Abstract

ADHD (attention deficit hyperactivity disorder) is a common neurodevelopmental disorder that affects how people think and behave. It’s been gaining attention lately as researchers look for better ways to detect and manage it early. Machine learning (ML) has emerged as a promising tool for predicting ADHD by analyzing brain signals, specifically electroencephalography (EEG), which records electrical activity in the brain. Studies have shown that features like attention continuity, a nonlinear aspect of EEG, can give clear insights into ADHD symptoms (Allahverdy et al., 2018). Using these EEG features, ML techniques like Logical regression, Decision tree, Randomforest are being employed to classify ADHD.

More advanced techniques, such as functional connectivity analysis (Springer, 2016) and dynamic brain network modeling (IEEE, 2012), have improved the detection process. Recent research, such as the one from ScienceDirect (2022), focuses on using ML to distinguish ADHD from other neurodevelopmental disorders by studying attention and cognitive function data. Combining ML with neuroimaging could further enhance the accuracy of these predictions. These advancements are paving the way for clinicians to use ML as a non-invasive and efficient tool for early ADHD diagnosis, and they could help doctors tailor treatments based on individual needs.

This is fascinating technology, especially ML, which is being used to solve real-world medical problems. The intersection of engineering and healthcare opens up so many possibilities for innovation, and it is exciting to see how these advancements can improve people’s lives, especially in areas like mental health and neurodevelopmental disorders.

List of Abbreviations

General Medical Terms

-ADHD – Attention Deficit Hyperactivity Disorder

- EEG – Electroencephalography

Data Visualization and Terms

-RF – Random Forest

-DT – Decision Tree

-LR – Logistic Regression

Programming and Statistical Terms:

-ML – Machine Learning

Table of Contents

CHAPTER NO. 1
INTRODUCTION

1.1 Introduction

Attention deficit hyperactivity disorder (ADHD) is one of the most frequent neurodevelopmental behavioural disorders in childhood. Children with ADHD have the following symptoms: hyperactivity, inattention, and impulsivity. According to the Centers for Disease Control (CDC) and prevention, the number of children in the USA who have been diagnosed with ADHD has fluctuated over time as follows: about 4.4 million children between the ages of 2 and 17 years were diagnosed with ADHD in 2003, 5.4 million children in 2007, 6.4 million children in 2011, and 6.1 million children in 2016. About 12.9% of male children and 5.6% of females were diagnosed with ADHD. Globally, the prevalence of adults with ADHD was 2.8% in 2016 and 0.96% in 2019; and 7.8% of children were diagnosed with ADHD in 2003, 9.5% in 2007, and 11% in 2007. There were 62% of children who had taken medication for ADHD, and 46.7% of those children had also received behavioural treatment. It is noted that the number of children with ADHD has been increasing day by day. Therefore, it is necessary to propose a model for the identification of the risk factors for ADHD.

Researchers are trying to determine the risk factors to reduce the number of children with ADHD. A study showed that genetic factors played a significant role and were linked with ADHD. Genetic factors are responsible for almost 75% of the risk of ADHD in younger children. Besides the genetic factors, there were several risk factors for ADHD such as brain injury, alcohol/tobacco use during pregnancy, and premature delivery. Previous studies also showed that age, sex, asthma, race, anxiety, depression, obesity, cigarette smoking, and socio-economic status were also associated with children with ADHD. These studies were conducted only to identify the risk factors for children with ADHD. It is necessary to propose a prediction model. In this regard, in comparison with classical approaches, machine learning (ML)-based models may be used for prediction. ML-based models have been also used for the identification and prediction in the field of medical imaging, healthcare, and mental health.

Despite the rapid development of ML-based classifiers, their application to ADHD diagnosis remains a difficult task. Yet, various ML-based classifiers have been utilized to predict children with ADHD in different countries using different ADHD datasets. However, the models’ performance has to be improved. The current study had the following objectives: (i) to extract the risk factors of children with ADHD; and (ii) to propose an ML-based classifier to classify and predict children as either having ADHD or healthy.

The overall layout of this study is as follows: Chapter 3 presents the materials and methods; we present descriptions of dataset, predictor and outcome variables, statistical analysis, machine learning techniques, and performance evaluation criteria. Results are presented in Chapter 5. Chapter 6 presents a detailed conclusion.

1.2 Problem Statements & Objectives

Problem Statement: Attention deficit hyperactivity disorder (ADHD) is a behaviour disorder characterized by inattention, impulsivity, and in some cases hyperactivity, typically diagnosed in childhood. It is a common childhood developmental disorder. Most patients with ADHD have a common brain-wave pattern that consists of an abundance of slow brain waves and a shortage of fast brain waves. This could be employed for automatic recognition from the characteristic brain wave. Even with the current progress, using EEG tests for ADHD detection needs a more precise approach in this area to get more accurate results. The amount of information in EEG signals is vast. It is also complicated for a human to detect abnormalities manually. This is where machine learning (ML) can be useful. Hence, this work uses EEG signals to detect ADHD in children.

