National College of Ireland

# Machine Learning and Eye-tracking Framework to Detect Engagement in Online Learning

MSc Research Project in Data Anlaytics
Configuration Manual

Yogalakshmi Chandrasekar
Student ID: x20221665@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Dr.Paul Stynes, Dr.Anu Sahni, Dr.Pramod Pathak

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Yogalakshmi Chandrasekar |
| **Student ID:** | x20221665@student.ncirl.ie |
| **Programme:** | Configuration Manual |
| **Year:** | 2022 |
| **Module:** | MSc Research Project in Data Anlaytics |
| **Supervisor:** | Dr.Paul Stynes, Dr.Anu Sahni, Dr.Pramod Pathak |
| **Submission Due Date:** | 15/08/2022 |
| **Project Title:** | Machine Learning and Eye-tracking Framework to Detect Engagement in Online Learning |
| **Word Count:** | 1255 |
| **Page Count:** | 15 |

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

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

| | |
|---|---|
| **Signature:** | Yogalakshmi Chandrasekar |
| **Date:** | 18th September 2022 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

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

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

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

# Machine Learning and Eye-tracking Framework to Detect Engagement in Online Learning

Yogalakshmi Chandrasekar
x20221665@student.ncirl.ie

## 1 Introduction

The report is aimed to outline the steps involved in the implementation of the research thesis 'Machine Learning and Eye-tracking Framework to Identify Engagement in Online Learning'. The configuration manual covers every step that is carried out in the entire project life cycle of this engagement detection research. This research study is targeted to identify participants engagement in a digital learning ecology. It is a framework that combines the machine learning algorithms and the data extracted from the SMI eye tracking device and retrieved from a BeGaze software in a machine readable form. There are totally three experiments conducted with 3 different sets of features and having the same dependent variable named 'Median split'. The data extracted from the Begaze software, Scores from the MCQ were used to classify the participants into two groups, and the data from the personality questionnaire. The report is structured into following sections, detailed below:

- Section 2 describes the specification computational resources. This section briefs about the tools used, the software and the hardware specifications of the machine employed.
- Section 3 describes the equipment used for conducting the primary experiment for the research.
- Section 4 details the data acquisition procedure and steps.
- Section 5 discusses the data pre-process that is done for the research with the data collected to make it ready to apply the machine learning models on the data-sets and the code snippets.
- Section 6 details the specifics of the three experiments that is done to identify the engagement and the code snippets.
- Section 7 provides the list of machine learning algorithm that is employed and the code snippets.
- The report concludes with section 7 with the overall summary of the document.

# 2 Computational Resource Information

The device configuration of the machine used in this research are below:

| | |
|---|---|
| Processor | Intel(R) Core(TM) i5-8145U CPU @ 2.10GHz 2.30 GHz |
| Installed RAM | 8.00 GB (7.85 GB usable) |
| System type | 64-bit operating system, x64-based processor |

Figure 1: Device specifications

| | |
|---|---|
| Edition Windows 11 Pro | |
| Version 21H2 | |
| OS build | 22000.856 |
| Experience | Windows Feature Experience Pack 1000.22000.856.0 |

Figure 2: Windows Specifications

To implement the machine learning models Jupyter notebook is used as an integrated development environment (IDE) and Python coding language is used. The version information are below:

| | |
|---|---|
| Anaconda Installers | Python 3.9 64-Bit Graphical Installer |
| Jupyter notebook | 6.4.8 |

Figure 3: Integrated Development Environment

| | |
|---|---|
| Coding Language | Python |
| Libraries | Pandas, NumPy, Seaborn, and Scikit learn |

Figure 4: Language and Libraries

# 3 Experimental Equipment

The experiment for this research is conducted on 26 participants 20 male participants and 6 female participants. Age of the participants ranged between 19 and 52. The participants involved in the experiment is asked to fill out a personality questionnaire in an online google form. They are posted with questions about their gender, age, behavioural pattern,

current mood with the help of Stanford Sleepiness Scale (SSS), and Karolinska Sleepiness Scale (KSS). After filling out the pre-experiment questionnaire they are asked to wear an eye tracking device to listen to an online video lecture on android development. Finally, they are asked to answer a few questions based on the tutorial to measure the score and the highest point is set to 15.

