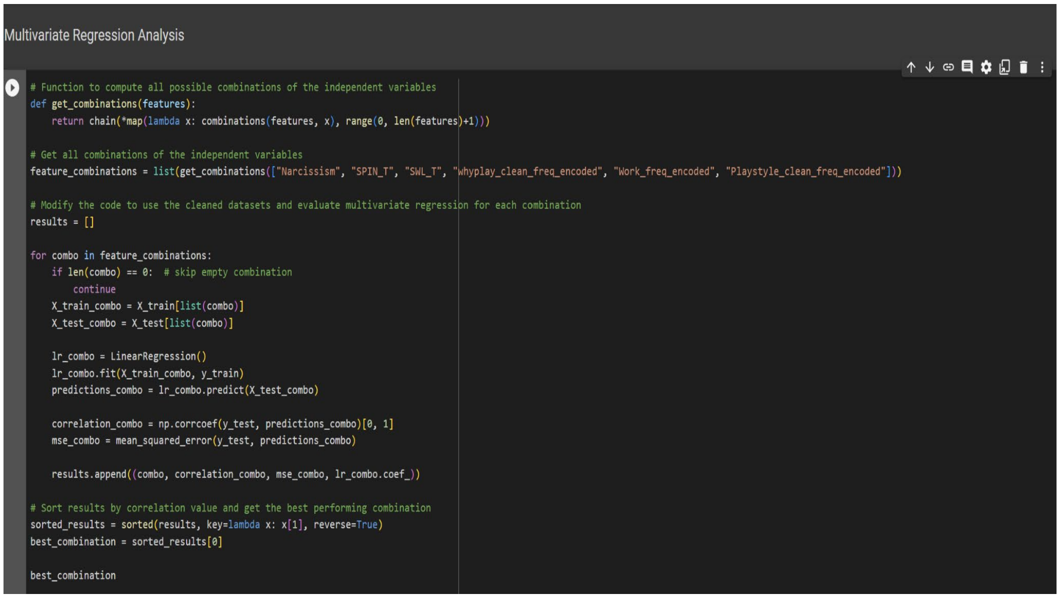
***Table 1: Linear Regression results for each independent Variable***

### j. Reporting the Prediction Accuracy:

From the above table, about **23.09%**(R2) of the variance of GAD_T(dependent variable) can be explained by SPIN_T (Independent variable). Similarly, about **0.88%** by Hours, **16.34%** by SWL_T, **1.37%** by whyplay_clean, **0.11%** by playstyle_clean etc.

## B. Multivariate Regression:

### a. Significance of Multivariate Regression in improving the Prediction Results:



Multivariate Regression Analysis

```python
# Function to compute all possible combinations of the independent variables
def get_combinations(features):
    return chain(*map(lambda x: combinations(features, x), range(0, len(features)+1)))

# Get all combinations of the independent variables
feature_combinations = list(get_combinations(["Narcissism", "SPIN_T", "SWL_T", "whyplay_clean_freq_encoded", "Work_freq_encoded", "Playstyle_clean_freq_encoded"]))

# Modify the code to use the cleaned datasets and evaluate multivariate regression for each combination
results = []

for combo in feature_combinations:
    if len(combo) == 0:  # skip empty combination
        continue
    X_train_combo = X_train[list(combo)]
    X_test_combo = X_test[list(combo)]

    lr_combo = LinearRegression()
    lr_combo.fit(X_train_combo, y_train)
    predictions_combo = lr_combo.predict(X_test_combo)

    correlation_combo = np.corrcoef(y_test, predictions_combo)[0, 1]
    mse_combo = mean_squared_error(y_test, predictions_combo)

    results.append((combo, correlation_combo, mse_combo, lr_combo.coef_))

# Sort results by correlation value and get the best performing combination
sorted_results = sorted(results, key=lambda x: x[1], reverse=True)
best_combination = sorted_results[0]

best_combination
```

Output:

**Best Combination Features: (('SPIN_T', 'SWL_T', 'whyplay_clean_freq_encoded')**

### b. Most predictive features (According to Training Data) & Coefficient for each feature in the best combination:

Most Predictive feature Ranking:

[('whyplay_clean_freq_encoded', -3.2539532000859888),

('SWL_T', -0.1789557702324269),

('SPIN_T', 0.12446524210259205)]

### c. Prediction Accuracy using: Correlation between Predicted vs. Real values & the mean square error between the two:

Best Combination Correlation: 0.5491065136412202

Best Combination MSE: 15.732124972082087

R2 value: 0.3009362043281292

Which indicates that the model explains about **30.09%** of the variance in the dependent variable(GAD_T) which is better than all the individual linear regression runs for each feature.

## C. Regularization:

### a. Results from Regularization:

For Regularization, we conducted Lasso & Ridge Regression and printed the regularization (Correlation , MSE & R2 respectively) results for univariate and multivariate regressions.

- **Regularization results for univariate:**
  'whyplay_clean_freq_encoded':
  - 'Ridge': (0.11865233399365668, 22.19663226968811, 0.013677499908410162),
  - 'Lasso': (1.4419670651941595e-16, 22.51248444486643, -0.0003576070076367621)
    'SPIN_T':
  - 'Ridge': (0.4823824036824144, 17.30870377478805, 0.23087593770667225),
  - 'Lasso': (0.48238240368241436, 17.344407194559363, 0.22928943188800566)
    'SWL_T':
  - 'Ridge': (0.40506393875118035, 18.82782034237062, 0.16337295650376504),
  - 'Lasso': (0.40506393875118035, 18.869840607218475, 0.16150575736401795)

- **Regularization results for Multivariate:**
  - 'Ridge': (0.5477978000626571, 15.760562488743263, 0.29966865207856996),
  - 'Lasso': (0.5462940040356477, 15.850063871042298, 0.2956915970876023)}

From the above report and previous reports, regularization didn’t have a lot of change for our prediction accuracy scores both for univariate and multivariate. For example, for **univariate before regularization** prediction accuracy for **SPIN_T** was **23.08%**, after regularization **with ridge**, the prediction accuracy is **23.09%** and **with Lasso** it is **22.92%**. Similarly for the rest of the variables it almost resembles very similar. And for **multivariate without regularization**, the prediction accuracy was **30.09%** and after **ridge** it was **29.97%** and with **lasso** it is **29.57%**.

## D. Repeating (A-C steps) for Random Test & Train Data:

### a. Results for 10 Random Runs:

It has values for correlation, MSE and r2 in each row.

