A Machine Learning based Eye Tracking Framework to detect Zoom Fatigue

MSc in Data Anlaytics (MSCDA - B)
Configuration Manual

Anjuli Patel
x19242581

School of Computing
National College of Ireland

Supervisor: Paul Stynes
Anu Sahni
Pramod Pathak

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anjuli Patel
Student ID:	19242581
Programme:	MSc in Data Analytics
Year:	2021
Module:	MSc Configuration Manual
Supervisor:	Paul Stynes, Anu Sahni, Pramod Pathak
Submission Due Date:	16/08/2021
Project Title:	A Machine Learning based Eye Tracking Framework to detect Zoom Fatigue
Word Count:	1330
Page Count:	12

Signature:	
Date:	16-08-2021

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.	☐

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

Configuration Manual
A Machine Learning based Eye Tracking Framework to detect Zoom Fatigue

Anjuli Patel
19242581
MSc Data Analytics
National College of Ireland

1 Introduction

The purpose of this document is to provide information about the steps involved in the implementation of the research project - 'A Machine Learning based Eye Tracking Framework to detect Zoom Fatigue'. The configuration manual describes the step-by-step details performed in the completion of the research. The research aims to determine to what extent can the features from the eye tracker be used to predict zoom fatigue?. For the implementation of this research, we have used five machine learning algorithms to detect Zoom fatigue and compare the performance of these machine learning algorithms. Below is the structure of this configuration manual describing the stages of implementation of the project:

- Section 2 Hardware and Software Requirements: In this section, we will describe the system configuration, tools, and technologies used in the research
- Section 3 Data Collection: In this section, we will discuss the steps involved in data collection for this research.
- Section 4 Data Pre-processing: Under this section, we will discuss the implementation, which includes, data preparation and storage, exploratory data analysis and Data transformation, and implementation of different machine learning models
- Section 5 Implementation: Under this section, we will discuss the steps taken for the implementation of the Machine Learning algorithm.
- Section 6 Conclusion: In this section, we will discuss the conclusion of the Configuration Manual

2 Hardware and Software Requirements

The detail of the system configuration used in the research is shown in 1.

Operating System	Windows 10 Home
Installed Memory (RAM)	8.0 GB
Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz

Table 1: System configuration

For implementation of the research, Python Programming Language has been used, using Jupyter Notebook as IDE. The detail of libraries and programming languages used for the research are mentioned below : - Python 3.8.5
- Jupyter Lab 3.0.14
- Pandas
- Numpy
- scikit learn
- os

3 Data Collection

In this section, we will discuss the steps involved in data collection for the research. An experiment was conducted with thirty-one participants, consisting of 12 girls and 19 boys with ages ranging between 22 to 35. In the experiment, the participants were asked to wear an eye tracker device (as shown in fig.1) while watching an online lecture. The movement of eyes and the stimulus of eyes towards the video was captured by the eye tracker device. The participants were asked to respond to two sets of the questionnaire for this experiment, where one set of the questionnaire contained questions regarding their personality, age, SSS (Stanford Sleepiness Scale), KSS (Karolinska Sleepiness Scale), and social cognition experiment. The second questionnaire consists of the response for the summary test for the lecture video.

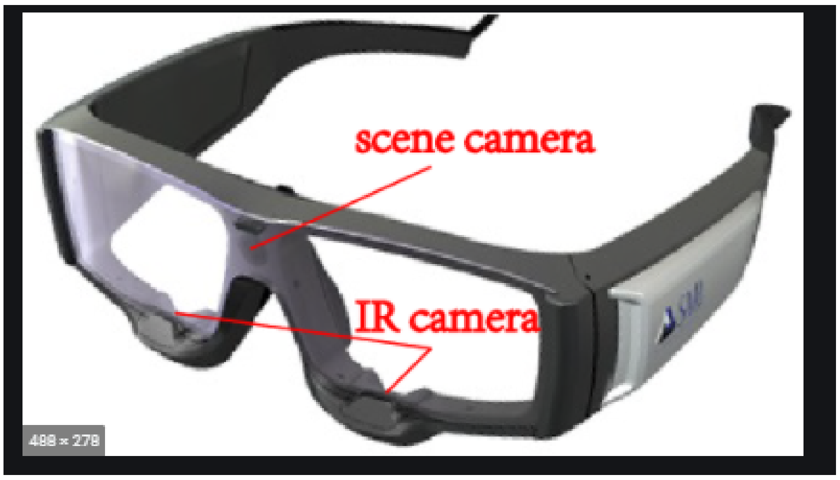
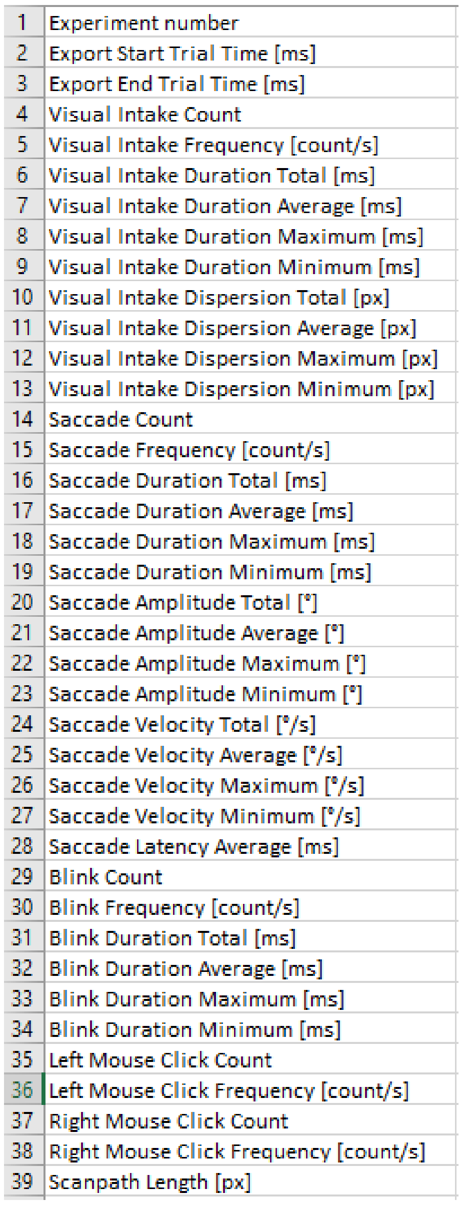


Figure 1: SMI Eye tracker glass

The data extracted from the questionnaire and eye tracker device for this research are available in github - Questionnaire_response [1]. The data extracted from eye tracker contains total of thirty-nine attributes. Figure 2 describes the list of attributes present in the dataset.