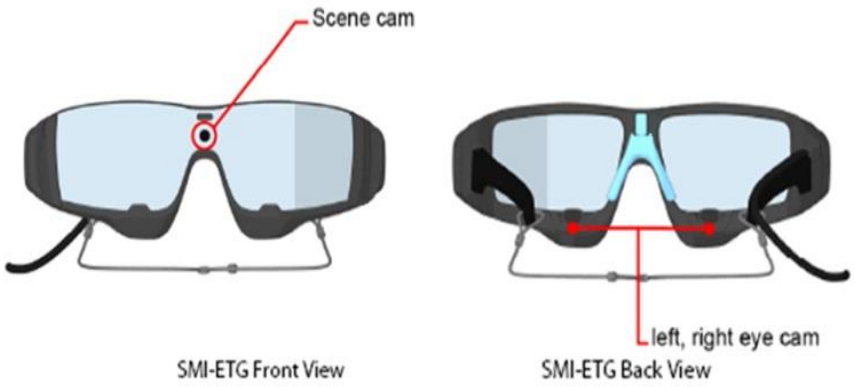


Figure 5: SMI eye tracker

A machine which is exclusive for functioning the BeGaze software is used specifically to store the raw data generated from the eye tracking experiment. The data for each video is stored inside a folder which contains audio, video, gaze collected, calibration and annotations details. The folder is saved under the participants name provided in the software at the beginning.

To extract the data in a machine readable format the entire folder must be opened via BeGaze software from the menus option.


Figure 6: BeGaze Software

Below figure shows the glimpse of how the metric export environment in BeGaze software looks like. Once the video is processed Gaze point option must be clicked and

metrics must be exported from the metrics option in the tool. Event statistics and trail summary are extracted and stored. The trial summary statistics is used in this research project.

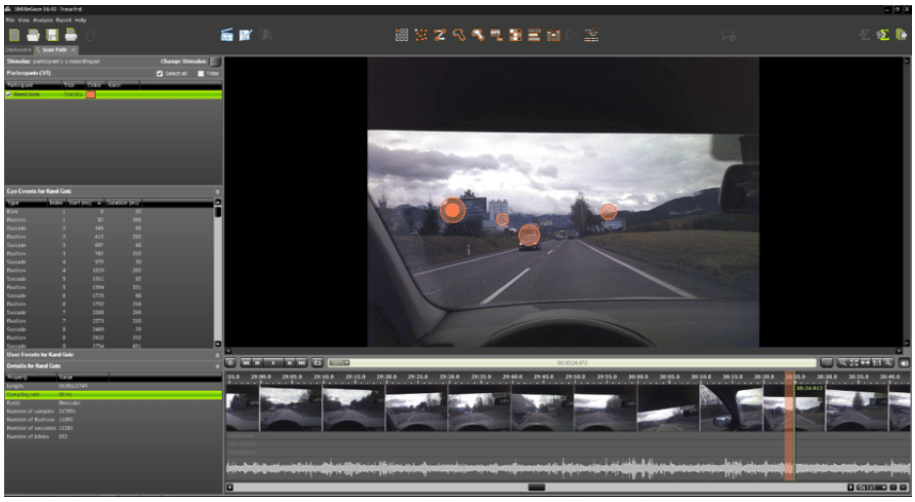

Figure 7: Metrics Export Environment

# 4 Data Acquisition

Below table provides the list of metrics that is generated by the BeGaze software with the help of the data captured by the SMI eye tracker.

