Configuration Manual

MSc Research Project
Data Analytics

Komal Riddhish Bharadva
Student ID: x19213051

School of Computing
National College of Ireland

Supervisor: Paul Stynes, Anu Sahni, and Pramod Pathak

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Komal Riddhish Bharadva………………………………………………………………		
Student ID:	x19213051………………………………………………………………………………		
Programme:	Data Analytics…………………………………………	**Year:**	2020-21………..
Module:	MSc Research Project……………………………………………………………………		
Supervisor:	Paul Stynes, Anu Sahni, and Pramod Pathak…………………………………		
Submission Due Date:	16/08/2021………………………………………………………………………………		
Project Title:	A Machine Learning framework to Detect Student's Online Engagement		
Word Count:	1678……………………………	**Page Count**	14……………………………………

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: Komal Riddhish Bharadva…………………………………………………………………

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.	□

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):	

Configuration Manual

A Machine Learning Framework to Detect Student's Online Engagement

Komal Riddhish Bharadva
x19213051

1 Introduction

This document gives a brief overview of all the steps undertaken in implementing this research project. The aim of this project is to detect student's engagement in an online environment using novel combination of machine learning model and eye-tracker device. The performance of machine learning model was evaluated, and the best model was selected for detection. The tools, techniques and libraries used is included in the other sections of this document.

This document is divided into following sections: the hardware specifications required for the implementation of this project is listed in Section 2 followed by software specifications required in Section 3. The environment setup for installing required tools is described in Section 4. The data gathering which is a crucial step in this project is illustrated in Section 5. The various steps involved in implementation of this project is shown in Section 6. Finally, the Section 7 concludes this document.

2 Hardware Specifications

The configuration of the system on which this project has been implemented is as mentioned below:

- Operating System: Windows 10 Home
- Hard Drive: 1024 GB HDD
- RAM: 8.0 GB
- Processor: Intel i5-10210U

3 Software Specifications

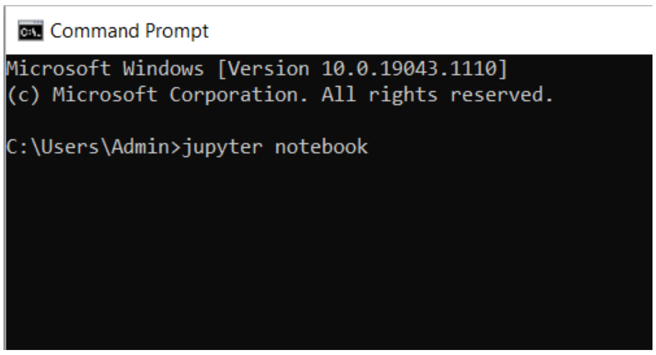
For the implementation of this project, Jupyter notebook has been used as the Integrated Development Environment (IDE) and the programming language used is Python. The visualizations have been performed using seaborn and matplotlib libraries. Below are the specific versions of these software.

- Python: 3.9.0
- Jupyter Notebook: 6.0.3

4 Environment Setup

This section includes the steps and guidelines that needs to be followed in order to setup the required tools, and software.

- The latest version of python can be downloaded and installed using following link[1].
- The jupyter notebook can be installed with the help of steps provided on the following link[2]. To start the jupyter notebook, below command as shown in **Fig. 1** needs to be executed in command prompt.



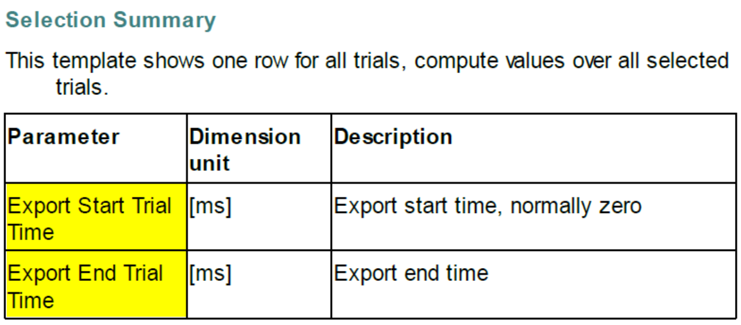
```
Microsoft Windows [Version 10.0.19043.1110]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>jupyter notebook
```

Fig. 1. Starting Jupyter notebook

- Various other required packages and libraries can be directly installed within Jupyter notebook using below command.
  $ *pip install <package_name>*

5 Data Gathering

The dataset has been prepared using data from an eye-tracker device, post-questionnaires, and calculating face time. The data from eye-tracker device consist of 38 features out of which only 8 variables were used for the analysis[3]. The **Fig. 2** below shows brief description of these 38 variables.



Selection Summary

This template shows one row for all trials, compute values over all selected trials.

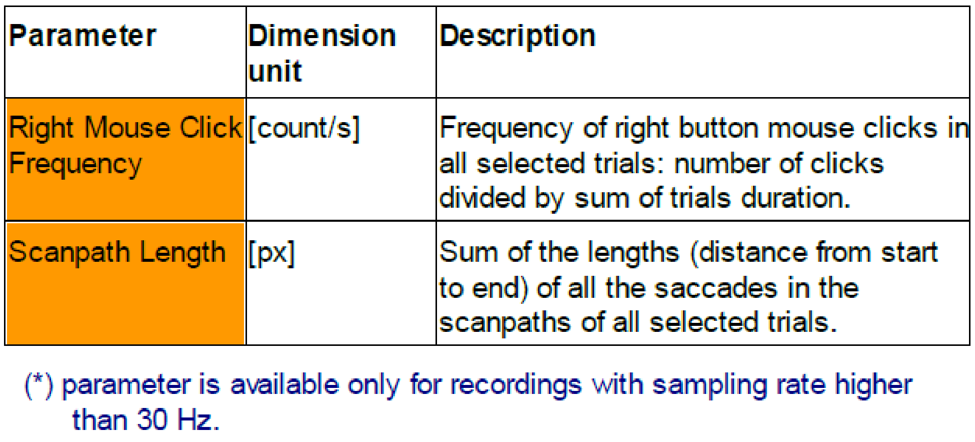
Parameter	Dimension unit	Description
Export Start Trial Time	[ms]	Export start time, normally zero
Export End Trial Time	[ms]	Export end time