**RUN 1:**

```
[{'Univariate': {'whyplay_clean_freq_encoded': (0.12409759792368609,

   21.949999159340322,

   0.015395936752964912),

  'SPIN_T': (0.44954933113754564, 17.78921051842815, 0.20203509661876218),

  'SWL_T': (0.3952037889202006, 18.81634212848481, 0.15596138384046743)},

 'Multivariate': (0.5414600617288107,
  15.776921073487037,
  0.29229971802194976),
 'Regularization': {'Ridge': (0.5414553821018578,
   15.777121681895142,
   0.29229071939501095),
  'Lasso': (0.5306907364620455, 16.08124995755724, 0.2786485350016784)}},
```

**RUN 2:**

```
 {'Univariate': {'whyplay_clean_freq_encoded': (0.09103303137422109,
   21.594690621127725,
   0.006418019732053026),
  'SPIN_T': (0.44955496727089883, 17.345963202312213, 0.20190398785586894),
  'SWL_T': (0.40041950808411275, 18.252473754565944, 0.16019500644722406)},
 'Multivariate': (0.5293586685496274, 15.645082297696861, 0.2801623267715576),
 'Regularization': {'Ridge': (0.5294229641452136,
   15.643598960093701,
   0.2802305758397666),
  'Lasso': (0.52853555765636, 15.690739169790282, 0.278061632384066)}},
```

**RUN 3:**

```
 {'Univariate': {'whyplay_clean_freq_encoded': (0.13009038757334504,
   20.852172769909398,
   0.015827727865422525),
  'SPIN_T': (0.45523699285393737, 16.801303245577845, 0.20701900121009598),
  'SWL_T': (0.38341560912101985, 18.09713492450261, 0.14585887071320613)},
 'Multivariate': (0.5341677462874562, 15.15555349388235, 0.28469442095476494),
 'Regularization': {'Ridge': (0.5341868487223487,
   15.155003175331805,
   0.284720394663321),
  'Lasso': (0.5225394505739197, 15.41356107900274, 0.27251708509256l)}},
```

**RUN 4:**

```
{'Univariate': {'whyplay_clean_freq_encoded': (0.1303314158204915,
   22.581692270620714,
   0.015448331251123992),
  'SPIN_T': (0.45693808818543513, 18.187057134187064, 0.20705245486002577),
  'SWL_T': (0.4174864196046097, 19.009997226520852, 0.17117263542586814)},
 'Multivariate': (0.5505157811711361,
  16.032779540411987,
  0.30097799305624995),
 'Regularization': {'Ridge': (0.5505200838476254,
   16.03282659562699,
   0.30097594146995166),
  'Lasso': (0.5397063246634192, 16.36014750691499, 0.2867048963435297)}},
```

**RUN 5:**

```
{'Univariate': {'whyplay_clean_freq_encoded': (0.1321355629931915,
   20.554403268736916,
   0.01682030854387473),
  'SPIN_T': (0.4247705668651109, 17.180158876817284, 0.17822069155528408),
  'SWL_T': (0.39170275532443116, 17.70314641184454, 0.1532046053862648)},
 'Multivariate': (0.5196247672065992, 15.281381340558704, 0.2690450024260347),
 'Regularization': {'Ridge': (0.519597515528005,
   15.281949710620976,
   0.26901781555540893),
  'Lasso': (0.5040688993332683, 15.597205155341118, 0.25393818775900145)}},
```

**RUN 6:**

```
 {'Univariate': {'whyplay_clean_freq_encoded': (0.1319520202977255,
   21.71164150688726,
   0.01734852924568253),
  'SPIN_T': (0.46612646307657907, 17.30339345396777, 0.2168622984499634),
  'SWL_T': (0.4037448026945115, 18.50360812176372, 0.16254154577189117)},
 'Multivariate': (0.5493119967348892,
  15.441584133020665,
  0.30112629419217507),
 'Regularization': {'Ridge': (0.5493229932465644,
   15.441427711489336,
   0.3011333737051529),
  'Lasso': (0.538866886303278, 15.731300230692954, 0.28801397611209945)}},
```

**RUN 7:**

```
 {'Univariate': {'whyplay_clean_freq_encoded': (0.1504428393552433,
   21.18519947475908,
   0.02132452318771383),
  'SPIN_T': (0.4267547298705094, 17.73305603112159, 0.18080039381860624),
  'SWL_T': (0.4046580366961039, 18.106326063724406, 0.16355674088191585)},
 'Multivariate': (0.5297148362501386,
  15.578056079799383,
  0.28035317864857767),
 'Regularization': {'Ridge': (0.5296982558580489,
   15.578406629189612,
   0.28033698460272805),
  'Lasso': (0.51353181522857, 15.948335608906795, 0.26324767557626605)}},
```

**RUN 8:**

```
 {'Univariate': {'whyplay_clean_freq_encoded': (0.126399655562209,
   21.976382411826023,
   0.01594391596901945),
  'SPIN_T': (0.4773138289098614, 17.27518544708523, 0.22645360717696206),
  'SWL_T': (0.43395711327623554, 18.199776698137395, 0.1850523597473538)},
 'Multivariate': (0.5611695512174788,
  15.326109999446771,
  0.31372909758942913),
 'Regularization': {'Ridge': (0.5612062072330207,
   15.325357852610692,
   0.31376277713942324),
  'Lasso': (0.5559067620971557, 15.526249677663856, 0.3047672646400924)}},
```

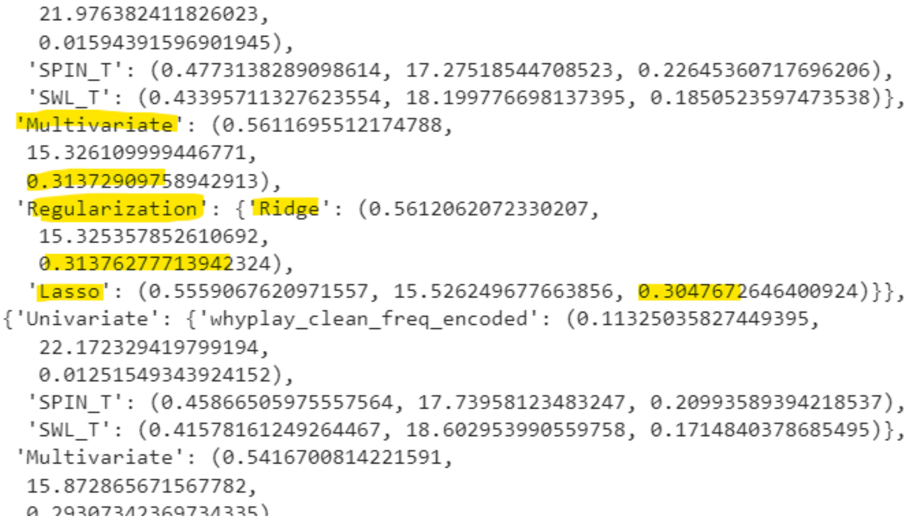
**RUN 9:**

```
 {'Univariate': {'whyplay_clean_freq_encoded': (0.11325035827449395,
   22.172329419799194,
   0.01251549343924152),
  'SPIN_T': (0.45866505975557564, 17.73958123483247, 0.20993589394218537),
  'SWL_T': (0.41578161249264467, 18.602953990559758, 0.1714840378685495)},
 'Multivariate': (0.5416700814221591,
  15.872865671567782,
  0.29307342369734335),
 'Regularization': {'Ridge': (0.5416774507643545,
   15.872750040972294,
   0.2930785735135565),
  'Lasso': (0.5348769251657716, 16.077294409977682, 0.28396882273037083)}},
```