| | |
|---|---|
| 1 | Trial |
| 2 | Stimulus |
| 3 | Export Start Trial Time [ms] |
| 4 | Export End Trial Time [ms] |
| 5 | Participant |
| 6 | Color |
| 7 | Visual Intake Count |
| 8 | Visual Intake Frequency [count/s] |
| 9 | Visual Intake Duration Total [ms] |
| 10 | Visual Intake Duration Average [ms] |
| 11 | Visual Intake Duration Maximum [ms] |
| 12 | Visual Intake Duration Minimum [ms] |
| 13 | Visual Intake Dispersion Total [px] |
| 14 | Visual Intake Dispersion Average [px] |
| 15 | Visual Intake Dispersion Maximum [px] |
| 16 | Visual Intake Dispersion Minimum [px] |
| 17 | Saccade Count |
| 18 | Saccade Frequency [count/s] |
| 19 | Saccade Duration Total [ms] |
| 20 | Saccade Duration Average [ms] |
| 21 | Saccade Duration Maximum [ms] |

| | |
|---|---|
| 21 | Saccade Duration Maximum [ms] |
| 22 | Saccade Duration Minimum [ms] |
| 23 | Saccade Amplitude Total [°] |
| 24 | Saccade Amplitude Average [°] |
| 25 | Saccade Amplitude Maximum [°] |
| 26 | Saccade Amplitude Minimum [°] |
| 27 | Saccade Velocity Total [°/s] |
| 28 | Saccade Velocity Average [°/s] |
| 29 | Saccade Velocity Maximum [°/s] |
| 30 | Saccade Velocity Minimum [°/s] |
| 31 | Saccade Latency Average [ms] |
| 32 | Blink Count |
| 33 | Blink Frequency [count/s] |
| 34 | Blink Duration Total [ms] |
| 35 | Blink Duration Average [ms] |
| 36 | Blink Duration Maximum [ms] |
| 37 | Blink Duration M |

Figure 8: Features (Metrics) from BeGaze software

# 5 Data Pre-process

This section of the report would detail every step involved in handling the data gathered through the eye tracking experiment.

## 5.1 Preparation

Most of the steps of data cleaning and preparation are done programmatically. The trial summary collected for each participants are stored in a 'txt' format. Ideally, the summary generated are a record for each participants but in some experiments it generated more than a record and they are included in the experiment. So final data set contains 45 records.

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

```
df= pd.read_csv("C:/Users/coding/Downloads/MCQ.csv", index_col=0)
```

```
df["median_split"] = (df.Score<df.Score.quantile()).replace({True:0, False:1})
df['Score'] = df['Score'].astype(int)
df.median split
```

Participants' were classified in to two group according to median split results. Records that has score equal to or above the median is considered enagaged and given the value '1' and remaining trials are classified under disengaged and to represent it the value given is '0'. Each record is given a unique serial number in order to make the concatenation across the dataframe possible.

```
data=pd.read_csv("C:/Users/coding/Downloads/Trial Summary.txt",sep='\t',index_col=0)
```

```
data.insert(0, 'Experiment no', range(1, 1 + len(data1)))
data.set_index('Experiment no',inplace=True)
```

```
data["median_split"] = (data.Score<data.Score.quantile()).replace({True:0, False:1})
```

```
data = data.drop(['Participant', 'Color','Stimulus'], axis=1)
```

```
data
```