[1] https://www.python.org/downloads/release/python-390
[2] https://jupyter.org/install
[3] https://psychologie.unibas.ch/fileadmin/user_upload/psychologie/Forschung/N-Lab/SMI_BeGaze_Manual.pdf

Parameter	**Dimension unit**	**Description**
Fixation Count		Number of fixations of all selected trials.
Fixation Frequency	[count/s]	Number of fixations of all selected trials divided by the sum of trial durations of all selected trials in seconds.
Fixation Duration Total	[ms]	Sum of duration of all fixations of all selected trials.
Fixation Duration Average	[ms]	Sum of duration of all fixations of all selected trials divided by the number of fixations of all selected trials.
Fixation Duration Maximum	[ms]	Longest fixation duration of all selected trials.
Fixation Duration Minimum	[ms]	Shortest fixation duration of all selected trials.
Fixation Dispersion Total	[px]	Sum of all fixation dispersions on X and Y of all selected trials.
Fixation Dispersion Average	[px]	Sum of dispersion of all fixations of all selected trials divided by the number of fixations of all selected trials.
Fixation Dispersion Maximum	[px]	Largest value for the sum of X and Y dispersions of fixation of all selected trials.
Fixation Dispersion Minimum	[px]	Smallest value for the sum of X and Y dispersions of fixation of all selected trials.
Saccade Count		Number of saccades of all selected trials.
Saccade Frequency	[count/s]	Number of saccades of all selected trials divided by the sum of trial durations of all selected trials in seconds.

Parameter	**Dimension unit**	**Description**
Saccade Duration Total	[ms]	Sum of all saccade duration of all selected trials.
Saccade Duration Average	[ms]	Sum of all saccade duration of all selected trials divided by the number of saccades of all selected trials.
Saccade Duration Maximum	[ms]	Longest saccade duration of all selected trials.
Saccade Duration Minimum	[ms]	Shortest saccade duration of all selected trials.
Saccade Amplitude Total	[°]	Sum of all saccades amplitude of all selected trials.
Saccade Amplitude Average	[°]	Sum of all saccades amplitude of all selected trials divided by the number of saccades of all selected trials.
Saccade Amplitude Maximum	[°]	Max. saccade amplitude of all selected trials.
Saccade Amplitude Minimum	[°]	Min. saccade amplitude of all selected trials.
Saccade Velocity Total	[°/s]	Sum of all saccades velocities of all selected trials. (*)
Saccade Velocity Average	[°/s]	Sum of all saccades velocities of all selected trials divided by the number of saccades of all selected trials. (*)
Saccade Velocity Maximum	[°/s]	Max. value of the saccade velocity of all selected trials. (*)

Parameter	Dimension unit	Description
Saccade Velocity Minimum	[°/s]	Min. value of the saccade velocity of all selected trials. (*)
Saccade Latency Average	[°/s]	saccade latency = time between the end of a saccade and the start of the next saccade. Saccade latency average = total saccade latency for all saccades / saccade count in all selected trials
Blink Count		Number of blinks of all selected trials.
Blink Frequency	[count/s]	Number of blinks of all selected trials divided by the sum of trial durations of all selected trials in seconds.
Blink Duration Total	[ms]	Sum of duration of all blinks of all selected trials.
Blink Duration Average	[ms]	Sum of duration of all blinks of all selected trials divided by the number of blinks of all selected trials.
Blink Duration Maximum	[ms]	Longest blink duration of all selected trials.
Blink Duration Minimum	[ms]	Shortest blink duration of all selected trials.
Left Mouse Click Count		Number of left button mouse clicks in all selected trials.
Left Mouse Click Frequency	[count/s]	Frequency of left button mouse clicks in all selected trials: number of clicks divided by sum of trial duration.
Right Mouse Click Count		Number of right button mouse clicks in all selected trials.



Parameter	Dimension unit	Description
Right Mouse Click Frequency	[count/s]	Frequency of right button mouse clicks in all selected trials: number of clicks divided by sum of trials duration.
Scanpath Length	[px]	Sum of the lengths (distance from start to end) of all the saccades in the scanpaths of all selected trials.

(*) parameter is available only for recordings with sampling rate higher than 30 Hz.

Fig. 2. Eye tracker data

Apart from data received from eye tracker, some other variables were also used for the analysis as shown below in **Table 1**.

Parameter	**Units**	**Description**
Video Test	Score out of 12	Test based on video shown
Face Time	Minutes	Time for which student looked at instructor’s face
Mind Test	Score out of 36	Reading the mind from the eyes test for cognitive process evaluation
Extraversion	Ranging between 0 to 7	Personality questionnaire variable 1
Agreeableness	Ranging between 0 to 7	Personality questionnaire variable 2
Conscientiousness	Ranging between 0 to 7	Personality questionnaire variable 3
Emotional stability	Ranging between 0 to 7	Personality questionnaire variable 4
Openness to Experiences	Ranging between 0 to 7	Personality questionnaire variable 5
Target	Either 1 or 0	Indicates whether a student is engaged or not.