**RUN 10:**

```
{'Univariate': {'whyplay_clean_freq_encoded': (0.16470290919524624,
   20.860904099258804,
   0.025140250531896724),
  'SPIN_T': (0.4780379923528406, 16.510043718462676, 0.2284621507051644),
  'SWL_T': (0.38706622000322255, 18.200384365210166, 0.14947012564415296)},
 'Multivariate': (0.5488469345141662, 14.957646460553752, 0.3010078848075437),
 'Regularization': {'Ridge': (0.5488276846152698,
   14.958045424132505,
   0.3009892406714805),
  'Lasso': (0.5360498330424274, 15.254800576728888, 0.28712144988269306)}}]
```

## b. Observations:



```
   21.976382411826023,
   0.01594391596901945),
  'SPIN_T': (0.4773138289098614, 17.27518544708523, 0.22645360717696206),
  'SWL_T': (0.43395711327623554, 18.199776698137395, 0.1850523597473538)},
 'Multivariate': (0.5611695512174788,
  15.326109999446771,
  0.31372909758942913),
 'Regularization': {'Ridge': (0.5612062072330207,
   15.325357852610692,
   0.3137627771394232 4),
  'Lasso': (0.5559067620971557, 15.526249677663856, 0.3047672646400924)}},
{'Univariate': {'whyplay_clean_freq_encoded': (0.11325035827449395,
   22.17232941979919 4,
   0.01251549343924152),
  'SPIN_T': (0.45866505975557564, 17.73958123483247, 0.20993589394218537),
  'SWL_T': (0.41578161249264467, 18.602953990559758, 0.1714840378685495)},
 'Multivariate': (0.5416700814221591,
  15.872865671567782,
  0.29307342369734335)
```

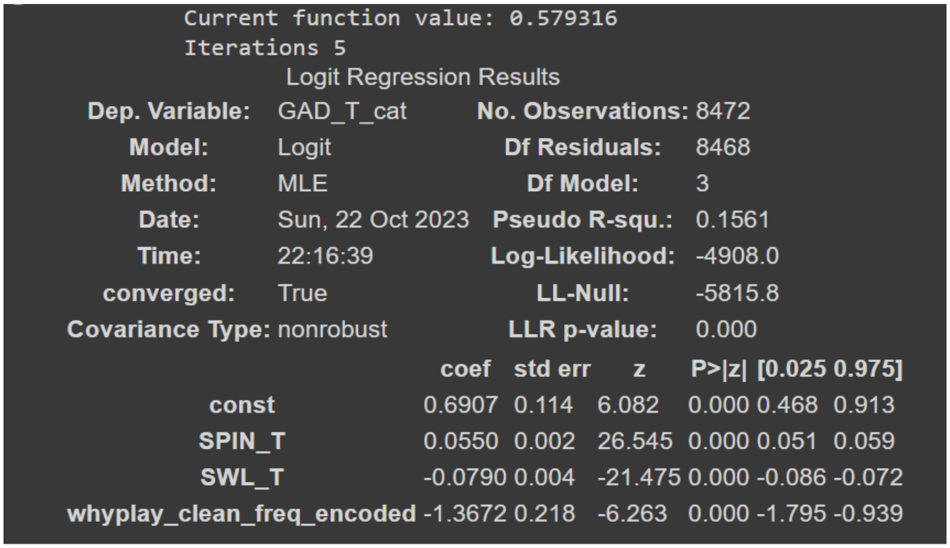
The multirun analysis results indicate the consistency in the performance of the models across different random splits of the data. The highest value for multivariate regression was for our **8th run** where we got accuracy of **31.37%** without regularization and **31.38%** with Ridge and **30.48%** with Lasso for which we have placed a screenshot highlighting the r2 values.

# 5. Question 2:

## A. Logistic Regression:

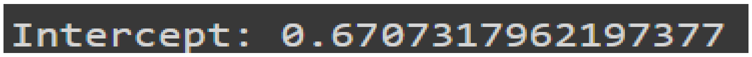
### a. Data Preparation Efforts:

We Categorized the dependent variables using the Median. (Values below the median were categorized as ***"0" (No Anxiety)*** and ***"1" (Has Anxiety)*** and since our other categorical variables in the feature matrix are already encoded using ***"Frequency Encoding"***, we were ready to perform logistic regression.



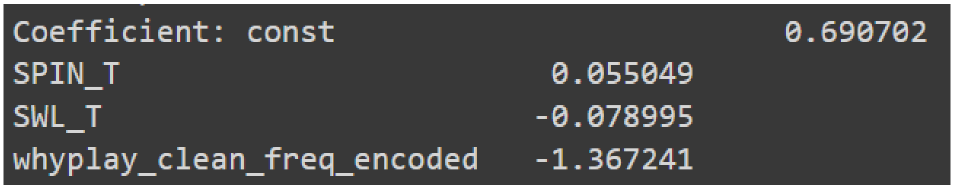
```
         Current function value: 0.579316
         Iterations 5
                    Logit Regression Results
    Dep. Variable:  GAD_T_cat         No. Observations: 8472
       Model:       Logit                Df Residuals:  8468
      Method:       MLE                     Df Model:   3
       Date:        Sun, 22 Oct 2023  Pseudo R-squ.:    0.1561
       Time:        22:16:39          Log-Likelihood:   -4908.0
     converged:     True                   LL-Null:     -5815.8
  Covariance Type: nonrobust             LLR p-value:   0.000
                                coef   std err    z     P>|z| [0.025 0.975]
            const              0.6907  0.114   6.082    0.000 0.468  0.913
           SPIN_T              0.0550  0.002   26.545   0.000 0.051  0.059
           SWL_T              -0.0790  0.004  -21.475   0.000 -0.086 -0.072
  whyplay_clean_freq_encoded  -1.3672  0.218  -6.263    0.000 -1.795 -0.939
```

### b. Intercept:

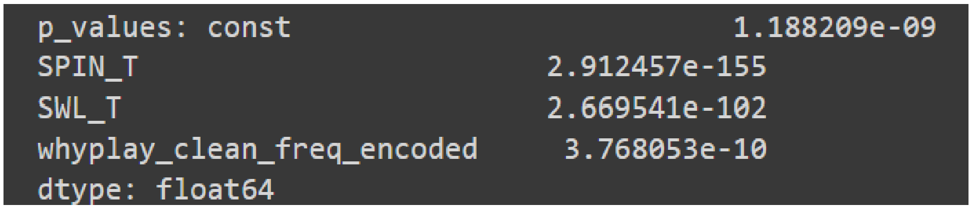


```
Intercept: 0.6707317962197377
```

### c. Feature-Wise Coefficients & Statistical Significance:

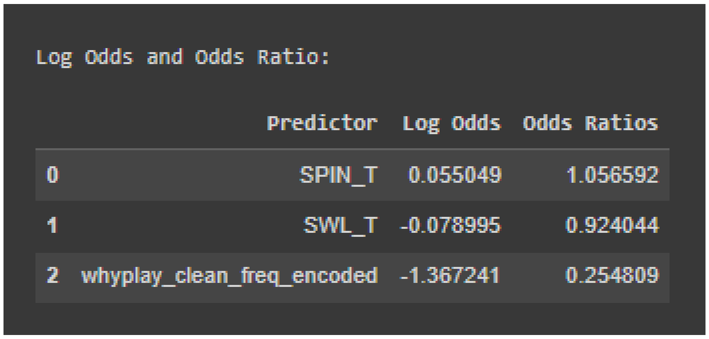


```
Coefficient: const                                0.690702
SPIN_T                             0.055049
SWL_T                             -0.078995
whyplay_clean_freq_encoded        -1.367241
```



```
p_values: const                           1.188209e-09
SPIN_T                          2.912457e-155
SWL_T                           2.669541e-102
whyplay_clean_freq_encoded       3.768053e-10
dtype: float64
```