Objectives:

- To generate the features of EEG signal of 19 channels
- To model the ADHD problem by methods namely logistic regression, Decision tree (DT) and Random Forest (RF)
- To tune the hyper parameters of DT and RF to avoid overfitting issues

1.3 Scope

This work uses the open-source dataset of EEG signal presented on IEEE website. The signals are processed in time domain only and frequency domain are left for future work. Only three features are used here and three machine learning methods (DT, RF and logistic) are compared.

2.2 Limitation In Existing System or Research Gap

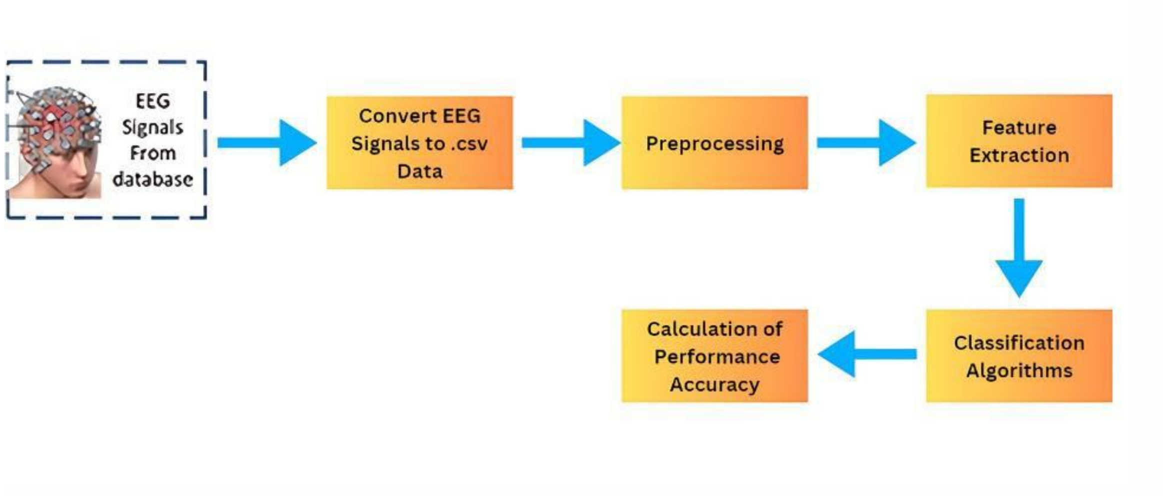
Some work related to detection of ADHD with EEG signal is found in literature with considering 2 to 5 channels only and none of the project have used all the 19 channels for feature generation which is done here. Also, this work tuned the hyperparameters of decision tree and random forest which is not reported in literature.

2.3 Mini Project Contribution

- Unlike other studies, we employed EEG signal from all the 19 channels.
- Three-time domain features are utilised for each channel, giving 51 inputs
- The model was trained on a large dataset of 121 children
- The hyperparameters of Decision tree and Random Forest algorithm to avoid overfitting.

CHAPTER NO. 3
PROPOSED SYSTEM

3.1 Architecture/Framework /Block Diagram


EEG Signals From database
Convert EEG Signals to .csv Data
Preprocessing
Feature Extraction
Classification Algorithms
Calculation of Performance Accuracy

In this study, EEG data were collected from a publicly available dataset containing 61 subjects who had ADHD & 60 normal subjects (10 girls + 50 boys, mean age: 9.85 ± 1.77 yr). There were no mental illnesses, epilepsy, or reports of high-risk behaviors among the children in the control group. Dataset's EEG recordings are captured at a rate of 128 Hz, and there are Nineteen channels available (Fz, F7, F8, Fp1, Fp2, Cz, T3, C4, T4, F3, T6, O1, F4, P3, Pz, C3, P4, T5, O2). Because visual attention is one of the deficiencies in ADHD youngsters, the EEG recording procedure was designed to address this. During cognitive activity, EEG signals were recorded while children engaged in an activity that involved counting the number of characters in displayed cartoon graphics as illustrated in Figure 2. The images are appropriately sized for visibility, and the count of characters in every picture varies randomly between 5 and 16. Given the diverse performance behaviors exhibited by children within cognitive tasks, the duration of EEG recordings exhibits variance across the dataset.