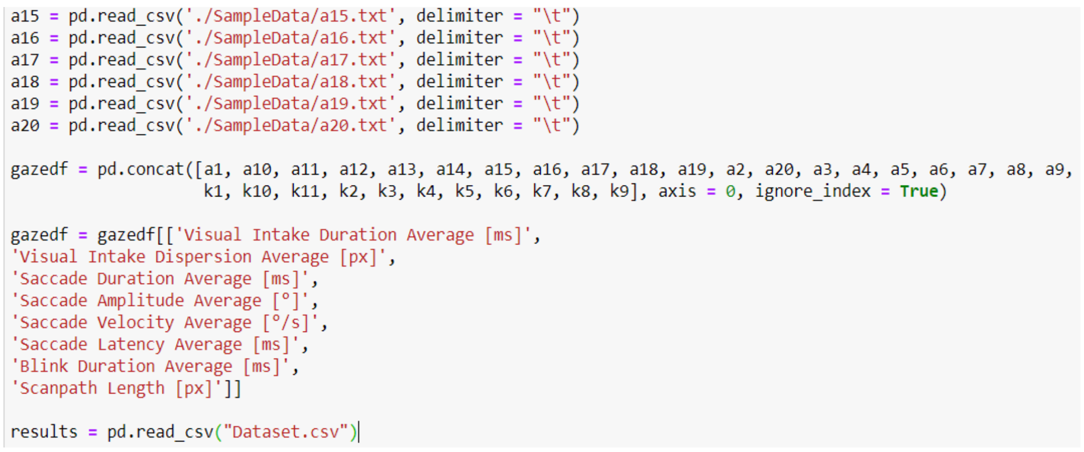
Table 1: Other variables used

6 Implementation

This section describes the various steps involved in implementation of this research project.

6.1 Data Preparation:

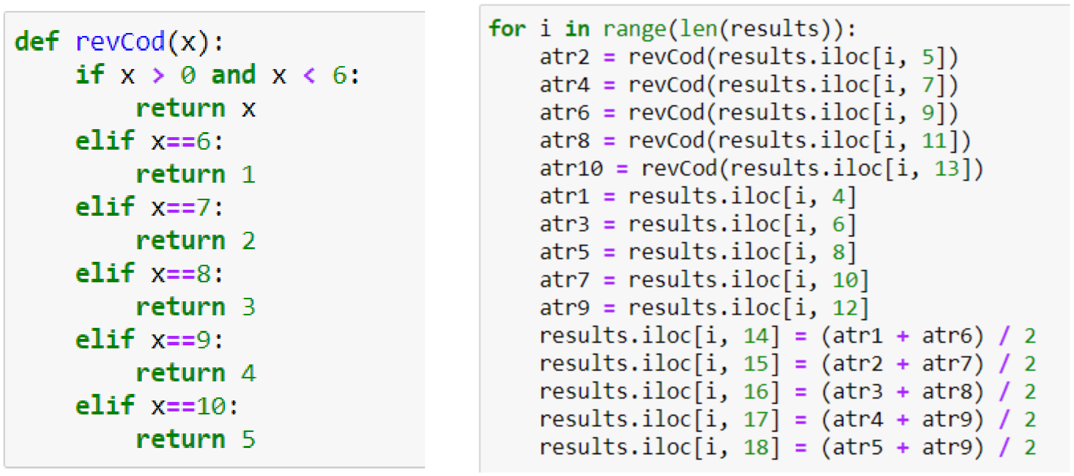
The data obtained from eye-tracker device is in text format for each student. All the data for each student has been stored in dataframe and then concatenated. Finally, a subset of required features are considered for further analysis as shown in **Fig 3** below. A dataset.csv file is also read in results dataframe that contains post questionnaire results and other variables considered.



```
a15 = pd.read_csv('./SampleData/a15.txt', delimiter = "\t")
a16 = pd.read_csv('./SampleData/a16.txt', delimiter = "\t")
a17 = pd.read_csv('./SampleData/a17.txt', delimiter = "\t")
a18 = pd.read_csv('./SampleData/a18.txt', delimiter = "\t")
a19 = pd.read_csv('./SampleData/a19.txt', delimiter = "\t")
a20 = pd.read_csv('./SampleData/a20.txt', delimiter = "\t")

gazedf = pd.concat([a1, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a2, a20, a3, a4, a5, a6, a7, a8, a9,
                    k1, k10, k11, k2, k3, k4, k5, k6, k7, k8, k9], axis = 0, ignore_index = True)

gazedf = gazedf[['Visual Intake Duration Average [ms]',
'Visual Intake Dispersion Average [px]',
'Saccade Duration Average [ms]',
'Saccade Amplitude Average [°]',
'Saccade Velocity Average [°/s]',
'Saccade Latency Average [ms]',
'Blink Duration Average [ms]',
'Scanpath Length [px]']]

results = pd.read_csv("Dataset.csv")
```

Fig. 3. Data Preparation

This is followed by preparation of data obtained from post questionnaires and face time variable. The personality questionnaire given to students consist of Ten item personality measure (TIPI) scale which uses reverse scoring[4]. Out of these 10 personality traits, 5 variables are obtained using reverse scoring as per the TIPI scale. This has been implemented in python as shown in **Fig. 4** below.

