FUNCTIONAL CONNECTIVITY – BASED PREDICTION OF AUTISM SPECTRUM DISORDER

by

HEMAVARSHINI C	**2019103526**
KAVIYA R V	**2019103538**
POOJASRI S	**2019103553**

A project report submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment of the requirements
for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI - 25

JUNE 2023

BONAFIDE CERTIFICATE

Certified that this project report titled **FUNCTIONAL CONNECTIVITY - BASED PREDICTION OF AUTISM SPECTRUM DISORDER** is the bonafide work of **HEMAVARSHINI C (2019103526), KAVIYA R V (2019103538) and POOJASRI S (2019103553)** who carried out the project work under my supervision, for the fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation based on which a degree or an award was conferred on an earlier occasion on these or any other candidates.

Place: Chennai

Date: 07/06/2023

Dr. Angelin Gladston
Associate Professor
Department of Computer Science and Engineering
Anna University, Chennai – 25

COUNTERSIGNED

Head of the Department,
Department of Computer Science and Engineering,
Anna University Chennai,
Chennai - 600025

ACKNOWLEDGEMENT

We express our deep gratitude to our guide, **Dr. Angelin Gladston**, Associate professor, Department of Computer Science and Engineering, for guiding us through every phase of the project. We appreciate her thoroughness, tolerance and ability to share her knowledge with us. We would also like to thank her for her kind support and for providing necessary facilities to carry out the work.

We are extremely grateful to **Dr. S. Valli**, Professor & Head of the Department, Department of Computer Science and Engineering, Anna University, Chennai – 25, for extending the facilities of the Department towards our project and for her unstinting support.

Our express our thanks to the panel of reviewers**, Dr. S. Bose,** Professor, Department of Computer Science and Engineering, **Dr. P. Geetha**, Professor, Department of Computer Science and Engineering for their valuable suggestions and critical reviews throughout the course of our project.

We express our thanks to all other teaching and non-teaching staff who helped us in one way or other for the successful completion of the project. We would also like to thank our parents, family and friends for their indirect contribution in the successful completion of this project.

HEMAVARSHINI C **KAVIYA R V** **POOJASRI S**

ABSTRACT - ENGLISH

Autism spectrum disorder includes conditions that were previously considered separate autism, Asperger's syndrome, childhood disintegrative disorder and an unspecified form of pervasive developmental disorder. Autism spectrum disorder begins in early childhood and eventually causes problems functioning in society, socially in school and at work. A small number of children appear to develop normally in the first year, and then go through a period of regression between 18 and 24 months of age when they develop autism symptoms. Therefore, the proposed system provides an effective solution to predict the presence of autism spectrum disorder in a more efficient way. The datasets are collected from ABIDE dataset. In data pre-processing technique, normalization is used to pre-process the datasets. In feature extraction process, ComBat Harmonization method will be used to extract the features. Then, the datasets will be trained using the algorithms such as SVM, CNN and ANN and the model file will be generated. When an input image is given for disease prediction process, it can effectively determine the presence of disease at earlier. Thus, this project helps in effective diagnosis of the autism spectrum disorder with higher accuracy than the existing models. CNN have better performance with high accuracy of about 84.62% in comparison to ANN that have about 82.05% accuracy and SVM that have about 48.72% accuracy.

திட்டப்பணிச் சுருக்கம்

ஆட்டிசம் ஸ்பெக்ட்ரம் கோளாறில் முன்பு தனி மன இறுக்கம், ஆஸ்பெர்ஜர் நோய்க்குறி, குழந்தைப் பருவ சிதைவுக் கோளாறு மற்றும் குறிப்பிடப்படாத பரவலான வளர்ச்சிக் கோளாறாகக் கருதப்பட்ட நிலைகள் அடங்கும். ஆட்டிசம் ஸ்பெக்ட்ரம் சீர்குலைவு குழந்தை பருவத்தில் தொடங்கி இறுதியில் சமூகத்தில், சமூக ரீதியாக பள்ளி மற்றும் வேலையில் செயல்படுவதில் சிக்கல்களை ஏற்படுத்துகிறது. ஒரு சிறிய எண்ணிக்கையிலான குழந்தைகள் முதல் வருடத்தில் சாதாரணமாக வளர்ச்சியடைந்து, பின்னர் 18 மற்றும் 24 மாதங்களுக்கு இடையில் ஆட்டிசம் அறிகுறிகளை உருவாக்கும் போது காலத்தை கடந்து செல்கின்றனர். எனவே, அமைப்பு ஆட்டிசம் ஸ்பெக்ட்ரம் கோளாறு இருப்பதை மிகவும் திறமையான முறையில் கணிக்க ஒரு பயனுள்ள தீர்வை வழங்குகிறது. ABIDE தரவுத்தொகுப்பிலிருந்து சேகரிக்கப்படுகின்றன. தரவு முன் செயலாக்க நுட்பத்தில், தரவுத்தொகுப்புகளை முன்கூட்டியே செயலாக்க இயல்பாக்கம் பயன்படுத்தப்படுகிறது. அம்சத்தைப் பிரித்தெடுக்கும் செயல்பாட்டில், அம்சங்களைப் பிரித்தெடுக்க காம்பாட் ஒத்திசைவு முறை பயன்படுத்தப்படும். பின்னர், தரவுத்தொகுப்புகள் SVM, CNN மற்றும் ANN போன்ற அல்காரிதம்களைப் பயன்படுத்தி பயிற்சியளிக்கப்பட்டு மாதிரி கோப்பு உருவாக்கப்படும். நோய் முன்னறிவிப்பு செயல்முறைக்கு ஒரு உள்ளீட்டு படம் கொடுக்கப்பட்டால், அது முந்தைய நோய் இருப்பதை திறம்பட தீர்மானிக்க முடியும். எனவே, இந்த திட்டம் தற்போதுள்ள மாதிரிகளை விட அதிக துல்லியத்துடன் ஆட்டிசம் ஸ்பெக்ட்ரம் கோளாறை திறம்பட கண்டறிய உதவுகிறது. சுமார் 82.05% துல்லியம் மற்றும் SVM 48.72% துல்லியம் கொண்ட ANN உடன் ஒப்பிடுகையில் CNN ஆனது சுமார் 84.62% அதிக துல்லியத்துடன் சிறந்த செயல்திறனைக் கொண்டுள்ளது.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

AQ	Autism Spectrum Quotient
ABIDE 1	Autism Brain Imaging Data Exchange
ANN	Artificial Neural Network
ASD	Autism Spectrum Disorder
CNN	Convolutional Neural Network
CVE	Common Vulnerabilities and Exposures
EEG	Electro Encephalo Graphy
MRI	Magnetic Resonance Imaging
NCREANN	Nonlinear Causal Relationship Estimation by Artificial Neural Network
PCA	Principal Component Analysis
SVM	Support Vector Machine
TD	Typically Developing
VR	Virtual Reality

CHAPTER 1

INTRODUCTION

This chapter presents the introductory part that encompasses the problem statement, overall objective of the proposed work and the challenges and applications of the project.

1.1 OVERVIEW

Autistic Spectrum Disorder (ASD) is a neurodevelopmental condition characterized by difficulties in social interaction, communication, and repetitive patterns of behavior. Early detection and accurate screening of ASD play a crucial role in enabling timely interventions and improving the quality of life for affected individuals. The prevalence of autism spectrum disorder (ASD) has been steadily rising, with estimates suggesting that approximately 1 in 54 children is affected by ASD. [1].

Engagement plays a vital role in the social and cognitive development of children, particularly those with autism spectrum disorder (ASD). Detecting and understanding engagement patterns in children with ASD can provide valuable insights into their interactions, preferences, and overall well-being [2]. Anxiety is a prevalent and debilitating condition among individuals with autism spectrum disorder (ASD). Timely detection and intervention are crucial for managing anxiety and improving the quality of life for individuals with ASD [3].

The detection and diagnosis of autism spectrum disorder (ASD) are challenging tasks due to the heterogeneity of the disorder and the lack of objective biomarkers. Various techniques have been explored to aid in the detection process, ranging from traditional clinical assessments to more advanced technological approaches. In recent years, deep learning techniques, a subset of machine learning algorithms, have gained considerable attention for their ability to automatically learn and extract features from complex datasets [4].

Traditional methods for diagnosing autism spectrum disorder (ASD) typically rely on subjective evaluations, clinical assessments, and parental reports. These approaches can be time-consuming, require specialized expertise, and may lack objectivity. Automated ASD detection using visual data, such as photographs, offers a promising alternative by leveraging the power of computer vision techniques and machine learning algorithms to identify characteristic patterns associated with ASD [5]. By leveraging machine learning algorithms and data-driven techniques, predictive models are developed that can effectively analyze and interpret diverse data sources to predict the likelihood of ASD in children [6].

Social interaction and communication deficits are core characteristics of autism spectrum disorders (ASD), impacting the daily lives and social integration of affected children. Virtual reality (VR) technology has emerged as a promising tool for promoting social interaction and communication skills in individuals with ASD [7]. Children with Autism Spectrum Disorder (ASD) face challenges in collaboration and verbal communication, which can significantly impact their social interactions and everyday interactions. The assessment of these skills is essential for understanding an individual's abilities, progress, and areas of improvement [8]. By selecting relevant features and leveraging advanced machine learning algorithms, our models aim to accurately classify individuals with ASD, contributing to early identification and intervention [9].

Functional connectivity analysis of rs-fMRI data offers a promising avenue for understanding the neural correlates of autism spectrum disorder. The outcomes of this research have the potential to enhance our understanding of the neurobiological basis of ASD and contribute to the development of more targeted interventions and treatments for individuals with ASD [10]. A novel approach called nCREANN uses an artificial neural network to estimate nonlinear causal relationships between brain regions based on fMRI data. This method allows for the identification of causal relationships between brain regions that may be relevant for ASD and can potentially be used as biomarkers for the disorder [11].

Machine learning approaches offer great potential for improving the detection and diagnosis of autism spectrum disorder. By leveraging diverse data sources and employing advanced algorithms, the accuracy, efficiency, and objectivity of ASD detection can be enhanced [12].

1.2 OBJECTIVES

- To develop a method using as **SVM**, **CNN** and **ANN** algorithms to easily determine the presence of the disease.
- Use **ABIDE** (Autism Brain Imaging Data Exchange) dataset for the detection of autism spectrum disorder.
- To help in effective diagnosis of disease with higher accuracy.

1.3 PROBLEM STATEMENT:

Autism has become the most prevalent childhood development disorder in United States. There is no easier system to predict Autism disease, which affects how a person perceives and socializes with others, causing problems in social interaction and communication. It lacks proper treatment due to inefficient prediction. It is not 100% clear on the causes or risk factors of ASD though it is believed that specific genes and chromosomes can increase the risk.

The diagnosis of ASD is typically performed using observation of behaviour as well as clinical interviews and questionnaires of the child and the parents. These techniques however may create disparities in diagnosis and therefore it has become crucial to identify objective pathological biomarkers of ASD that can support clinical diagnosis, especially in ambiguity, as well as an aid in predicting the risk of ASD before the manifestation of behavioural symptoms.

On the part of our contribution two algorithms namely CNN and SVM has been implemented with an objective of predicting Autism spectrum disorder with higher accuracy than the existing ones from the base paper. In preprocessing step normalization have been performed as multi-site imaging dataset is used. With these implemented algorithms CNN's performance was better than the prevalent ones with an accuracy of about 84.62%.

1.4 CHALLENGES AND APPLICATIONS

Challenges of Prediction of Autism Spectrum Disorder:

1. **Heterogeneity of ASD**: Autism Spectrum Disorder is a highly heterogeneous condition, encompassing a wide range of symptoms. This heterogeneity poses challenges for prediction models as they need to account for the diverse clinical presentations and variations in the data.
2. **Data Integration**: ASD prediction often requires integrating data from multiple sources, such as clinical assessments, behavioural observations, genetic markers, and neuroimaging data. Integrating and harmonizing these diverse data types can be challenging due to differences in data formats, quality, and compatibility.
3. **Generalization**: Developing prediction models that can generalize well to diverse populations and different settings is a significant challenge. Models trained on one dataset may not perform optimally when applied to new, independent datasets.

Applications of Prediction of Autism Spectrum Disorder:

1. **Early Detection and Intervention**: One of the key applications of ASD prediction is the early detection of the disorder in infants or young children. Early identification allows for timely intervention and support, leading to improved outcomes and quality of life for individuals with ASD.
2. **Screening and Assessment Tools**: ASD prediction models can serve as screening tools to identify individuals who may be at risk for ASD and require further evaluation. These models can augment traditional

assessment methods and provide objective measurements to support clinical decision-making.

3. **Resource Allocation and Planning**: ASD prediction models can assist policymakers, healthcare providers, and educators in resource allocation and planning. By identifying populations at higher risk for ASD, they can help allocate resources and develop targeted interventions to support individuals with ASD and their families.

1.5 ORGANIZATION OF THESIS

The organization of this thesis is as follows.

- ✓ **Chapter 2** provides a detailed summary of the literature that were referred while undertaking this project.
- ✓ **Chapter 3** provides the architecture diagram which conveys the entire flow of the project. The next section presents the proposed system for this project. The last section in Chapter 3 provides a brief description about the detailed module design which delivers the algorithm steps to implement that module indicating the intermediate deliverables along with the implementation details.
- ✓ **Chapter 4** details about the results and discussions. The first section of Chapter 4 provides a description about the dataset. The second section of Chapter 4 discusses about the results of the implementation of each module. The third section of Chapter 4 provides for all possible test cases in tabular form. The fourth section discusses about the performance metrics adopted for each algorithm. The last section of Chapter 4 provides a comparative analysis of the implemented algorithms.
- ✓ **Chapter 5** gives a conclusion of this project and discusses on the directions of future work.

CHAPTER 2

LITERATURE SURVEY

A. Baranwal et al., (2020) proposed Autism Spectrum Disorder screening: Prediction with machine learning models. It is observed Autistic Spectrum Disorder (ASD) is a developmental disorder that can be observed in all age groups. This paper uses ASD screening dataset for analysis and prediction of probable cases in adults, children and adolescents. The dataset for each of the age groups are analysed and inferences are drawn from them. Machine learning algorithms like Artificial Neural Networks (ANN), Random Forest, Logistic Regression, Decision Tree and Support Vector Machines (SVM) are used for prediction and comparison [1].

A. Chorianopoulou et al., (2017) proposed engagement detection for children with Autism Spectrum Disorder. Children with Autism Spectrum Disorder (ASD) face several difficulties in social communication. Hence, analysing social interaction can provide insight on their social and cognitive skills. In this paper, the degree of engagement of children in interactions with their parents is investigated. Features are derived from both participants including acoustic, linguistic and dialogue act features are explored. The effect of visual cues is also investigated. The task of engagement detection using video-recorded sessions consisting of interactions of typically developing (TD) and ASD children is experimented. Results show that engagement is easier to predict for TD children than for ASD children, and that the parent's actions/movements are better predictors of the child's degree of engagement [2].

A. Puli et al., (2020) proposed an automatic anxiety detection in autism: a real time algorithm for detecting physiological arousal in the presence of motion. It is identified that anxiety is a significant clinical concern in autism spectrum disorder (ASD) due to its negative impact on physical and psychological health. This paper introduced a novel multiple model Kalman-like filter to integrate heart rate and accelerometery signals. The filter tracks user heart rate under different motion assumptions and chooses the appropriate model for anxiety detection based on user motion conditions. The objective of this work was to address this challenge by proposing an approach for real-time detection and mitigation of physical activity effects. A critical limitation of existing anxiety detection systems is that physiological arousal is not specific to anxiety and can occur with other user states such as physical activity. This can result in false positives which can hinder the operation of these systems in real-world situations. The proposed method can reduce false detections due to user motion, and effectively detect arousal states during movement periods [3].

A. Sharma et al., (2020) proposed deep analysis of Autism Spectrum Disorder detection techniques. It is observed that Autism spectrum disorder (ASD) is a complex, complicated and lifelong development disability which includes problem that are characterized by repetitive behaviour, non-verbal communication, lack of concentration. In recent years, ASD is increasing at a higher momentum which needs early diagnosis. Detecting Autism through various Screening tool are very time consuming and costly. In last few years, various mathematical models also called as predictive analytics are widely used for predictions. For medical science, Machine learning and pattern recognition are various multidisciplinary research areas which provide effective techniques to diagnose ASD. The main aim of this paper is to analyse various Machine learning algorithms, used by various researcher like SVM (support Vector Machine), Random Forest Scan, decision trees, logistic regression and compare the result based on their accuracy and efficiency [4].

A. Z. Guo (2023) proposed an automated autism detection based on characterizing observable patterns from photos. This paper investigated the feasibility of developing an automated method to analyse the visual cues of autism using the photos taken by people with ASD, comparing to photos taken by people without ASD, in different scenarios. It was inspired by a recent study based on manual inspection of the photos. The key challenge is what and how to characterize the photos taken by people with ASD, to facilitate an automated separation from normal people. People with ASD show atypical attentions to social stimuli and gaze at human faces and complex scenes in an unusual way, and their facial expressions are often atypical as well. Several features are proposed to characterize the observable behaviours for ASD with experimental validations. This is the first work to perform an automatic analysis of the photos taken by people with ASD, achieving a prediction accuracy of 85.8% [5].

D. D. Jemima et all., (2021) suggested a study on diagnosis of Autism Spectrum Disorder for children. It is observed that ASD is one of the behavioural and developmental disorders in human being. Disorder in speech, difficulties in having interactions with social world, any restrained and repetitive actions in human will affect the well-being. It also includes the disorder in the development of brain, which in turn affects the brain structural and functional capabilities. Hence there are lots of literatures in diagnosing autism spectrum disorder automatically. They do report early that diagnosis of ASD is possible due to the improvement made in computer advancements and the diagnostic criteria. A complete survey on diagnosing autism spectrum disorder for children using machine learning and deep learning algorithms, ICT tools, humanoid robot, VR has been studied and their performance metrics have been deliberated in this paper [6].

H. Zhao et al., (2018) proposed a hand-in-hand: a communication- enhancement collaborative virtual reality system for promoting social interaction in children with Autism Spectrum Disorders. It is identified that children with autism spectrum disorders (ASD) often exhibit impairments in communication and social interaction, and thus face various social challenges in collaborative activities. The development of CVE technology for ASD intervention may lead to the 15 Internal creation of a novel low-cost intervention environment that will foster collaboration with peers and provide flexibility in communication [7].

L. Zhang et al., (2021) proposed of designing an intelligent agent to measure collaboration and verbal communication skills of children with Autism Spectrum Disorder in collaborative puzzle games. It is identified that collaborative puzzle games are interactive activities that can be played to foster the collaboration and verbal-communication skills of children with ASD. This paper designed an intelligent agent that can play collaborative puzzle games with children and verbally communicate with them as if it is another human player. This intelligent agent is also able to automatically measure children's task-performance and verbal communication behaviours throughout game play. Two preliminary studies were conducted with children with ASD to evaluate the feasibility and performance of the intelligent agent [8].

M. B. Mohammed et al., (2021) suggested about identifying Autism Spectrum Disorder through feature selection based machine learning. It is observed that ASD is a developmental disability that is likely to be perceived at a young age, persisting throughout a lifetime. The goal of this study is to detect ASD more efficiently with the use of Machine Learning methods. This paper worked with the AQ-10 Adult dataset. Different data synthesisation techniques have been and a few feature selection techniques and eventually implemented them with other classifiers. Throughout the analysis, the usage of Neural Network has some significant effect due to a smaller data set can be seen, the best performance was provided by the combination of classifiers and feature selection methods to develop the prediction model. After evaluation, they deduced that a model with Principal Component Analysis (PCA) feature selection method using the AdaBoost classifier gave the best results [9].

M. Ingalhalikar et al., (2021) proposed functional connectivity-based prediction of Autism on site harmonized ABIDE datsaset. This paper worked with larger sample sizes available from multi-site publicly available neuroimaging data repositories makes machine-learning based diagnostic classification of mental disorders more feasible by alleviating the curse of dimensionality. However, since multi-site data are aggregated post-hoc, i.e., they were acquired from different scanners with different acquisition parameters, non-neural inter-site variability may mask inter-group differences. The results show that higher classification accuracies across multiple classification models can be obtained (especially for models based on artificial neural networks) from multi-site data post harmonization with the ComBat technique as compared to without harmonization, outperforming earlier results from existing studies using ABIDE [10].

N. Talebi et al., (2019) proposed NCREANN: Nonlinear Casual Relationship Estimation by Artificial Neural Network that is applied for Autism connectivity study. The causal relationship estimator called “nCREANN” identifies both linear and nonlinear components of effective connectivity in the brain. Furthermore, it can distinguish between these two types of connectivity components by calculating the linear and nonlinear parts of the network input-output mapping. The nCREANN performance has been verified using synthesized data and then it has been applied on EEG data collected during rest in children with autism spectrum disorder (ASD) and Typically Developing (TD) children [11].

N. Zaman et al., (2021) suggested Autism Spectrum Disorder detection using machine learning approach. It is always a complex procedure to diagnosis autism spectrum disorder (ASD) because there is no particular medical test for autism, like a blood test, to make a diagnosis for the disorder. It is a behaviourally diagnosed condition. To make a diagnosis, doctors look at the child's developmental history and behaviour. Apparently, most children do not attain a proper diagnosis for autism until it is too late This lateness in diagnosis hinders a child's ability to get the help they need to keep developing. It is important to diagnose ASD as early as possible through monitoring, screening, and evaluating a child's development so that a proper care and support is ensured for an autistic child to help them reach their full potential. So, a system that will have the ability to diagnosis Autism is developed with a reliable and effective conclusion even without the help of a professional [12].

CHAPTER 3

SYSTEM DESIGN AND IMPLEMENTATION

Chapter 3 contains the system architecture diagram of the proposed model. It also contains a detailed description of all the modules in the system along with module design diagram and pseudo code.

3.1 SYSTEM ARCHITECTURE

The system architecture diagram provides a high-level overview of the structure and components of a software system. It illustrates the various elements of the system and how they interact with each other to fulfil the system's functionalities. The system architecture diagram for the proposed system is shown in figure 3.1. Figure 3.1. have a total of 7 modules which elaborates the flow of the project implementation.