| Experiment no | Export Start Trial Time [ms] | Score | Export End Trial Time [ms] | Visual Intake Count | Visual Intake Frequency [count/s] | Visual Intake Duration Total [ms] | Visual Intake Duration Average [ms] | Visual Intake Duration Maximum [ms] | Visual Intake Duration Minimum [ms] | Visual Intake Dispersion Total [px] | ... | Blink Duration Total [ms] | Blink Duration Average [ms] | Blink Duration Maximum [ms] | Blink Duration Minimum [ms] | L Mou Cl Cou |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.0 | 15 | 94361.4 | 111 | 1.2 | 59592.8 | 536.9 | 3750.7 | 99.4 | 6756.4 | ... | 8264.7 | 551.0 | 4746.2 | 165.8 | |
| 2 | 0.0 | 15 | 526819.8 | 156 | 0.3 | 20084.1 | 128.7 | 348.9 | 66.2 | 16880.2 | ... | 447642.5 | 7338.4 | 82528.5 | 414.5 | |
| 3 | 0.0 | 6 | 11932.0 | 36 | 3.0 | 9576.4 | 266.0 | 995.7 | 99.3 | 1727.3 | ... | 745.8 | 149.2 | 165.7 | 132.8 | |
| 4 | 0.0 | 6 | 441586.0 | 1119 | 2.5 | 354798.9 | 317.1 | 12098.2 | 82.7 | 48621.3 | ... | 43215.2 | 219.4 | 1825.4 | 83.0 | |

Figure 9: Loading the raw data to Pandas Dataframe

## 5.2 Transformation

Each record has participants' name under 2 columns 'stimulus' and 'participant' they were dropped.

```
data = data.drop(['Left Mouse Click Count','Left Mouse Click Frequency [count/s]',
                  'Right Mouse Click Count','Right Mouse Click Frequency [count/s]'], axis=1)
```

Figure 10: Removing personal information of the participants

A variable named 'perclose' is calculated to measure the number blinks within the total time.

```
#Perclos= blink time + closing time / total time interval

def perclos():
    for i in range(len(data1)) :
        tot_blink=data.iloc[i,30] #total blink duration
        tot_interval= data.iloc[i,6] #total time interval
        print(data.iloc[i,0],tot_interval)
        PERCLOS=tot_blink/tot_interval
        data.iloc[i,36]=PERCLOS


perclos()
```

Figure 11: Calculating Perclos

The personality questionnaire is pulled using python code and the values in the gender column are transformed in to '0' if the value is 'Woman' and '1' for 'Man.

```
df_q=pd.read_csv("C:/Users/coding/Downloads/personality questionnaire Responses - Sheet1.csv")
```

```
df_q
```

| | Gender | Age | Extraverted, enthusiastic. | Critical, quarrelsome. | Dependable, self-disciplined. | Anxious, easily upset | Open to new experiences, complex | Reserved, quiet | Sympathetic, warm | Disorganized, careless | Calm, emotionally stable | Conventional, uncreative | KS Karolins Sleepine Scal This is alertne tes under tl we need rate c self fro degree sleepine as follow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Woman | 26.0 | 5.0 | 3.0 | 4.0 | 5.0 | 3.0 | 4.0 | 5.0 | 4.0 | 4.0 | 4.0 | |
| 1 | Woman | 26.0 | 5.0 | 3.0 | 4.0 | 5.0 | 3.0 | 4.0 | 5.0 | 4.0 | 4.0 | 4.0 | |
| 2 | Man | 29.0 | 4.0 | 2.0 | 6.0 | 5.0 | 3.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | |
| 3 | Man | 29.0 | 4.0 | 2.0 | 6.0 | 5.0 | 3.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | |
| 4 | Man | 29.0 | 4.0 | 2.0 | 6.0 | 5.0 | 3.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | |

```
##transformation
df_q.loc[df_q["Gender"] == "Man", "Gender"] = 1
df_q.loc[df_q["Gender"] == "Woman", "Gender"] = 0
```

```
df_q
```