### d. Log-Odds (TBD) & Odd-Ratios of Outcome (Per Unit Increase in each Independent Variable):

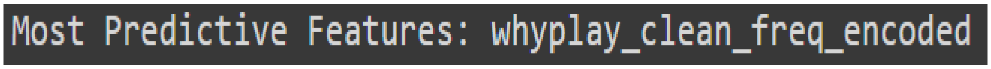


Log Odds and Odds Ratio:

|   | Predictor | Log Odds | Odds Ratios |
|---|---|---|---|
| 0 | SPIN_T | 0.055049 | 1.056592 |
| 1 | SWL_T | -0.078995 | 0.924044 |
| 2 | whyplay_clean_freq_encoded | -1.367241 | 0.254809 |

- From the above table, for every point rise in SPIN_T, the log odds of having a higher GAD_T score (above median) goes up by around **0.055**. So, the Social Phobia Inventory tends to be linked to a higher GAD_T score.
- Furthermore, As SWL_T goes up by one point, the log odds of having a higher GAD_T score drops by about **0.079.** This suggests that those who are more satisfied with life might be less prone to higher GAD_T scores.
- Also, Every single unit rise in whyplay_clean_freq_encoded results in the log odds of a higher GAD_T score dropping substantially by around **1.369.**

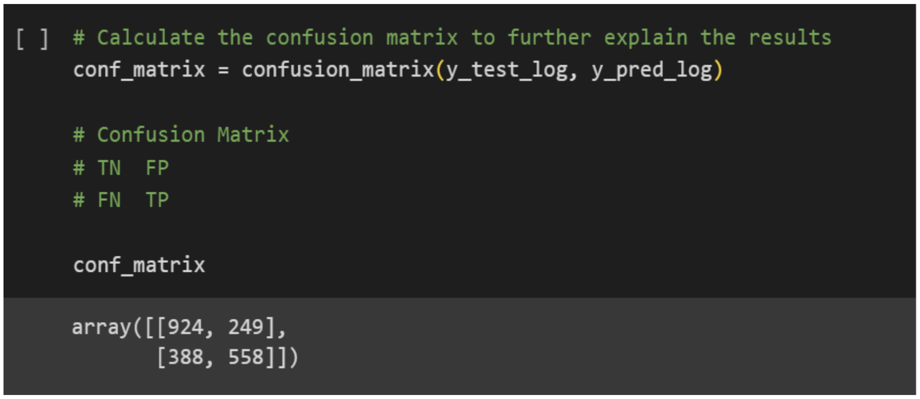
### e. Most predictive features (According to Training Data):



```
Most Predictive Features: whyplay_clean_freq_encoded
```

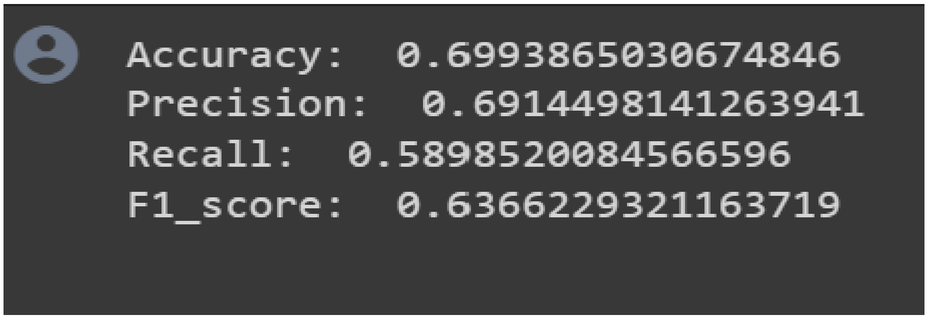
### f. Observations (Trained against Test):

i. Firstly, we calculated the ***"Confusion Matrix"*** on the Testing Data



```
[ ]  # Calculate the confusion matrix to further explain the results
     conf_matrix = confusion_matrix(y_test_log, y_pred_log)

     # Confusion Matrix
     # TN  FP
     # FN  TP

     conf_matrix

     array([[924, 249],
            [388, 558]])
```

**ii.** Based on the confusion matrix, we calculated the ***"Accuracy"***, ***"Precision"***, ***"Recall"*** and ***"F1_Score"*** for the Logistic Model



```
Accuracy:  0.6993865030674846
Precision:  0.6914498141263941
Recall:  0.5898520084566596
F1_score:  0.6366229321163719
```

**iii.** The F1_Score of ***63.6%*** indicates that the model is ***"Relatively Balanced".***

### g. Prediction on Testing Dataset:

**Accuracy:** The logistic regression model achieved an accuracy of approximately **69.90%**

**Confusion Matrix:**

**True Positives (TP):** 558

**True Negatives (TN):** 924

**False Positives (FP):** 249

**False Negatives (FN):** 388

From the confusion matrix, we can infer that the model correctly predicted 558 positive instances and 924 negative instances. However, it also incorrectly predicted 249 positive instances (which were actually negative) and 388 negative instances (which were actually positive).

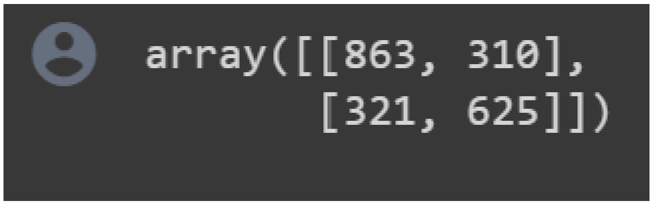
**Inference:**

Given the dataset and the features used, the logistic regression model provides reasonably good accuracy. The model might benefit from further tuning.

## B. Classification (Using Naïve Bayes):

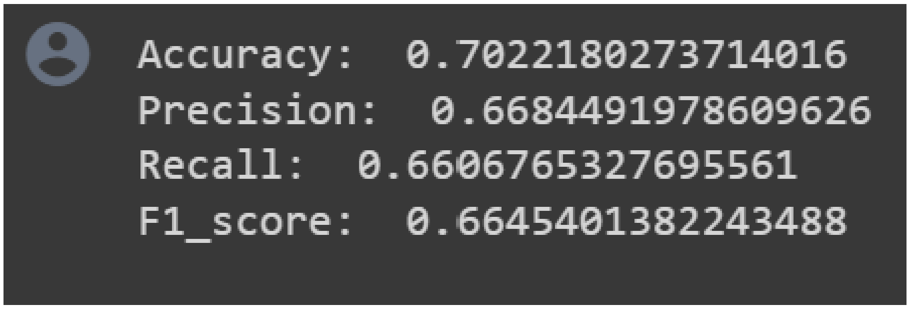
### a. Observations (From Training the Classifier):