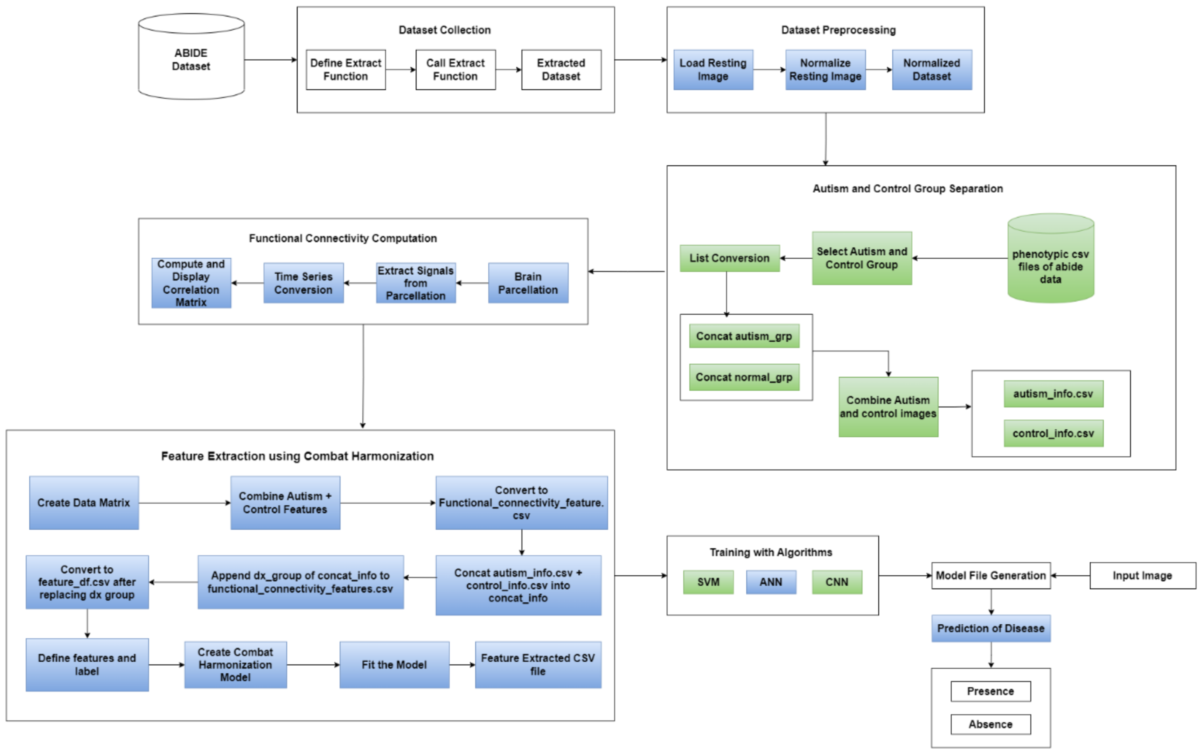

ABIDE Dataset
Dataset Collection
Define Extract Function
Call Extract Function
Extracted Dataset
Dataset Preprocessing
Load Resting Image
Normalize Resting Image
Normalized Dataset
Autism and Control Group Separation
List Conversion
Select Autism and Control Group
phenotypic csv files of abide data
Concat autism_grp
Concat normal_grp
Combine Autism and control images
autism_info.csv
control_info.csv
Functional Connectivity Computation
Compute and Display Correlation Matrix
Time Series Conversion
Extract Signals from Parcellation
Brain Parcellation
Feature Extraction using Combat Harmonization
Create Data Matrix
Combine Autism + Control Features
Convert to Functional_connectivity_feature.csv
Convert to feature_df.csv after replacing dx group
Append dx_group of concat_info to functional_connectivity_features.csv
Concat autism_info.csv + control_info.csv into concat_info
Define features and label
Create Combat Harmonization Model
Fit the Model
Feature Extracted CSV file
Training with Algorithms
SVM
ANN
CNN
Model File Generation
Input Image
Prediction of Disease
Presence
Absence

Figure 3.1 Overall Architecture Diagram

3.2 PROPOSED SYSTEM

The proposed system provides an effective solution to predict the presence of autism spectrum disorder in a more efficient way. In this project, Deep Learning and Machine Learning algorithms such as CNN, SVM and ANN are used to determine the presence of disorder at an early stage. The datasets are collected from ABIDE dataset which is an open access, multisite image repository comprising structural and functional scans of ASD and matched typically developing (TD). The datasets are in '.tgz' format which is a compressed archive file format that combines multiple files and directories into a single file. Hence the datasets are extracted by defining an "extract" function. Then, in data pre-processing technique normalization is performed to pre-process the datasets. Normalization is performed in order to make the dataset compatible with these algorithms. The Autism and Control groups must be separated to proceed further for feature extraction process. Separating the autism group from the control group enables direct comparisons between the two groups. In feature extraction process, ComBat Harmonization method is used to extract the features after combining the autism images and control images. The features are extracted after computing the functional connectivity matrix for each image. The extracted features are appended and a csv file containing the required features will be generated. Then, the generated csv file will be trained using deep learning and machine learning algorithms such as CNN, SVM and ANN and the model file will be generated. When an input image is given for disease prediction process, it can effectively determine the presence of disease at earlier. Thus, this project helps in effective diagnosis of the autism spectrum disorder with higher accuracy than the existing models.

CNN and SVM have been implemented apart from the base paper in this project with an objective of improving the accuracy.

3.3 DETAILED MODULE DESIGN WITH IMPLEMENTAION

3.3.1 DATASET COLLECTION

Figure 3.2 shows the module design of collecting the datasets which is in .tgz format initially. Hence an extraction process is needed to collect the dataset.

ALGORITHM STEPS

- Download the abide dataset which is an open access multisite image repository comprising structural and functional scans of ASD and matched typically developing (TD) controls.
- Extract resting images from the downloaded file by defining the 'extract' function and call it.

INPUT: ABIDE dataset

OUTPUT: Extracted dataset

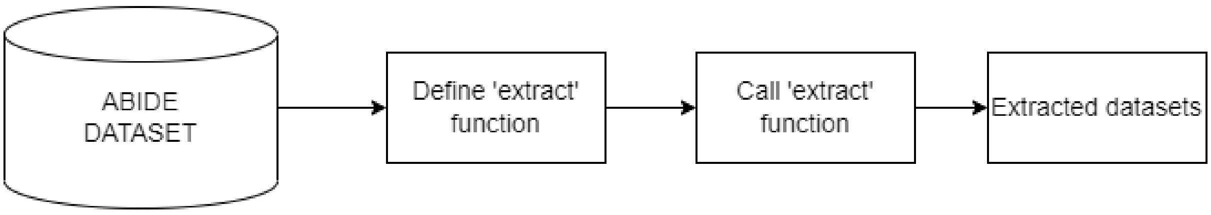
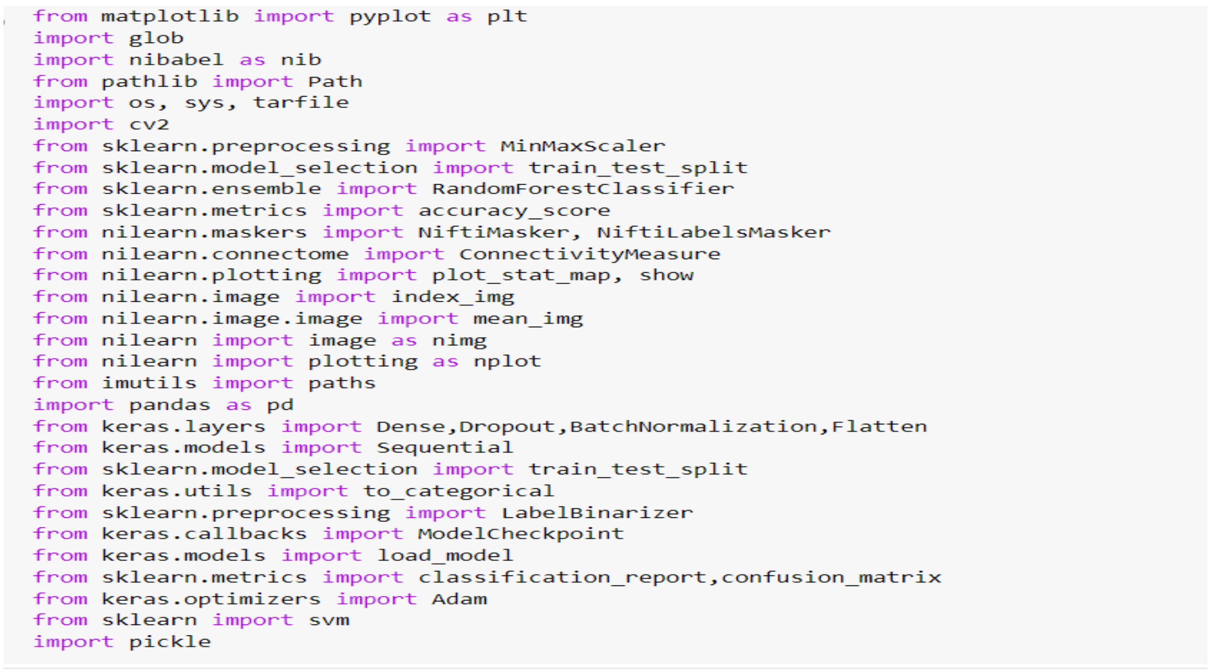


Figure 3.2 Dataset Collection

IMPLEMENTATION

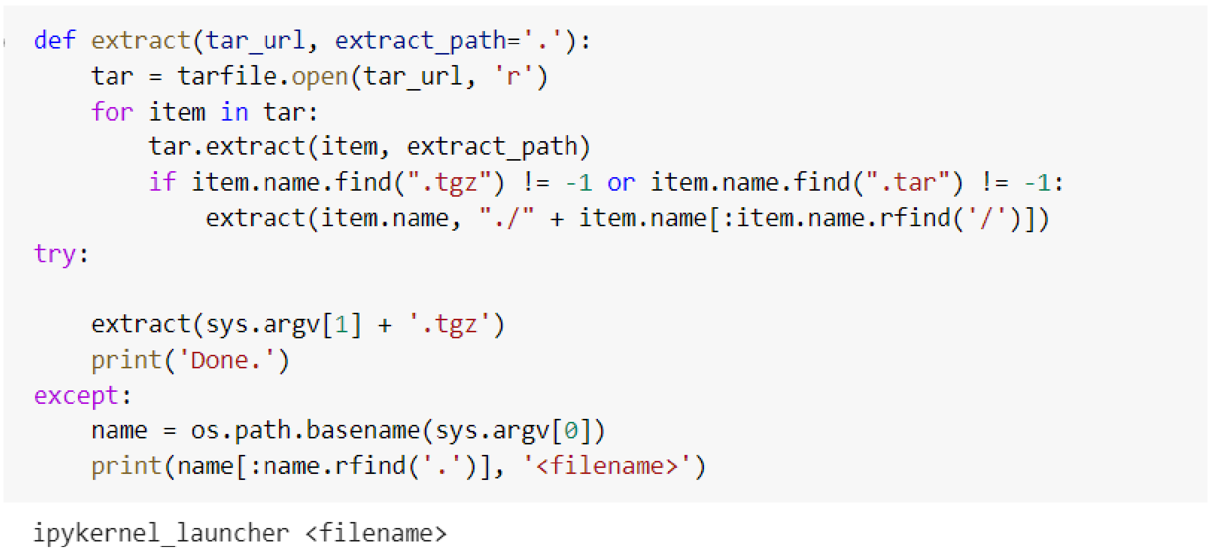
Figure 3.3 shows the required libraries that are imported- Nilearn and NiBabel are popular Python libraries used in the field of neuroimaging for processing and analyzing brain imaging data. Scikit-learn, often abbreviated as sklearn, is a popular open-source machine learning library for Python used for classifying autism and control groups. Keras is an open-source deep learning framework written in Python. It provides a high-level interface for building and training neural networks.



```python
from matplotlib import pyplot as plt
import glob
import nibabel as nib
from pathlib import Path
import os, sys, tarfile
import cv2
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from nilearn.maskers import NiftiMasker, NiftiLabelsMasker
from nilearn.connectome import ConnectivityMeasure
from nilearn.plotting import plot_stat_map, show
from nilearn.image import index_img
from nilearn.image.image import mean_img
from nilearn import image as nimg
from nilearn import plotting as nplot
from imutils import paths
import pandas as pd
from keras.layers import Dense,Dropout,BatchNormalization,Flatten
from keras.models import Sequential
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
from sklearn.metrics import classification_report,confusion_matrix
from keras.optimizers import Adam
from sklearn import svm
import pickle
```

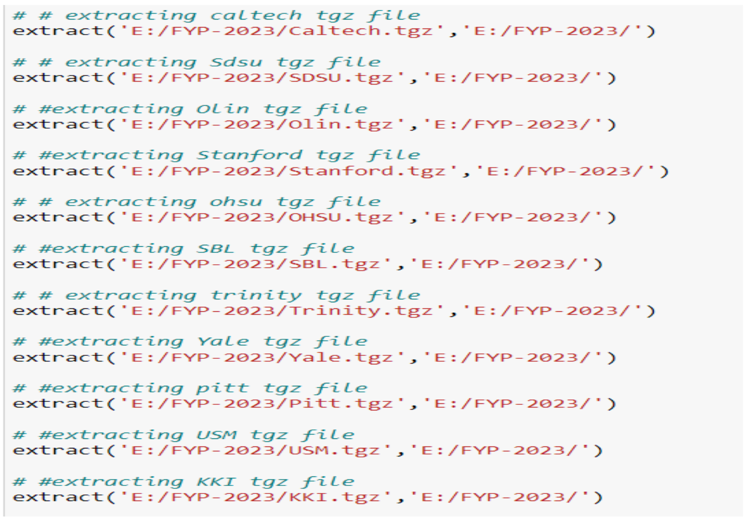
Figure 3.3 Importing libraries

Figure 3.4 shows the implementation of defining and then calling "extract" function. The "extract" function is defined to extract the .tgz files. Once the "extract" function is invoked the datasets will be extracted.

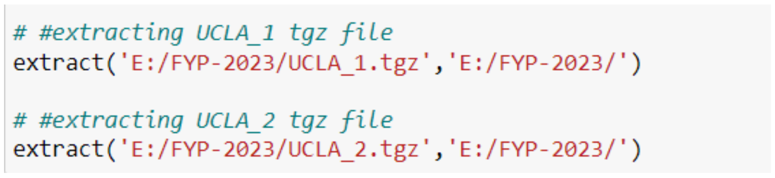


```python
def extract(tar_url, extract_path='.'):
    tar = tarfile.open(tar_url, 'r')
    for item in tar:
        tar.extract(item, extract_path)
        if item.name.find(".tgz") != -1 or item.name.find(".tar") != -1:
            extract(item.name, "./" + item.name[:item.name.rfind('/')])
try:

    extract(sys.argv[1] + '.tgz')
    print('Done.')
except:
    name = os.path.basename(sys.argv[0])
    print(name[:name.rfind('.')], '<filename>')
```

```
ipykernel_launcher <filename>
```



```
# # extracting caltech tgz file
extract('E:/FYP-2023/Caltech.tgz','E:/FYP-2023/')

# # extracting Sdsu tgz file
extract('E:/FYP-2023/SDSU.tgz','E:/FYP-2023/')

# #extracting Olin tgz file
extract('E:/FYP-2023/Olin.tgz','E:/FYP-2023/')

# #extracting Stanford tgz file
extract('E:/FYP-2023/Stanford.tgz','E:/FYP-2023/')

# # extracting ohsu tgz file
extract('E:/FYP-2023/OHSU.tgz','E:/FYP-2023/')

# #extracting SBL tgz file
extract('E:/FYP-2023/SBL.tgz','E:/FYP-2023/')

# # extracting trinity tgz file
extract('E:/FYP-2023/Trinity.tgz','E:/FYP-2023/')

# #extracting Yale tgz file
extract('E:/FYP-2023/Yale.tgz','E:/FYP-2023/')

# #extracting pitt tgz file
extract('E:/FYP-2023/Pitt.tgz','E:/FYP-2023/')

# #extracting USM tgz file
extract('E:/FYP-2023/USM.tgz','E:/FYP-2023/')

# #extracting KKI tgz file
extract('E:/FYP-2023/KKI.tgz','E:/FYP-2023/')
```



```
# #extracting UCLA_1 tgz file
extract('E:/FYP-2023/UCLA_1.tgz','E:/FYP-2023/')

# #extracting UCLA_2 tgz file
extract('E:/FYP-2023/UCLA_2.tgz','E:/FYP-2023/')
```

Figure 3.4 Define & call "extract"

3.3.2 DATASET PREPROCESSING

Figure 3.5 shows the module design of dataset pre-processing. In pre-processing technique normalization is inhibited. Normalization is necessary because the intensity values in MRI images can vary between different scans, due to factors such as differences in scanner hardware, acquisition protocols, and patient positioning. This variability can lead to inaccuracies in subsequent analyses of feature extraction.

ALGORITHM STEPS

- Load the anatomy and resting images from the extracted dataset.
- Normalize resting images using MinMaxScaler.
- Visualizing resting normalized images.

INPUT: Extracted dataset

OUTPUT: Normalized dataset

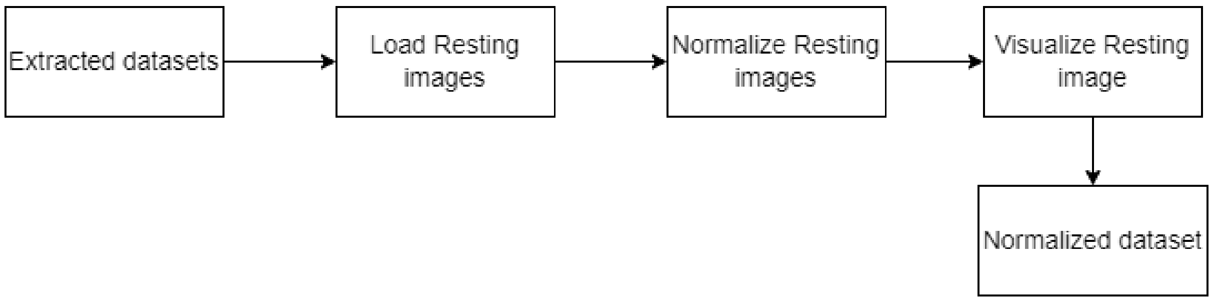
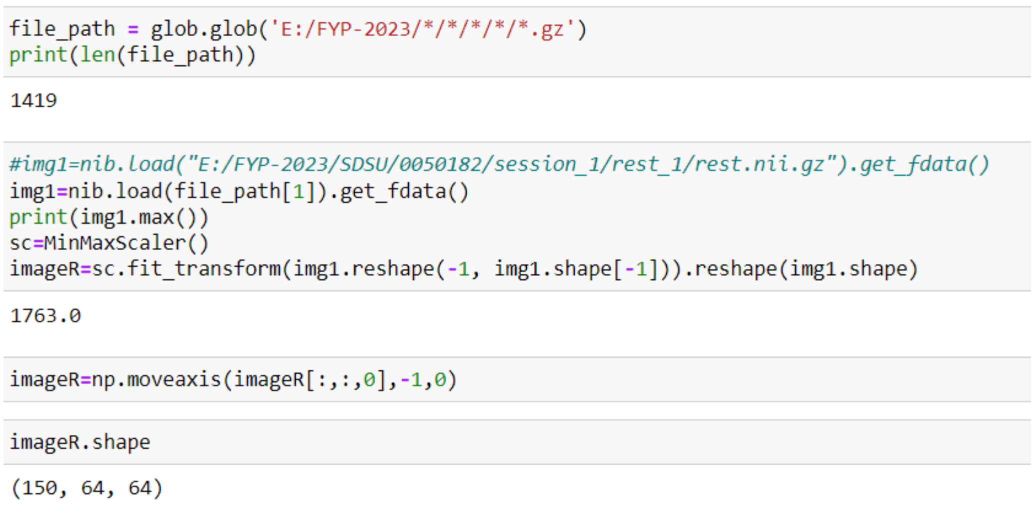


Figure 3.5. Dataset Pre-processing

IMPLEMENTATION

Figure 3.6 shows the implementation details of Dataset pre-processing. The datasets are loaded using load() of nibabel library and then normalized using MinMaxScalar(). Once normalized the resting images are visualized.



```
file_path = glob.glob('E:/FYP-2023/*/*/*/*/*.gz')
print(len(file_path))
```

```
1419
```

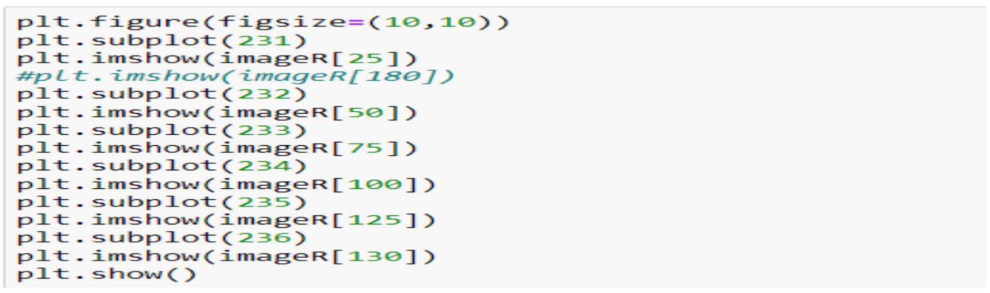
```
#img1=nib.load("E:/FYP-2023/SDSU/0050182/session_1/rest_1/rest.nii.gz").get_fdata()
img1=nib.load(file_path[1]).get_fdata()
print(img1.max())
sc=MinMaxScaler()
imageR=sc.fit_transform(img1.reshape(-1, img1.shape[-1])).reshape(img1.shape)
```

```
1763.0
```

```
imageR=np.moveaxis(imageR[:,:,0],-1,0)
```

```
imageR.shape
```

```
(150, 64, 64)
```



```
plt.figure(figsize=(10,10))
plt.subplot(231)
plt.imshow(imageR[25])
#plt.imshow(imageR[180])
plt.subplot(232)
plt.imshow(imageR[50])
plt.subplot(233)
plt.imshow(imageR[75])
plt.subplot(234)
plt.imshow(imageR[100])
plt.subplot(235)
plt.imshow(imageR[125])
plt.subplot(236)
plt.imshow(imageR[130])
plt.show()
```

Figure 3.6 Dataset pre-processing (Implementation)

3.3.2 AUTISM & CONTROL GROUP SEPERATION

Figure 3.7 shows the module design of Autism and Control group separation. If the ASD and control groups are not separated, the model may develop biases and not generalize well to new, unseen data. This could lead to inaccurate predictions, misdiagnosis, and inappropriate treatments. Moreover, separating the ASD and control groups is necessary to identify the specific biomarkers or features that differentiate individuals with ASD from typically developing individuals. If the two groups are not separated, the model may identify irrelevant features that do not contribute to the diagnosis of ASD.

ALGORITHM STEPS

- Load the csv files-phenotypic_CALTECH.csv, phenotypic_KKI.csv, phenotypic_SBL.csv, phenotypic_PITT.csv, phenotypic_OHSU.csv, phenotypic_USM, phenotypic_YALE.csv, phenotypic_SDSU.csv, phenotypic_OLIN.csv, phenotypic_STANFORD.csv, phenotypic_UCLA_1.csv, phenotypic_UCLA_2.csv, phenotypic_TRINITY.csv
- Select autism & control group based on DX_group.