| | Gender | Age | Extraverted, enthusiastic. | Critical, quarrelsome. | Dependable, self-disciplined. | Anxious, easily upset | Open to new experiences, complex | Reserved, quiet | Sympathetic, warm | Disorganized, careless | Calm, emotionally stable | Conventional, uncreative | KS Karolins Sleepine Scal This is alertne tes under tl we need rate c self fro degree sleepine as follow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 26.0 | 5.0 | 3.0 | 4.0 | 5.0 | 3.0 | 4.0 | 5.0 | 4.0 | 4.0 | 4.0 | |
| 1 | 0 | 26.0 | 5.0 | 3.0 | 4.0 | 5.0 | 3.0 | 4.0 | 5.0 | 4.0 | 4.0 | 4.0 | |
| 2 | 1 | 29.0 | 4.0 | 2.0 | 6.0 | 5.0 | 3.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | |
| 3 | 1 | 29.0 | 4.0 | 2.0 | 6.0 | 5.0 | 3.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | |
| 4 | 1 | 29.0 | 4.0 | 2.0 | 6.0 | 5.0 | 3.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | |

Figure 12: Personality Questionnaire Dataset Transformation

Finally, all the records are joined as a single dataset and exported to CSV.

```
df_exp2= pd.concat([data1, df_q], axis=1, join='inner')
display(df_exp2)
```

| Export End Trial Time [ms] | Visual Intake Count | Visual Intake Frequency [count/s] | Visual Intake Duration Total [ms] | Visual Intake Duration Average [ms] | Visual Intake Duration Maximum [ms] | Visual Intake Duration Minimum [ms] | Visual Intake Dispersion Total [px] | ... | Dependable, self-disciplined. | Anxious, easily upset | Open to new experiences, complex | Reserved, quiet | Sympathetic, warm |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 94361.4 | 111.0 | 1.2 | 59592.8 | 536.9 | 3750.7 | 99.4 | 6756.4 | ... | 4.0 | 5.0 | 3.0 | 4.0 | 5.0 |
| 526819.8 | 156.0 | 0.3 | 20084.1 | 128.7 | 348.9 | 66.2 | 16880.2 | ... | 4.0 | 5.0 | 3.0 | 4.0 | 5.0 |
| 11932.0 | 36.0 | 3.0 | 9576.4 | 266.0 | 995.7 | 99.3 | 1727.3 | ... | 6.0 | 5.0 | 3.0 | 3.0 | 4.0 |

Figure 13: Final Dataset for Experiment 3

## 5.3 Storage

The raw data of eye tracker is retrieved and stored in organization's one drive cloud solution which could be helpful for future researchers. The masked and fully transformed data-sets are stored in a github repository and it made available publicly after removing the sensitive information and as well as the Jupyter notebook is uploaded in the same location if future researchers want to replicate it.

## 5.4 Data Exploration and Visualisation

The variables are checked for their correlation and whichever feature had high correlation were removed from the model application.

```
cor_v1=data.corr()
cor_v1

sns.set(style="whitegrid", font_scale=13)

plt.figure(figsize=(300,300))
plt.title( 'Pearson Correlation Matrix', fontsize=25)
sns.heatmap(data1.corr(),linewidth=0.25, vmax=0.7, square=True, cmap="GnBu", linecolor='w',
annot=True, annot_kws={"size":10}, cbar_kws={"shrink":.7})
```

```
cor_v1=df_exp2.corr()
cor_v1

sns.set(style="whitegrid", font_scale=13)

plt.figure(figsize=(300,300))
plt.title( 'Pearson Correlation Matrix', fontsize=25)
sns.heatmap(df_exp2.corr(),linewidth=0.25, vmax=0.7, square=True, cmap="GnBu", linecolor='w',
annot=True, annot_kws={"size":10}, cbar_kws={"shrink":.7})
```