**Confusion Matrix:**



```
array([[863, 310],
       [321, 625]])
```

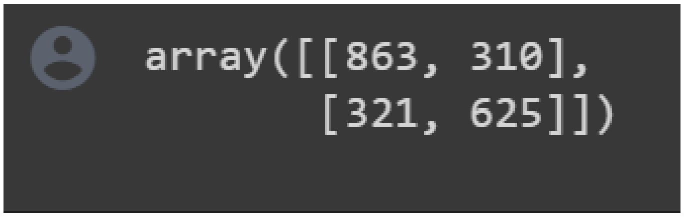
**Results:**



```
Accuracy:  0.7022180273714016
Precision:  0.6684491978609626
Recall:  0.6606765327695561
F1_score:  0.6645401382243488
```

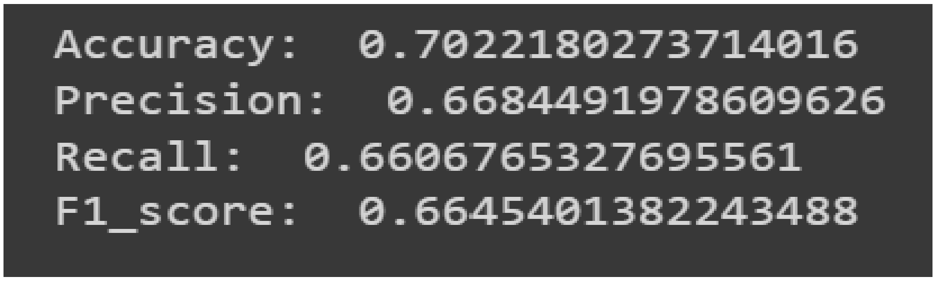
### b. Observations (Using Laplace Estimator):

**Confusion Matrix:**



```
array([[863, 310],
       [321, 625]])
```

**Results:**



```
Accuracy:  0.7022180273714016
Precision:  0.6684491978609626
Recall:  0.6606765327695561
F1_score:  0.6645401382243488
```

**Summary of Findings:**

Naïve Bayes Classification Results:

Without Laplace Smoothing:

True Positives (TP): 625

True Negatives (TN): 863

False Positives (FP): 310

False Negatives (FN): 321

With Laplace Smoothing:

True Positives (TP): 625

True Negatives (TN): 863

False Positives (FP): 310

False Negatives (FN): 321

**Inference:**

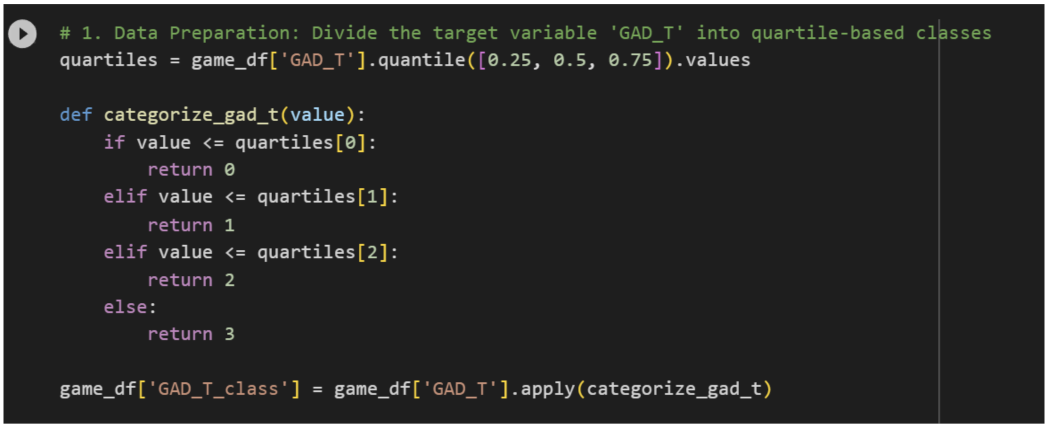
Upon observation, the confusion matrix values for both models ***(with and without Laplace smoothing)*** are the same. This means that, for this particular dataset and feature set, the Laplace estimator did not improve the results.

# 6. Question 3:

## A. Decision Trees & Random Forests:

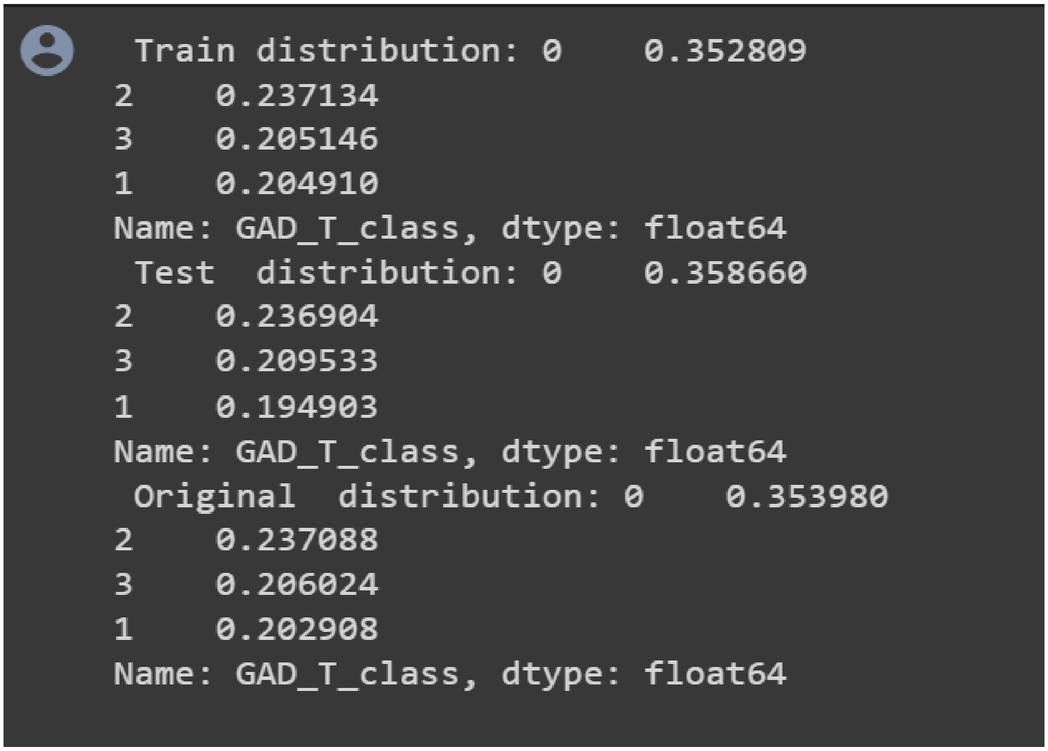
### a. Data Preparation Efforts:

Firstly, we divided the target variable ***(GAD_T)*** into **"*Quartile-Based Classes*" (0,1,2,3)** since our target variable is not binary.



```
# 1. Data Preparation: Divide the target variable 'GAD_T' into quartile-based classes
quartiles = game_df['GAD_T'].quantile([0.25, 0.5, 0.75]).values

def categorize_gad_t(value):
    if value <= quartiles[0]:
        return 0
    elif value <= quartiles[1]:
        return 1
    elif value <= quartiles[2]:
        return 2
    else:
        return 3

game_df['GAD_T_class'] = game_df['GAD_T'].apply(categorize_gad_t)
```