  If dx_grp=1 it specifies autism group

  If dx_grp=2 it specifies control group.
- Convert the selected groups to list using tolist().
- Concat autism_group and control_group.
- Convert both groups to csv files.
- Combine autism & control images groups.
- autism_info.csv & control_info.csv will be created.

INPUT:phenotypic_CALTECH.csv,phenotypic_KKI.csv, phenotypic_SBL.csv,phenotypic_PITT.csv,phenotypic_OHSU.csv, phenotypic_USM, phenotypic_YALE.csv, phenotypic_SDSU.csv, phenotypic_OLIN.csv, phenotypic_STANFORD.csv, phenotypic_UCLA_1.csv, phenotypic_UCLA_2.csv, phenotypic_TRINITY.csv

OUTPUT: autism_info.csv & control_info.csv

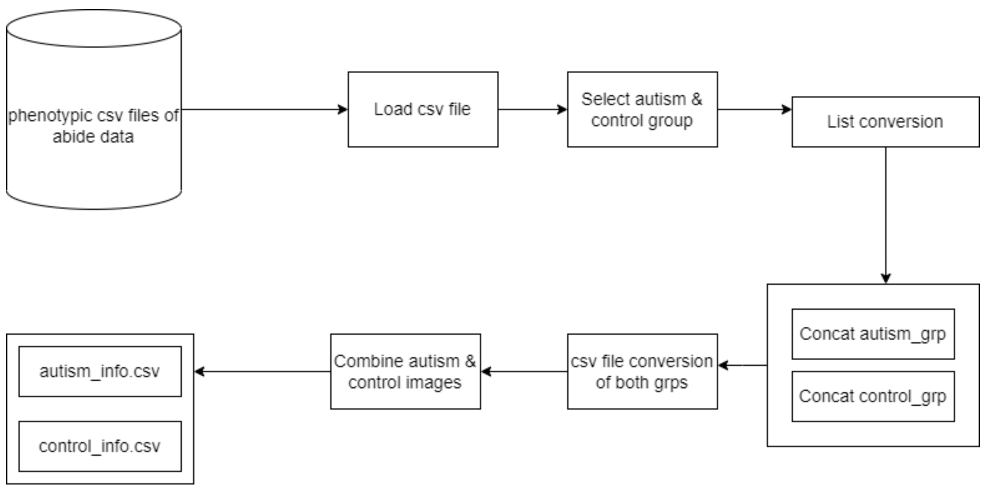
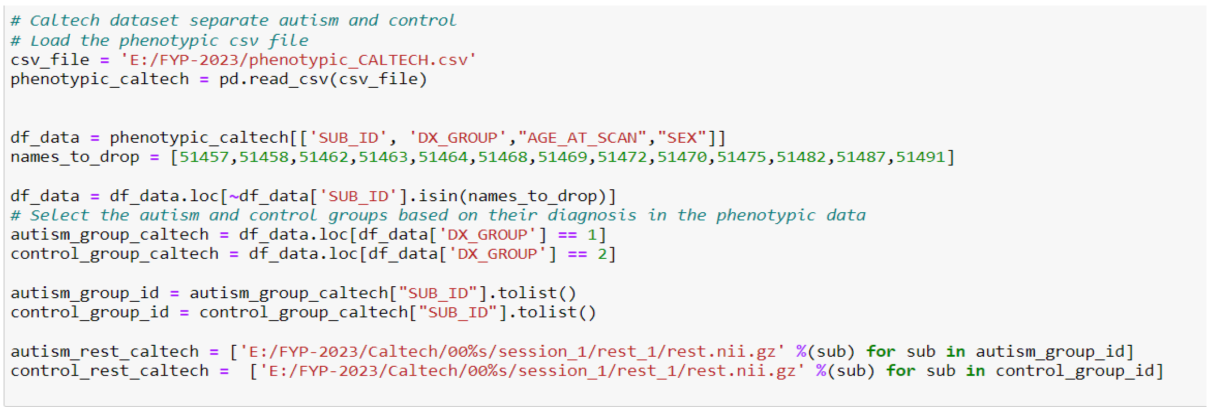

phenotypic csv files of abide data
Load csv file
Select autism & control group
List conversion
Concat autism_grp
Concat control_grp
csv file conversion of both grps
Combine autism & control images
autism_info.csv
control_info.csv

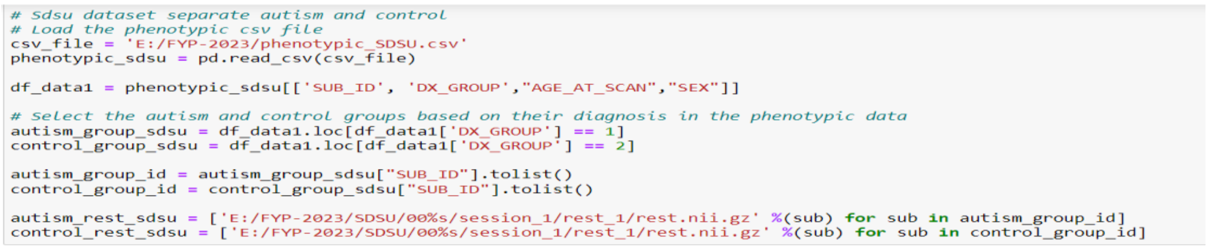
Figure 3.7 Autism and Control group separation

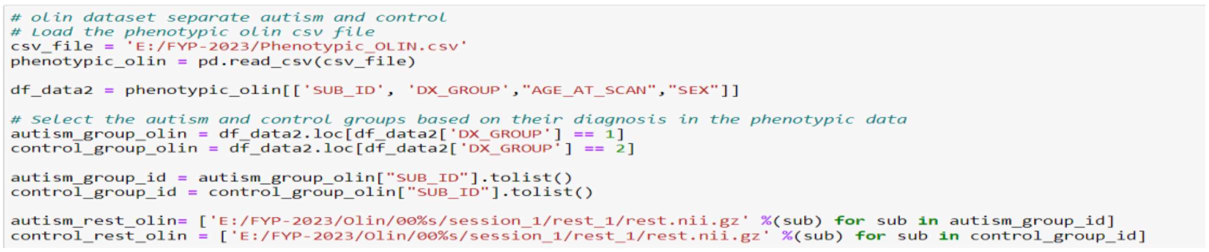
IMPLEMENTATION

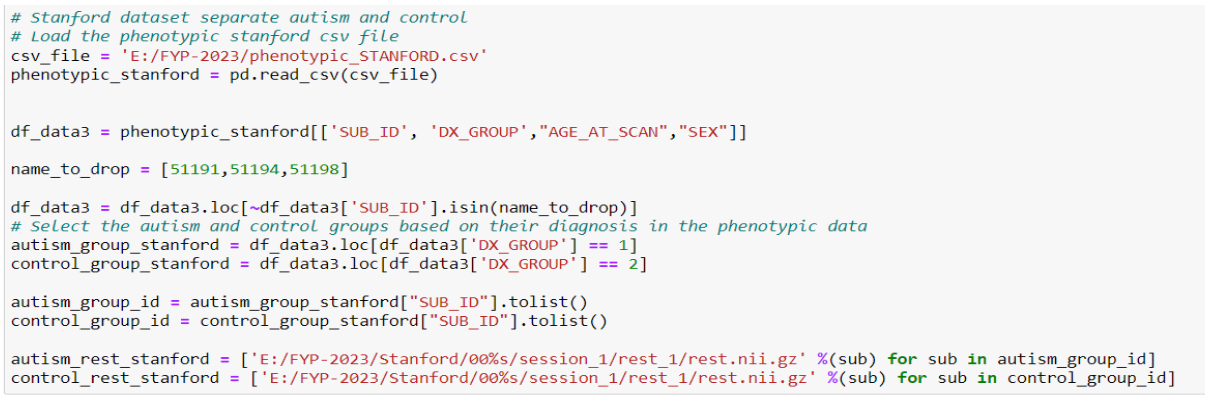
Figure 3.8, 3.9 shows the implementation of autism and control group separation. In Figure 3.8 the phenotypic csv files of corresponding university sites are loaded. The autism and control groups are then selected based on the DX-GROUP and the group ids are converted to list. The autism_rest_* and control_rest_* holds the subids of the autism and control images respectively.

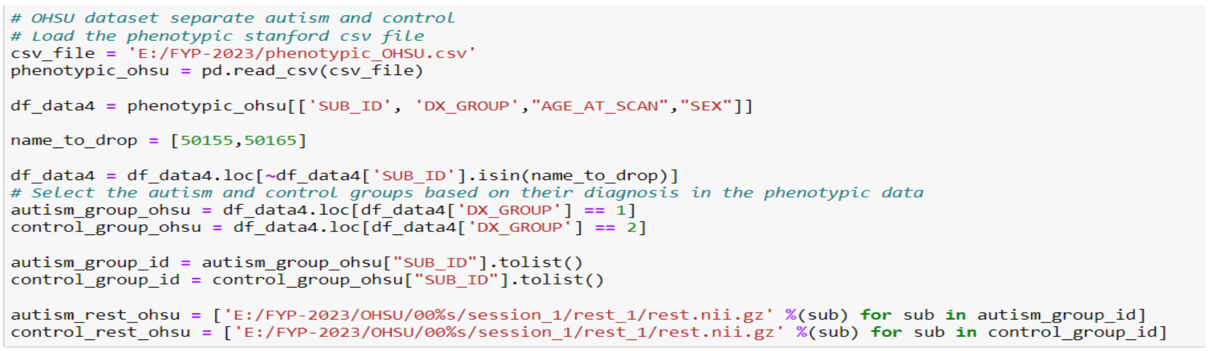


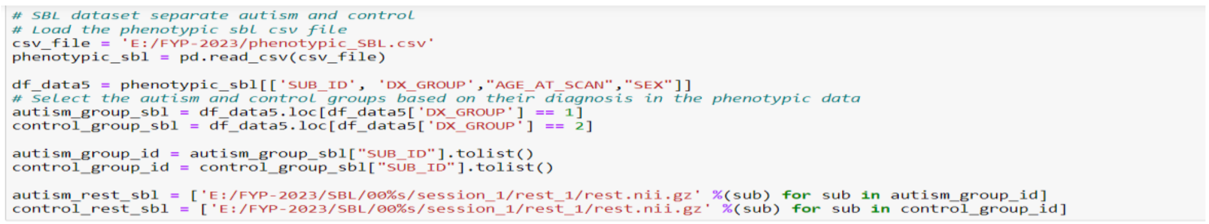
```
# Caltech dataset separate autism and control
# Load the phenotypic csv file
csv_file = 'E:/FYP-2023/phenotypic_CALTECH.csv'
phenotypic_caltech = pd.read_csv(csv_file)


df_data = phenotypic_caltech[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]
names_to_drop = [51457,51458,51462,51463,51464,51468,51469,51472,51470,51475,51482,51487,51491]

df_data = df_data.loc[~df_data['SUB_ID'].isin(names_to_drop)]
# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_caltech = df_data.loc[df_data['DX_GROUP'] == 1]
control_group_caltech = df_data.loc[df_data['DX_GROUP'] == 2]

autism_group_id = autism_group_caltech["SUB_ID"].tolist()
control_group_id = control_group_caltech["SUB_ID"].tolist()

autism_rest_caltech = ['E:/FYP-2023/Caltech/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_caltech =  ['E:/FYP-2023/Caltech/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```

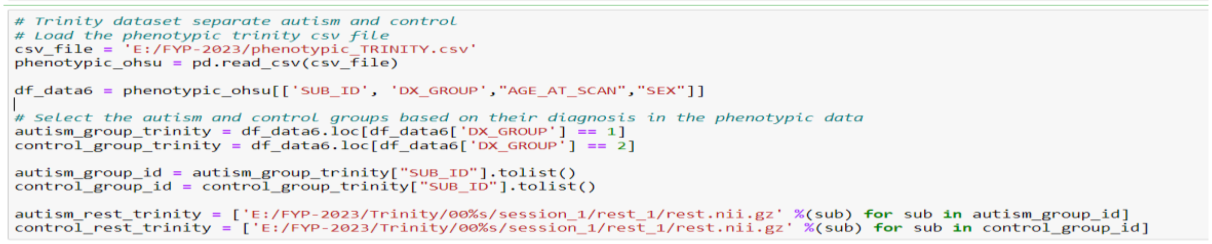

```
# Sdsu dataset separate autism and control
# Load the phenotypic csv file
csv_file = 'E:/FYP-2023/phenotypic_SDSU.csv'
phenotypic_sdsu = pd.read_csv(csv_file)

df_data1 = phenotypic_sdsu[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_sdsu = df_data1.loc[df_data1['DX_GROUP'] == 1]
control_group_sdsu = df_data1.loc[df_data1['DX_GROUP'] == 2]

autism_group_id = autism_group_sdsu["SUB_ID"].tolist()
control_group_id = control_group_sdsu["SUB_ID"].tolist()

autism_rest_sdsu = ['E:/FYP-2023/SDSU/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_sdsu = ['E:/FYP-2023/SDSU/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```

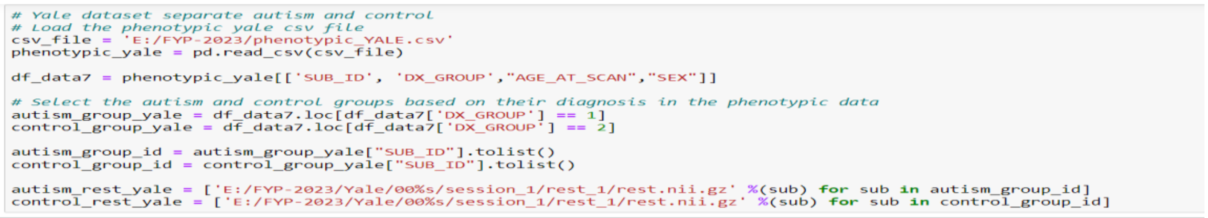

```
# olin dataset separate autism and control
# Load the phenotypic olin csv file
csv_file = 'E:/FYP-2023/Phenotypic_OLIN.csv'
phenotypic_olin = pd.read_csv(csv_file)

df_data2 = phenotypic_olin[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_olin = df_data2.loc[df_data2['DX_GROUP'] == 1]
control_group_olin = df_data2.loc[df_data2['DX_GROUP'] == 2]

autism_group_id = autism_group_olin["SUB_ID"].tolist()
control_group_id = control_group_olin["SUB_ID"].tolist()

autism_rest_olin= ['E:/FYP-2023/Olin/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_olin = ['E:/FYP-2023/Olin/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```


```
# Stanford dataset separate autism and control
# Load the phenotypic stanford csv file
csv_file = 'E:/FYP-2023/phenotypic_STANFORD.csv'
phenotypic_stanford = pd.read_csv(csv_file)


df_data3 = phenotypic_stanford[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

name_to_drop = [51191,51194,51198]

df_data3 = df_data3.loc[~df_data3['SUB_ID'].isin(name_to_drop)]
# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_stanford = df_data3.loc[df_data3['DX_GROUP'] == 1]
control_group_stanford = df_data3.loc[df_data3['DX_GROUP'] == 2]

autism_group_id = autism_group_stanford["SUB_ID"].tolist()
control_group_id = control_group_stanford["SUB_ID"].tolist()

autism_rest_stanford = ['E:/FYP-2023/Stanford/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_stanford = ['E:/FYP-2023/Stanford/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```

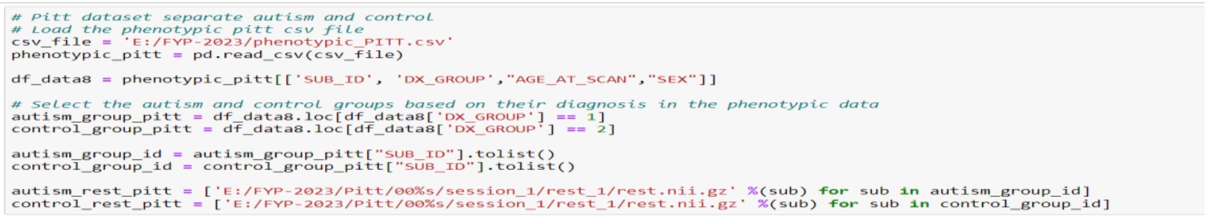

```
# OHSU dataset separate autism and control
# Load the phenotypic stanford csv file
csv_file = 'E:/FYP-2023/phenotypic_OHSU.csv'
phenotypic_ohsu = pd.read_csv(csv_file)

df_data4 = phenotypic_ohsu[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

name_to_drop = [50155,50165]

df_data4 = df_data4.loc[~df_data4['SUB_ID'].isin(name_to_drop)]
# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_ohsu = df_data4.loc[df_data4['DX_GROUP'] == 1]
control_group_ohsu = df_data4.loc[df_data4['DX_GROUP'] == 2]

autism_group_id = autism_group_ohsu["SUB_ID"].tolist()
control_group_id = control_group_ohsu["SUB_ID"].tolist()

autism_rest_ohsu = ['E:/FYP-2023/OHSU/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_ohsu = ['E:/FYP-2023/OHSU/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```


```
# SBL dataset separate autism and control
# Load the phenotypic sbl csv file
csv_file = 'E:/FYP-2023/phenotypic_SBL.csv'
phenotypic_sbl = pd.read_csv(csv_file)

df_data5 = phenotypic_sbl[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]
# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_sbl = df_data5.loc[df_data5['DX_GROUP'] == 1]
control_group_sbl = df_data5.loc[df_data5['DX_GROUP'] == 2]

autism_group_id = autism_group_sbl["SUB_ID"].tolist()
control_group_id = control_group_sbl["SUB_ID"].tolist()

autism_rest_sbl = ['E:/FYP-2023/SBL/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_sbl = ['E:/FYP-2023/SBL/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```

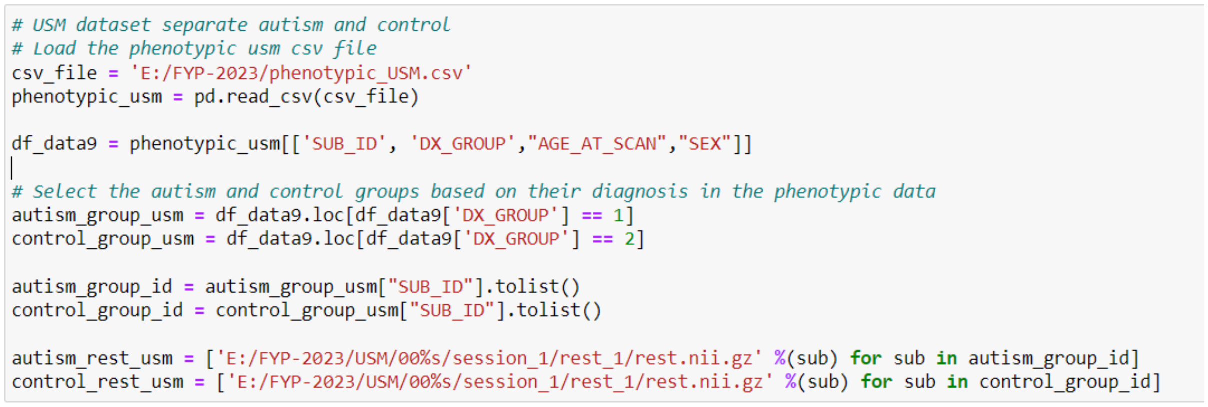

```
# Trinity dataset separate autism and control
# Load the phenotypic trinity csv file
csv_file = 'E:/FYP-2023/phenotypic_TRINITY.csv'
phenotypic_ohsu = pd.read_csv(csv_file)

df_data6 = phenotypic_ohsu[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_trinity = df_data6.loc[df_data6['DX_GROUP'] == 1]
control_group_trinity = df_data6.loc[df_data6['DX_GROUP'] == 2]

autism_group_id = autism_group_trinity["SUB_ID"].tolist()
control_group_id = control_group_trinity["SUB_ID"].tolist()

autism_rest_trinity = ['E:/FYP-2023/Trinity/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_trinity = ['E:/FYP-2023/Trinity/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```



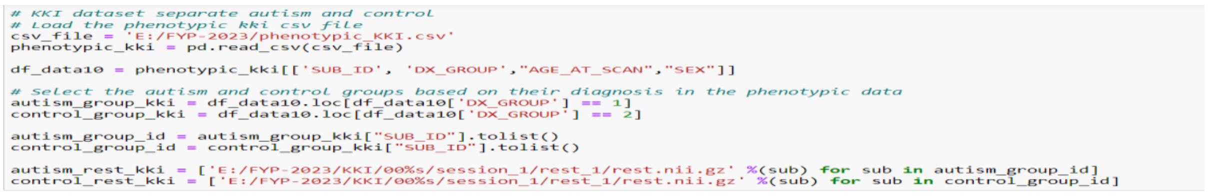
```
# Yale dataset separate autism and control
# Load the phenotypic yale csv file
csv_file = 'E:/FYP-2023/phenotypic_YALE.csv'
phenotypic_yale = pd.read_csv(csv_file)

df_data7 = phenotypic_yale[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_yale = df_data7.loc[df_data7['DX_GROUP'] == 1]
control_group_yale = df_data7.loc[df_data7['DX_GROUP'] == 2]

autism_group_id = autism_group_yale["SUB_ID"].tolist()
control_group_id = control_group_yale["SUB_ID"].tolist()

autism_rest_yale = ['E:/FYP-2023/Yale/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_yale = ['E:/FYP-2023/Yale/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```