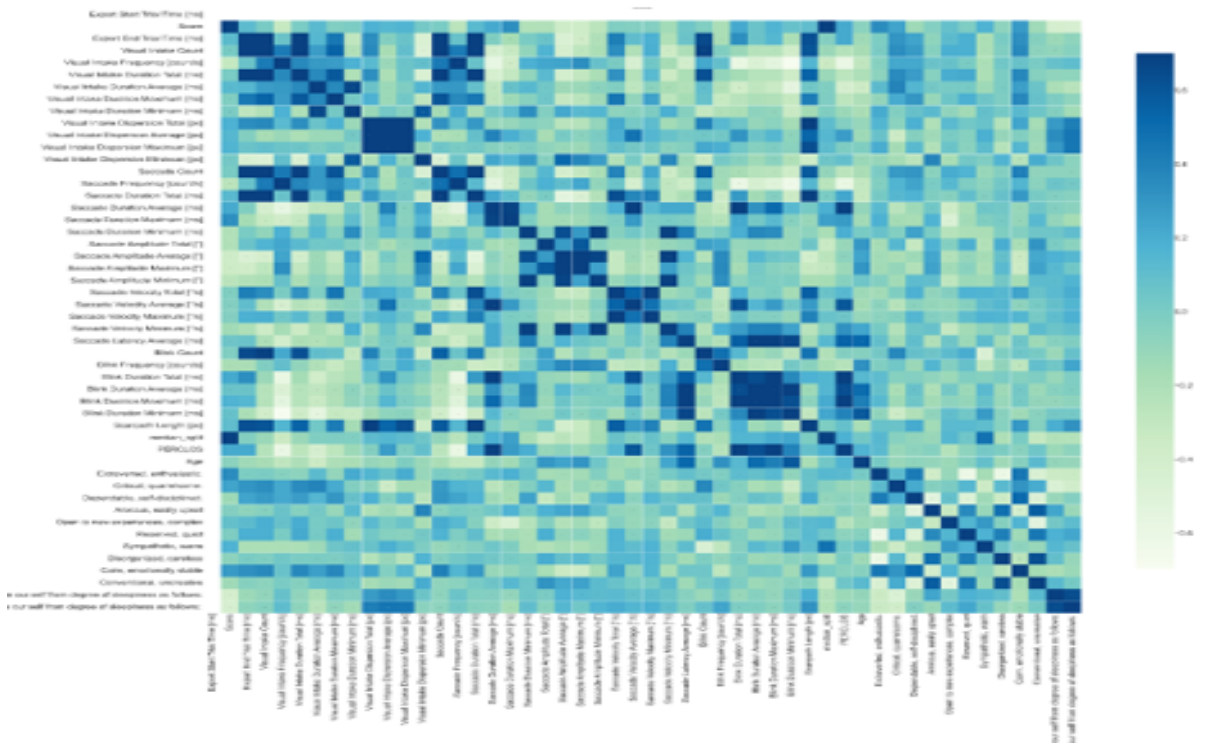

Figure 14: Pearson Correlation

# 6 Experiments

This section explains the three different experiments carried out in this research project and the rationale of each. In all the three experiment the target variable is 'Median split'.

## 6.1 Experiment 1

This experiment has included all the metrics extracted from the BeGaze software that showed no to less correlation between each other.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X1= con_df[['Visual Intake Count', 'Visual Intake Frequency [count/s]',
       'Visual Intake Duration Total [ms]',
       'Visual Intake Duration Average [ms]',
       'Visual Intake Duration Maximum [ms]',
       'Visual Intake Duration Minimum [ms]',
       'Visual Intake Dispersion Total [px]',
       'Visual Intake Dispersion Average [px]',
       'Visual Intake Dispersion Maximum [px]',
       'Visual Intake Dispersion Minimum [px]', 'Saccade Count',
       'Saccade Frequency [count/s]', 'Saccade Duration Total [ms]',
       'Saccade Duration Average [ms]', 'Saccade Duration Maximum [ms]',
       'Saccade Duration Minimum [ms]', 'Saccade Amplitude Total [°]',
       'Saccade Amplitude Average [°]', 'Saccade Amplitude Maximum [°]',
       'Saccade Amplitude Minimum [°]', 'Saccade Velocity Total [°/s]',
       'Saccade Velocity Average [°/s]', 'Saccade Velocity Maximum [°/s]',
       'Saccade Velocity Minimum [°/s]', 'Saccade Latency Average [ms]',
       'Blink Count', 'Blink Frequency [count/s]', 'Blink Duration Total [ms]',
       'Blink Duration Average [ms]', 'Blink Duration Maximum [ms]',
       'Blink Duration Minimum [ms]', 'Scanpath Length [px]']]