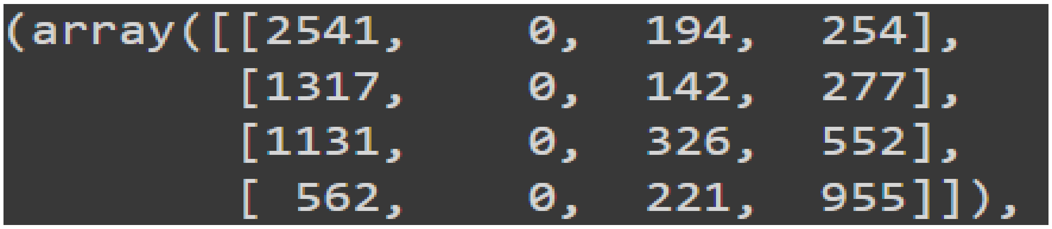
### b. Train Data vs. Test Data vs. Original Distribution:



```
 Train distribution: 0    0.352809
2    0.237134
3    0.205146
1    0.204910
Name: GAD_T_class, dtype: float64
 Test  distribution: 0    0.358660
2    0.236904
3    0.209533
1    0.194903
Name: GAD_T_class, dtype: float64
 Original  distribution: 0    0.353980
2    0.237088
3    0.206024
1    0.202908
Name: GAD_T_class, dtype: float64
```

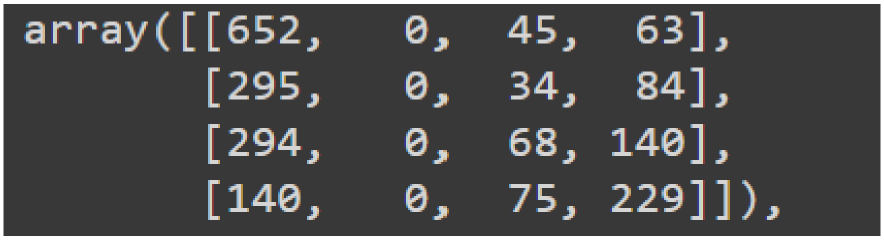
c. **Results (From Trained Decision Tree):**

**Confusion Matrix (Train Data):**



```
(array([[2541,     0,   194,   254],
        [1317,     0,   142,   277],
        [1131,     0,   326,   552],
        [ 562,     0,   221,   955]]),
```

**Confusion Matrix (Test Data):**



```
array([[652,    0,   45,   63],
       [295,    0,   34,   84],
       [294,    0,   68,  140],
       [140,    0,   75,  229]]),
```

**Correctly Classified Samples for Train Data:** 0.4511331444759207

**Incorrectly Classified Samples for Train Data:**0.5488668555240793

**Correctly Classified Samples for Test Data:** 0.44785276073619634

**Incorrectly Classified Samples for Test Data:** 0.5521472392638036

## d. Observations (From Trained Decision Tree):

As you can see from the confusion matrices, we were having a problem with Class 1 since all of them were 0 so we tried to understand what might have caused this issue and we found out that:

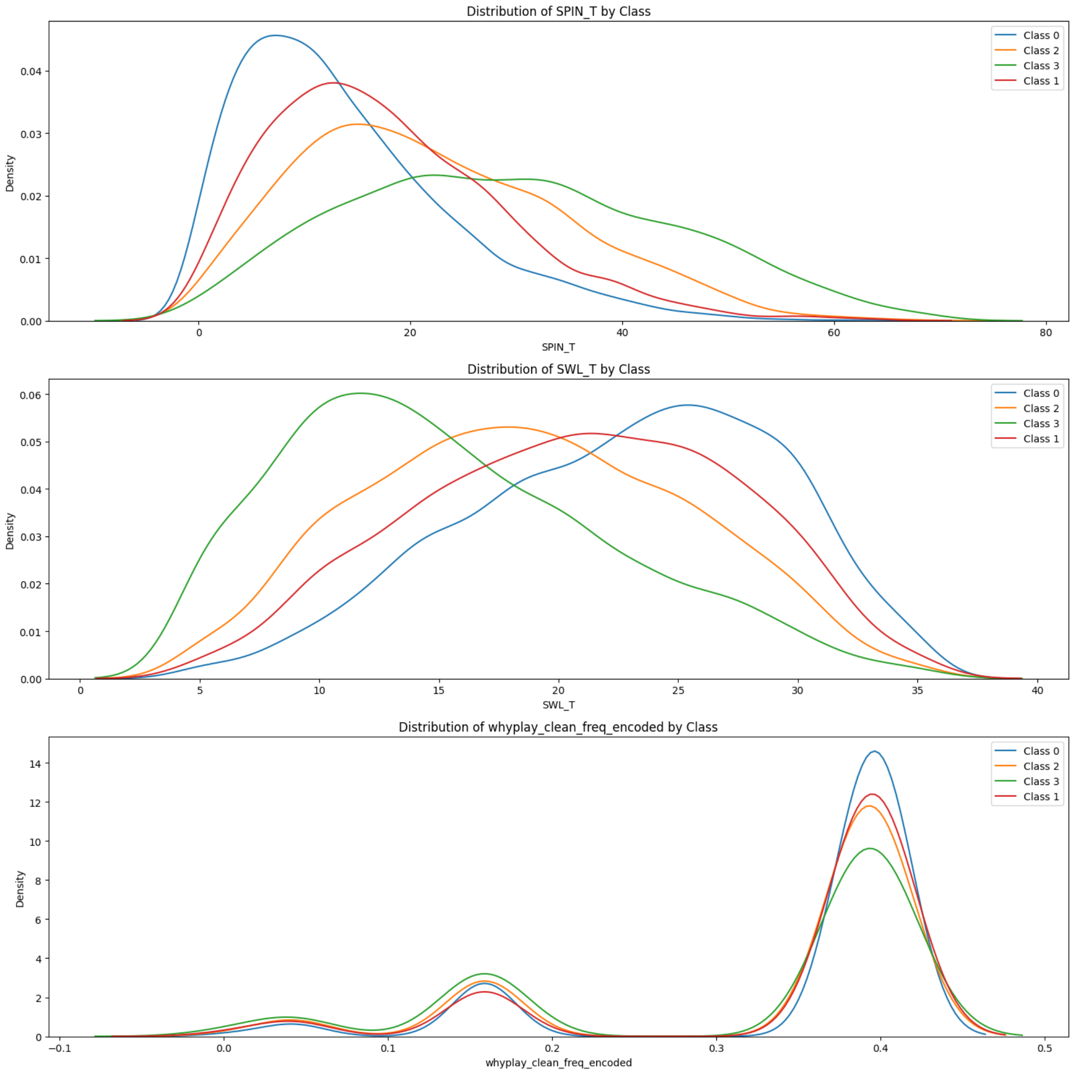
The training and testing accuracies are quite close, which suggests that the model isn't overfitting significantly.

Conclusion:

The decision tree's classification accuracy is almost similar for both the training and testing sets. The tree, however, seems to struggle with distinguishing between the classes, as seen in the confusion matrices

- While there is some imbalance, it's not extreme. Class 1 has slightly fewer samples than Class 0 and Class 2, but it's comparable to Class 3. (Imbalance).
- Class 1 is smaller than Class 0 and Class 2 but is very similar in size to Class 3. The model's difficulty with Class 1 is not solely due to its relative size in the dataset(Relative Size).