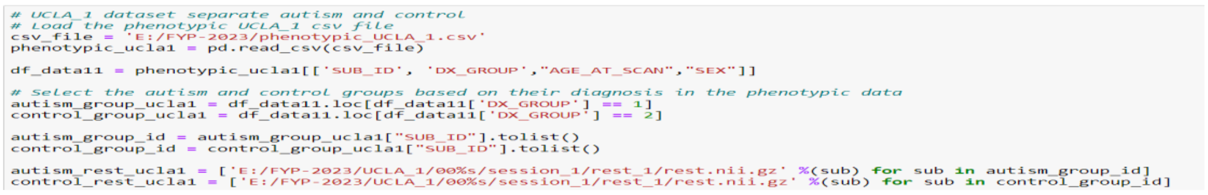
```
# Pitt dataset separate autism and control
# Load the phenotypic pitt csv file
csv_file = 'E:/FYP-2023/phenotypic_PITT.csv'
phenotypic_pitt = pd.read_csv(csv_file)

df_data8 = phenotypic_pitt[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_pitt = df_data8.loc[df_data8['DX_GROUP'] == 1]
control_group_pitt = df_data8.loc[df_data8['DX_GROUP'] == 2]

autism_group_id = autism_group_pitt["SUB_ID"].tolist()
control_group_id = control_group_pitt["SUB_ID"].tolist()

autism_rest_pitt = ['E:/FYP-2023/Pitt/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_pitt = ['E:/FYP-2023/Pitt/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```



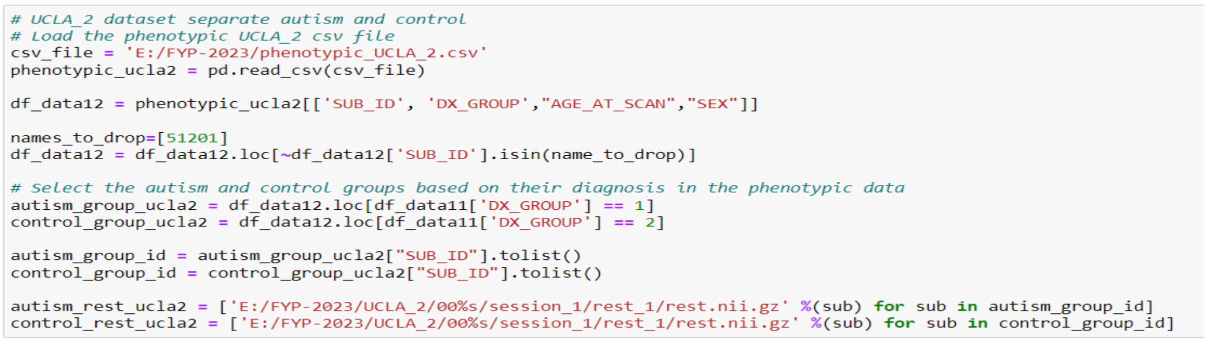
```
# USM dataset separate autism and control
# Load the phenotypic usm csv file
csv_file = 'E:/FYP-2023/phenotypic_USM.csv'
phenotypic_usm = pd.read_csv(csv_file)

df_data9 = phenotypic_usm[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_usm = df_data9.loc[df_data9['DX_GROUP'] == 1]
control_group_usm = df_data9.loc[df_data9['DX_GROUP'] == 2]

autism_group_id = autism_group_usm["SUB_ID"].tolist()
control_group_id = control_group_usm["SUB_ID"].tolist()

autism_rest_usm = ['E:/FYP-2023/USM/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_usm = ['E:/FYP-2023/USM/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```



```
# KKI dataset separate autism and control
# Load the phenotypic kki csv file
csv_file = 'E:/FYP-2023/phenotypic_KKI.csv'
phenotypic_kki = pd.read_csv(csv_file)

df_data10 = phenotypic_kki[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_kki = df_data10.loc[df_data10['DX_GROUP'] == 1]
control_group_kki = df_data10.loc[df_data10['DX_GROUP'] == 2]

autism_group_id = autism_group_kki["SUB_ID"].tolist()
control_group_id = control_group_kki["SUB_ID"].tolist()

autism_rest_kki = ['E:/FYP-2023/KKI/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_kki = ['E:/FYP-2023/KKI/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```



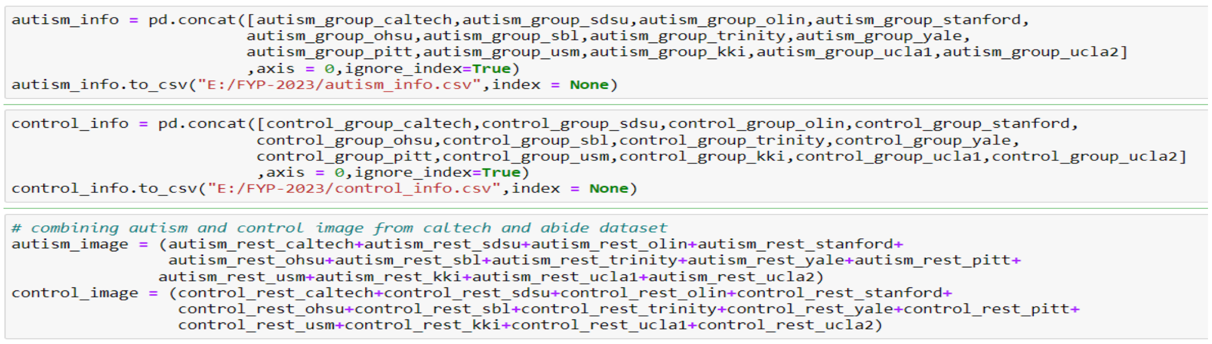
```
# UCLA_1 dataset separate autism and control
# Load the phenotypic UCLA_1 csv file
csv_file = 'E:/FYP-2023/phenotypic_UCLA_1.csv'
phenotypic_ucla1 = pd.read_csv(csv_file)

df_data11 = phenotypic_ucla1[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_ucla1 = df_data11.loc[df_data11['DX_GROUP'] == 1]
control_group_ucla1 = df_data11.loc[df_data11['DX_GROUP'] == 2]

autism_group_id = autism_group_ucla1["SUB_ID"].tolist()
control_group_id = control_group_ucla1["SUB_ID"].tolist()

autism_rest_ucla1 = ['E:/FYP-2023/UCLA_1/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_ucla1 = ['E:/FYP-2023/UCLA_1/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```



```
# UCLA_2 dataset separate autism and control
# Load the phenotypic UCLA_2 csv file
csv_file = 'E:/FYP-2023/phenotypic_UCLA_2.csv'
phenotypic_ucla2 = pd.read_csv(csv_file)

df_data12 = phenotypic_ucla2[['SUB_ID', 'DX_GROUP',"AGE_AT_SCAN","SEX"]]

names_to_drop=[51201]
df_data12 = df_data12.loc[~df_data12['SUB_ID'].isin(name_to_drop)]

# Select the autism and control groups based on their diagnosis in the phenotypic data
autism_group_ucla2 = df_data12.loc[df_data11['DX_GROUP'] == 1]
control_group_ucla2 = df_data12.loc[df_data11['DX_GROUP'] == 2]

autism_group_id = autism_group_ucla2["SUB_ID"].tolist()
control_group_id = control_group_ucla2["SUB_ID"].tolist()

autism_rest_ucla2 = ['E:/FYP-2023/UCLA_2/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in autism_group_id]
control_rest_ucla2 = ['E:/FYP-2023/UCLA_2/00%s/session_1/rest_1/rest.nii.gz' %(sub) for sub in control_group_id]
```

Figure 3.8 Autism & Control group separation (Implementation I)

In Figure 3.9 the autism information is concated from various university sites based on the diagnosis in phenotypic data. Similarly, the control information is concated from various university sites based on the diagnosis in phenotypic data. Then all autism and control images are concated based on the subids of the resting images in autism_image and control_image respectively.



```
autism_info = pd.concat([autism_group_caltech,autism_group_sdsu,autism_group_olin,autism_group_stanford,
                        autism_group_ohsu,autism_group_sbl,autism_group_trinity,autism_group_yale,
                        autism_group_pitt,autism_group_usm,autism_group_kki,autism_group_ucla1,autism_group_ucla2]
                        ,axis = 0,ignore_index=True)
autism_info.to_csv("E:/FYP-2023/autism_info.csv",index = None)
```

```
control_info = pd.concat([control_group_caltech,control_group_sdsu,control_group_olin,control_group_stanford,
                         control_group_ohsu,control_group_sbl,control_group_trinity,control_group_yale,
                         control_group_pitt,control_group_usm,control_group_kki,control_group_ucla1,control_group_ucla2]
                         ,axis = 0,ignore_index=True)
control_info.to_csv("E:/FYP-2023/control_info.csv",index = None)
```

```
# combining autism and control image from caltech and abide dataset
autism_image = (autism_rest_caltech+autism_rest_sdsu+autism_rest_olin+autism_rest_stanford+
                autism_rest_ohsu+autism_rest_sbl+autism_rest_trinity+autism_rest_yale+autism_rest_pitt+
               autism_rest_usm+autism_rest_kki+autism_rest_ucla1+autism_rest_ucla2)
control_image = (control_rest_caltech+control_rest_sdsu+control_rest_olin+control_rest_stanford+
                 control_rest_ohsu+control_rest_sbl+control_rest_trinity+control_rest_yale+control_rest_pitt+
                 control_rest_usm+control_rest_kki+control_rest_ucla1+control_rest_ucla2)
```

Figure 3.9 Autism & Control group separation (Implementation II)

3.3.4 FUNTIONAL CONNECTIVITY COMPUTATION

Figure 3.10 shows the module design of Functional Connectivity Computation. Functional connectivity in ASD prediction refers to the measurement of the temporal correlations between spatially remote neurophysiological events in the brain, such as between different brain regions, that are thought to be functionally connected. It is a method used in the analysis of functional magnetic resonance imaging (fMRI) data to investigate the patterns of activation and communication between different brain regions.

ALGORITHM STEPS

- Define the regions from which the signals must be extracted.
- Extract signals from parcellation using nilearn.maskers.NiftiLabelsMasker.
- Convert it to time-series using NiftiLabelsMasker.fit_transform() method.
- Compute the correlation matrix.
- Display the correlation matrix.

INPUT: normalized dataset

OUTPUT: functional connectivity matrix

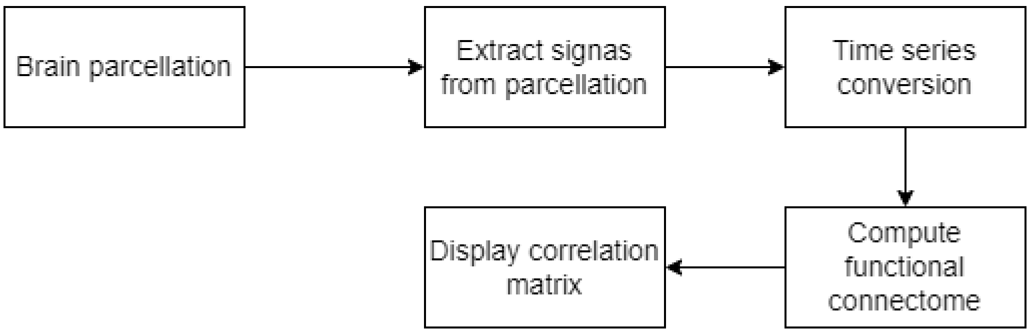
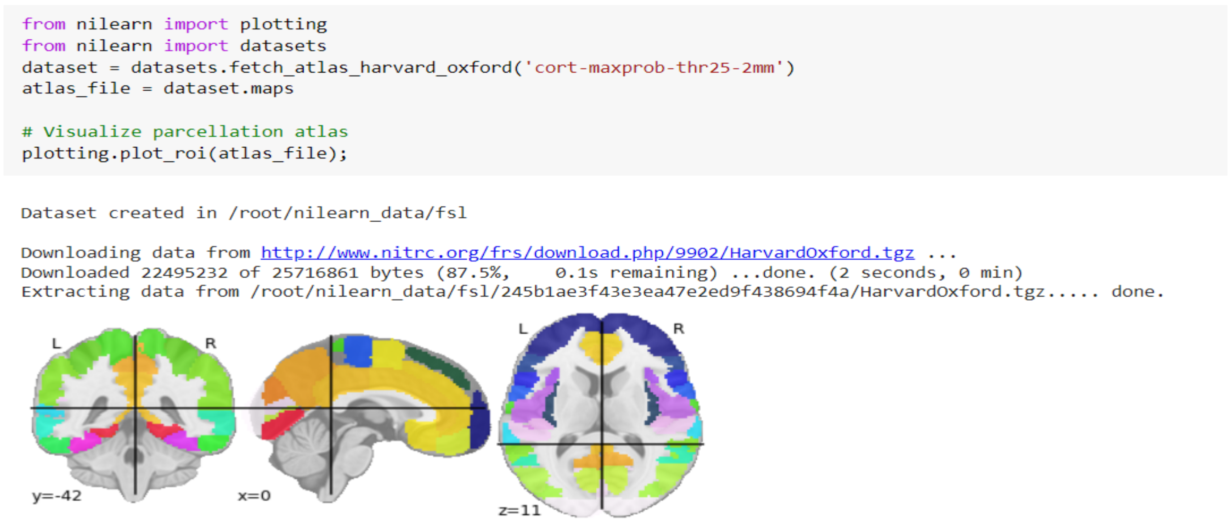

Brain parcellation
Extract signas from parcellation
Time series conversion
Compute functional connectome
Display correlation matrix

Figure 3.10 Functional Connectivity Computation

IMPLEMENTATION

Figure 3.11 shows the parcellation of brain. Brain parcellation refers to the process of dividing the brain into distinct regions or parcels based on specific criteria, such as anatomical, functional, or connectivity properties. This code imports the datasets module from the Nilearn library and then uses the fetch_atlas_harvard_oxford function to download the Harvard-Oxford parcellation atlas dataset. The 'cort-maxprob-thr25-2mm' parameter specifies the specific version of the atlas to download.

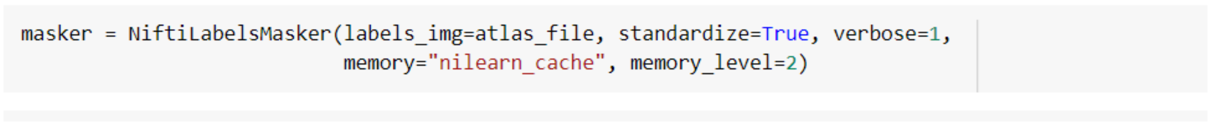


```
from nilearn import plotting
from nilearn import datasets
dataset = datasets.fetch_atlas_harvard_oxford('cort-maxprob-thr25-2mm')
atlas_file = dataset.maps

# Visualize parcellation atlas
plotting.plot_roi(atlas_file);
```

```
Dataset created in /root/nilearn_data/fsl

Downloading data from http://www.nitrc.org/frs/download.php/9902/HarvardOxford.tgz ...
Downloaded 22495232 of 25716861 bytes (87.5%,    0.1s remaining) ...done. (2 seconds, 0 min)
Extracting data from /root/nilearn_data/fsl/245b1ae3f43e3ea47e2ed9f438694f4a/HarvardOxford.tgz..... done.
```

Figure 3.11 Brain parcellation

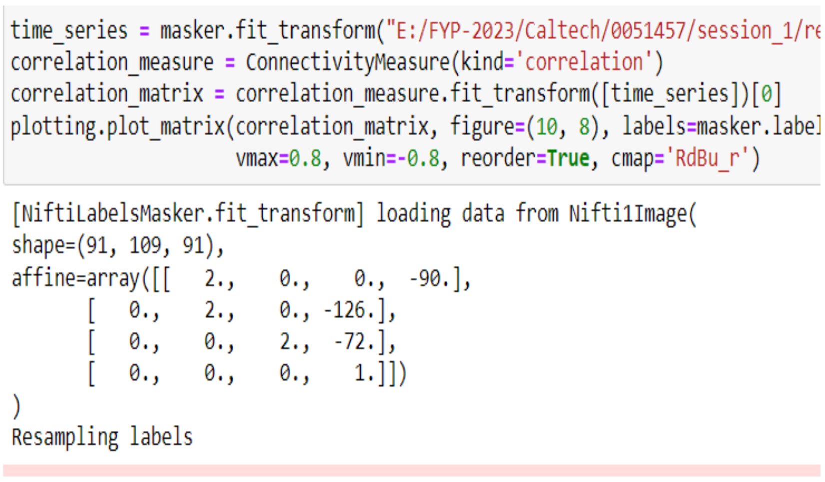
Figure 3.12 shows the extraction of signals from parcellation using nilearn.maskers.NiftiLabelsMasker.



```
masker = NiftiLabelsMasker(labels_img=atlas_file, standardize=True, verbose=1,
                           memory="nilearn_cache", memory_level=2)
```

Figure 3.12 Extract signal from parcellation

Figure 3.13 implements the time series conversion and then the correlation matrix is computed and plotted.



```
time_series = masker.fit_transform("E:/FYP-2023/Caltech/0051457/session_1/re
correlation_measure = ConnectivityMeasure(kind='correlation')
correlation_matrix = correlation_measure.fit_transform([time_series])[0]
plotting.plot_matrix(correlation_matrix, figure=(10, 8), labels=masker.label
                     vmax=0.8, vmin=-0.8, reorder=True, cmap='RdBu_r')
```

```
[NiftiLabelsMasker.fit_transform] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [   0.,    2.,    0., -126.],
       [   0.,    0.,    2.,  -72.],
       [   0.,    0.,    0.,    1.]])
)
Resampling labels
```

Figure 3.13 Compute & display correlation matrix

3.3.5 FEATURE EXTRACTION USING COMBAT HARMONIZATION

Figure 3.14 shows the module design of Feature Extraction Using COMBAT Harmonization. ComBat is a popular method used to harmonize data from different sources, such as different imaging centres, to reduce unwanted variation and improve the quality of analysis. In the context of ASD prediction, it is used to address the issue of site-to-site variability in functional connectivity data.

ALGORITHM STEPS

- Create the data matrix which contains the features to be harmonized along with the correlation matrix.
    - ✓ Compute mask
    - ✓ Flatten matrix to form autism_features and control_features matrix.
- Combine the autism and control features.
- Convert to functional_connectivity_features.csv with the combined features.
- Replace the dx_grp by 1 and 0.
- Convert to features_df.csv after replacing.
- Define the features & labels.
- Create combat harmonization model
- Fit the model.

INPUT: data matrix & correlation matrix

OUTPUT: features_df.csv

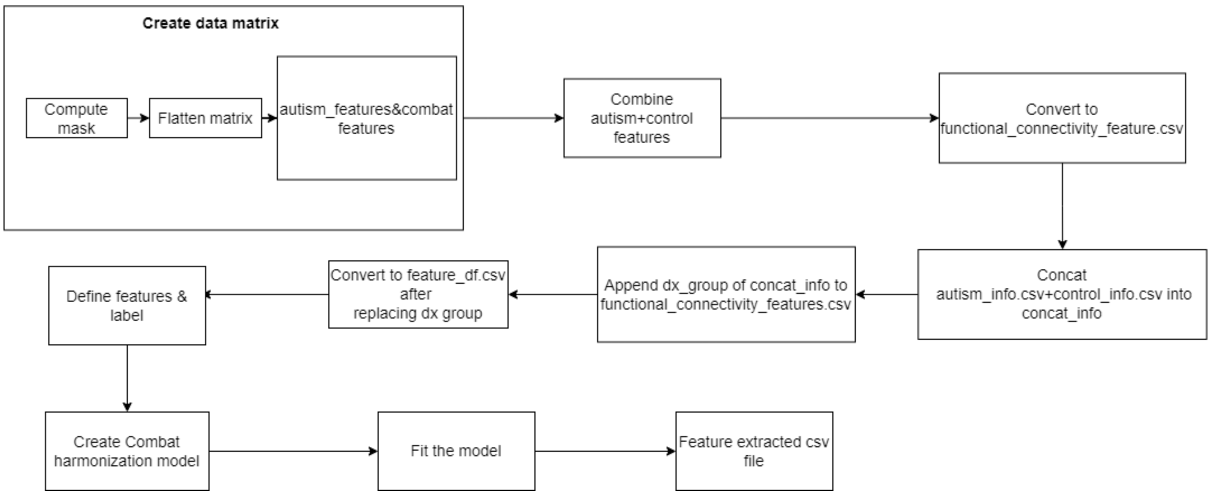
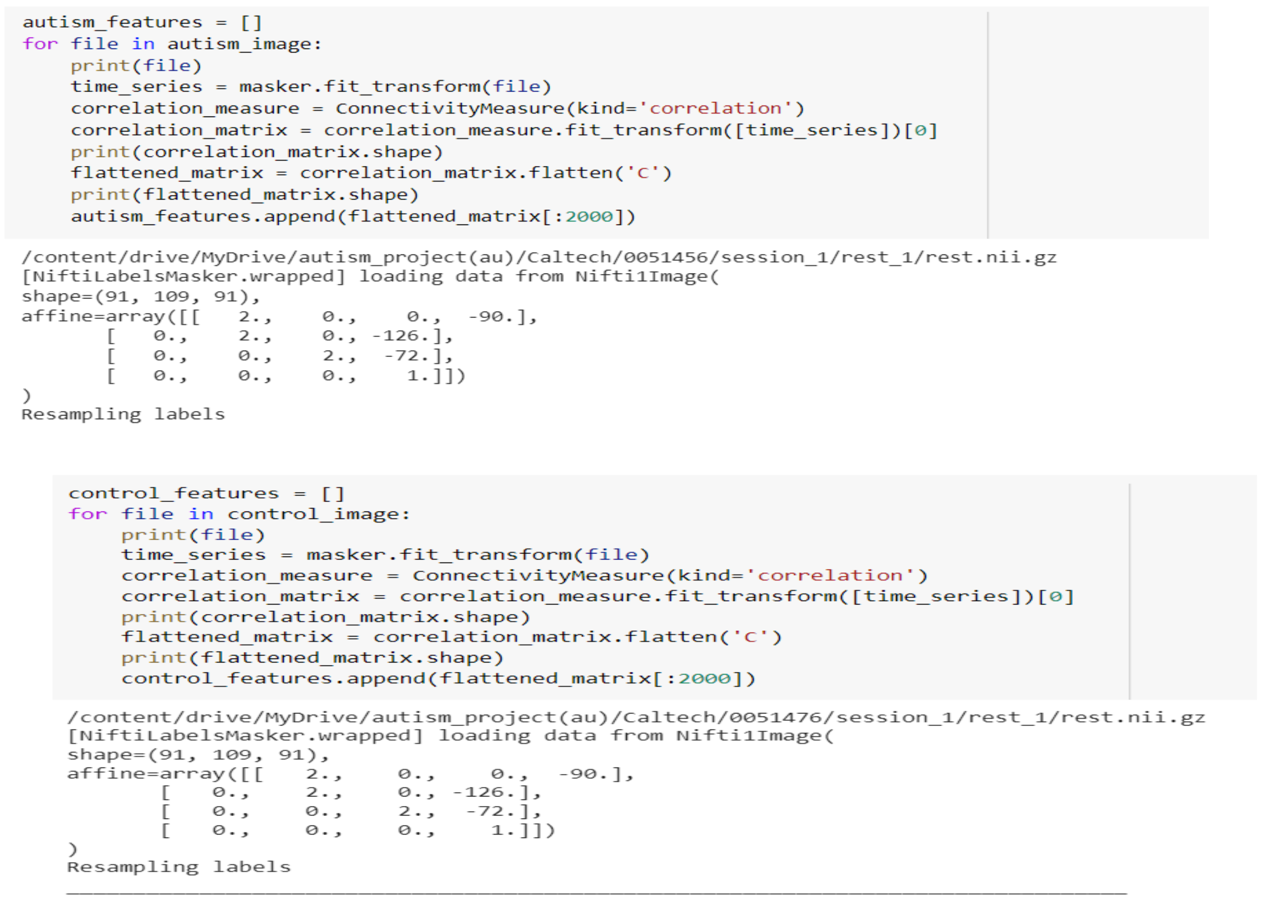

Create data matrix
Compute mask
Flatten matrix
autism_features&combat features
Combine autism+control features
Convert to functional_connectivity_feature.csv
Concat autism_info.csv+control_info.csv into concat_info
Append dx_group of concat_info to functional_connectivity_features.csv
Convert to feature_df.csv after replacing dx group
Define features & label
Create Combat harmonization model
Fit the model
Feature extracted csv file