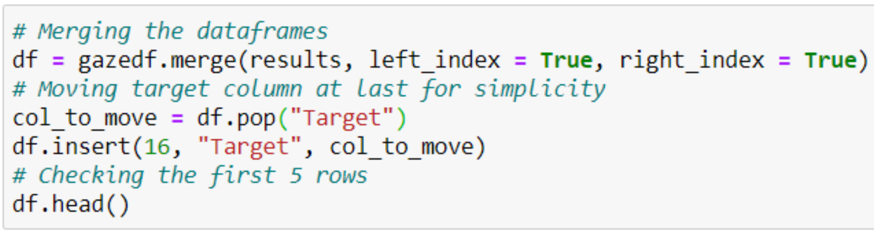


```
def revCod(x):
    if x > 0 and x < 6:
        return x
    elif x==6:
        return 1
    elif x==7:
        return 2
    elif x==8:
        return 3
    elif x==9:
        return 4
    elif x==10:
        return 5
```

```
for i in range(len(results)):
    atr2 = revCod(results.iloc[i, 5])
    atr4 = revCod(results.iloc[i, 7])
    atr6 = revCod(results.iloc[i, 9])
    atr8 = revCod(results.iloc[i, 11])
    atr10 = revCod(results.iloc[i, 13])
    atr1 = results.iloc[i, 4]
    atr3 = results.iloc[i, 6]
    atr5 = results.iloc[i, 8]
    atr7 = results.iloc[i, 10]
    atr9 = results.iloc[i, 12]
    results.iloc[i, 14] = (atr1 + atr6) / 2
    results.iloc[i, 15] = (atr2 + atr7) / 2
    results.iloc[i, 16] = (atr3 + atr8) / 2
    results.iloc[i, 17] = (atr4 + atr9) / 2
    results.iloc[i, 18] = (atr5 + atr9) / 2
```

Fig. 4. TIPI reverse scoring

Finally, both these dataframe: gazedf and results are merged from which a final dataframe is obtained as shown in **Fig. 5** below. The face time variable is calculated manually looking at the video obtained from eye-tracker device having eye movements of the students.



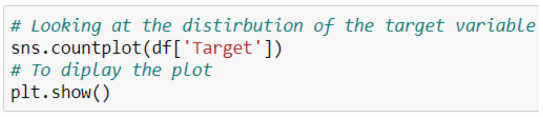
```
# Merging the dataframes
df = gazedf.merge(results, left_index = True, right_index = True)
# Moving target column at last for simplicity
col_to_move = df.pop("Target")
df.insert(16, "Target", col_to_move)
# Checking the first 5 rows
df.head()
```

Fig. 5. Final merged dataset

[4] https://gosling.psy.utexas.edu/scales-weve-developed/ten-item-personality-measure-tipi/

6.2 Exploratory Data Analysis (EDA):

Before starting with the data cleaning and data pre-processing steps, it is important to understand the distribution, type, and behaviour of the dataset. To check the distribution of target variable, a bar graph was plotted using the seaborn package in python as shown in **Fig. 6** below.



```
# Looking at the distribution of the target variable
sns.countplot(df['Target'])
# To diplay the plot
plt.show()
```

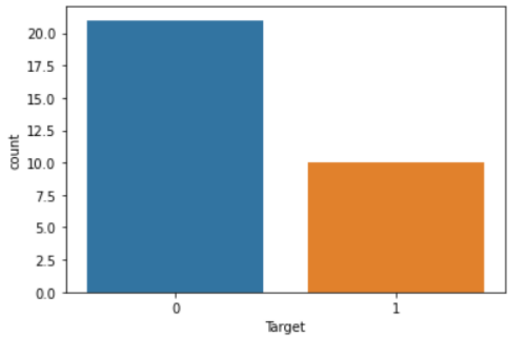
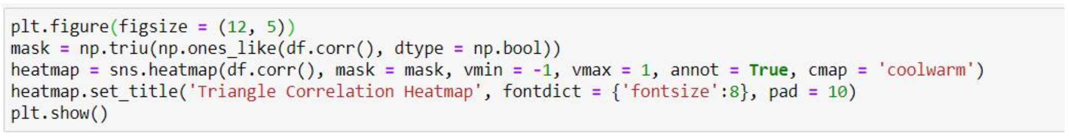
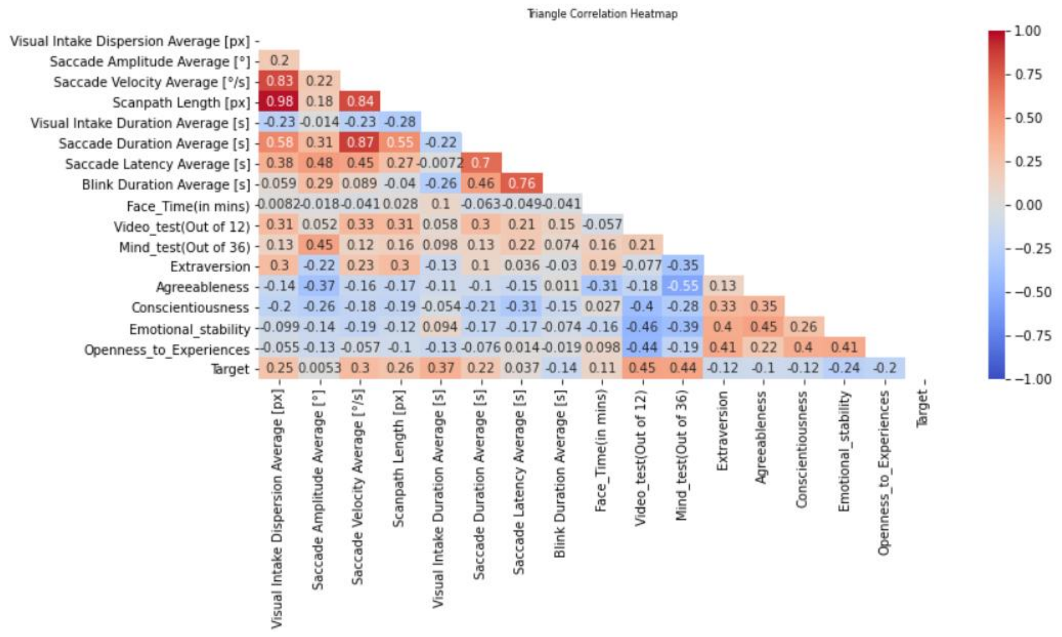


Fig. 6. Bar plot of Target variable

As there are large number of variables considered in the final dataset, there is a possibility that there is a correaltion between variables. In order to verify that, below correlation plot was done with the help of seaborn package in python. The **Fig. 7** shows the correlation matrix plot between all the variables included in the dataset.