X1_train, X1_test, y_train, y_test = train_test_split(X1,y,test_size=0.2,
                                                      random_state=123,
                                                      shuffle=True,
                                                      )
scaler = StandardScaler()
X1_train=scaler.fit_transform(X1_train)
X1_test=scaler.fit_transform(X1_test)
```

Figure 15: Features included in the first experiment and test-train split

## 6.2 Experiment 2

This experiment has included all the metrics extracted from the BeGaze software that showed no to less correlation between each other and the calculated 'perclose' from the metrics blink count and total time interval .

```
y=df['median_split']
X=df[['Visual Intake Count',
       'Visual Intake Frequency [count/s]',
       'Visual Intake Duration Total [ms]',
       'Visual Intake Duration Average [ms]',
       'Visual Intake Duration Maximum [ms]',
       'Visual Intake Duration Minimum [ms]',
       'Visual Intake Dispersion Total [px]',
       'Visual Intake Dispersion Average [px]',
       'Visual Intake Dispersion Maximum [px]',
       'Visual Intake Dispersion Minimum [px]', 'Saccade Count',
       'Saccade Frequency [count/s]', 'Saccade Duration Total [ms]',
       'Saccade Duration Average [ms]', 'Saccade Duration Maximum [ms]',
       'Saccade Duration Minimum [ms]', 'Saccade Amplitude Total [°]',
       'Saccade Amplitude Average [°]', 'Saccade Amplitude Maximum [°]',
       'Saccade Amplitude Minimum [°]', 'Saccade Velocity Total [°/s]',
       'Saccade Velocity Average [°/s]', 'Saccade Velocity Maximum [°/s]',
       'Saccade Velocity Minimum [°/s]', 'Saccade Latency Average [ms]',
       'Blink Count', 'Blink Frequency [count/s]', 'Blink Duration Total [ms]',
       'Blink Duration Average [ms]', 'Blink Duration Maximum [ms]',
       'Blink Duration Minimum [ms]', 'Scanpath Length [px]','PERCLOS']]
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler



X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=123,
                                                    shuffle=True,
                                                    )



scaler = StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.fit_transform(X_test)
```

Figure 16: Features included in the second experiment and test-train split

## 6.3 Experiment 3

This experiment has included all the metrics extracted from the BeGaze software that showed no to less correlation between each other and the data collected from the personality questionnaire.

```
df3_final= pd.concat([df_exp3, df_ques], axis=1, join='inner')
display(df3_final)
```

| port End Trial Time [ms] | Visual Intake Count | Visual Intake Frequency [count/s] | Visual Intake Duration Total [ms] | Visual Intake Duration Average [ms] | Visual Intake Duration Maximum [ms] | Visual Intake Duration Minimum [ms] | Visual Intake Dispersion Total [px] | Visual Intake Dispersion Average [px] | Visual Intake Dispersion Maximum [px] | ... | Dependable, self-disciplined. | Anxious, easily upset | Open to new experiences, complex | Reserved, quiet |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 61.4 | 111.0 | 1.2 | 59592.8 | 536.9 | 3750.7 | 99.4 | 6756.4 | 60.9 | 357.1 | ... | 4.0 | 5.0 | 3.0 | 4.0 |
| 19.8 | 156.0 | 0.3 | 20084.1 | 128.7 | 348.9 | 66.2 | 16880.2 | 108.2 | 672.5 | ... | 6.0 | 5.0 | 3.0 | 3.0 |
| 32.0 | 36.0 | 3.0 | 9576.4 | 266.0 | 995.7 | 99.3 | 1727.3 | 48.0 | 302.4 | ... | 6.0 | 5.0 | 3.0 | 3.0 |
| 86.0 | 1119.0 | 2.5 | 354798.9 | 317.1 | 12098.2 | 82.7 | 48621.3 | 43.5 | 19673.3 | ... | 6.0 | 5.0 | 3.0 | 3.0 |