Figure 3.14 Feature extraction using Combat Harmonization

IMPLEMENTATION

Figure 3.15 shows the creation of data matrix for extracting the features. The autism and control features are obtained by computing the mask and then the functional connectivity matrix is computed, and the matrix is flattened for each autism and control image respectively and appended to autism_features and control_features array respectively.



```
autism_features = []
for file in autism_image:
    print(file)
    time_series = masker.fit_transform(file)
    correlation_measure = ConnectivityMeasure(kind='correlation')
    correlation_matrix = correlation_measure.fit_transform([time_series])[0]
    print(correlation_matrix.shape)
    flattened_matrix = correlation_matrix.flatten('C')
    print(flattened_matrix.shape)
    autism_features.append(flattened_matrix[:2000])
```

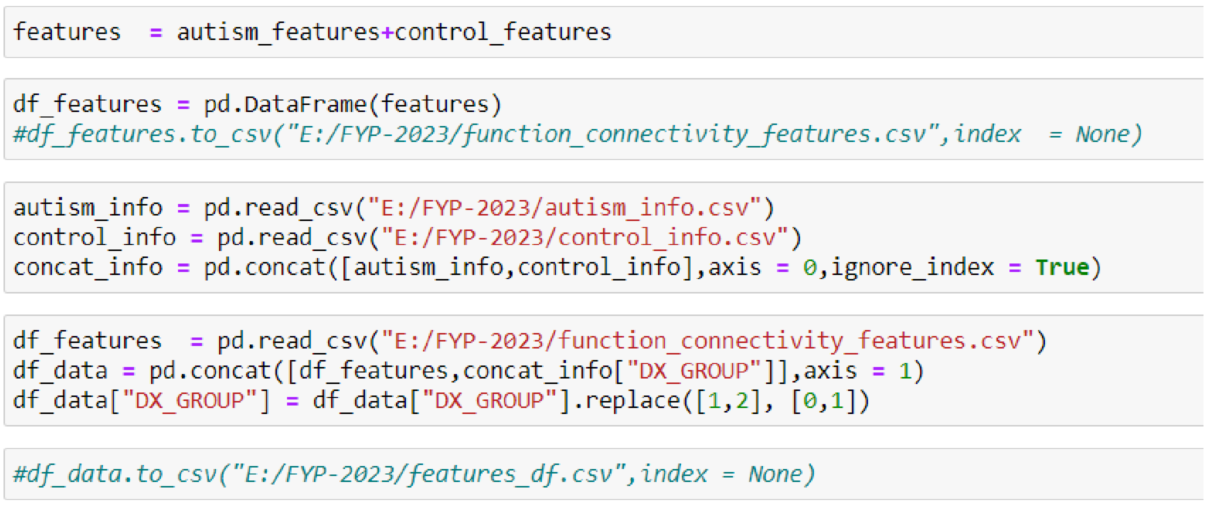
```
/content/drive/MyDrive/autism_project(au)/Caltech/0051456/session_1/rest_1/rest.nii.gz
[NiftiLabelsMasker.wrapped] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [   0.,    2.,    0., -126.],
       [   0.,    0.,    2.,  -72.],
       [   0.,    0.,    0.,    1.]])
)
Resampling labels
```

```
control_features = []
for file in control_image:
    print(file)
    time_series = masker.fit_transform(file)
    correlation_measure = ConnectivityMeasure(kind='correlation')
    correlation_matrix = correlation_measure.fit_transform([time_series])[0]
    print(correlation_matrix.shape)
    flattened_matrix = correlation_matrix.flatten('C')
    print(flattened_matrix.shape)
    control_features.append(flattened_matrix[:2000])
```

```
/content/drive/MyDrive/autism_project(au)/Caltech/0051476/session_1/rest_1/rest.nii.gz
[NiftiLabelsMasker.wrapped] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [   0.,    2.,    0., -126.],
       [   0.,    0.,    2.,  -72.],
       [   0.,    0.,    0.,    1.]])
)
Resampling labels
```

Figure 3.15 Data matrix creation

In figure 3.16 the extracted autism and control features are concated and converted to functional_connectivity_features.csv. Then the autism_info and control_info are concated into concat_info and the dx_group in concat_info is appended with functional_connectivity_features.csv. In functional_connectivity_features.csv the dx_group are replaced with 0 and 1. After replacing it is finally converted into features_df.csv.



```
features  = autism_features+control_features
```

```
df_features = pd.DataFrame(features)
#df_features.to_csv("E:/FYP-2023/function_connectivity_features.csv",index  = None)
```

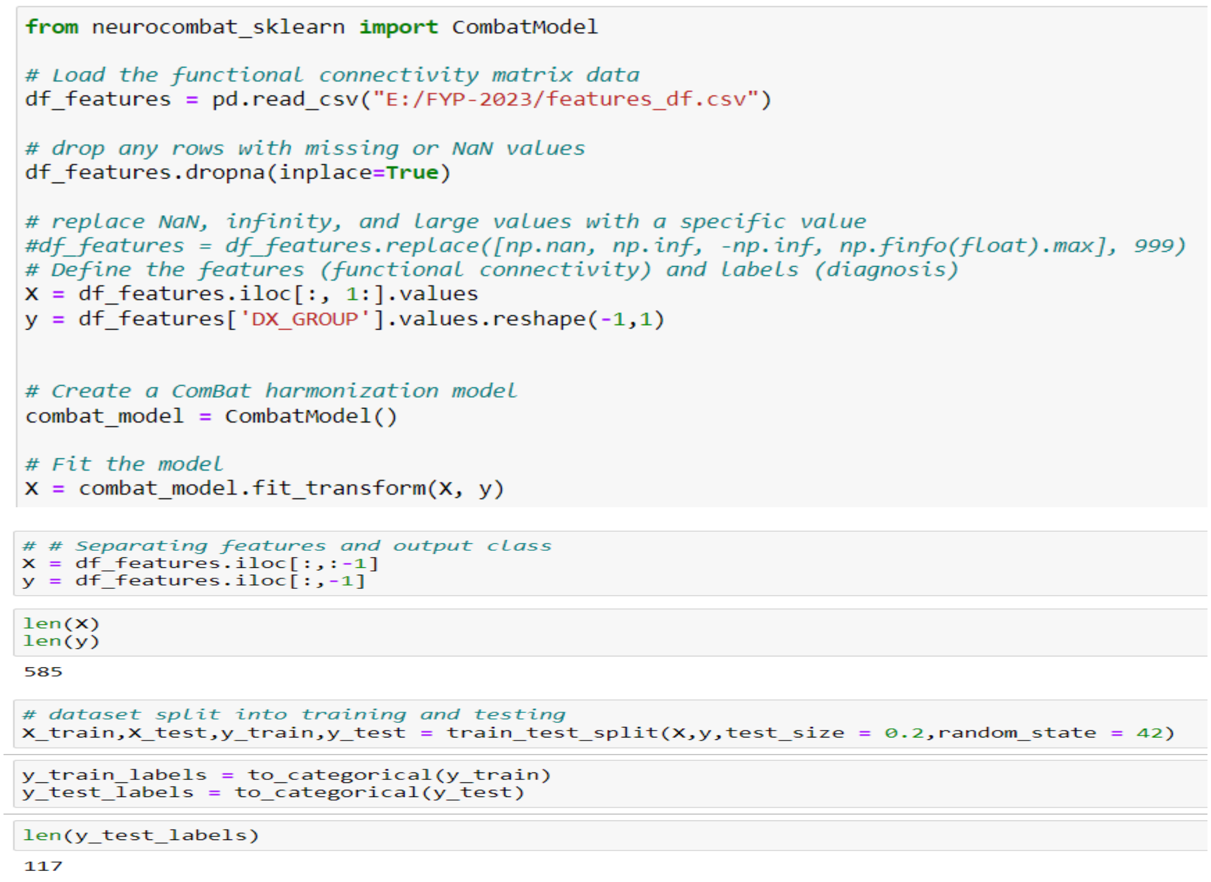
```
autism_info = pd.read_csv("E:/FYP-2023/autism_info.csv")
control_info = pd.read_csv("E:/FYP-2023/control_info.csv")
concat_info = pd.concat([autism_info,control_info],axis = 0,ignore_index = True)
```

```
df_features  = pd.read_csv("E:/FYP-2023/function_connectivity_features.csv")
df_data = pd.concat([df_features,concat_info["DX_GROUP"]],axis = 1)
df_data["DX_GROUP"] = df_data["DX_GROUP"].replace([1,2], [0,1])
```

```
#df_data.to_csv("E:/FYP-2023/features_df.csv",index = None)
```

Figure 3.16 features_df.csv file generation

Figure 3.17 the features_df.csv file read and a combat harmonization model is created and then the model is fitted. The features and the labels are separated. The train and test set ate splitted. The total number of images are 585.



```
from neurocombat_sklearn import CombatModel

# Load the functional connectivity matrix data
df_features = pd.read_csv("E:/FYP-2023/features_df.csv")

# drop any rows with missing or NaN values
df_features.dropna(inplace=True)

# replace NaN, infinity, and large values with a specific value
#df_features = df_features.replace([np.nan, np.inf, -np.inf, np.finfo(float).max], 999)
# Define the features (functional connectivity) and labels (diagnosis)
X = df_features.iloc[:, 1:].values
y = df_features['DX_GROUP'].values.reshape(-1,1)


# Create a ComBat harmonization model
combat_model = CombatModel()

# Fit the model
X = combat_model.fit_transform(X, y)
```

```
# # Separating features and output class
x = df_features.iloc[:,:-1]
y = df_features.iloc[:,-1]
```

```
len(x)
len(y)
```

```
585
```

```
# dataset split into training and testing
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 42)
```

```
y_train_labels = to_categorical(y_train)
y_test_labels = to_categorical(y_test)
```

```
len(y_test_labels)
```

```
117
```

Figure 3.17 Combat harmonization

3.3.6 TRAINING USING ML & DL ALGORITHMS (SVM, ANN, CNN)

Figure 3.18 shows the module design of training using machine learning and deep learning algorithms.

ALGORITHM STEPS

- Separate features & labels.
- Split the dataset into train and testing sets.
- Train the data with SVM, ANN and CNN algorithms.
- Evaluate the results.
- A model file will be generated.

INPUT: features_df.csv

OUTPUT: ann_weight.h5(ANN), svm_weight.p, cnn_weight.h5

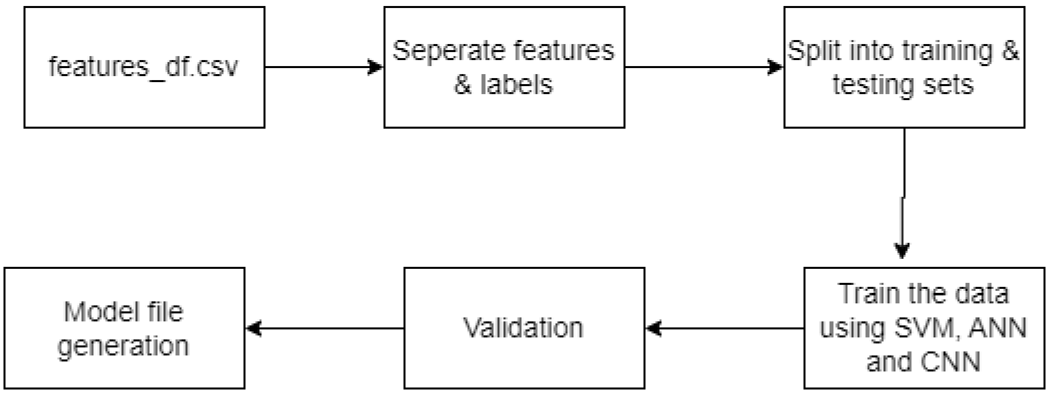

features_df.csv
Seperate features & labels
Split into training & testing sets
Train the data using SVM, ANN and CNN
Validation
Model file generation

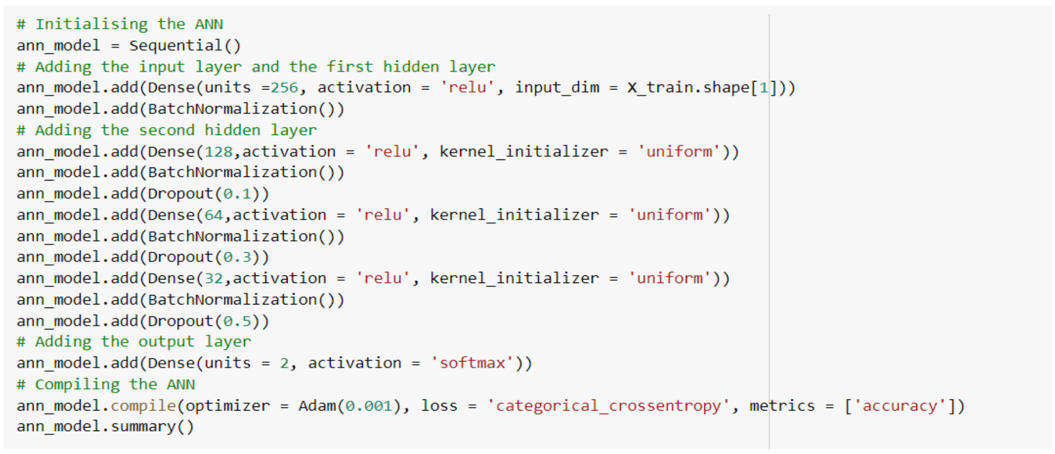
Figure 3.18 Training with algorithms

IMPLEMENTATION

TRAIN USING ANN

Figure 3.19 shows the initialization of ANN model by adding the input, hidden and output layer and compiled finally.

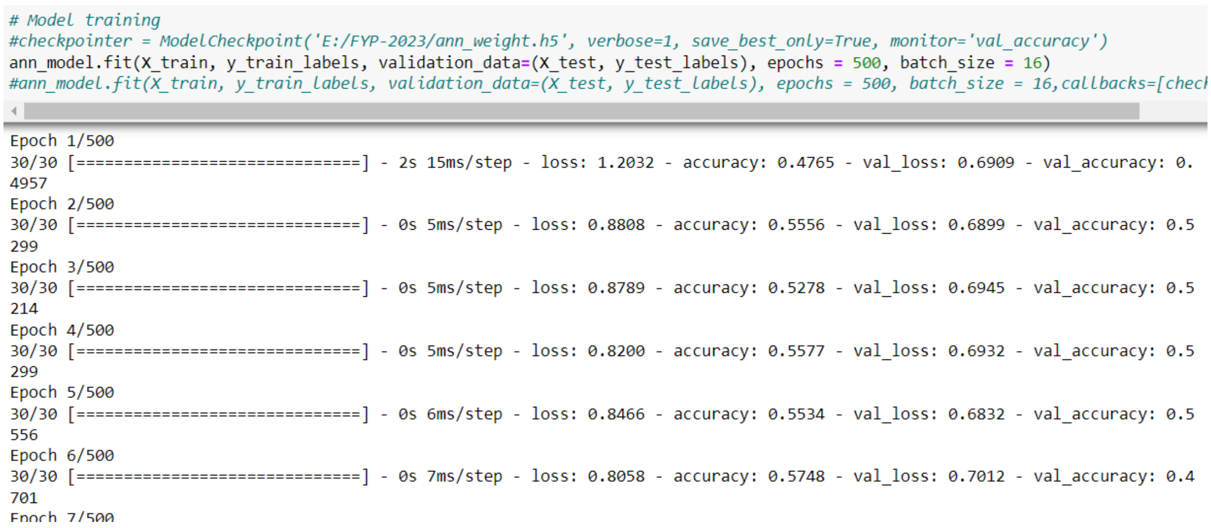
The ann_model.summary() is used to display the summary of the ANN model.



```
# Initialising the ANN
ann_model = Sequential()
# Adding the input layer and the first hidden layer
ann_model.add(Dense(units =256, activation = 'relu', input_dim = X_train.shape[1]))
ann_model.add(BatchNormalization())
# Adding the second hidden layer
ann_model.add(Dense(128,activation = 'relu', kernel_initializer = 'uniform'))
ann_model.add(BatchNormalization())
ann_model.add(Dropout(0.1))
ann_model.add(Dense(64,activation = 'relu', kernel_initializer = 'uniform'))
ann_model.add(BatchNormalization())
ann_model.add(Dropout(0.3))
ann_model.add(Dense(32,activation = 'relu', kernel_initializer = 'uniform'))
ann_model.add(BatchNormalization())
ann_model.add(Dropout(0.5))
# Adding the output layer
ann_model.add(Dense(units = 2, activation = 'softmax'))
# Compiling the ANN
ann_model.compile(optimizer = Adam(0.001), loss = 'categorical_crossentropy', metrics = ['accuracy'])
ann_model.summary()
```

Figure 3.19 Initializing ANN model

Figure 3.20 shows the training of the dataset using fit function and the model file - ann_weight.h5 is generated.



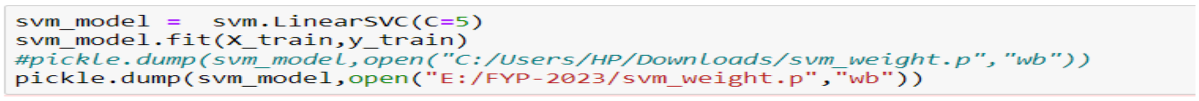
```
# Model training
#checkpointer = ModelCheckpoint('E:/FYP-2023/ann_weight.h5', verbose=1, save_best_only=True, monitor='val_accuracy')
ann_model.fit(X_train, y_train_labels, validation_data=(X_test, y_test_labels), epochs = 500, batch_size = 16)
#ann_model.fit(X_train, y_train_labels, validation_data=(X_test, y_test_labels), epochs = 500, batch_size = 16,callbacks=[checl
```

```
Epoch 1/500
30/30 [==============================] - 2s 15ms/step - loss: 1.2032 - accuracy: 0.4765 - val_loss: 0.6909 - val_accuracy: 0.
4957
Epoch 2/500
30/30 [==============================] - 0s 5ms/step - loss: 0.8808 - accuracy: 0.5556 - val_loss: 0.6899 - val_accuracy: 0.5
299
Epoch 3/500
30/30 [==============================] - 0s 5ms/step - loss: 0.8789 - accuracy: 0.5278 - val_loss: 0.6945 - val_accuracy: 0.5
214
Epoch 4/500
30/30 [==============================] - 0s 5ms/step - loss: 0.8200 - accuracy: 0.5577 - val_loss: 0.6932 - val_accuracy: 0.5
299
Epoch 5/500
30/30 [==============================] - 0s 6ms/step - loss: 0.8466 - accuracy: 0.5534 - val_loss: 0.6832 - val_accuracy: 0.5
556
Epoch 6/500
30/30 [==============================] - 0s 7ms/step - loss: 0.8058 - accuracy: 0.5748 - val_loss: 0.7012 - val_accuracy: 0.4
701
Epoch 7/500
```

Figure 3.20 Train using ANN model

TRAIN USING SVM

Figure 3.21 shows the training of dataset using SVM model and “svm_weight.p” model file is generated .

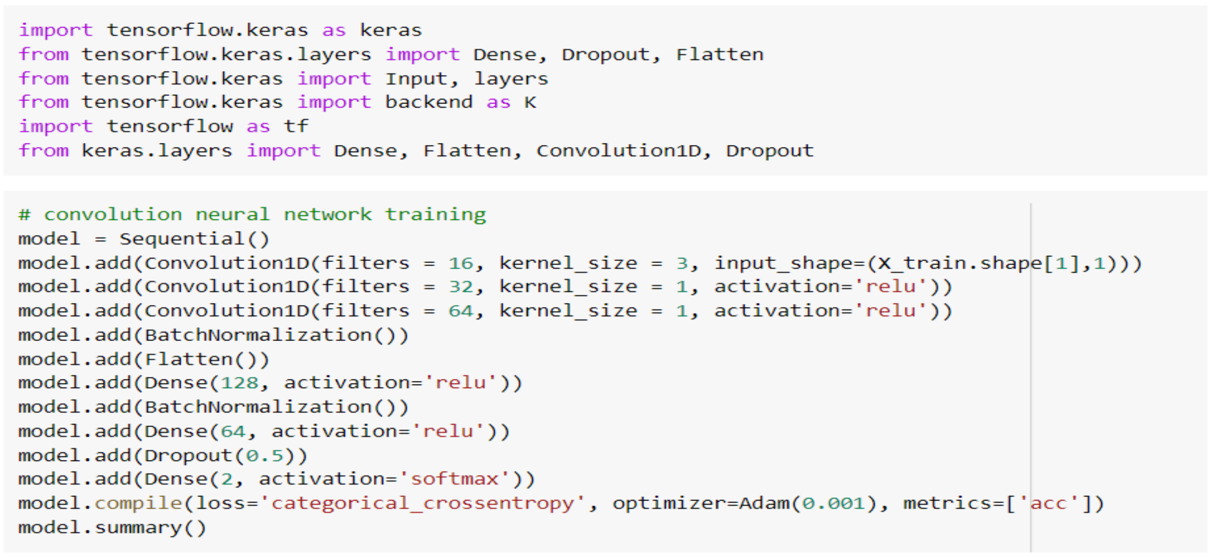


```
svm_model =  svm.LinearSVC(C=5)
svm_model.fit(X_train,y_train)
#pickle.dump(svm_model,open("C:/Users/HP/Downloads/svm_weight.p","wb"))
pickle.dump(svm_model,open("E:/FYP-2023/svm_weight.p","wb"))
```

Figure 3.21 Train using SVM

TRAIN USING CNN

Figure 3.22 shows the initialization of CNN model. It consists of three convolutional layers followed by two fully connected layers. The model.summary() displays the summary of CNN model.