1	Experiment number
2	Export Start Trial Time [ms]
3	Export End Trial Time [ms]
4	Visual Intake Count
5	Visual Intake Frequency [count/s]
6	Visual Intake Duration Total [ms]
7	Visual Intake Duration Average [ms]
8	Visual Intake Duration Maximum [ms]
9	Visual Intake Duration Minimum [ms]
10	Visual Intake Dispersion Total [px]
11	Visual Intake Dispersion Average [px]
12	Visual Intake Dispersion Maximum [px]
13	Visual Intake Dispersion Minimum [px]
14	Saccade Count
15	Saccade Frequency [count/s]
16	Saccade Duration Total [ms]
17	Saccade Duration Average [ms]
18	Saccade Duration Maximum [ms]
19	Saccade Duration Minimum [ms]
20	Saccade Amplitude Total [°]
21	Saccade Amplitude Average [°]
22	Saccade Amplitude Maximum [°]
23	Saccade Amplitude Minimum [°]
24	Saccade Velocity Total [°/s]
25	Saccade Velocity Average [°/s]
26	Saccade Velocity Maximum [°/s]
27	Saccade Velocity Minimum [°/s]
28	Saccade Latency Average [ms]
29	Blink Count
30	Blink Frequency [count/s]
31	Blink Duration Total [ms]
32	Blink Duration Average [ms]
33	Blink Duration Maximum [ms]
34	Blink Duration Minimum [ms]
35	Left Mouse Click Count
36	Left Mouse Click Frequency [count/s]
37	Right Mouse Click Count
38	Right Mouse Click Frequency [count/s]
39	Scanpath Length [px]

Figure 2: Attributes in dataset collected from eye tracker device

4 Data Pre-processing

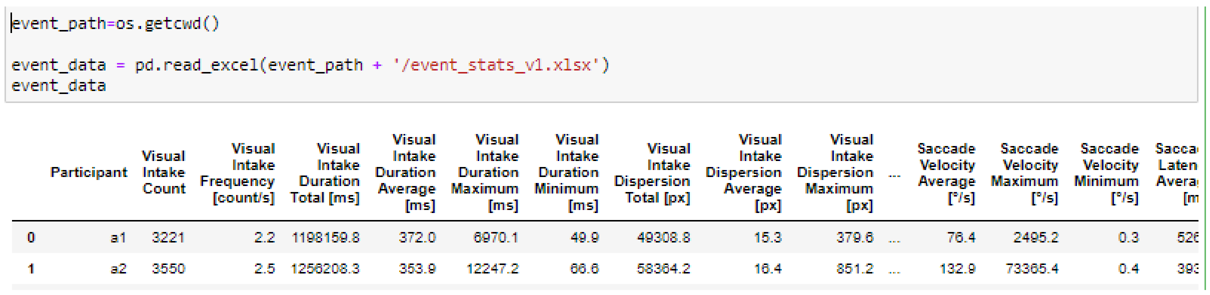
In this section we will discuss step by step process of data pre-processing for the research.

4.1 Data Preparation and Storage

The data was collected from two different excel files and merged to prepare the dataset for the research. The data in file event_stats_v1.xlsx contains the data extracted from the

[1]`https://github.com/anjuli22/Thesis_MS_19242581`

eye tracker device, see fig. 3.

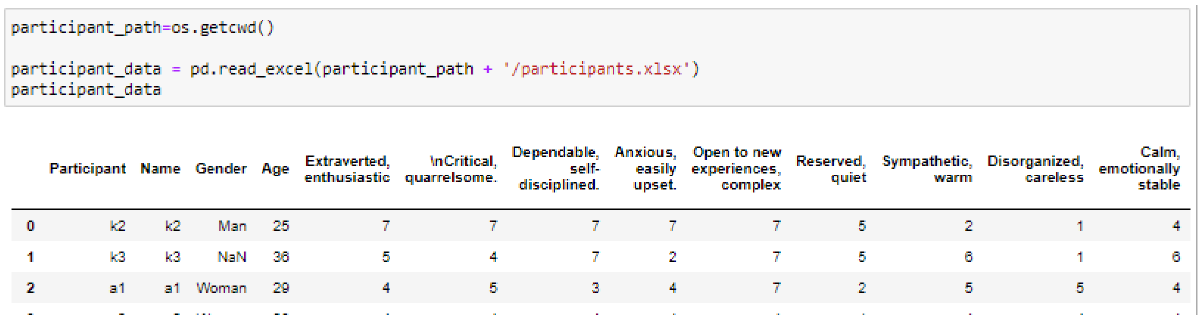


```
event_path=os.getcwd()

event_data = pd.read_excel(event_path + '/event_stats_v1.xlsx')
event_data
```

	Participant	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]	...	Saccade Velocity Average [°/s]	Saccade Velocity Maximum [°/s]	Saccade Velocity Minimum [°/s]	Sacca Laten Avera [m
0	a1	3221	2.2	1198159.8	372.0	6970.1	49.9	49308.8	15.3	379.6	...	76.4	2495.2	0.3	526
1	a2	3550	2.5	1256208.3	353.9	12247.2	66.6	58364.2	16.4	851.2	...	132.9	73365.4	0.4	393

Figure 3: Data set containing data from eye tracker device

The data in file participants.xlsx contains the data extracted from the questionnaire, see fig. 4.



```
participant_path=os.getcwd()

participant_data = pd.read_excel(participant_path + '/participants.xlsx')
participant_data
```