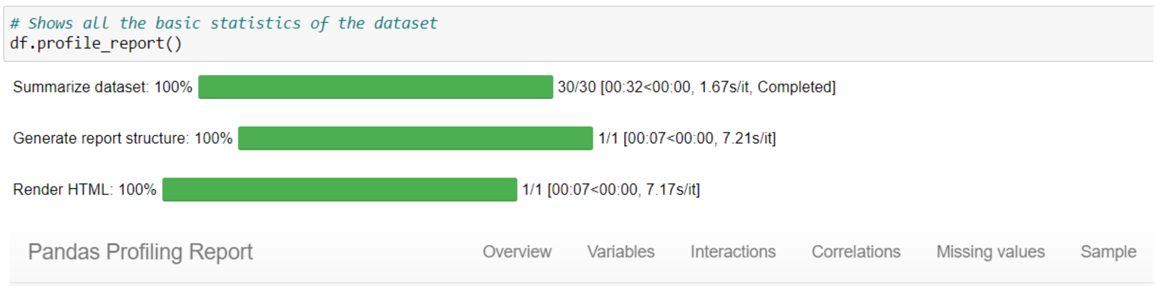
```
plt.figure(figsize = (12, 5))
mask = np.triu(np.ones_like(df.corr(), dtype = np.bool))
heatmap = sns.heatmap(df.corr(), mask = mask, vmin = -1, vmax = 1, annot = True, cmap = 'coolwarm')
heatmap.set_title('Triangle Correlation Heatmap', fontdict = {'fontsize':8}, pad = 10)
plt.show()
```



	Visual Intake Dispersion Average [px]	Saccade Amplitude Average [°]	Saccade Velocity Average [°/s]	Scanpath Length [px]	Visual Intake Duration Average [s]	Saccade Duration Average [s]	Saccade Latency Average [s]	Blink Duration Average [s]	Face_Time(in mins)	Video_test(Out of 12)	Mind_test(Out of 36)	Extraversion	Agreeableness	Conscientiousness	Emotional_stability	Openness_to_Experiences	Target
Visual Intake Dispersion Average [px]																	
Saccade Amplitude Average [°]	0.2																
Saccade Velocity Average [°/s]	0.83	0.22															
Scanpath Length [px]	0.98	0.18	0.84														
Visual Intake Duration Average [s]	-0.23	-0.014	-0.23	-0.28													
Saccade Duration Average [s]	0.58	0.31	0.87	0.55	-0.22												
Saccade Latency Average [s]	0.38	0.48	0.45	0.27	-0.0072	0.7											
Blink Duration Average [s]	-0.059	0.29	0.089	-0.04	-0.26	0.46	0.76										
Face_Time(in mins)	-0.0082	-0.018	-0.041	0.028	0.1	-0.063	-0.049	-0.041									
Video_test(Out of 12)	0.31	0.052	0.33	0.31	0.058	0.3	0.21	0.15	-0.057								
Mind_test(Out of 36)	0.13	0.45	0.12	0.16	0.098	0.13	0.22	0.074	0.16	0.21							
Extraversion	0.3	-0.22	0.23	0.3	-0.13	0.1	0.036	-0.03	0.19	-0.077	-0.35						
Agreeableness	-0.14	-0.37	-0.16	-0.17	-0.11	-0.1	-0.15	0.011	-0.31	-0.18	-0.55	0.13					
Conscientiousness	-0.2	-0.26	-0.18	-0.19	-0.054	-0.21	-0.31	-0.15	0.027	-0.4	-0.28	0.33	0.35				
Emotional_stability	-0.099	-0.14	-0.19	-0.12	0.094	-0.17	-0.17	-0.074	-0.16	-0.46	-0.39	0.4	0.45	0.26			
Openness_to_Experiences	-0.055	-0.13	-0.057	-0.1	-0.13	-0.076	0.014	-0.019	0.098	-0.44	-0.19	0.41	0.22	0.4	0.41		
Target	0.25	0.0053	0.3	0.26	0.37	0.22	0.037	-0.14	0.11	0.45	0.44	-0.12	-0.1	-0.12	-0.24	-0.2	

Fig. 7. Correlation Matrix Plot

Moreover, pandas profiling package was used to get the basic statistics of the dataset. It summarizes each variable and also indicates if there is any missing value as shown in Fig. 8 below.



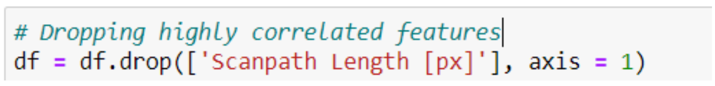
```
# Shows all the basic statistics of the dataset
df.profile_report()
```

```
Summarize dataset: 100%  30/30 [00:32<00:00, 1.67s/it, Completed]

Generate report structure: 100%  1/1 [00:07<00:00, 7.21s/it]

Render HTML: 100%  1/1 [00:07<00:00, 7.17s/it]
```

Fig. 8. Pandas Profiling

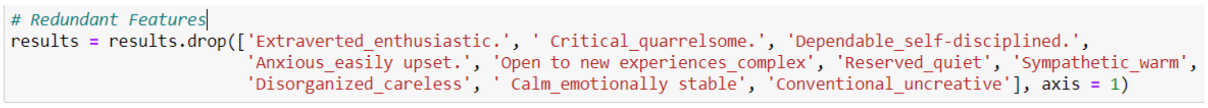
6.3 Data Pre-processing and Transformation:

Before applying machine learning models to the dataset, data should be cleaned and transformed. The variables which are highly correlated (p-value greater than 0.90) with each other are dropped as shown in **Fig 9**. Also, any redundant features are dropped as shown in **Fig. 10** below. These are the 10 original personality traits parameters obtained which are already converted into 5 new columns.



```
# Dropping highly correlated features
df = df.drop(['Scanpath Length [px]'], axis = 1)
```