We tried to visualize the above imbalance and relative size problem for our data and below are the plots for



the same:

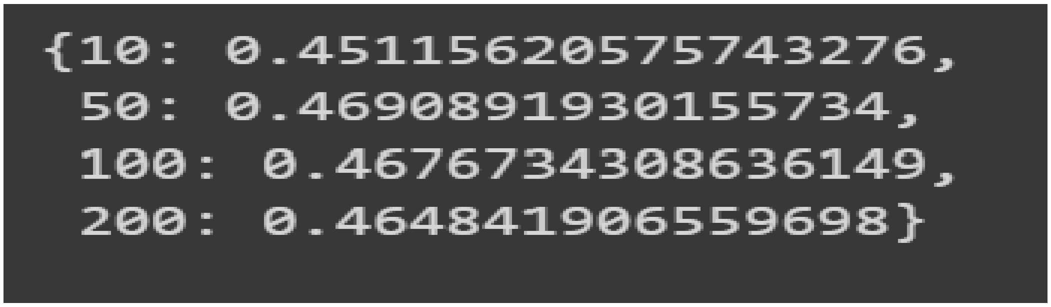
- **SPIN_T:** While Class 1 has a broader distribution in "SPIN_T", its significant overlap with Classes 2 and 3 makes differentiation challenging.
- **SWL_T:** "SWL_T" shows Class 1 peaking around 25, similar to Classes 0 and 2, leading to potential confusion between them.
- **whyplay_clean_freq_encoded:** This feature offers limited distinction across classes, particularly failing to differentiate Class 1 effectively.

**Conclusion:**

The decision tree's classification accuracy is almost similar for both the training and testing sets. The tree, however, seems to struggle with distinguishing between the classes, as seen in the confusion matrices

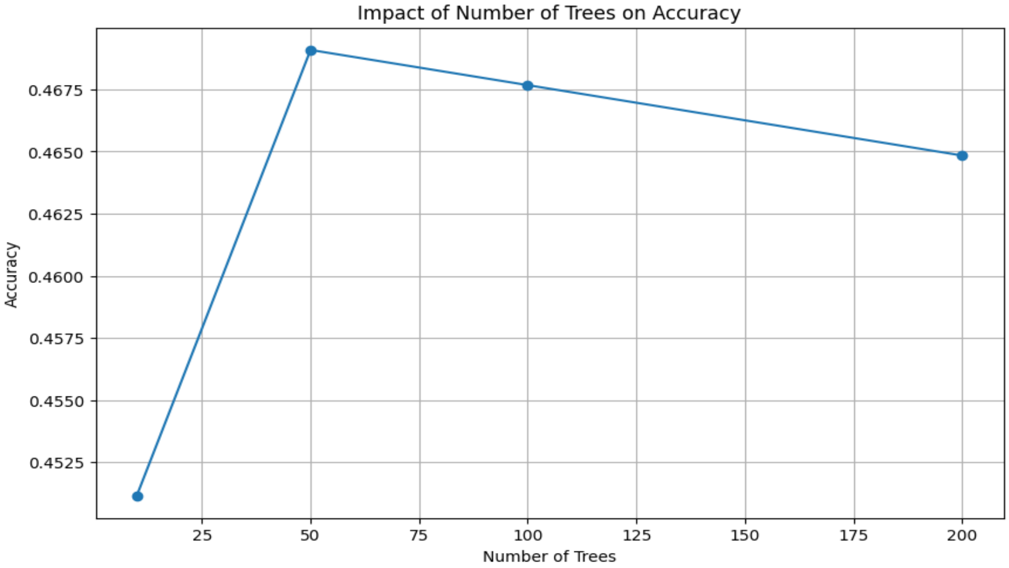
## e. Observations (From Gradient Boosted Decision Trees):

For Gradient Boosting, we chose **10, 50, 100, 200 (till 200) trees** and then we got the following accuracy results for boosting:



```
{10: 0.45115620575743276,
 50: 0.4690891930155734,
 100: 0.4676734308636149,
 200: 0.4648419065596 98}
```

Next, we plotted the Boosting Results:



**Result:**

- The accuracy generally increases as the number of trees increases, especially from 10 to 50 trees.
- Beyond 50 trees, the accuracy plateaus shows a slight decrease when moving to 200 trees.

## f. Observations from Comparative Analysis (Bagging & Random Forests):

For Bagging & Random Forests, we chose **10, 50, 100, 200 (till 200) trees** and then we got the following accuracy results for boosting:

**Bagging Results:**

({10: 0.38414346389806514,

50: 0.3954695611137329,

100: 0.39263803680981596,

200: 0.4011326097215668}

**Random Forest Results:**

{10: 0.38508730533270413,

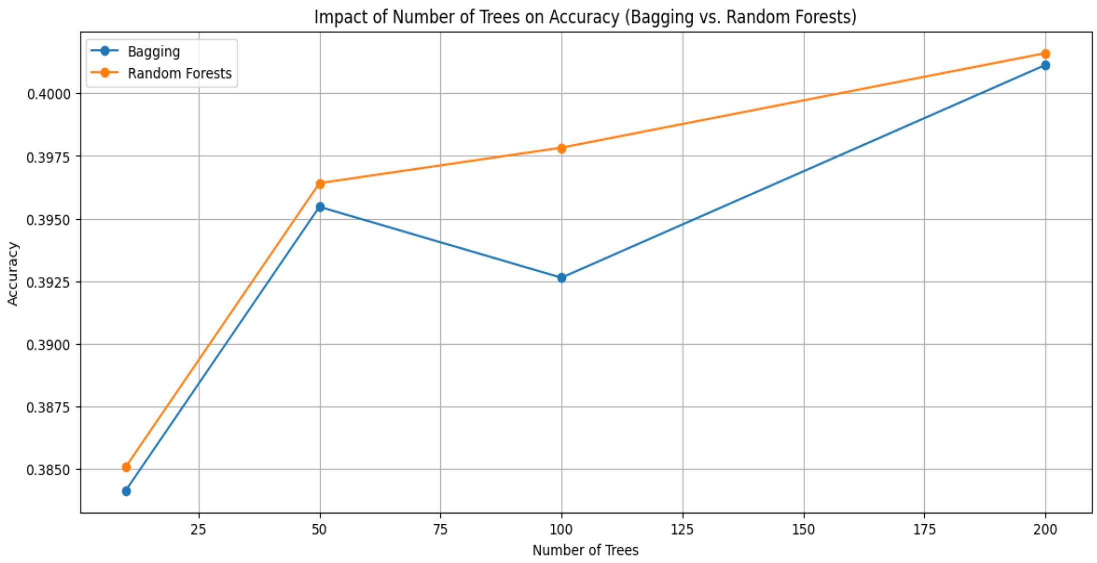
50: 0.3964134025483719,

100: 0.39782916470033036,

200: 0.4016045304388863},

**Inference:**

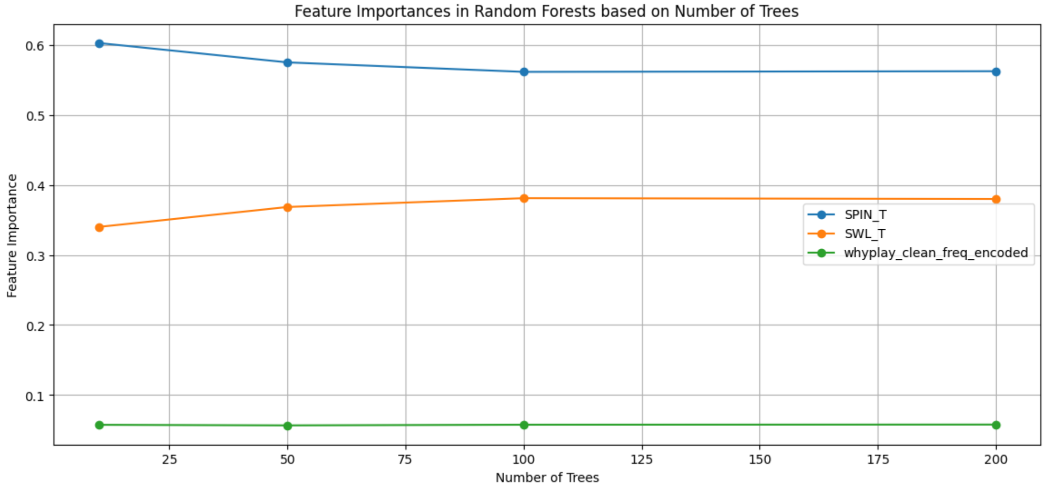
The Random Forest classifier generally outperforms the Bagging classifier as the number of trees increases, especially beyond 50 trees. We also visualized the plot for feature importance in both Bagging vs. Random Forests and also impact on the number of trees on accuracy for both bagging and random forests.



## g. Most Important Features from Random Forest:

### i. Feature Importances for Each:

```
{10: array([0.6024941 , 0.33995189, 0.05755401]),
 50: array([0.57482615, 0.36843685, 0.056737  ]),
 100: array([0.56129486, 0.38102854, 0.0576766 ]),
 200: array([0.56222445, 0.37987567, 0.05789988])})
```



# 7. Question 4:

## Comparative Analysis (All Classifiers):

## A. Linear Regression:

**Univariate Analysis:**

Among the individual features, SWL_T has the highest prediction accuracy of 23.09%. This indicates that it might be the most influential individual feature in predicting the target.

**Strongest feature Correlation:**

Negative: whyplay_clean :- -2.945
Positive: Work:- 0.6648
No Correlation:- playstyle_clean

**Multivariate Analysis:**

The multivariate model without regularization has a prediction accuracy of 30.09%, which is higher than any individual feature's accuracy. This shows the benefit of using multiple features in prediction.

**Regularization:**

Introducing Ridge regularization, the model's accuracy changed to 29.97%, indicating that Ridge regularization didn’t have much impact on the accuracy scores.

Lasso regularization also slightly reduces the model's accuracy to 29.57%. This suggests that while Lasso might be simplifying the model by zeroing out some feature coefficients, it might be doing so at the cost of predictive accuracy.

**Multirun Analysis:**

The multirun analysis results indicate the consistency in the performance of the models across different random splits of the data. The difference in performance between the models remains relatively consistent, indicating robustness in our models' predictive capability.
The results of the best run results:

Multivariate:

- Without regularization:- 31.37%
- Ridge:- 31.38%
- Lasso:- 30.48%

**Conclusion:**

The best result we obtained was a prediction accuracy of 31.38% with ridge regression on one of our multi-run analyses for multivariate regression.

## B. Logistic Regression:

**Logistic Regression:**

- Our analysis shows that for every unit increase in the (SPIN_T) score, the chances of having a higher (GAD_T) score (indicating more anxiety) increases. To be precise,it increases by around 0.055 (log odds).

- Conversely, as the SWL_T score (satisfaction with life) goes up by one point, the likelihood of having a higher (GAD_T) score decreases by about 0.079 (log odds). This implies that individuals more satisfied with their lives might be less prone to anxiety.

- Another interesting observation for every unit increase in `whyplay_clean_freq_encoded`, the likelihood of having a higher GAD_T score decreases significantly, by around 1.369 (log odds).

**Model Performance:**

Accuracy: 69.90%.

F1_Score: 63.6%. (Relatively Balanced and not overly biased towards one class).

**Conclusion:**

Overall, the Logistic Regression model offers promising results in predicting the presence of anxiety among individuals based on the provided features. However, further tuning of the model can improve the prediction accuracy.

**Naïve Bayes :**

**Naïve Bayes without Laplace Smoothing:**

**Model Performance:**

- **Accuracy:** The model accurately predicted the outcome in approximately 70.22% of the instances.
- **Precision:** Out of all the positive predictions made by the model, about 66.84% were indeed correct.
- **Recall:** Out of all the actual positive instances, the model managed to identify 66.06% of them.
- **F1_Score:** The balance between precision and recall, represented by the F1 score, stood at 66.45%.

**Naïve Bayes with Laplace Smoothing:**

**Model Performance:**

- **Accuracy:** Even with Laplace smoothing, the accuracy remained the same at 70.22%.
- **Precision:** Precision was consistent at 66.84%.
- **Recall:** Recall saw a minute change, registering at 66.07%.

- **F1_Score:** The F1 score remained unchanged at 66.45%.

**Conclusion:**

From our analysis it is observed that the performance metrics remained consistent, whether or not Laplace smoothing was applied. This suggests that, for this specific dataset and set of features, the Laplace estimator did not have a significant impact on the model's predictive capabilities.

## C. Decision Trees & Random Forests:

For a single decision tree model, the accuracy is around around 44.78% on the test set. This will serve as our baseline for comparison. As we increased the number of trees in boosting:

- With just 10 trees, the accuracy is roughly the same as the single tree.
- A significant jump in accuracy is observed when using 50 trees, reaching 46.91%.
- The accuracy slightly dips at 100 trees (46.77%) and further reduces at 200 trees (46.48%).

**Key Observations:**

- The single tree model has very similar performance on both the training and test sets, indicating it might not be overfitting significantly. However, overfitting is seen in the boosted models (especially those with more trees) as there is a decrease in accuracy after 50 trees.
- Boosting improved the model's accuracy from 44.78% to 46.91% (with 50 trees). However, after that point, adding more trees doesn't seem to benefit the model, and can even harm its performance.
- Based on our results, the model with 50 trees seems to strike the best balance between complexity and performance.

Furthermore, The results of decision trees and random forests are explained in detail in our plots in the Bagging and Random Forests section of our report.

**Conclusion:**

Overall, boosting provides a way to improve the performance of a single decision tree). However, like all methods, it has its limitations as mentioned above.

# 8. Contributions:

***Code:*** *Rajeevan (Linear and Data Cleaning) , Pranav(Logistic  & NB), Asmita (Decision trees and Random forest)*

***Report Documentation:*** *Rajeevan, Pranav*

***Presentation Slide Deck:*** *Rajeevan, Asmita*

***Presentation:*** *Rajeevan, Pranav, Asmita*

**-THE END-**