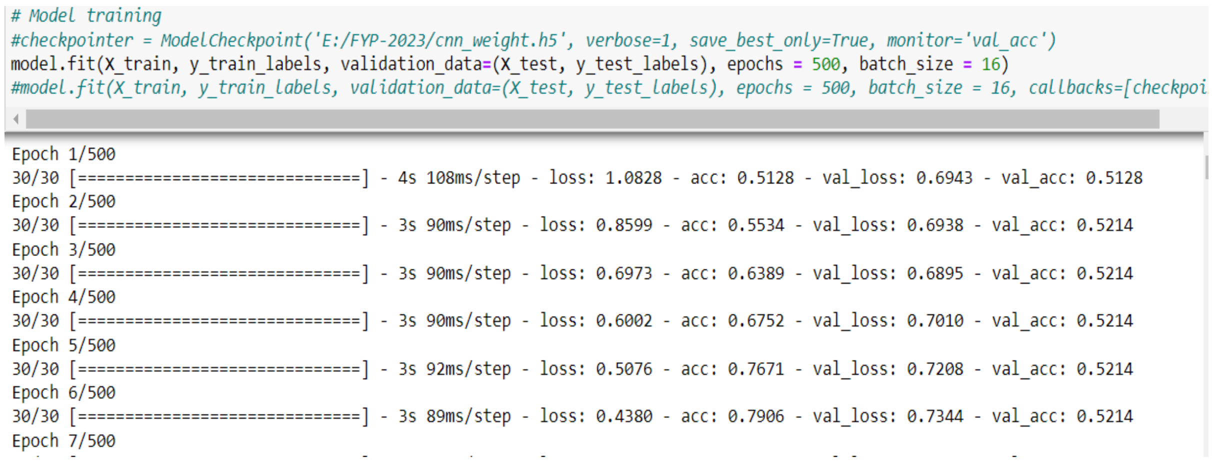


```
import tensorflow.keras as keras
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras import Input, layers
from tensorflow.keras import backend as K
import tensorflow as tf
from keras.layers import Dense, Flatten, Convolution1D, Dropout
```

```
# convolution neural network training
model = Sequential()
model.add(Convolution1D(filters = 16, kernel_size = 3, input_shape=(X_train.shape[1],1)))
model.add(Convolution1D(filters = 32, kernel_size = 1, activation='relu'))
model.add(Convolution1D(filters = 64, kernel_size = 1, activation='relu'))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(0.001), metrics=['acc'])
model.summary()
```

Figure 3.22 Initialising CNN model

Figure 3.23 shows the training of the dataset using fit function and the model file - cnn_weight.h5 is generated.



```
# Model training
#checkpointer = ModelCheckpoint('E:/FYP-2023/cnn_weight.h5', verbose=1, save_best_only=True, monitor='val_acc')
model.fit(X_train, y_train_labels, validation_data=(X_test, y_test_labels), epochs = 500, batch_size = 16)
#model.fit(X_train, y_train_labels, validation_data=(X_test, y_test_labels), epochs = 500, batch_size = 16, callbacks=[checkpoi
```

```
Epoch 1/500
30/30 [==============================] - 4s 108ms/step - loss: 1.0828 - acc: 0.5128 - val_loss: 0.6943 - val_acc: 0.5128
Epoch 2/500
30/30 [==============================] - 3s 90ms/step - loss: 0.8599 - acc: 0.5534 - val_loss: 0.6938 - val_acc: 0.5214
Epoch 3/500
30/30 [==============================] - 3s 90ms/step - loss: 0.6973 - acc: 0.6389 - val_loss: 0.6895 - val_acc: 0.5214
Epoch 4/500
30/30 [==============================] - 3s 90ms/step - loss: 0.6002 - acc: 0.6752 - val_loss: 0.7010 - val_acc: 0.5214
Epoch 5/500
30/30 [==============================] - 3s 92ms/step - loss: 0.5076 - acc: 0.7671 - val_loss: 0.7208 - val_acc: 0.5214
Epoch 6/500
30/30 [==============================] - 3s 89ms/step - loss: 0.4380 - acc: 0.7906 - val_loss: 0.7344 - val_acc: 0.5214
Epoch 7/500
```

Figure 3.23 Train using CNN model

3.3.7 PREDICTION

Figure 3.24 shows the module design of prediction which identifies the presence of autism spectrum disorder or typically developing controls.

ALGORITHM STEPS

- Define ‘feature_extract’ function.
- An image will be given as input as file path.
- Predict the disease based on the model file generated.

INPUT: Input path

OUTPUT: Presence of autism / normal group.

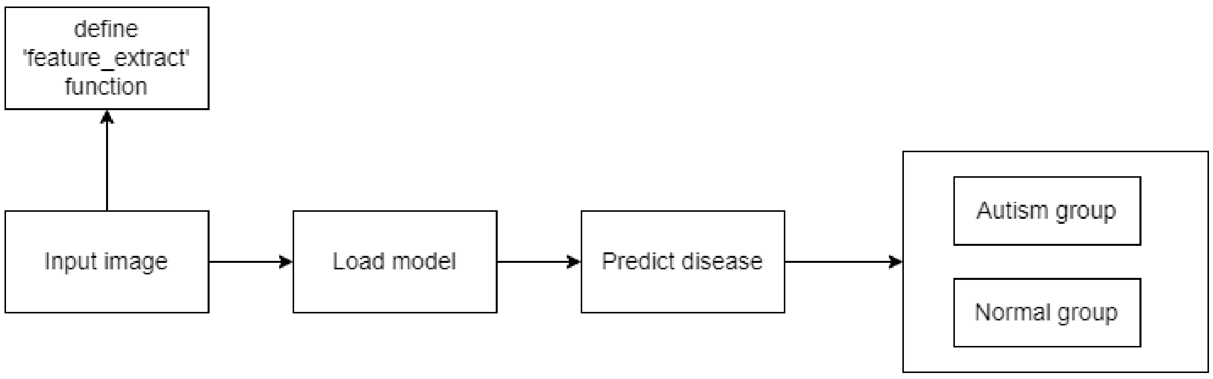
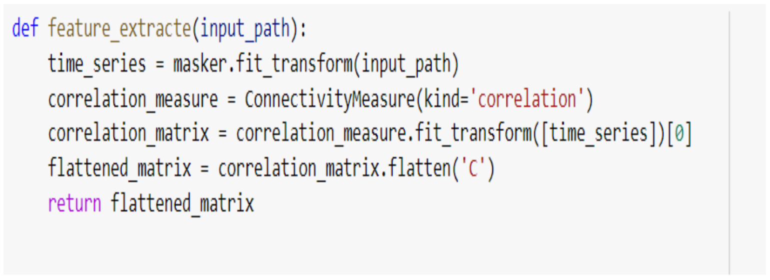


Figure 3.24 Prediction

IMPLEMENTATION

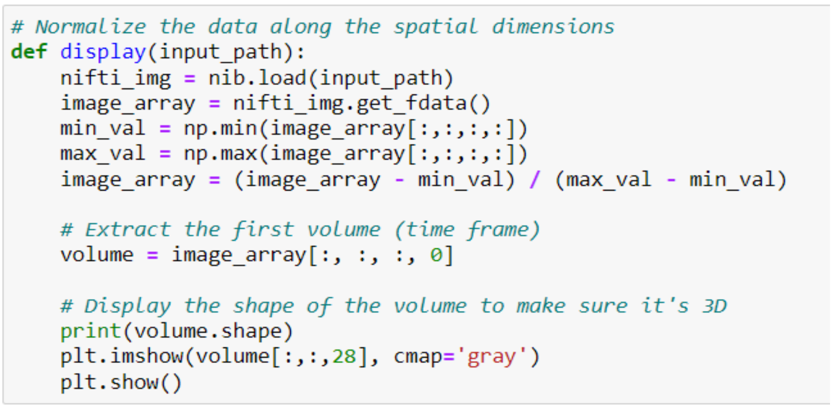
Figure 3.25 defines the feature_extracte function in which the input path is passed as the parameter. The correlation matrix is computed and flattened for the given input image path.



```python
def feature_extracte(input_path):
    time_series = masker.fit_transform(input_path)
    correlation_measure = ConnectivityMeasure(kind='correlation')
    correlation_matrix = correlation_measure.fit_transform([time_series])[0]
    flattened_matrix = correlation_matrix.flatten('C')
    return flattened_matrix
```

Figure 3.25 Define feature_extracte()

Figure 3.26 shows the definition if display function where the input path is passed as parameter. The data is normalized along the spatial dimensions. Then the first volume is extracted and finally the shape of the volume is displayed to make sure it’s 3D and then displayed.



```python
# Normalize the data along the spatial dimensions
def display(input_path):
    nifti_img = nib.load(input_path)
    image_array = nifti_img.get_fdata()
    min_val = np.min(image_array[:,:,:,:])
    max_val = np.max(image_array[:,:,:,:])
    image_array = (image_array - min_val) / (max_val - min_val)

    # Extract the first volume (time frame)
    volume = image_array[:, :, :, 0]

    # Display the shape of the volume to make sure it's 3D
    print(volume.shape)
    plt.imshow(volume[:,:,28], cmap='gray')
    plt.show()
```

Figure 3.26 Define display()

CHAPTER 4

RESULTS AND DISCUSSION

Results and discussion are essential components of any project as they communicate the outcomes of the study and provide an interpretation and analysis of the findings. The results and discussion sections are crucial in providing a clear and comprehensive understanding of the study's outcomes, its significance, and its potential implications.

4.1 DATASET DESCRIPTION

The Autism Brain Imaging Data Exchange I (ABIDE I) represents the first ABIDE initiative. ABIDE-I encompasses rs-fMRI and T1 structural brain images that were acquired at 13 sites. The image acquisition parameters and protocol information can be found at https://fcon_1000.projects.nitrc.org/indi/abide. A total of 585 MRI images have been acquired from these 13 sites. The phenotypic- csv files have been made used to locate the subsid's of the image.

Images representing AUTISM: (Figure 4.1, Figure 4.2)

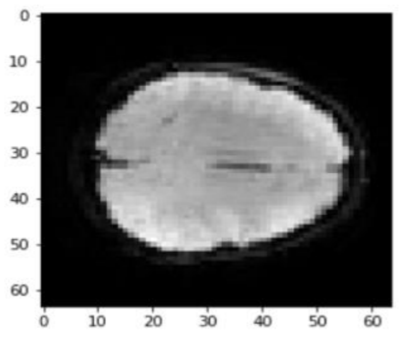


Figure 4.1

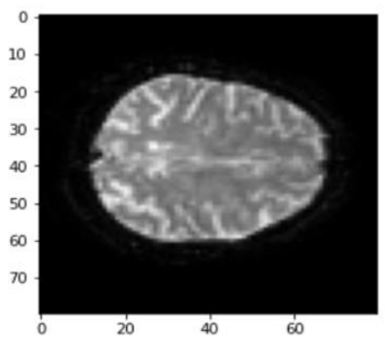


Figure 4.2

Images representing CONTROL: (Figure 4.3, Figure 4.4)

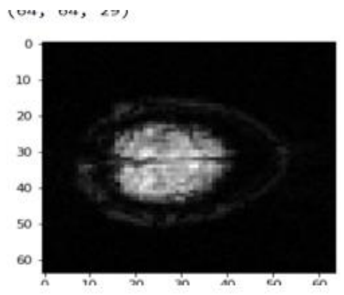


Figure 4.3

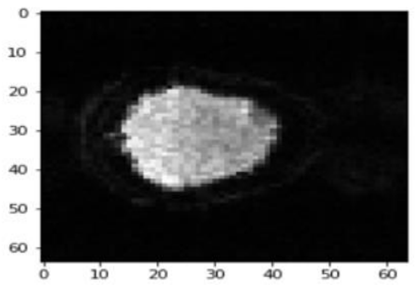


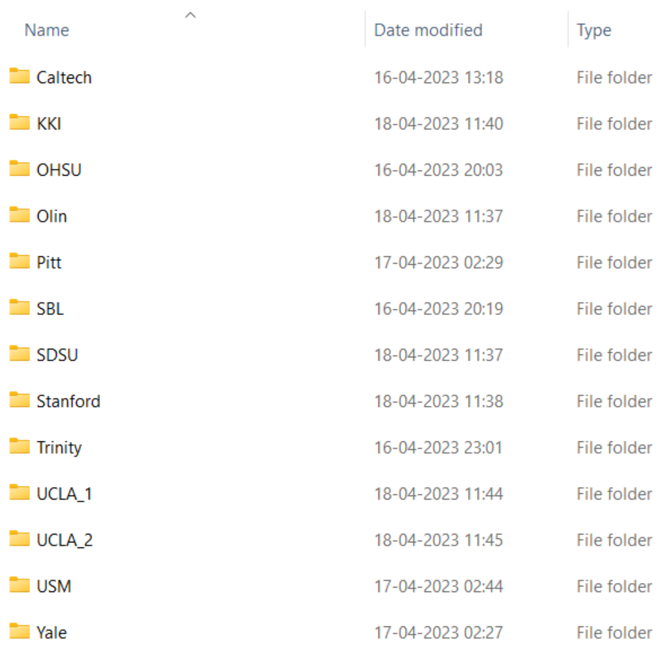
Figure 4.4

4.2 RESULTS

Results discusses about the practical results of each module obtained while implementing the project.

Module 1-Dataset Collection

The dataset has been collected for the project and the figure 4.5 shows the extracted datasets after calling the "extract" function.



Name	Date modified	Type
Caltech	16-04-2023 13:18	File folder
KKI	18-04-2023 11:40	File folder
OHSU	16-04-2023 20:03	File folder
Olin	18-04-2023 11:37	File folder
Pitt	17-04-2023 02:29	File folder
SBL	16-04-2023 20:19	File folder
SDSU	18-04-2023 11:37	File folder
Stanford	18-04-2023 11:38	File folder
Trinity	16-04-2023 23:01	File folder
UCLA_1	18-04-2023 11:44	File folder
UCLA_2	18-04-2023 11:45	File folder
USM	17-04-2023 02:44	File folder
Yale	17-04-2023 02:27	File folder

Figure 4.5 Extracted datasets

Module 2- Dataset Pre-processing

Figure 4.6 visualizes the resting 4d Fmri brain image after normalizing the images using MinMaxScalar().

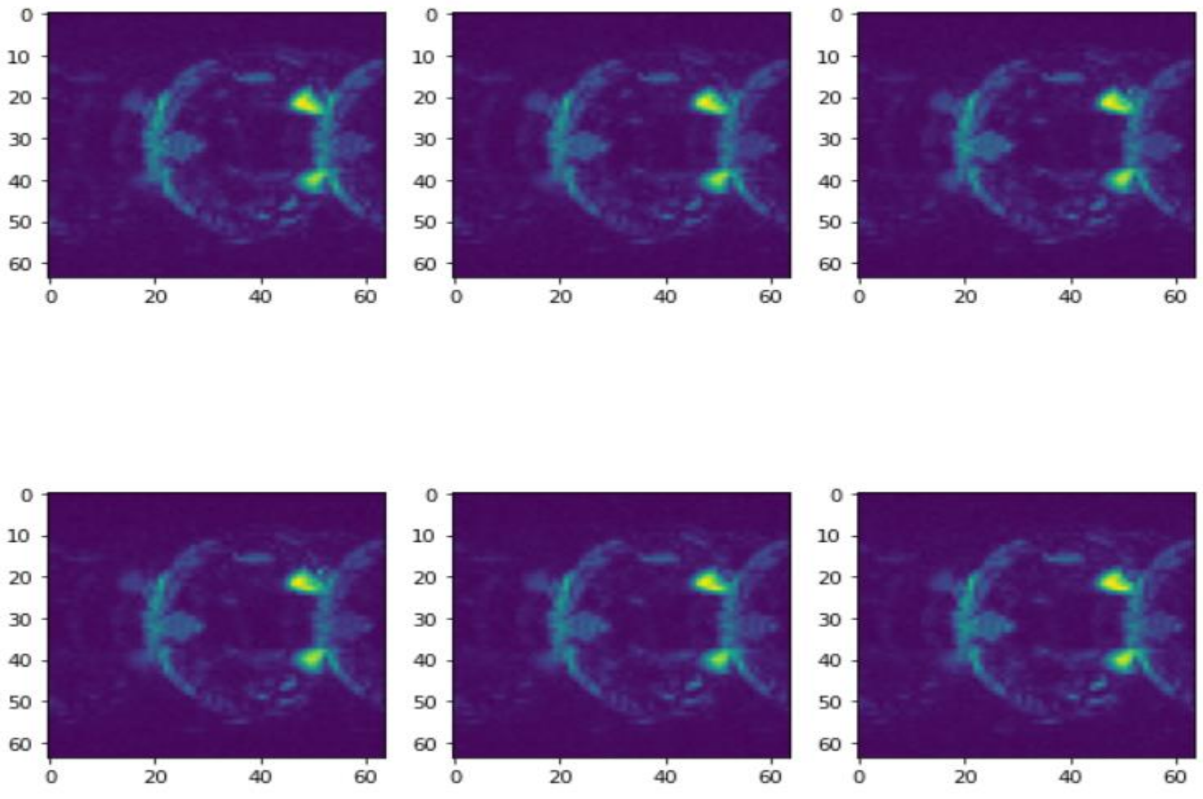
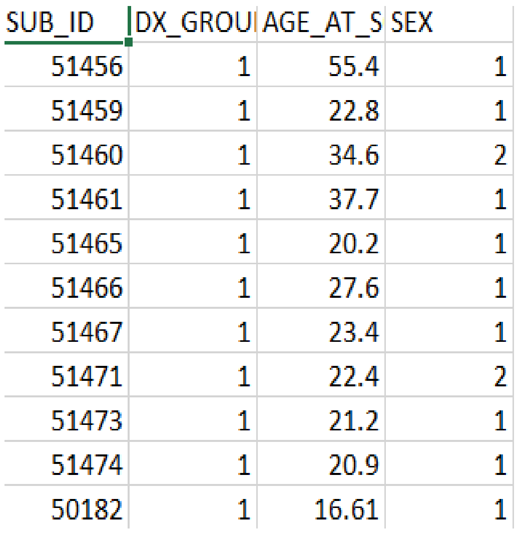


Figure 4.6 View resting 4d Fmri brain image

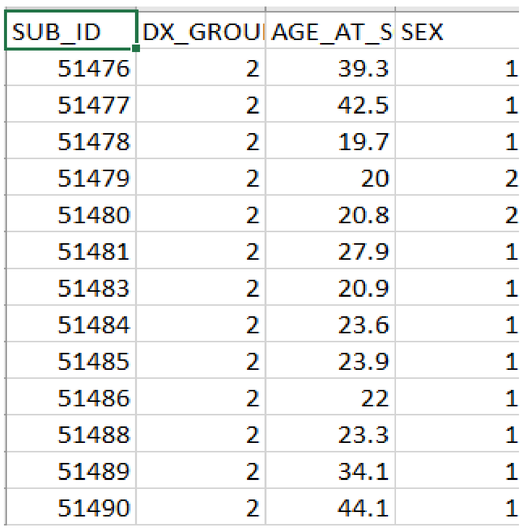
Module 3- Autism and Control group separation

Figures 4.7 and 4.8 display a few rows of the autism_info.csv and control_info.csv file after concating from various university sites based on the diagnosis in phenotypic data.



SUB_ID	DX_GROU	AGE_AT_S	SEX
51456	1	55.4	1
51459	1	22.8	1
51460	1	34.6	2
51461	1	37.7	1
51465	1	20.2	1
51466	1	27.6	1
51467	1	23.4	1
51471	1	22.4	2
51473	1	21.2	1
51474	1	20.9	1
50182	1	16.61	1

Figure 4.7 autism_info.csv file



SUB_ID	DX_GROU	AGE_AT_S	SEX
51476	2	39.3	1
51477	2	42.5	1
51478	2	19.7	1
51479	2	20	2
51480	2	20.8	2
51481	2	27.9	1
51483	2	20.9	1
51484	2	23.6	1
51485	2	23.9	1
51486	2	22	1
51488	2	23.3	1
51489	2	34.1	1
51490	2	44.1	1

Figure 4.8 control_info.csv file

Module 4- Functional Connectivity computation

Figure 4.9 displays the functional connectivity of the image path specified.

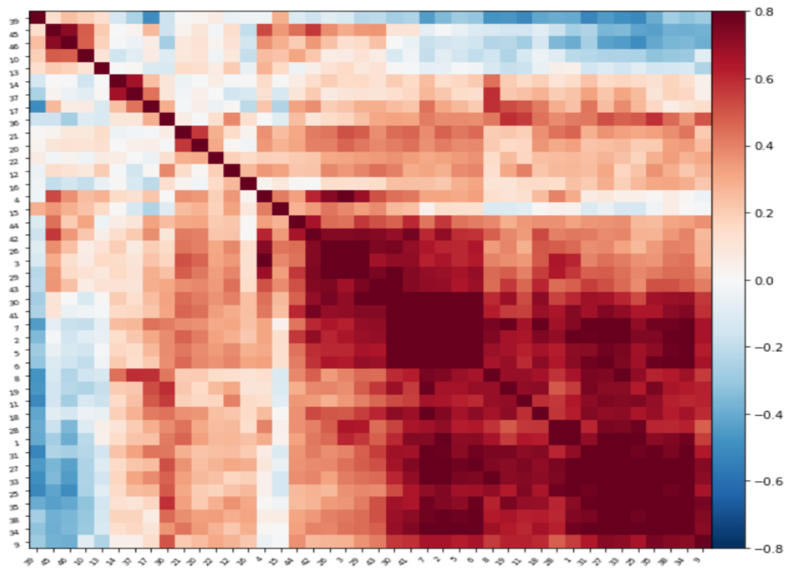
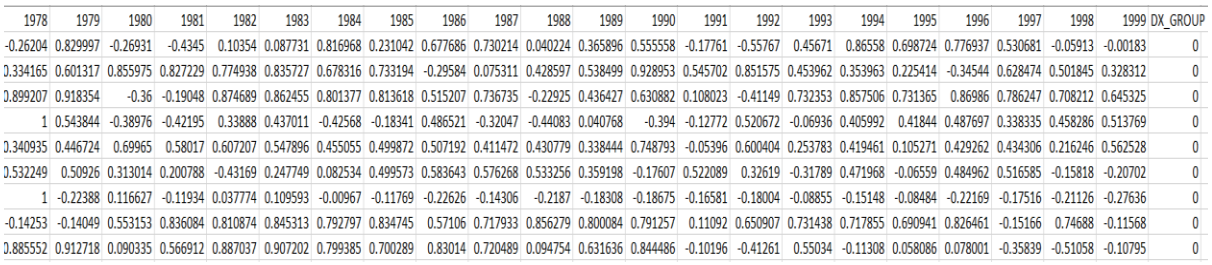


Figure 4.9 Functional connectivity matrix

Module 5-Feature extraction using Combat harmonization

Figure 4.10 displays a few rows of features_df.csv file which is generated after extracting the features from autism and control images.