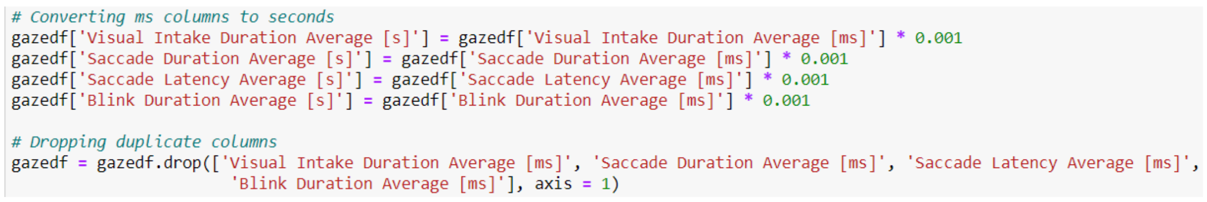
Fig. 9. Dropping highly correlated features



```
# Redundant Features
results = results.drop(['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'], axis = 1)
```

Fig. 10. Dropping redundant features

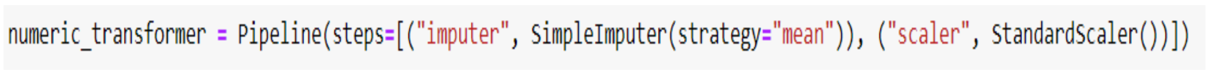
Some columns were in milliseconds, and some were in seconds. Therefore, all the milliseconds' columns are converted into seconds and redundant columns are dropped as shown in **Fig. 11** below.



```
# Converting ms columns to seconds
gazedf['Visual Intake Duration Average [s]'] = gazedf['Visual Intake Duration Average [ms]'] * 0.001
gazedf['Saccade Duration Average [s]'] = gazedf['Saccade Duration Average [ms]'] * 0.001
gazedf['Saccade Latency Average [s]'] = gazedf['Saccade Latency Average [ms]'] * 0.001
gazedf['Blink Duration Average [s]'] = gazedf['Blink Duration Average [ms]'] * 0.001

# Dropping duplicate columns
gazedf = gazedf.drop(['Visual Intake Duration Average [ms]', 'Saccade Duration Average [ms]', 'Saccade Latency Average [ms]',
                      'Blink Duration Average [ms]'], axis = 1)
```

Fig. 11. Converting milliseconds column to seconds

Finally, all the data are converted to a common scale using Standard Scaler library in python. A helper function is defined that uses a pipeline which includes a Simple Imputer and Standard Scaler as shown in **Fig. 12** below.

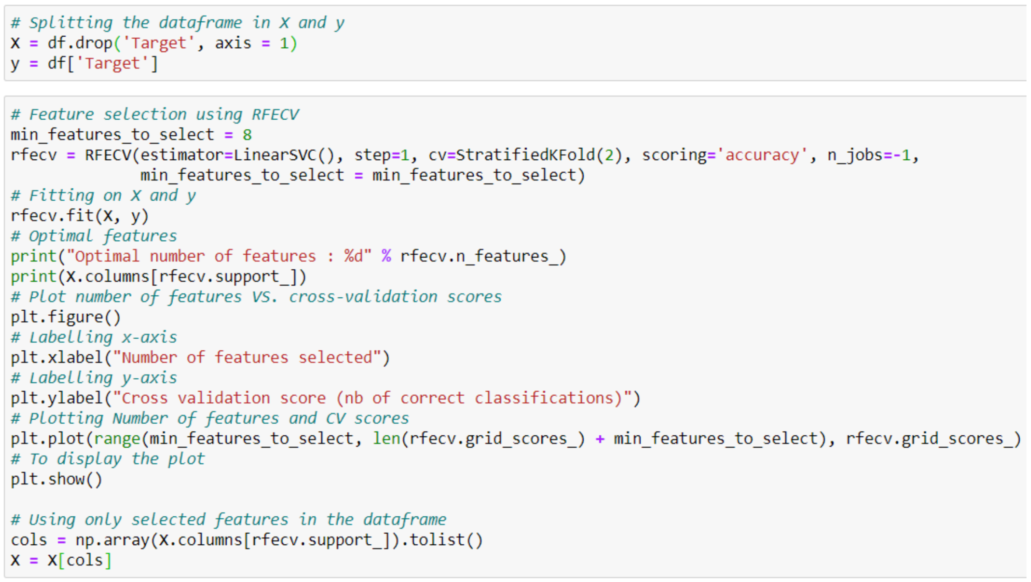


```
numeric_transformer = Pipeline(steps=[("imputer", SimpleImputer(strategy="mean")), ("scaler", StandardScaler())])
```

Fig. 12. Pipeline

6.4 Feature Selection:

The feature selection is an important step which reduces the number of input variables by selecting the most relevant features contributing for good model. Recursive Feature Elimination and Cross-Validation Selection (RFECV) technique is used which removes irrelevant features based on validation scores. The implementation of this is as shown in **Fig. 13** below.