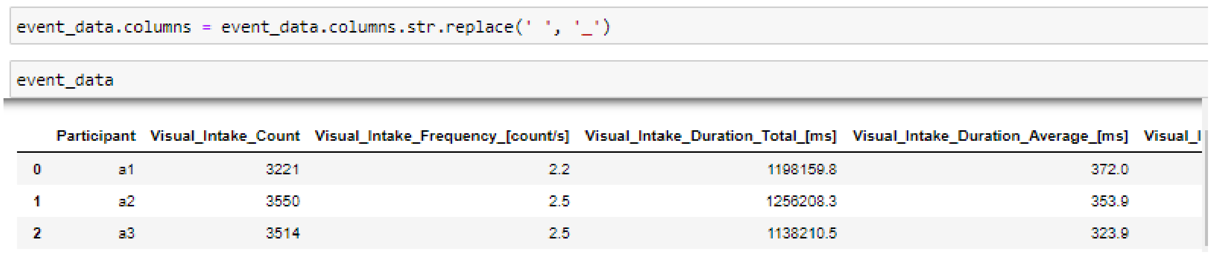
	Participant	Name	Gender	Age	Extraverted, enthusiastic	\nCritical, quarrelsome.	Dependable, self-disciplined.	Anxious, easily upset.	Open to new experiences, complex	Reserved, quiet	Sympathetic, warm	Disorganized, careless	Calm, emotionally stable
0	k2	k2	Man	25	7	7	7	7	7	5	2	1	4
1	k3	k3	NaN	36	5	4	7	2	7	5	6	1	6
2	a1	a1	Woman	29	4	5	3	4	7	2	5	5	4

Figure 4: Data set containing response from questionnaire

4.2 Data Transformation

In this section we will discuss the data transformation performed for the research. This includes change in format, structure or logical interpretation of values.

The attributes extracted from eye tracker device had space in the name of columns which was replaced by '_', as shown in fig.5.



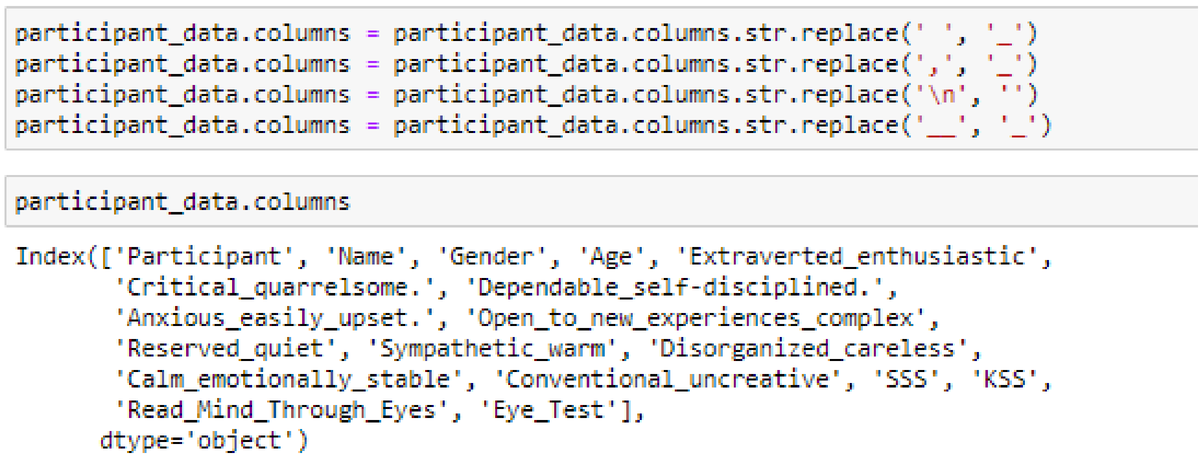
```
event_data.columns = event_data.columns.str.replace(' ', '_')
```

```
event_data
```

	Participant	Visual_Intake_Count	Visual_Intake_Frequency_[count/s]	Visual_Intake_Duration_Total_[ms]	Visual_Intake_Duration_Average_[ms]	Visual_I
0	a1	3221	2.2	1198159.8	372.0	
1	a2	3550	2.5	1256208.3	353.9	
2	a3	3514	2.5	1138210.5	323.9	

Figure 5: Replacing space with '_' in event_data column names

Similarly, the data extracted from the questionnaire, also had multiple special characters in the column names, which was replaced by '_', as shown in fig.6.



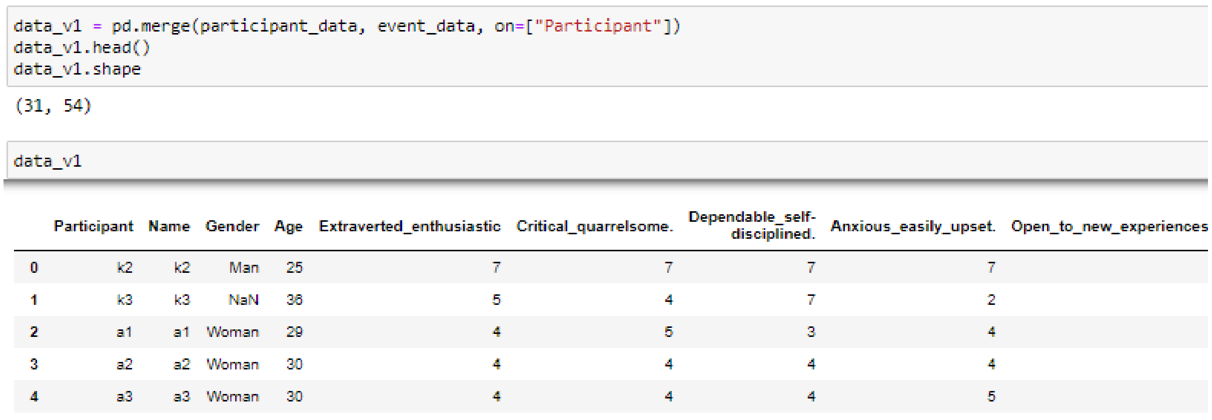
```
participant_data.columns = participant_data.columns.str.replace(' ', '_')
participant_data.columns = participant_data.columns.str.replace(',', '_')
participant_data.columns = participant_data.columns.str.replace('\n', '')
participant_data.columns = participant_data.columns.str.replace('__', '_')
```

```
participant_data.columns
```

```
Index(['Participant', 'Name', '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', 'SSS', 'KSS',
       'Read_Mind_Through_Eyes', 'Eye_Test'],
      dtype='object')
```