1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	DX_GROUP
-0.26204	0.829997	-0.26931	-0.4345	0.10354	0.087731	0.816968	0.231042	0.677686	0.730214	0.040224	0.365896	0.555558	-0.17761	-0.55767	0.45671	0.86558	0.698724	0.776937	0.530681	-0.05913	-0.00183	0
.334165	0.601317	0.855975	0.827229	0.774938	0.835727	0.678316	0.733194	-0.29584	0.075311	0.428597	0.538499	0.928953	0.545702	0.851575	0.453962	0.353963	0.225414	-0.34544	0.628474	0.501845	0.328312	0
.899207	0.918354	-0.36	-0.19048	0.874689	0.862455	0.801377	0.813618	0.515207	0.736735	-0.22925	0.436427	0.630882	0.108023	-0.41149	0.732353	0.857506	0.731365	0.86986	0.786247	0.708212	0.645325	0
1	0.543844	-0.38976	-0.42195	0.33888	0.437011	-0.42568	-0.18341	0.486521	-0.32047	-0.44083	0.040768	-0.394	-0.12772	0.520672	-0.06936	0.405992	0.41844	0.487697	0.338335	0.458286	0.513769	0
.340935	0.446724	0.69965	0.58017	0.607207	0.547896	0.455055	0.499872	0.507192	0.411472	0.430779	0.338444	0.748793	-0.05396	0.600404	0.253783	0.419461	0.105271	0.429262	0.434306	0.216246	0.562528	0
.532249	0.50926	0.313014	0.200788	-0.43169	0.247749	0.082534	0.499573	0.583643	0.576268	0.533256	0.359198	-0.17607	0.522089	0.32619	-0.31789	0.471968	-0.06559	0.484962	0.516585	-0.15818	-0.20702	0
1	-0.22388	0.116627	-0.11934	0.037774	0.109593	-0.00967	-0.11769	-0.22626	-0.14306	-0.2187	-0.18308	-0.18675	-0.16581	-0.18004	-0.08855	-0.15148	-0.08484	-0.22169	-0.17516	-0.21126	-0.27636	0
-0.14253	-0.14049	0.553153	0.836084	0.810874	0.845313	0.792797	0.834745	0.57106	0.717933	0.856279	0.800084	0.791257	0.11092	0.650907	0.731438	0.717855	0.690941	0.826461	-0.15166	0.74688	-0.11568	0
.885552	0.912718	0.090335	0.566912	0.887037	0.907202	0.799385	0.700289	0.83014	0.720489	0.094754	0.631636	0.844486	-0.10196	-0.41261	0.55034	-0.11308	0.058086	0.078001	-0.35839	-0.51058	-0.10795	0

Figure 4.10 features_df.csv file

Module 6- Training using ML & DL algorithms (SVM, ANN, CNN):

ANN

Figure 4.11 displays the summary of ANN model. From figure 4.11 the following data can be inferred:

1.**Number of layers**: The model has 5 dense layers.

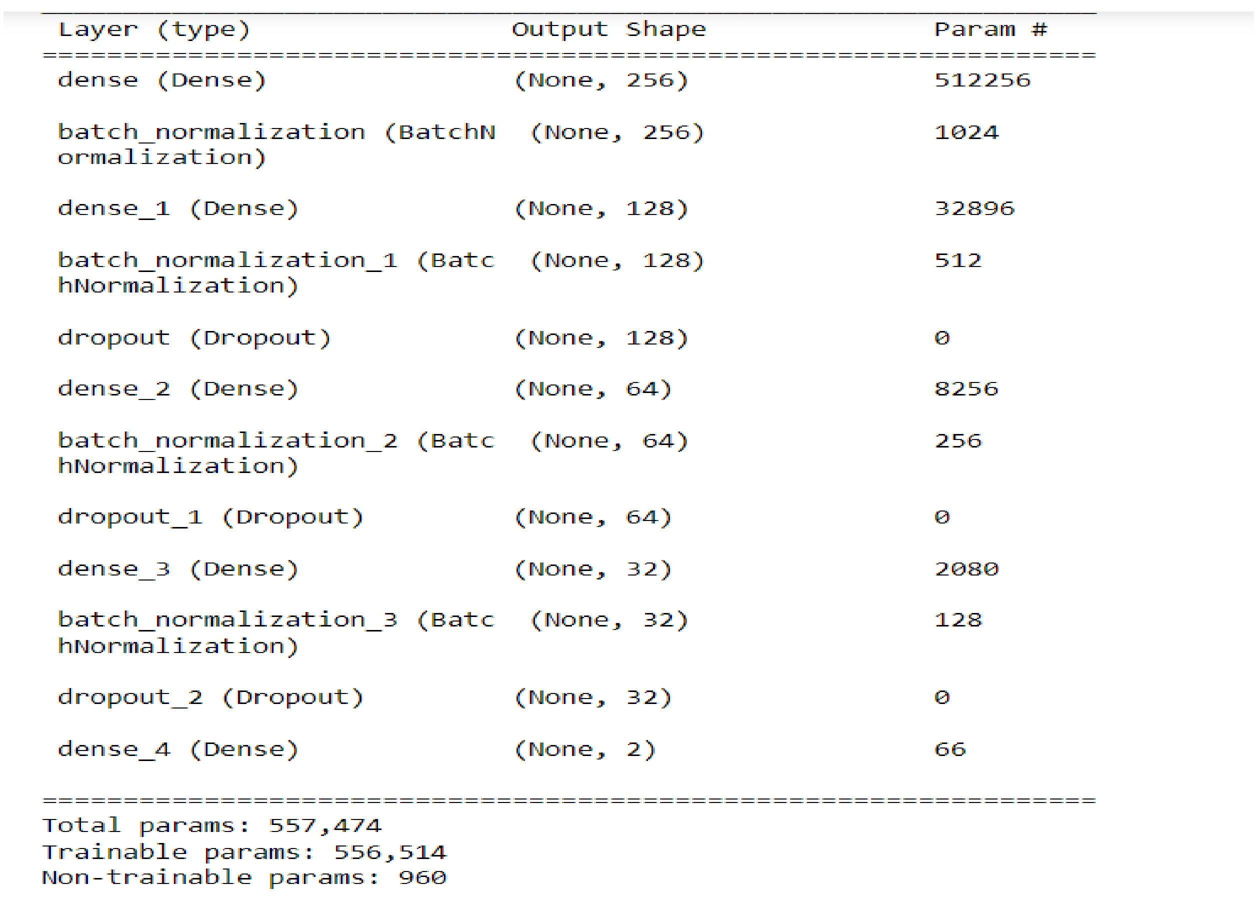
2.**Number of neurons in each layer**: The number of neurons in each layer is as follows: 256, 128, 64, 32, and 2.

3.**Activation function**: The activation function used in this model is ReLU (Rectified Linear Unit) activation function for all the dense layers.

4.**Batch normalization**: The model uses batch normalization after each dense layer, which helps in faster convergence and generalization.

5.**Dropout rate:** The model uses dropout with a rate of 0.5 after the second, third, and fourth dense layers to reduce overfitting.

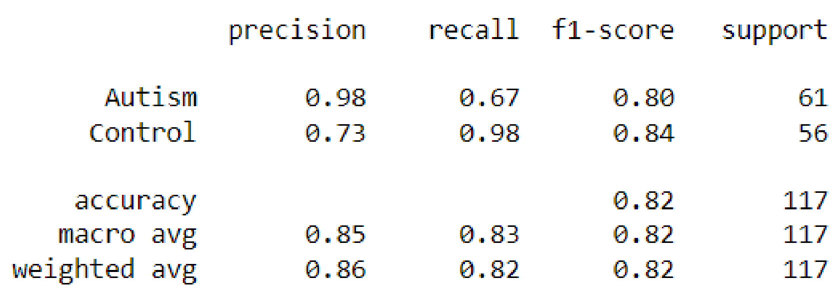
6.**Total number of trainable parameters:** The model has a total of 556,514 trainable parameters.



```
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 256)               512256    
                                                                 
 batch_normalization (BatchN  (None, 256)              1024      
 ormalization)                                                   
                                                                 
 dense_1 (Dense)             (None, 128)               32896     
                                                                 
 batch_normalization_1 (Batc  (None, 128)              512       
 hNormalization)                                                 
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_2 (Dense)             (None, 64)                8256      
                                                                 
 batch_normalization_2 (Batc  (None, 64)               256       
 hNormalization)                                                 
                                                                 
 dropout_1 (Dropout)         (None, 64)                0         
                                                                 
 dense_3 (Dense)             (None, 32)                2080      
                                                                 
 batch_normalization_3 (Batc  (None, 32)               128       
 hNormalization)                                                 
                                                                 
 dropout_2 (Dropout)         (None, 32)                0         
                                                                 
 dense_4 (Dense)             (None, 2)                 66        
                                                                 
=================================================================
Total params: 557,474
Trainable params: 556,514
Non-trainable params: 960
```

Figure 4.11 ANN model summary

Figure 4.12 shows the classification report for trained ANN model. It shows how well ANN model is performed which is represented by three metrics such Precision, Recall, F1 score. The accuracy is about 82%.



```
              precision    recall  f1-score   support

      Autism       0.98      0.67      0.80        61
     Control       0.73      0.98      0.84        56

    accuracy                           0.82       117
   macro avg       0.85      0.83      0.82       117
weighted avg       0.86      0.82      0.82       117
```

Figure 4.12 Classification report for ANN

Figure 4.13 shows confusion matrix for trained ANN model. From the confusion matrix the number of false test cases is 21.

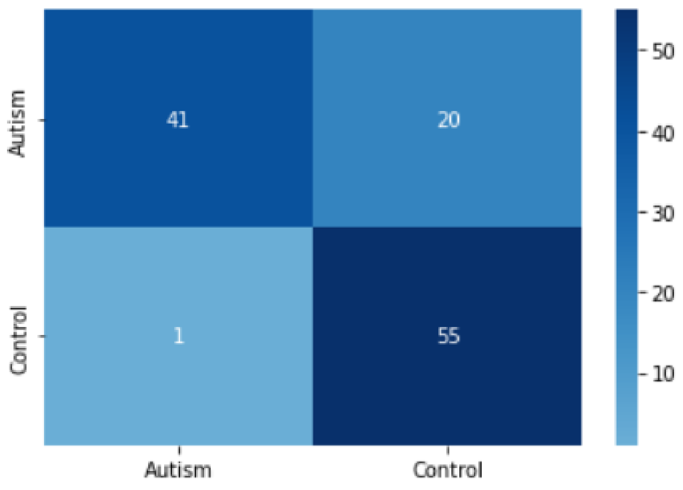
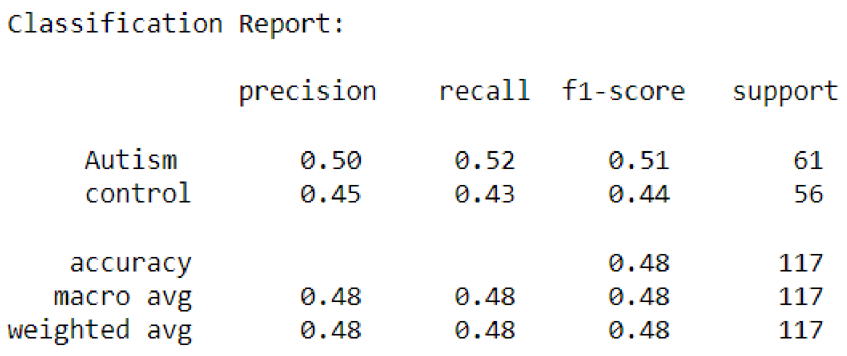


Figure 4.13 Confusion matrix of ANN

SVM

Figure 4.14 shows the classification report for trained SVM model. It shows how well SVM model is performed which is represented by three metrics such Precision, Recall, F1 score. The accuracy is about 48%.



```
Classification Report:

              precision    recall  f1-score   support

     Autism       0.50      0.52      0.51        61
     control      0.45      0.43      0.44        56

    accuracy                          0.48       117
   macro avg      0.48      0.48      0.48       117
weighted avg      0.48      0.48      0.48       117
```

Figure 4.14 Classification report for SVM

Figure 4.15 shows confusion matrix for trained SVM model. From the confusion matrix the number of false test cases is 61.

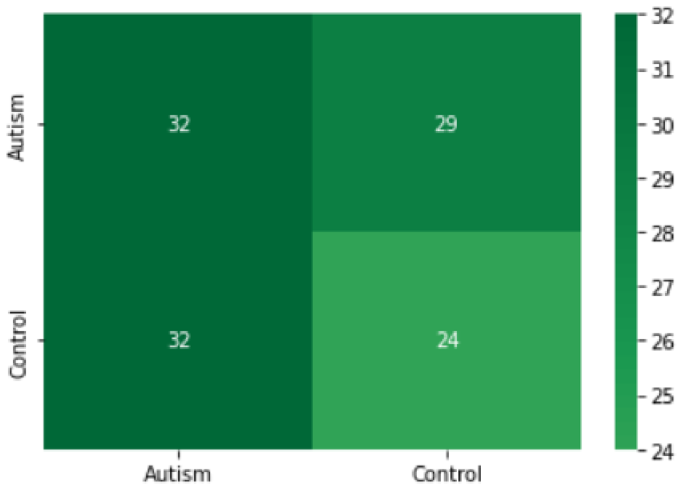


Figure 4.15 Confusion matrix of SVM

CNN

Figure 4.16 displays the summary of CNN model. From figure 4.16 the following data can be inferred:

It consists of three convolutional layers followed by two fully connected layers. The details of the layers and their output shapes are as follows:

1.**Conv1D layers**: The model has three 1D convolutional layers with 16, 32, and 64 filters respectively, and kernel size of 3. The output shapes of these layers are (None, 1998, 16), (None, 1998, 32), and (None, 1998, 64) respectively.

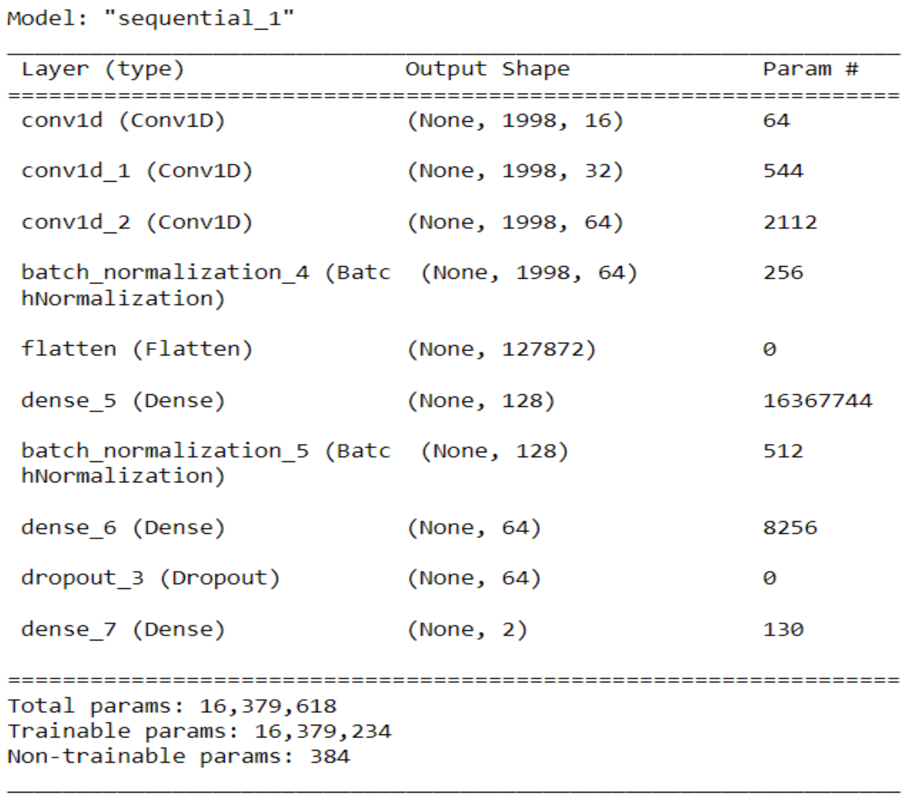
2.**Batch normalization**: The model includes batch normalization after the third convolutional layer and after the first fully connected layer to accelerate convergence and reduce overfitting.

3.**Flatten layer**: The output of the last convolutional layer is flattened to a 1D tensor of shape (None, 127872).

4.**Dense layers**: The model has two fully connected layers with 128 and 64 neurons respectively. Both layers use ReLU activation function. The output shape of the last dense layer is (None, 2), indicating a binary classification task.

5.**Dropout**: The model uses dropout with a rate of 0.5 after the second fully connected layer.

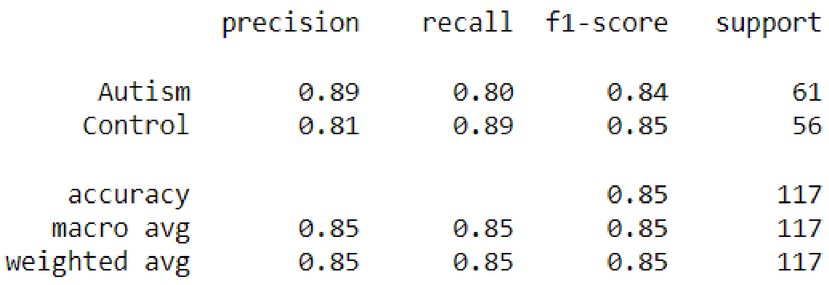
6.**Total number of parameters:** The model has 16,379,618 parameters in total, out of which 16,379,234 are trainable and 384 are non-trainable.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1d (Conv1D)             (None, 1998, 16)          64        
                                                                 
 conv1d_1 (Conv1D)           (None, 1998, 32)          544       
                                                                 
 conv1d_2 (Conv1D)           (None, 1998, 64)          2112      
                                                                 
 batch_normalization_4 (Batc  (None, 1998, 64)         256       
 hNormalization)                                                 
                                                                 
 flatten (Flatten)           (None, 127872)            0         
                                                                 
 dense_5 (Dense)             (None, 128)               16367744  
                                                                 
 batch_normalization_5 (Batc  (None, 128)              512       
 hNormalization)                                                 
                                                                 
 dense_6 (Dense)             (None, 64)                8256      
                                                                 
 dropout_3 (Dropout)         (None, 64)                0         
                                                                 
 dense_7 (Dense)             (None, 2)                 130       
                                                                 
=================================================================
Total params: 16,379,618
Trainable params: 16,379,234
Non-trainable params: 384
_________________________________________________________________
```

Figure 4.16 CNN model summary

Figure 4.17 shows the classification report for trained CNN model. It shows how well CNN model is performed which is represented by three metrics such Precision, Recall, F1 score. The accuracy is about 85%.



```
              precision    recall  f1-score   support

      Autism       0.89      0.80      0.84        61
     Control       0.81      0.89      0.85        56

    accuracy                           0.85       117
   macro avg       0.85      0.85      0.85       117
weighted avg       0.85      0.85      0.85       117
```

Figure 4.17 Classification report for CNN

Figure 4.18 shows confusion matrix for trained CNN model. From the confusion matrix the number of false test cases is 18.

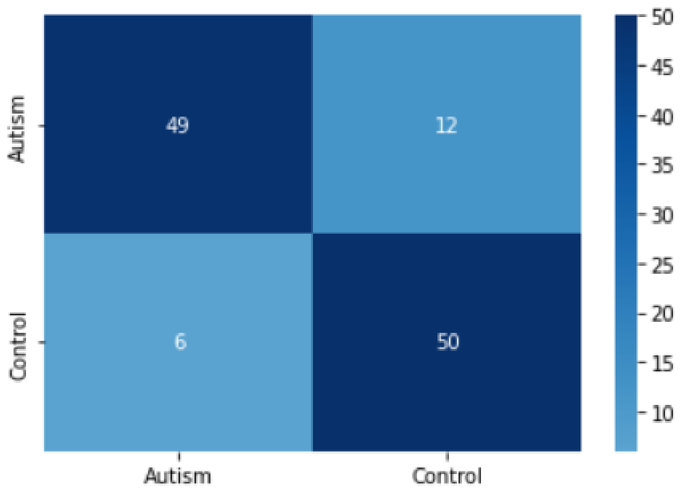
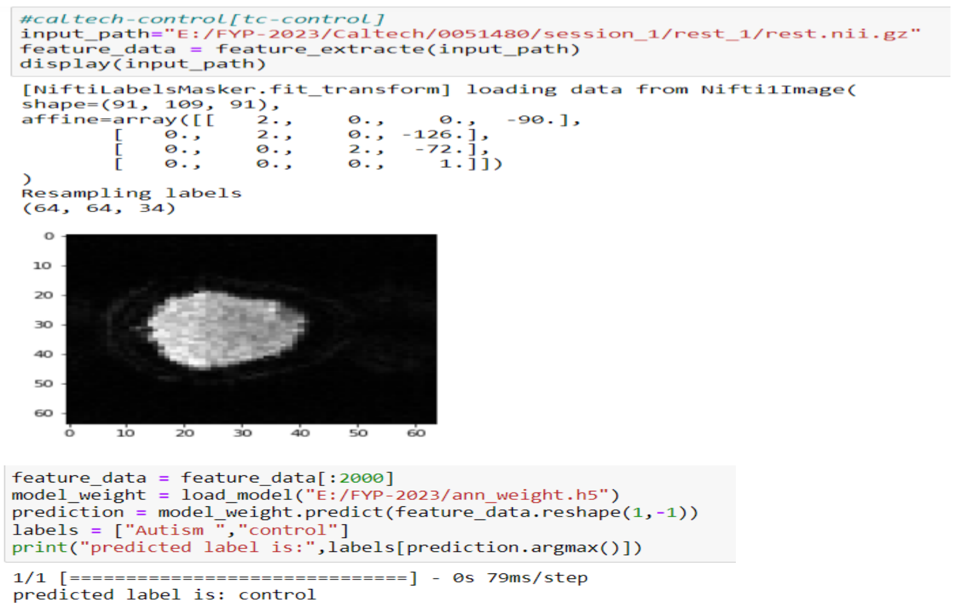


Figure 4.18 Confusion matrix of CNN

Module 7- Prediction

Prediction using ANN

Figure 4.19 predicts control and autism group using ANN model. The image with subid 0051480 and 0050188 indicates control and autism group respectively. These groups are predicted correctly using ANN model.

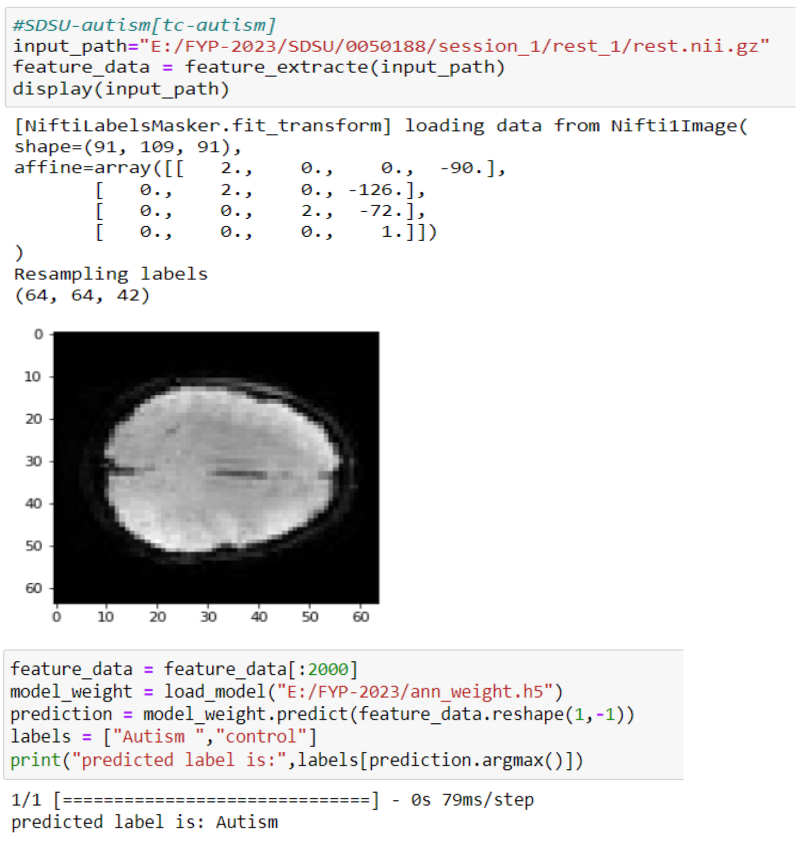


```
#caltech-control[tc-control]
input_path="E:/FYP-2023/Caltech/0051480/session_1/rest_1/rest.nii.gz"
feature_data = feature_extracte(input_path)
display(input_path)
```

```
[NiftiLabelsMasker.fit_transform] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [    0.,    2.,    0., -126.],
       [    0.,    0.,    2.,  -72.],
       [    0.,    0.,    0.,    1.]])
)
Resampling labels
(64, 64, 34)
```

```
feature_data = feature_data[:2000]
model_weight = load_model("E:/FYP-2023/ann_weight.h5")
prediction = model_weight.predict(feature_data.reshape(1,-1))
labels = ["Autism ","control"]
print("predicted label is:",labels[prediction.argmax()])
```

```
1/1 [==============================] - 0s 79ms/step
predicted label is: control
```



```
#SDSU-autism[tc-autism]
input_path="E:/FYP-2023/SDSU/0050188/session_1/rest_1/rest.nii.gz"
feature_data = feature_extracte(input_path)
display(input_path)
```

```
[NiftiLabelsMasker.fit_transform] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [   0.,    2.,    0., -126.],
       [   0.,    0.,    2.,  -72.],
       [   0.,    0.,    0.,    1.]])
)
Resampling labels
(64, 64, 42)
```

```
feature_data = feature_data[:2000]
model_weight = load_model("E:/FYP-2023/ann_weight.h5")
prediction = model_weight.predict(feature_data.reshape(1,-1))
labels = ["Autism ","control"]
print("predicted label is:",labels[prediction.argmax()])
```

```
1/1 [==============================] - 0s 79ms/step
predicted label is: Autism
```

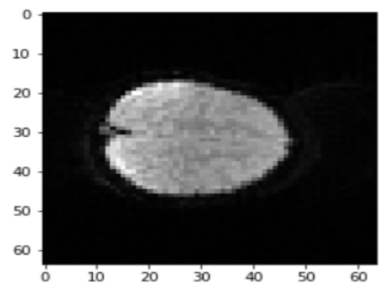
Figure 4.19 Prediction using ANN

Prediction using SVM

Figure 4.20 predicts control and autism group using SVM model. The image with subid 0050169 and 0051581 indicates control and autism group respectively. These groups are predicted correctly using SVM model.