```
y3=df3_final['median_split']
X3=df3_final.drop('median_split',axis = 1)
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler



X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3,
                                                        test_size=0.2,
                                                        random_state=123,
                                                        shuffle=True,
                                                        )



scaler = StandardScaler()
X3_train=scaler.fit_transform(X3_train)
X3_test=scaler.fit_transform(X3_test)
```

Figure 17: Features included in the Third experiment and test-train split

# 7 Model Implementation

## 7.1 Support Vector Machine

SVM algorithm is applied across all the experiment and below is the code snippet for the model application and the model evaluation  validation through confusion matrix.

```
from sklearn.svm import SVC

svclassifier2 = SVC(kernel='linear')
svclassifier2.fit(X_train, y_train)
```

```
SVC(kernel='linear')
```

```
y_pred = svclassifier2.predict(X_test)
```

```
y_pred
```

```
array([1., 0., 1., 0., 0., 0., 1., 1., 0.])
```

```
from sklearn.metrics import classification_report, confusion_matrix

print('-'*30)
print('Classification Report for SVM for Experiment 2 :')
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print('-'*30)
```

Figure 18: Implementation of Support Vector Machine

## 7.2 Logistic Regression

Logistic regression algorithm is applied across all the experiment and below is the code snippet for the model application and the model evaluation validation through confusion matrix.

```
Logistic regression
```

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train,y_train)
pred_lr2 = model.predict(X_test)
print('-'*40)
print('Classification Result for Logistic Regression  :')
print(classification_report(y_test, pred_lr2))
print('-'*40)
```

Figure 19: Implementation of Logistic Regression

## 7.3 K-Nearest Neighbour

KNN algorithm is applied across all the experiment and below is the code snippet for the model application and the model evaluation  validation through confusion matrix.

```
KNN
```

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets
model.fit(X_train,y_train)

#Predict Output
y_knn_pred= model.predict(X_test)
print(y_knn_pred)
```

```
[0. 1. 1. 0. 0. 1. 1. 0. 1.]
```

```
print('-'*30)
print('Classification Report for KNN for Experiment 2 :')
print(confusion_matrix(y_test,y_knn_pred))
print(classification_report(y_test,y_knn_pred))
print('-'*30)
```

Figure 20: Implementation of KNN

## 7.4 Decision Tree

Decision Tree algorithm is applied across all the experiment and below is the code snippet for the model application and the model evaluation  validation through confusion matrix.

```
Decision tree
```

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
pred_dt2 = dt.predict(X_test)


print('-'*40)
print('Classification result for Decision Tree for second experiment :')
print(classification_report(y_test, pred_dt2))
print('-'*40)
```

Figure 21: Implementation of Decision Tree

### 7.5 Adaboost

Adaboost algorithm is applied across all the experiment and below is the code snippet for the model application and the model evaluation  validation through confusion matrix.

```
Adaboost
```

```
 from sklearn.ensemble import AdaBoostClassifier

abc = AdaBoostClassifier(n_estimators=50,
                         learning_rate=3
                         )
# Train Adaboost Classifer
model2 = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred_abc = model2.predict(X_test)

print('-'*40)
print('Classification Results for AdaBoost for second Experiment:')
print(classification_report(y_test, y_pred_abc))
print('-'*40)
```

Figure 22: Implementation of Adaboost

## 8 Conclusion

To summarise, this report illustrates step by step procedure of this research project. The sections are divided chronologically and each step is explained in full length. The entire code and the data-sets are available on the github repository