Figure 6: Replacing special character with '_' in participant_data column names

The data from both the files were merged to create the final dataset, as shown in fig.7.



```
data_v1 = pd.merge(participant_data, event_data, on=["Participant"])
data_v1.head()
data_v1.shape
```

```
(31, 54)
```

```
data_v1
```

	Participant	Name	Gender	Age	Extraverted_enthusiastic	Critical_quarrelsome.	Dependable_self-disciplined.	Anxious_easily_upset.	Open_to_new_experiences
0	k2	k2	Man	25	7	7	7	7	
1	k3	k3	NaN	36	5	4	7	2	
2	a1	a1	Woman	29	4	5	3	4	
3	a2	a2	Woman	30	4	4	4	4	
4	a3	a3	Woman	30	4	4	4	5	

Figure 7: Merging data

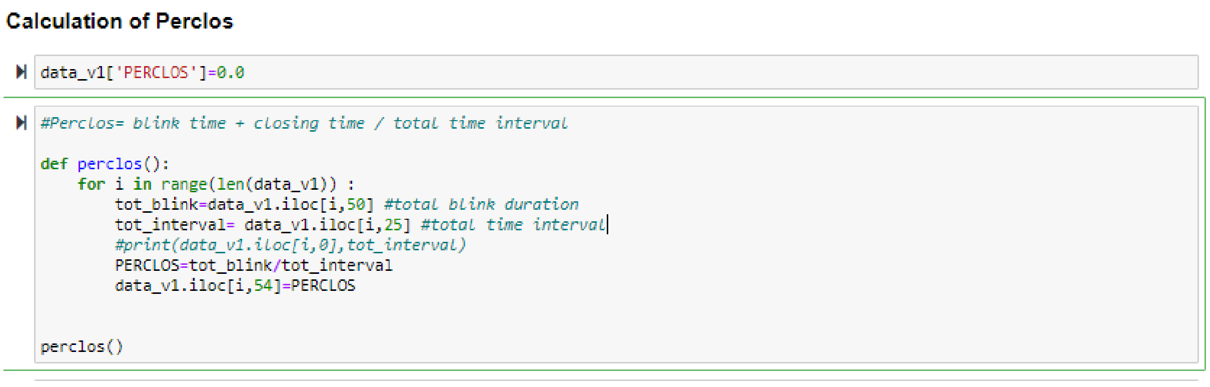
For the research we have calculated PERCLOS, which is the percentage of the total blink duration and total interval, see fig.8.

$$PERCLOS = \frac{blink + CLOS}{interval}$$

Where CLOS is the time interval for the process of blinking of eyes, blink is the time interval for blinking. Interval can be defined as the sum of time for blinking, fixation, saccade and CLOS.

$$interval = blink + fixation + scaccade + CLOS$$

The dataset, contained some attributes in milliseconds, and some attributes in seconds. So to equalise the granularity of these variables, we have transformed the variables with milliseconds to seconds, see fig.9.

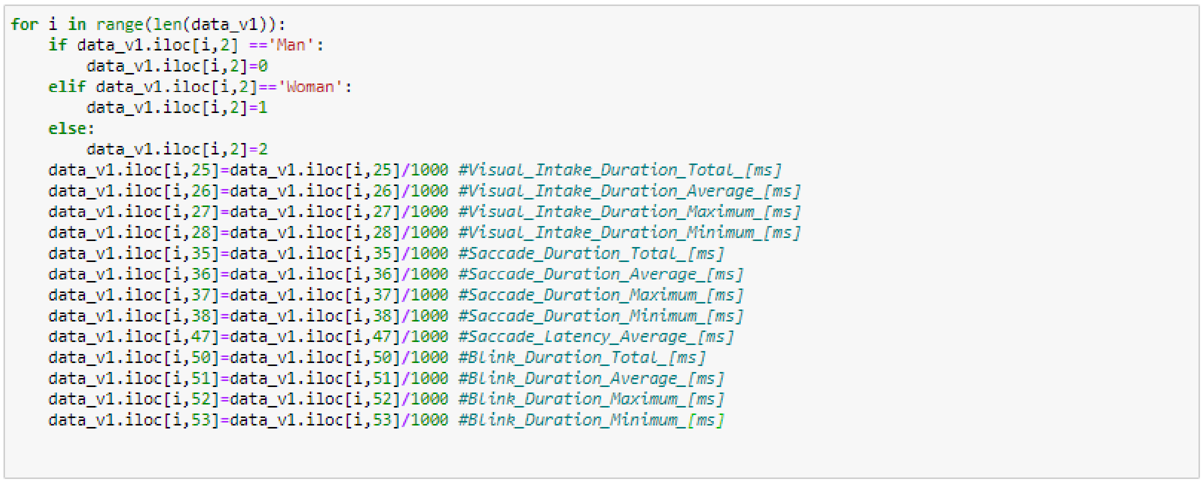


Calculation of Perclos

```
data_v1['PERCLOS']=0.0
```

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

def perclos():
    for i in range(len(data_v1)) :
        tot_blink=data_v1.iloc[i,50] #total blink duration
        tot_interval= data_v1.iloc[i,25] #total time interval
        #print(data_v1.iloc[i,0],tot_interval)
        PERCLOS=tot_blink/tot_interval
        data_v1.iloc[i,54]=PERCLOS