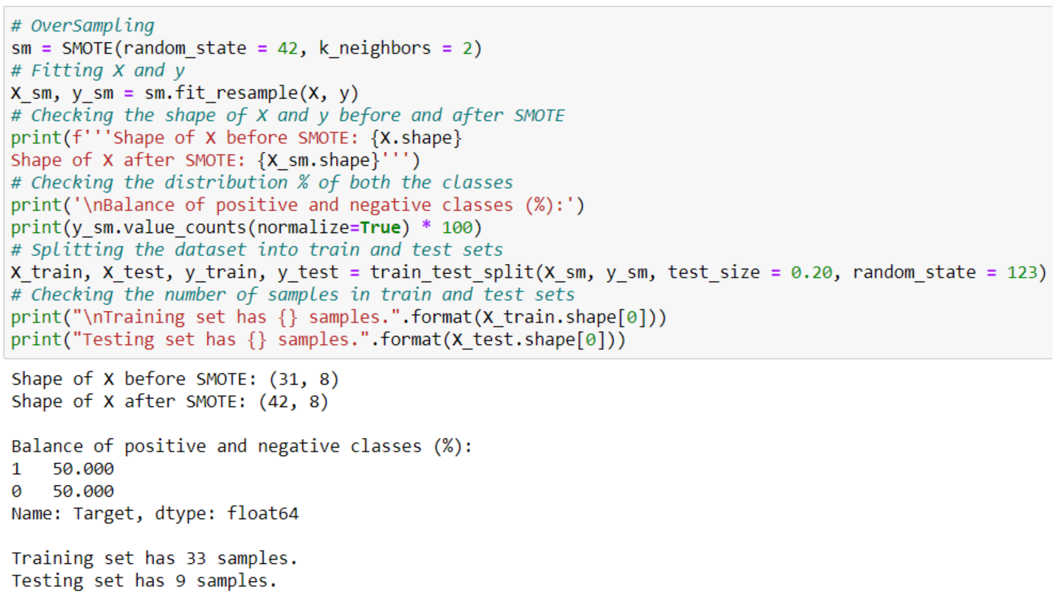
```
# Splitting the dataframe in X and y
X = df.drop('Target', axis = 1)
y = df['Target']
```

```
# Feature selection using RFECV
min_features_to_select = 8
rfecv = RFECV(estimator=LinearSVC(), step=1, cv=StratifiedKFold(2), scoring='accuracy', n_jobs=-1,
              min_features_to_select = min_features_to_select)
# Fitting on X and y
rfecv.fit(X, y)
# Optimal features
print("Optimal number of features : %d" % rfecv.n_features_)
print(X.columns[rfecv.support_])
# Plot number of features VS. cross-validation scores
plt.figure()
# Labelling x-axis
plt.xlabel("Number of features selected")
# Labelling y-axis
plt.ylabel("Cross validation score (nb of correct classifications)")
# Plotting Number of features and CV scores
plt.plot(range(min_features_to_select, len(rfecv.grid_scores_) + min_features_to_select), rfecv.grid_scores_)
# To display the plot
plt.show()

# Using only selected features in the dataframe
cols = np.array(X.columns[rfecv.support_]).tolist()
X = X[cols]
```

Fig. 13. Feature Selection Code

6.5 Sampling Technique:

Once feature selection is done, the next step is sampling of the dataset. Since the target variable is highly imbalanced, it may lead to overfitting. Synthetic Minority Oversampling Technique (SMOTE) which overcomes the overfitting problem by random oversampling of the minority class is used. Below **Fig. 14** shows the implementation of the same.



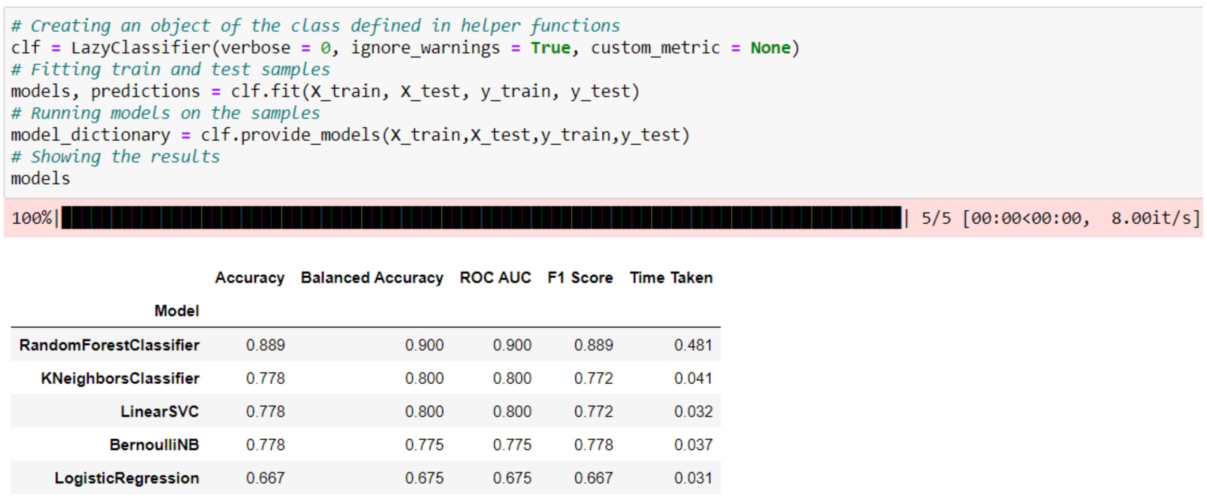
```
# OverSampling
sm = SMOTE(random_state = 42, k_neighbors = 2)
# Fitting X and y
X_sm, y_sm = sm.fit_resample(X, y)
# Checking the shape of X and y before and after SMOTE
print(f'''Shape of X before SMOTE: {X.shape}
Shape of X after SMOTE: {X_sm.shape}''')
# Checking the distribution % of both the classes
print('\nBalance of positive and negative classes (%):')
print(y_sm.value_counts(normalize=True) * 100)
# Splitting the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_sm, y_sm, test_size = 0.20, random_state = 123)
# Checking the number of samples in train and test sets
print("\nTraining set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

```
Shape of X before SMOTE: (31, 8)
Shape of X after SMOTE: (42, 8)

Balance of positive and negative classes (%):
1    50.000
0    50.000
Name: Target, dtype: float64