```
#ohsu-control[tc-control]
input_path = 'E:/FYP-2023/OHSU/0050169/session_1/rest_1/rest.nii.gz'
feature_data = feature_extracte(input_path)
display(input_path)
```

```
[NiftiLabelsMasker.fit_transform] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [   0.,    2.,    0., -126.],
       [   0.,    0.,    2.,  -72.],
       [   0.,    0.,    0.,    1.]])
)
Resampling labels
(64, 64, 36)
```

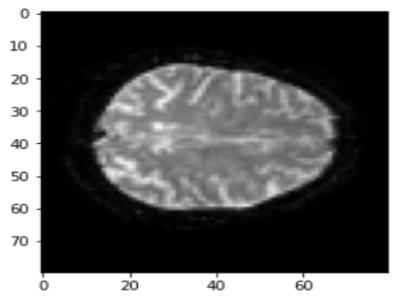


```
feature_data = feature_data[:2000]
model_weight = pickle.load(open("E:FYP-2023/svm_weight.p","rb"))
prediction = model_weight.predict(feature_data.reshape(1,-1))
labels = ["Autism ","control"]
print("predicted label is:",labels[prediction[0]])
```

```
predicted label is: control
```

```
#sbl autism[tc-autism]
input_path = 'E:/FYP-2023/SBL/0051581/session_1/rest_1/rest.nii.gz'
feature_data = feature_extracte(input_path)
display(input_path)
```

```
[NiftiLabelsMasker.fit_transform] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [   0.,    2.,    0., -126.],
       [   0.,    0.,    2.,  -72.],
       [   0.,    0.,    0.,    1.]])
)
(80, 80, 38)
```



```
feature_data = feature_data[:2000]
model_weight = pickle.load(open("E:FYP-2023/svm_weight.p","rb"))
prediction = model_weight.predict(feature_data.reshape(1,-1))
labels = ["Autism ","control"]
print("predicted label is:",labels[prediction[0]])
```

```
predicted label is: Autism
```

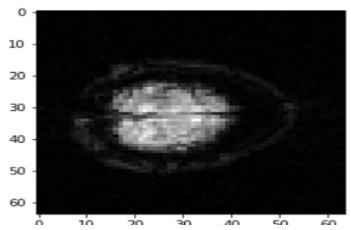
Figure 4.20 Prediction using SVM

Prediction using CNN

Figure 4.21 predicts control and autism group using CNN model. The image with subid 0050030 and 0050477 indicates control and autism group respectively. These groups are predicted correctly using CNN model.

```
#Pitt control[tc-control]
input_path = "E:/FYP-2023/Pitt/0050030/session_1/rest_1/rest.nii.gz"
feature_data = feature_extracte(input_path)
display(input_path)
```

```
[NiftiLabelsMasker.fit_transform] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [   0.,    2.,    0., -126.],
       [   0.,    0.,    2.,  -72.],
       [   0.,    0.,    0.,    1.]])
)
Resampling labels
(64, 64, 29)
```

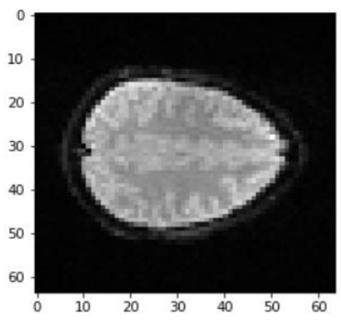


```
feature_data = feature_data[:2000]
model_weight = load_model("E:/FYP-2023/cnn_weight.h5")
prediction = model_weight.predict(feature_data.reshape(1,-1))
labels = ["Autism ","control"]
print("predicted label is:",labels[prediction.argmax()])
```

```
1/1 [==============================] - 0s 331ms/step
predicted label is: control
```

```
#usm-autism[tc-autism]
input_path = 'E:/FYP-2023/USM/0050477/session_1/rest_1/rest.nii.gz'
feature_data = feature_extracte(input_path)
display(input_path)
```

```
[NiftiLabelsMasker.fit_transform] loading data from Nifti1Image(
shape=(91, 109, 91),
affine=array([[   2.,    0.,    0.,  -90.],
       [   0.,    2.,    0., -126.],
       [   0.,    0.,    2.,  -72.],
       [   0.,    0.,    0.,    1.]])
)
Resampling labels
```



```
feature_data = feature_data[:2000]
model_weight = load_model("E:/FYP-2023/cnn_weight.h5")
prediction = model_weight.predict(feature_data.reshape(1,-1))
labels = ["Autism ","control"]
print("predicted label is:",labels[prediction.argmax()])
```

```
1/1 [==============================] - 1s 631ms/step
predicted label is: Autism
```

Figure 4.21 Prediction using CNN

4.3 POSSIBLE TEST CASES:

ANN:

Prediction using ANN:

Table 4.1. shows all possible test cases with both classes-Autism and Control groups predicted with ANN algorithm.

From Table 4.1 we see two images have been predicted with wrong class.

Prediction using ANN

UNIVERSITY	SUBID	TRUE CLASS	PREDICTED CLASS
Stanford	0051180	Control	Control
Stanford	0051160	Autism	Control
Stanford	0051174	Autism	Autism
Caltech	0051461	Autism	Autism
Caltech	0051480	Control	Control
Olin	0050123	Autism	Control
Olin	0050115	Control	Control
Olin	0050124	Autism	Autism
SDSU	0050188	Autism	Autism
SDSU	0050199	Control	Control

Table 4.1.

SVM:

Prediction using SVM

Table 4.2. shows all possible test cases with both classes-Autism and Control groups predicted with SVM algorithm.

From Table 4.2. we see three images have been predicted with wrong class.

Prediction using SVM

UNIVERSITY	SUBID	TRUE CLASS	PREDICTED CLASS
OHSU	0050169	Control	Control
OHSU	0050153	Autism	Autism
SBL	0051571	Autism	Control
SBL	0051581	Autism	Control
SBL	0051557	Control	Control

Trinity	0050237	Autism	Autism
Trinity	0050269	Control	Control
Yale	0050614	Autism	Autism
Yale	0050559	Control	Control
Stanford	0051179	Autism	Control

Table 4.2

CNN:

Prediction using CNN:

Table 4.3. shows all possible test cases with both classes-Autism and Control groups predicted with CNN algorithm.

From Table 4.3. we see one image have been predicted with wrong class.

Prediction using CNN

UNIVERSITY	**SUBID**	**TRUE CLASS**	**PREDICTED CLASS**
Pitt	0050014	Autism	Autism
Pitt	0050030	Control	Control
USM	0050477	Autism	Autism
USM	0050453	Control	Autism
USM	0050432	Control	Control
KKI	0050791	Autism	Autism
KKI	0050774	Control	Control
UCLA_1	0051202	Autism	Autism
UCLA_1	0051255	Control	Control
UCLA_2	0051294	Autism	Autism
UCLA_2	0051301	Autism	Autism

Table 4.3.

From Table 4.1., 4.2., 4.3., we infer that CNN have predicted better in comparison to SVM and ANN as CNN have high performance metrics in comparison to other two algorithms.

4.4 PERFORMANCE EVALUATION

Performance metrics are used to evaluate the effectiveness of a model or algorithm in achieving a specific task. In the context of ASD prediction, performance metrics can help to determine the accuracy, precision, recall, and other measures of how well a model is able to predict the presence or absence of ASD in a given population.

4.4.1 Metrics for Evaluation

- **Accuracy:**

  Accuracy represents the number of correctly classified data instances over the total number of data instances.

$$Accuracy = \frac{\mathrm{TN+TP}}{TN+FP+TP+FN)} \quad \text{———————}(1)$$

- **Precision:**

  Precision is a metric that quantifies the number of correct positive predictions made.

$$Precision = \frac{TP}{TP+FP} \quad \text{———————} (2)$$

- **Recall:**

  Recall is a metric that quantifies the number of correct positive predictions made of all positive predictions that could have been made.

$$Recall = \frac{TP}{TP+FN} \quad \text{———————} (3)$$

- **F1-score:**

  The F1 score combines precision and recall using their harmonic mean and maximizing the F1 score implies simultaneously maximizing both precision and recall. Thus, the F1 score has become the choice of researchers for evaluating their models in conjunction with accuracy.

$$\mathrm{f1}score = \frac{2*precision*recall}{precision+recall} \quad \text{———————} (4)$$

4.5 COMPARATIVE ANALYSIS

Table 4.4. shows the performance metrics -Accuracy, Recall, Precision and F1 score calculated for the implemented algorithms.

From Table 4.4 we conclude that CNN has the highest accuracy of about 84.62% while ANN has an accuracy of about 82.05% which is nearly closer to CNN's performance. But SVM has the lowest accuracy of about 48.72% which is not suited for this prediction of ASD as SVM is not good for training dataset with large number of features. Thus, CNN can be effectively used for the prediction of ASD disorder.

Performance metrics

ALGORITHM	CLASS	PRECISION	RECALL	F1-SCORE
ANN	Autism	0.98	0.67	0.80
	Control	0.73	0.98	0.84
	Accuracy	82.05		
SVM	Autism	0.51	0.54	0.52
	Control	0.46	0.43	0.44
	Accuracy	48.72		
CNN	Autism	0.89	0.80	0.84
	Control	0.81	0.89	0.85
	Accuracy	84.62		

Table 4.4

Accuracy Comparison:

Figure 4.22 provides an accuracy comparison of different algorithms. From the Figure 4.22 CNN has the highest accuracy of about 84.62%. and hence it can be effectively used for the prediction of ASD.

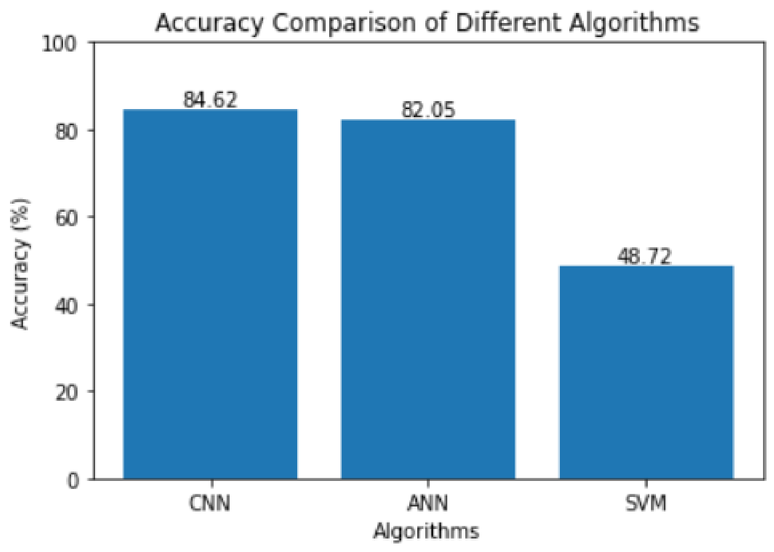


Figure 4.22. Accuracy Comparison

Precision Comparison:

Figure 4.23 shows a bar graph showing the precision comparison among the three algorithms.

From figure 4.23 the following data can be inferred:

- ❖ The ANN model has a high precision for detecting autism (0.98) and a lower precision for detecting control (0.73).
- ❖ The SVM model has a relatively low precision for detecting both autism (0.51) and control (0.46).
- ❖ The CNN model has a high precision for detecting both autism (0.89) and control (0.81).

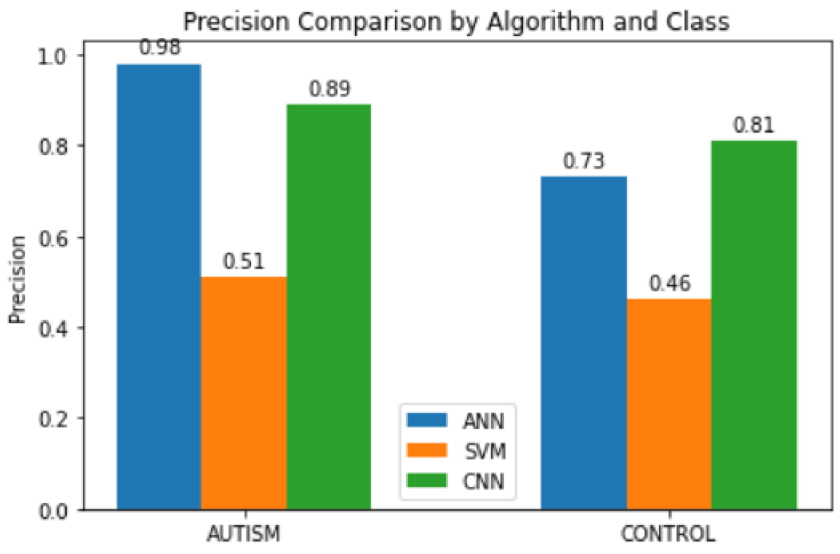


Figure 4.23 Precision Comparison

Recall Comparison:

Figure 4.24 shows a bar graph showing the recall comparison among the three algorithms.

From figure 4.24 the following data can be inferred:

- ❖ For ANN, the model has better recall for the control group (0.98) compared to the autism group (0.67).
- ❖ For SVM, the model has slightly better recall for the autism group (0.54) compared to the control group (0.43).
- ❖ For CNN, the model has better recall for the control group (0.89) compared to the autism group (0.80).

Overall, we can see that the CNN model has the highest recall values for both the autism and control groups, indicating that it is better at correctly identifying true positive cases.

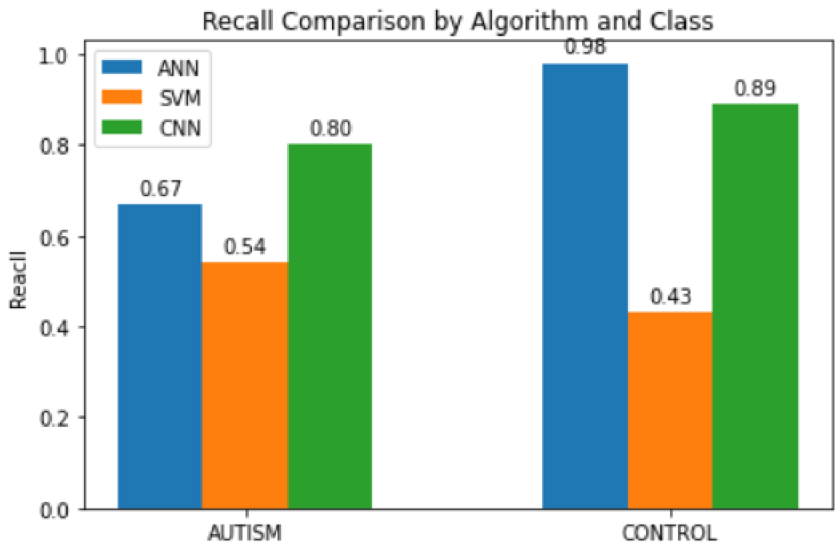


Figure 4.24. Recall Comparison

F1-score Comparison:

Figure 4.25 shows a bar graph showing the fi-score comparison among the three algorithms.

From figure 4.25 the following data can be inferred:

Based on the given f1-score values, the ANN and CNN models have relatively high performance in both predicting Autism and Control groups compared to the SVM model. The ANN model has the highest f1-score for predicting Autism (0.80) while the CNN model has the highest f1-score for predicting the Control group (0.85). The SVM model has the lowest f1-score for both Autism and Control groups, suggesting that it has lower overall performance compared to the ANN and CNN models.

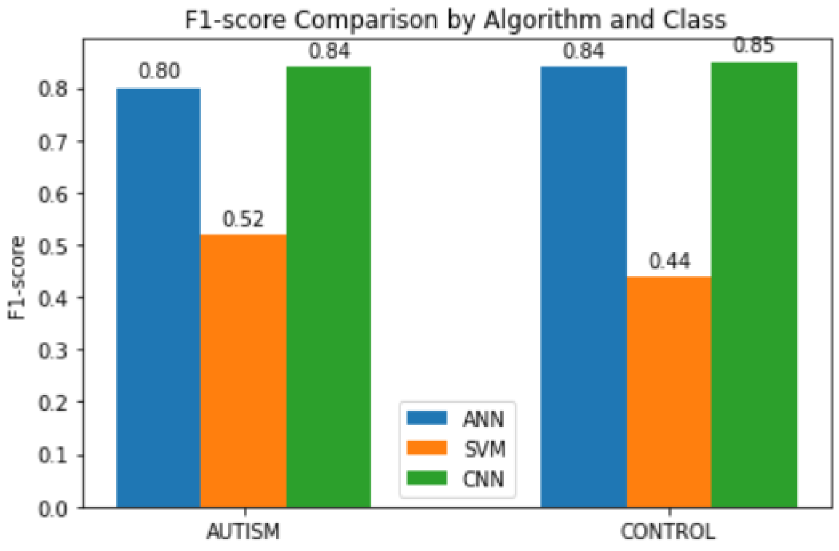


Figure 4.25. F1-score Comparison

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION:

The project has been successfully implemented to predict the presence of the autism spectrum disorder and determine whether the person has disorder and provide prior measures to avoid the disease. The algorithms- SVM, CNN and ANN are used to determine the presence of the disease. CNN and SVM have been implemented apart from the base paper. CNN have better performance in comparison to other two algorithms with an accuracy of about 84.62%. Thus, CNN can effectively use to determine the presence of autism spectrum disorder.

5.2 FUTURE WORK

In the coming future, the application of the autism spectrum disorder determine technology in the healthcare field can be reviewed and it can promote for detecting the disease with more accuracy. Thus, this project has an efficient scope in coming future where manual predicting can be converted to computerized production in a cheap way.

REFERENCES

[1] A. Baranwal and M. Vanitha, (2020), “Autistic spectrum disorder screening: Prediction with machine learning models,” 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1–7.

[2] A. Chorianopoulou, E. Tzinis, E. Iosif, A. Papoulidi, C. Papailiou, and A. Potamianos, (2017), “Engagement detection for children with autism spectrum disorder,” 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 5055–5059.

[3] A. Puli and A. Kushki, (2020), Toward automatic anxiety detection in autism: A real-time algorithm for detecting physiological arousal in the presence of motion, *IEEE Trans. Biomed. Eng.*, vol. 67, no. 3, pp. 646–657.

[4] A. Sharma and P. Tanwar, (2020), “Deep analysis of autism spectrum disorder detection techniques,” 2020 International Conference on Intelligent Engineering and Management (ICIEM), 2020, pp. 455–459.

[5] A. Z. Guo, (2023), Automated autism detection based on characterizing observable patterns from photos, *IEEE Trans. Affect. Comput.*, vol. 14, no. 1, pp. 836–841.

[6] D. D. Jemima, A. G. Selvarani, and J. D. Louis Lovenia, (2021), “A study on diagnosis of autism spectrum disorder for children,” 2021 3rd International Conference on Signal Processing and Communication (ICPSC), 2021, pp. 547–551.

[7] H. Zhao, A. R. Swanson, A. S. Weitlauf, Z. E. Warren, and N. Sarkar, (2018), Hand-in-hand: A Communication-Enhancement collaborative virtual reality system for promoting social interaction in children with autism spectrum disorders, *IEEE Trans. Hum. Mach. Syst.*, vol. 48, no. 2, pp. 136–148.

[8] L. Zhang et al., (2021), Design of an intelligent agent to measure collaboration and verbal-communication skills of children with autism spectrum disorder in collaborative puzzle games, *IEEE Trans. Learn. Technol.*, vol. 14, no. 3, pp. 338–352.

[9] M. B. Mohammed, L. Salsabil, M. Shahriar, S. S. Tanaaz, and A. Fahmin, (2021), "Identification of autism spectrum disorder through feature selection-based machine learning," 2021 24th International Conference on Computer and Information Technology (ICCIT), 2021, pp. 1–6.

[10] M. Ingalhalikar, S. Shinde, A. Karmarkar, A. Rajan, D. Rangaprakash, and G. Deshpande, (2021), Functional connectivity-based prediction of autism on site harmonized ABIDE dataset, *IEEE Trans. Biomed. Eng.*, vol. 68, no. 12, pp. 3628–3637.

[11] N. Talebi, A. M. Nasrabadi, I. Mohammad-Rezazadeh, and R. Coben, (2019), NCREANN: Nonlinear Causal Relationship Estimation by Artificial Neural Network; Applied for autism connectivity study, *IEEE Trans. Med. Imaging, vol.* 38, no. 12, pp. 2883–2890.

[12] N. Zaman, J. Ferdus, and A. Sattar, (2021), "autism spectrum disorder detection using machine learning approach," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2021, pp. 1–6.