perclos()
```

Figure 8: Calculation of PERCLOS



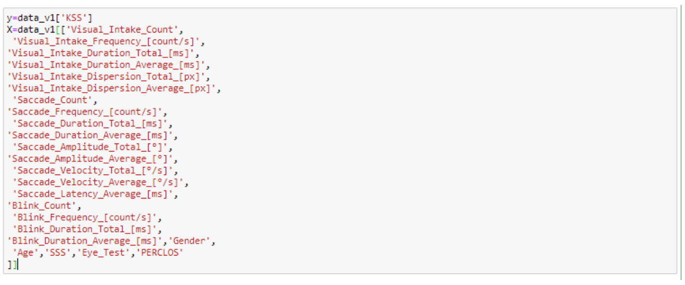
```
for i in range(len(data_v1)):
    if data_v1.iloc[i,2] =='Man':
        data_v1.iloc[i,2]=0
    elif data_v1.iloc[i,2]=='Woman':
        data_v1.iloc[i,2]=1
    else:
        data_v1.iloc[i,2]=2
    data_v1.iloc[i,25]=data_v1.iloc[i,25]/1000 #Visual_Intake_Duration_Total_[ms]
    data_v1.iloc[i,26]=data_v1.iloc[i,26]/1000 #Visual_Intake_Duration_Average_[ms]
    data_v1.iloc[i,27]=data_v1.iloc[i,27]/1000 #Visual_Intake_Duration_Maximum_[ms]
    data_v1.iloc[i,28]=data_v1.iloc[i,28]/1000 #Visual_Intake_Duration_Minimum_[ms]
    data_v1.iloc[i,35]=data_v1.iloc[i,35]/1000 #Saccade_Duration_Total_[ms]
    data_v1.iloc[i,36]=data_v1.iloc[i,36]/1000 #Saccade_Duration_Average_[ms]
    data_v1.iloc[i,37]=data_v1.iloc[i,37]/1000 #Saccade_Duration_Maximum_[ms]
    data_v1.iloc[i,38]=data_v1.iloc[i,38]/1000 #Saccade_Duration_Minimum_[ms]
    data_v1.iloc[i,47]=data_v1.iloc[i,47]/1000 #Saccade_Latency_Average_[ms]
    data_v1.iloc[i,50]=data_v1.iloc[i,50]/1000 #Blink_Duration_Total_[ms]
    data_v1.iloc[i,51]=data_v1.iloc[i,51]/1000 #Blink_Duration_Average_[ms]
    data_v1.iloc[i,52]=data_v1.iloc[i,52]/1000 #Blink_Duration_Maximum_[ms]
    data_v1.iloc[i,53]=data_v1.iloc[i,53]/1000 #Blink_Duration_Minimum_[ms]
```

Figure 9: Data Transformation

In the dataset the gender values are Man, Woman and Null. These values are transformed into 0, 1 and 2 respectively as shown in fig.9.

4.3 Exploratory Data Analysis

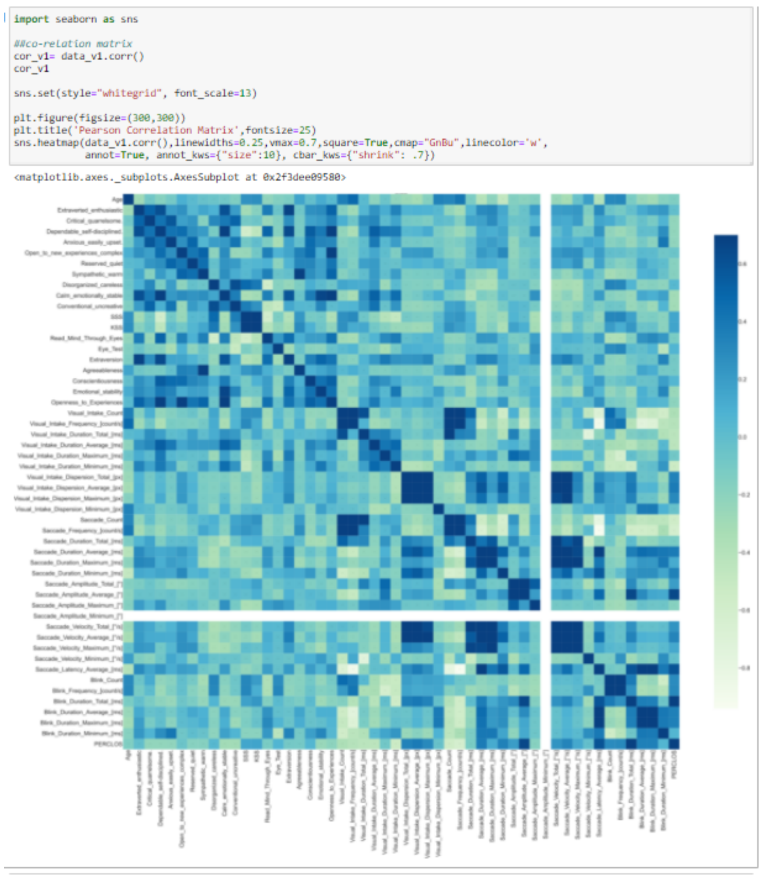
The dataset for research consist of 54 columns, so there might be a possibility that some of the variables in the dataset are correlated. This will impact the performance of the machine learning algorithm, so we have plotted Pearson correlation matrix to check the correlation between the variables, as shown in fig.11.