Training set has 33 samples.
Testing set has 9 samples.
```

Fig. 14. Oversampling of the dataset

6.6 Data Modelling:

Five different machine learning models were applied like Logistic Regression, Random Forest Classifier, Linear SVC, Bernoulli Naïve Bayes, and K-Neighbors Classifier. A helper class named ModLazyClassifier was defined for performing classification. This is a modified version of predefined package available named LazyClassifier. The implementation of this class is as shown in **Fig. 15**. The machine learning models were then evaluated using different evaluation metrics such as accuracy, balanced accuracy, AUC-ROC score, and F1-score.



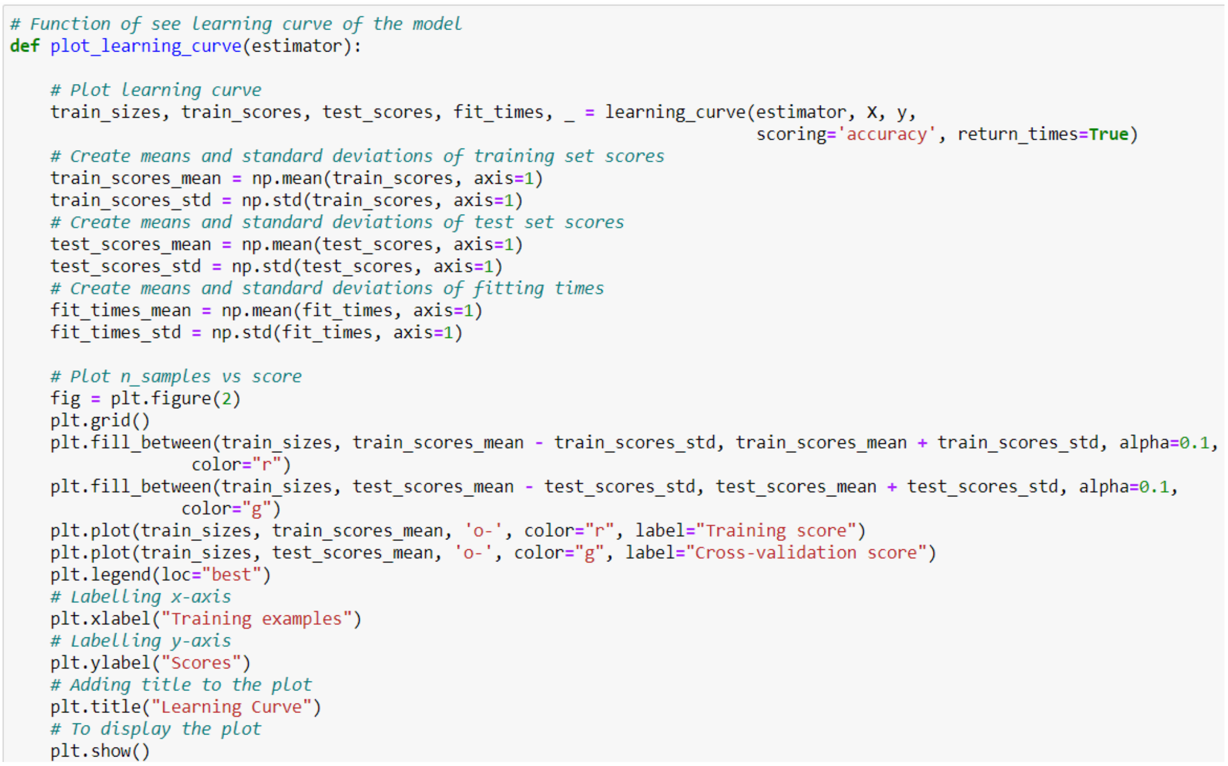
```
# Creating an object of the class defined in helper functions
clf = LazyClassifier(verbose = 0, ignore_warnings = True, custom_metric = None)
# Fitting train and test samples
models, predictions = clf.fit(X_train, X_test, y_train, y_test)
# Running models on the samples
model_dictionary = clf.provide_models(X_train,X_test,y_train,y_test)
# Showing the results
models
```

```
100%|██████████████████████████████████████████████████| 5/5 [00:00<00:00,  8.00it/s]
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
RandomForestClassifier	0.889	0.900	0.900	0.889	0.481
KNeighborsClassifier	0.778	0.800	0.800	0.772	0.041
LinearSVC	0.778	0.800	0.800	0.772	0.032
BernoulliNB	0.778	0.775	0.775	0.778	0.037
LogisticRegression	0.667	0.675	0.675	0.667	0.031

Fig. 15. Modelling of the dataset

The learning curve method was also defined which track the learning of the machine learning models. This shows how efficiently the model is able to learn from the training data provided. The **Fig. 16** shows the implementation of this learning curve method.



```python
# Function of see learning curve of the model
def plot_learning_curve(estimator):

    # Plot learning curve
    train_sizes, train_scores, test_scores, fit_times, _ = learning_curve(estimator, X, y,
                                                                          scoring='accuracy', return_times=True)
    # Create means and standard deviations of training set scores
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    # Create means and standard deviations of test set scores
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    # Create means and standard deviations of fitting times
    fit_times_mean = np.mean(fit_times, axis=1)
    fit_times_std = np.std(fit_times, axis=1)

    # Plot n_samples vs score
    fig = plt.figure(2)
    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, alpha=0.1,
                     color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")
    plt.legend(loc="best")
    # Labelling x-axis
    plt.xlabel("Training examples")
    # Labelling y-axis
    plt.ylabel("Scores")
    # Adding title to the plot
    plt.title("Learning Curve")
    # To display the plot
    plt.show()
```

Fig. 16. Learning curve of the model

7 Conclusion

All the steps involved in the implementation of this project are briefly described in this document. The tools, techniques, and software requirement are also mentioned in the respective section. The environment setup is quite easy and can be implemented step by step with the help of links cited. The data gathering for this project is a bit challenging, but each step taken is precisely documented in Section 5. Moreover, the packages, and libraries required for several data analysis phases are stated wherever required.