National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Timothy Antony Rajkumar
Student ID: 21172200

School of Computing
National College of Ireland

Supervisor: Mr Abubakr Siddig

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Timothy Antony Rajkumar		
Student ID:	21172200		
Programme:	MSc in Data Analytics	**Year:**	2023
Module:	MSc Research Project/ Internship		
Lecturer:	Abubakr siddig		
Submission Due Date:	14/08/2023		
Project Title:	Detection of Autism Spectrum Disorder using Deep Neural Network		
Word Count:	697 **Page Count:** 8		

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: Timothy Antony Rajkumar

Date: 14/08/2023

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

Timothy Antony Rajkumar
Student ID: x21172200

1 Introduction

The configuration manual contains the information about all the technologies and tools used to implement in the research project. Software and all the technologies used in the research are described in section-2. Section -3 describes the step by step procedure of implementation, execution, and output of the research project along with the source code.

2 Technologies and Software

Python of version 3.9.13 was used for the development of deep neural network algorithm to detect ASD in Toddlers.

```
(base) C:\Users\Timothy>python --version
Python 3.9.13
```

Figure 1: Version of python

Jupyter Notebook is used to develop all the artificial neural network models. The Jupyter notebook is an open-source tool that supports python programming languages which produces an interactive output for the user.

```
(base) C:\Users\Timothy>jupyter notebook --version
6.4.12
```

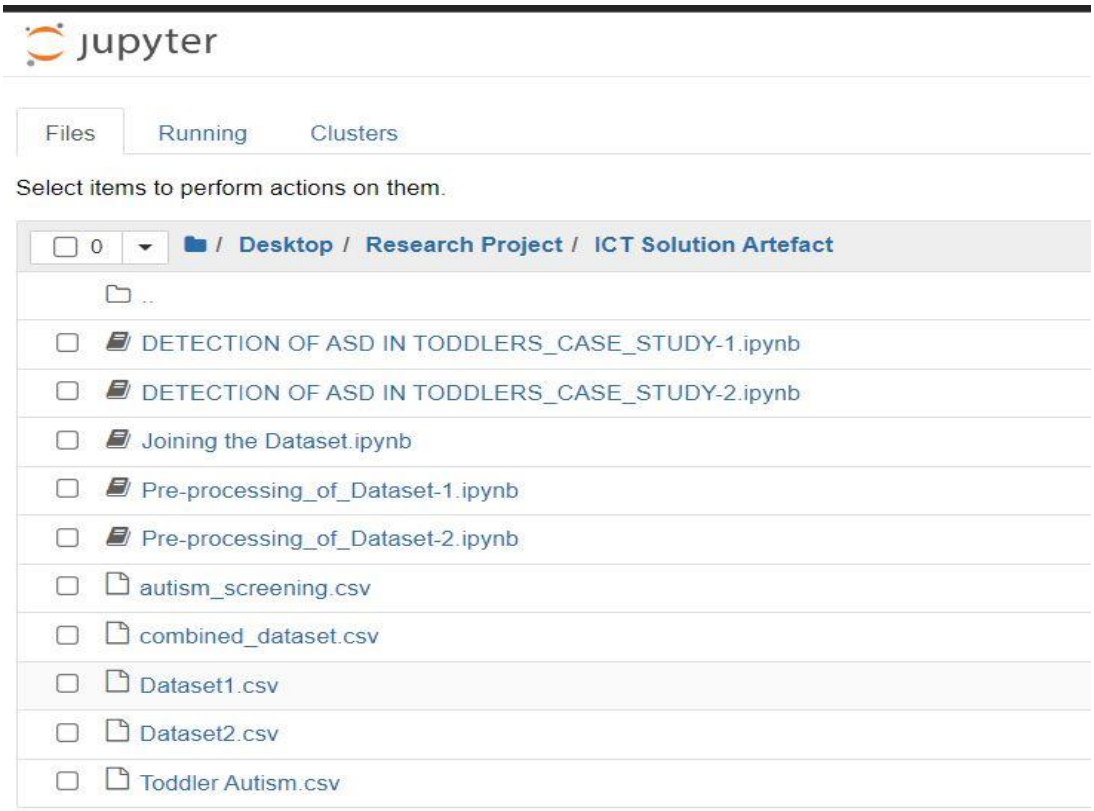
Figure 2: Jupyter notebook Version

The specification of laptop used for this research is shown below

Device name	Timothy
Processor	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz
Installed RAM	24.0 GB (23.8 GB usable)

3 Implementation

- Step 1: Install the Jupyter notebook into the systems and launch Jupyter notebook for the folder having all the source code and datasets.



- Step 2: Initially the dataset should be imported for data pre-processing
- Step 3: Import the necessary libraries such as pandas, numpy, TensorFlow and etc.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import plot_model
from sklearn.model_selection import train_test_split, GridSearchCV

sns.set_style("darkgrid")
pd.set_option("display.max_columns", None)   # setting to display all columns
pd.options.plotting.backend = "plotly"
```

Figure 3: Importing the libraries

```
#importing the dataset
df = pd.read_csv("autism_screening.csv")
df.head()
```

```
#importing the dataset
df = pd.read_csv("Toddler Autism.csv")
df.head()
```

Figure 4: Loading the datasets

- Step 5: Performing pre-processing methods on both the datasets individually

```
#Finding any NULL Values
import numpy as np
np.seterr(divide='ignore', invalid='ignore')
pd.DataFrame(fd.isnull().sum(), columns=["Missing Values"]).style.bar(color = "red")
```