```
y=data_v1['KSS']
X=data_v1[['Visual_Intake_Count',
 'Visual_Intake_Frequency_[count/s]',
'Visual_Intake_Duration_Total_[ms]',
'Visual_Intake_Duration_Average_[ms]',
'Visual_Intake_Dispersion_Total_[px]',
'Visual_Intake_Dispersion_Average_[px]',
 'Saccade_Count',
'Saccade_Frequency_[count/s]',
 'Saccade_Duration_Total_[ms]',
'Saccade_Duration_Average_[ms]',
 'Saccade_Amplitude_Total_[°]',
'Saccade_Amplitude_Average_[°]',
 'Saccade_Velocity_Total_[°/s]',
 'Saccade_Velocity_Average_[°/s]',
 'Saccade_Latency_Average_[ms]',
'Blink_Count',
 'Blink_Frequency_[count/s]',
 'Blink_Duration_Total_[ms]',
'Blink_Duration_Average_[ms]','Gender',
 'Age','SSS','Eye_Test','PERCLOS'
]]
```

Figure 10: Independent and dependent variable

Feature selection for this research is done after analysing the correlation matrix plotted for the dataset. The details of the final dataset and independent and dependent variables is shown in fig.10



```
import seaborn as sns

##co-relation matrix
cor_v1= data_v1.corr()
cor_v1

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

plt.figure(figsize=(300,300))
plt.title('Pearson Correlation Matrix',fontsize=25)
sns.heatmap(data_v1.corr(),linewidths=0.25,vmax=0.7,square=True,cmap="GnBu",linecolor='w',
            annot=True, annot_kws={"size":10}, cbar_kws={"shrink": .7})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2f3dee09580>
```

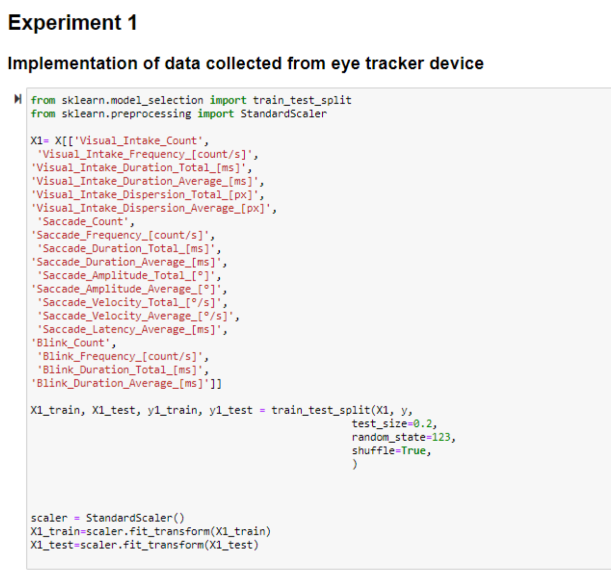
Figure 11: Pearson Correlation Matrix

5 Implementation

The research was conducted with four experiment, in the first experiment the data extracted from eye tracker device was considered, in the second experiment the data extracted from eye tracker device and calculated PERCLOS was considered, in the third experiment the data extracted from eye tracker device and questionnaire was considered and in the fourth experiment the data extracted from eye tracker device, questionnaire and calculated PERCLOS was considered.

5.1 Experiment 1 : Implementation of data collected from eye tracker device

In this experiment we have selected data extracted from eye tracker device. The data Structure and scalar transform for experiment 1, shown in fig.12.



Experiment 1

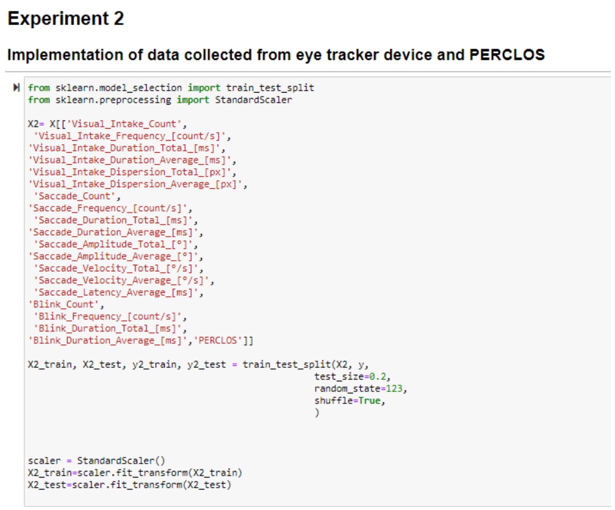
Implementation of data collected from eye tracker device

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

X1= X[['Visual_Intake_Count',
 'Visual_Intake_Frequency_[count/s]',
'Visual_Intake_Duration_Total_[ms]',
'Visual_Intake_Duration_Average_[ms]',
'Visual_Intake_Dispersion_Total_[px]',
'Visual_Intake_Dispersion_Average_[px]',
 'Saccade_Count',
'Saccade_Frequency_[count/s]',
 'Saccade_Duration_Total_[ms]',
'Saccade_Duration_Average_[ms]',
 'Saccade_Amplitude_Total_[°]',
'Saccade_Amplitude_Average_[°]',
 'Saccade_Velocity_Total_[°/s]',
 'Saccade_Velocity_Average_[°/s]',
 'Saccade_Latency_Average_[ms]',
'Blink_Count',
 'Blink_Frequency_[count/s]',
 'Blink_Duration_Total_[ms]',
'Blink_Duration_Average_[ms]']]

X1_train, X1_test, y1_train, y1_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 12: Data Structure and Transform for Experiment 1

5.2 Experiment 2: Implementation of data collected from eye tracker device and PERCLOS

In this experiment we have selected data extracted from eye tracker device and calculated PERCLOS. The data Structure and scalar transform for experiment 2, shown in fig.13 .



Experiment 2

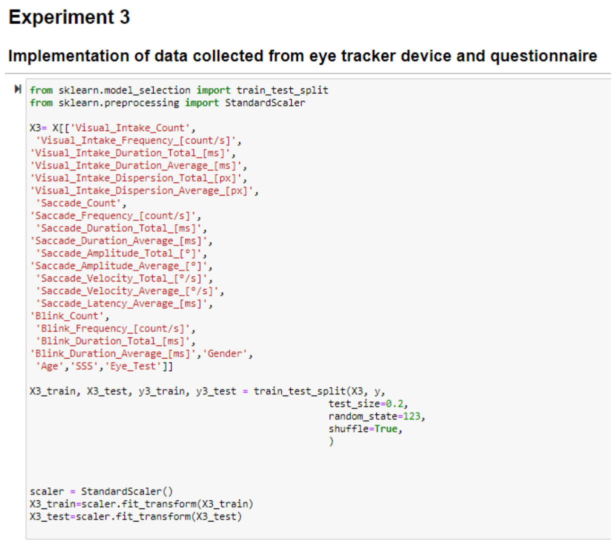
Implementation of data collected from eye tracker device and PERCLOS

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

X2= X[['Visual_Intake_Count',
 'Visual_Intake_Frequency_[count/s]',
'Visual_Intake_Duration_Total_[ms]',
'Visual_Intake_Duration_Average_[ms]',
'Visual_Intake_Dispersion_Total_[px]',
'Visual_Intake_Dispersion_Average_[px]',
 'Saccade_Count',
'Saccade_Frequency_[count/s]',
 'Saccade_Duration_Total_[ms]',
'Saccade_Duration_Average_[ms]',
 'Saccade_Amplitude_Total_[°]',
'Saccade_Amplitude_Average_[°]',
 'Saccade_Velocity_Total_[°/s]',
 'Saccade_Velocity_Average_[°/s]',
 'Saccade_Latency_Average_[ms]',
'Blink_Count',
 'Blink_Frequency_[count/s]',
 'Blink_Duration_Total_[ms]',
'Blink_Duration_Average_[ms]','PERCLOS']]

X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y,
                                                    test_size=0.2,
                                                    random_state=123,
                                                    shuffle=True,
                                                    )



scaler = StandardScaler()
X2_train=scaler.fit_transform(X2_train)
X2_test=scaler.fit_transform(X2_test)
```

Figure 13: Data Structure and Transform for Experiment 2

5.3 Experiment 3: Implementation of data collected from eye tracker device and questionnaire

In this experiment we have selected data extracted from eye tracker device and the response from the questionnaire. The data Structure and scalar transform for experiment 3, shown in fig.14.



Experiment 3

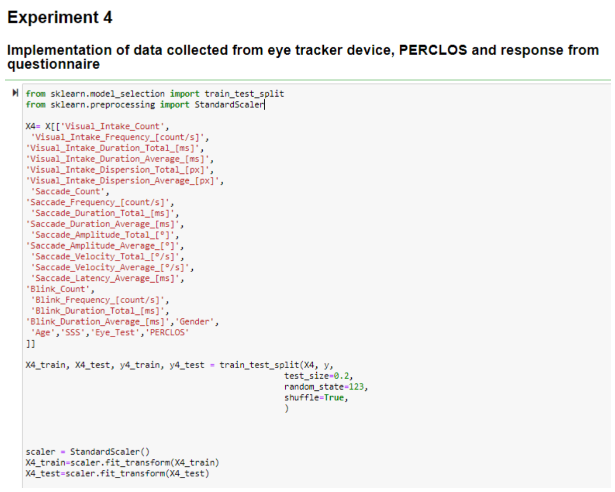
Implementation of data collected from eye tracker device and questionnaire

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

X3= X[['Visual_Intake_Count',
 'Visual_Intake_Frequency_[count/s]',
'Visual_Intake_Duration_Total_[ms]',
'Visual_Intake_Duration_Average_[ms]',
'Visual_Intake_Dispersion_Total_[px]',
'Visual_Intake_Dispersion_Average_[px]',
 'Saccade_Count',
'Saccade_Frequency_[count/s]',
 'Saccade_Duration_Total_[ms]',
'Saccade_Duration_Average_[ms]',
 'Saccade_Amplitude_Total_[°]',
'Saccade_Amplitude_Average_[°]',
 'Saccade_Velocity_Total_[°/s]',
 'Saccade_Velocity_Average_[°/s]',
 'Saccade_Latency_Average_[ms]',
'Blink_Count',
 'Blink_Frequency_[count/s]',
 'Blink_Duration_Total_[ms]',
'Blink_Duration_Average_[ms]','Gender',
 'Age','SSS','Eye_Test']]

X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y,
                                                    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 14: Data Structure and Transform for Experiment 3

5.4 Experiment 4: Implementation of data collected from eye tracker device, PERCLOS and response from questionnaire

In this experiment we have selected data extracted from eye tracker device, the response from the questionnaire and the calculated PERCLOS. The data Structure and scalar transform for experiment 4, shown in fig.15 .



Experiment 4

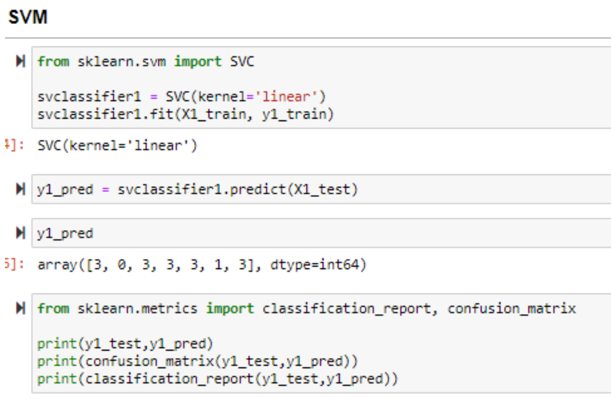
Implementation of data collected from eye tracker device, PERCLOS and response from questionnaire

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

X4= X[['Visual_Intake_Count',
 'Visual_Intake_Frequency_[count/s]',
'Visual_Intake_Duration_Total_[ms]',
'Visual_Intake_Duration_Average_[ms]',
'Visual_Intake_Dispersion_Total_[px]',
'Visual_Intake_Dispersion_Average_[px]',
 'Saccade_Count',
'Saccade_Frequency_[count/s]',
 'Saccade_Duration_Total_[ms]',
'Saccade_Duration_Average_[ms]',
 'Saccade_Amplitude_Total_[°]',
'Saccade_Amplitude_Average_[°]',
 'Saccade_Velocity_Total_[°/s]',
 'Saccade_Velocity_Average_[°/s]',
 'Saccade_Latency_Average_[ms]',
'Blink_Count',
 'Blink_Frequency_[count/s]',
 'Blink_Duration_Total_[ms]',
'Blink_Duration_Average_[ms]','Gender',
 'Age','SSS','Eye_Test','PERCLOS'
]]