```
# Handling The Missing Values
print(f"Maximum age is data: {df['Age in Months'].max()}\n")

print(f"Minimum age is data: {df['Age in Months'].min()}")
```

```
# dropping record number 52
df.drop(index = 52, inplace = True)

# resetting index
df.reset_index(inplace = True)
```

```
# Replacing the NULL values in age column with the mean value of age
df['Age in Months'] = df['Age in Months'].fillna(np.round(df['Age in Months'].mean(), 0))
```

- Step 6: checking the outliers and removing them

```
def find_outliers_IQR(df):

   q1=df.quantile(0.25)

   q3=df.quantile(0.75)

   IQR=q3-q1

   outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]

   return outliers
```

```
outliers = find_outliers_IQR(df['Age in Months'])

print('number of outliers: '+ str(len(outliers)))

print('max outlier value: '+ str(outliers.max()))

print('min outlier value: '+ str(outliers.min()))

outliers
```

```
def drop_outliers_IQR(df):

   q1=df.quantile(0.25)

   q3=df.quantile(0.75)

   IQR=q3-q1

   not_outliers = df[~((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]

   outliers_dropped = outliers.dropna().reset_index()

   return outliers_dropped
```

- Step 7: Dropping unwanted columns

```
#Dropping Unwanted Columns
df.drop(['age_desc', 'Who_completed_the test', 'index' ], axis = 1, inplace = True)
df.head()
```

- Step 8: Checking the categorical features

```
#Checking for the unique values in categorical Feature
for col in df.select_dtypes('O').columns:
    print("---------")
    print(f'Column name: {col}\n')
    print(f'Unique values:\n{df[col].unique()}\n\n')
```

- Step 9: combining the two datasets

```
concatenated_df = pd.concat([df1, df2], ignore_index=True)
concatenated_df.reset_index(drop=True, inplace=True)
concatenated_df.to_csv('combined-dataset.csv', index=False)
fd= pd.read_csv("combined_dataset.csv")
```

- Step 10: The data is Normalized using one-hot encoding

```
#since the order is not proper, one-hot encoding process is performed
X = pd.get_dummies(X)
Y = pd.get_dummies(Y)
```

- Step 11 : In case study-1 all the features are selected and in case study-2 selected features are used for model implementation

```
#Splitting the Data
X = fd.drop("Class/ASD Traits ", axis = 1)     # select all other feature except "Class/ASD Traits" for training
Y = fd['Class/ASD Traits ']
```

Figure 5: Case study-1 selecting all features

```
target_columns = 'Class/ASD Traits '
target = fd[target_columns]
```

```
feature_columns = ['A1', 'A2', 'A3','A4','A6', 'A8','A9','A10','Age_Mons','Sex', 'Ethnicity','Jaundice','Family_mem_with_ASD']
features = fd[feature_columns]
```

Figure 6: Case study -2 some features are removed

- For both case studies the model implementations are same which are follows:

- Step 12: The dataset is further split into test and train datasets, where the dataset is divided into 25% and 75% for training the model.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25)
```

- Step 13: The dataset are trained using the ANN model

```
input_dim = X.shape[1]
```

```
model = Sequential()
model.add(Dense(128, input_dim = input_dim, kernel_initializer='normal', activation='relu'))
model.add(Dense(64, input_dim = input_dim, kernel_initializer='normal', activation='relu'))
model.add(Dense(32, activation = "relu", kernel_initializer='normal'))
model.add(Dense(16, activation = "relu", kernel_initializer='normal'))
model.add(Dense(8, activation = "relu", kernel_initializer='normal'))
model.add(Dense(2, activation = 'sigmoid'))
```

Figure 7: ANN Model

- Step 14: Compiling the Model

```
# compiling model
model.compile(optimizer = Adam(learning_rate = 0.001),
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])
```

- Step 15: The model is trained and saved as a model file

```
result = model.fit(X_train, Y_train, epochs = 20, batch_size = 10,validation_split=0.25)
```

Figure 8: Model file created

- Step 16: The model is evaluated by confusion matrix and classification report

```
from sklearn.metrics import confusion_matrix, classification_report

class_report = classification_report(Y_test[[1]], prediction)
print(class_report)
```

Figure 9: Classification Report

```
cm = confusion_matrix(Y_test[[1]], prediction
```

Figure 10: Confusion matrix

- Step 17: Next, the models are tuned using hyper-parameters

```
def build_model(hp):
    model = Sequential()

    model.add(Dense(units=hp.Int('units_1', min_value=32, max_value=256, step=32),
                    input_dim=input_dim,
                    kernel_initializer='normal',
                    activation='relu'))

    for i in range(hp.Int('num_layers', min_value=1, max_value=5)):
        model.add(Dense(units=hp.Int(f'units_{i+2}', min_value=16, max_value=128, step=16),
                        activation='relu',
                        kernel_initializer='normal'))

    model.add(Dense(2, activation='sigmoid'))

    model.compile(optimizer=Adam(learning_rate=hp.Float('learning_rate', min_value=1e-4, max_value=1e-2, sampling='log')),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

    return model
```

Figure 11: Model for Hyper-parameter tuning

- Step 18: The hyperband algorithm is used for tuning and the model saved.

```
tuner = kt.Hyperband(build_model,
                     objective='val_accuracy',
                     max_epochs=20,
                     factor=3,
                     directory='my_dir',
                     project_name='hyperparam_tuning')

tuner.search(X_train, Y_train, validation_data=(X_test, Y_test))

best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
best_model = tuner.hypermodel.build(best_hps)

best_model.fit(X_train, Y_train, epochs=20, validation_data=(X_test, Y_test))
```

- Step 19: Finally the model is evaluated using confusion matrix and classification report.