X4_train, X4_test, y4_train, y4_test = train_test_split(X4, y,
                                                    test_size=0.2,
                                                    random_state=123,
                                                    shuffle=True,
                                                    )



scaler = StandardScaler()
X4_train=scaler.fit_transform(X4_train)
X4_test=scaler.fit_transform(X4_test)
```

Figure 15: Data Structure and Transform for Experiment 4

5.5 SVM

SVM machine learning algorithm was implemented across all the experiment. The implementation and confusion matrix of SVM for experiments in research is shown in fig.16



```
SVM

from sklearn.svm import SVC

svclassifier1 = SVC(kernel='linear')
svclassifier1.fit(X1_train, y1_train)
```

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

```
y1_pred = svclassifier1.predict(X1_test)
```

```
y1_pred
```

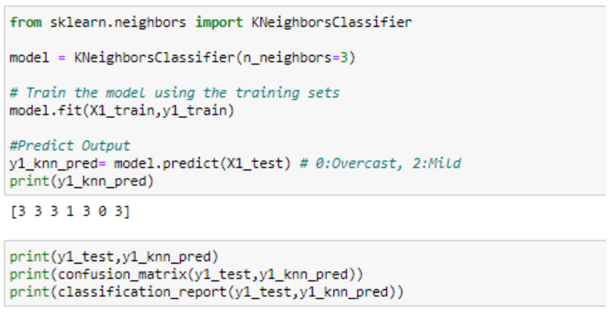
```
]: array([3, 0, 3, 3, 3, 1, 3], dtype=int64)
```

```
from sklearn.metrics import classification_report, confusion_matrix

print(y1_test,y1_pred)
print(confusion_matrix(y1_test,y1_pred))
print(classification_report(y1_test,y1_pred))
```

Figure 16: SVM implementation and confusion matrix for Experiment

5.6 KNN

KNN machine learning algorithm was implemented across all the experiment. The implementation and confusion matrix of KNN for experiments in research is shown in fig.17



```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets
model.fit(X1_train,y1_train)

#Predict Output
y1_knn_pred= model.predict(X1_test) # 0:Overcast, 2:Mild
print(y1_knn_pred)
```

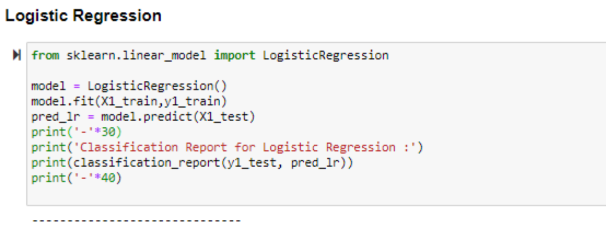
```
[3 3 3 1 3 0 3]
```

```
print(y1_test,y1_knn_pred)
print(confusion_matrix(y1_test,y1_knn_pred))
print(classification_report(y1_test,y1_knn_pred))
```

Figure 17: KNN implementation and confusion matrix for Experiment

5.7 Logistic Regression

Logistic Regression algorithm was implemented across all the experiment. The implementation and confusion matrix of Logistic Regression for experiments in research is shown in fig.18



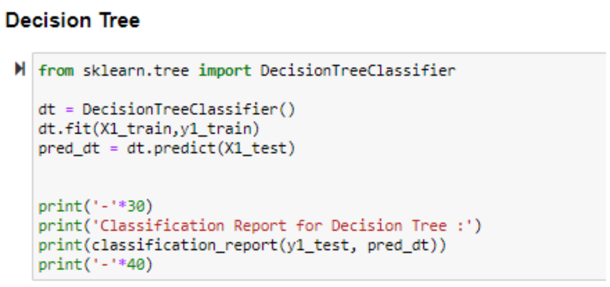
```
Logistic Regression

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X1_train,y1_train)
pred_lr = model.predict(X1_test)
print('-'*30)
print('Classification Report for Logistic Regression :')
print(classification_report(y1_test, pred_lr))
print('-'*40)
```

```
------------------------------
```

Figure 18: Logistic Regression implementation and confusion matrix for Experiment

5.8 Decision Tree

Decision Tree algorithm was implemented across all the experiment. The implementation and confusion matrix of Decision Tree for experiments in research is shown in fig.19



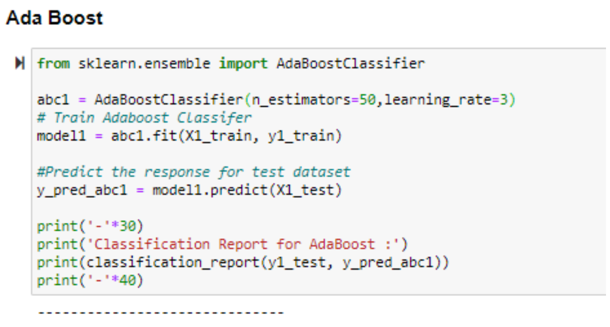
Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(X1_train,y1_train)
pred_dt = dt.predict(X1_test)


print('-'*30)
print('Classification Report for Decision Tree :')
print(classification_report(y1_test, pred_dt))
print('-'*40)
```

Figure 19: Decision Tree implementation and confusion matrix for Experiment

5.9 Ada-Boost

Ada-Boost algorithm was implemented across all the experiment. The implementation and confusion matrix of Ada-Boost for experiments in research is shown in fig.20



Ada Boost

```
from sklearn.ensemble import AdaBoostClassifier

abc1 = AdaBoostClassifier(n_estimators=50,learning_rate=3)
# Train Adaboost Classifer
model1 = abc1.fit(X1_train, y1_train)

#Predict the response for test dataset
y_pred_abc1 = model1.predict(X1_test)

print('-'*30)
print('Classification Report for AdaBoost :')
print(classification_report(y1_test, y_pred_abc1))
print('-'*40)
------------------------------
```

Figure 20: Ada-Boost implementation and confusion matrix for Experiment

6 Conclusion

In conclusion, the information mentioned in this report, explains the complete step by step implementation of the research. The report is divided into sections and has been explained in detail and sequential form. The entire code has been published in the github repository[2].

[2]https://github.com/anjuli22/Thesis_MS_19242581