3.2 Algorithm

- **Logistic Regression (LR):**

  Logistic regression is a statistical method used for binary classification, predicting whether an outcome belongs to one of two classes (e.g., ADHD or healthy). It calculates the probability of an event occurring by applying a logistic function to a weighted sum of the input features. The output is a value between 0 and 1, which is used to classify data into categories.

- **Decision Tree (DT):**

  A decision tree is a flowchart-like structure where each internal node represents a decision based on a feature, and each leaf node represents a class label (e.g., ADHD or healthy). The tree is built by recursively splitting the data based on features that provide the best classification at each step (usually measured by metrics like Gini impurity or entropy).

- **Random Forest (RF):**

  Random forest is an ensemble method that builds multiple decision trees using different subsets of the data and features. Each tree votes for a class, and the final prediction is based on the majority vote of all trees. This approach improves accuracy and reduces overfitting compared to using a single decision tree.

3.3 Details Of Hardware and Software

<u>Hardware Details:</u>

- Laptop with minimum 8 Gigabytes RAM, good enough storage (512 Gigabytes). Good internet connection.

<u>Software Details:</u>

1. Programming Languages: -

- Python: For data processing, analysis, and visualization due to its rich ecosystem of libraries.

2. Data Visualization Libraries: -

- Matplotlib: For creating static, interactive, and animated visualizations in Python.

- Seaborn: For statistical data visualization built on top of Matplotlib, making it easier to generate complex visualizations.

3. Data Manipulation Libraries: -

- Pandas: For data manipulation and analysis, providing data structures like DataFrames to handle tabular data easily.

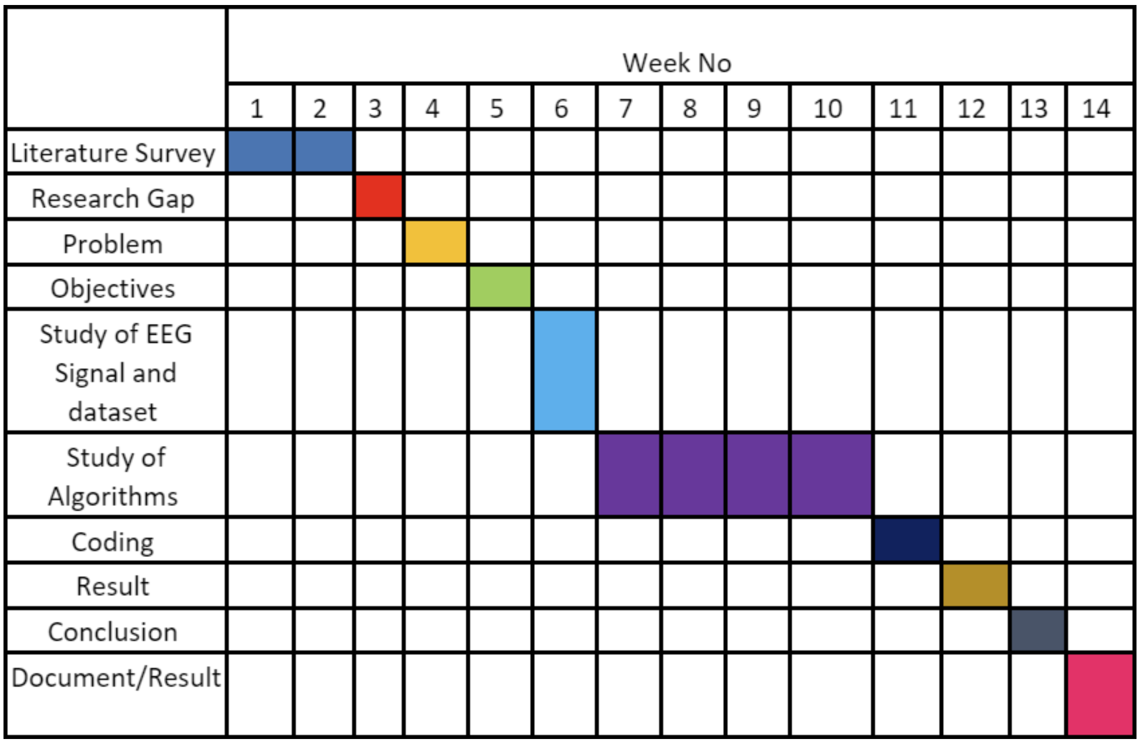
- NumPy: For numerical computations and handling large, multi-dimensional arrays and matrices.

4. Integrated Development Environment (IDE): -

- Jupyter Notebook: For interactive coding and data visualization, allowing easy sharing and presentation of analyses.

CHAPTER NO. 4
IMPLEMENTATION PLAN

4.1 Gantt Chart (Term I and Term II)



	Week No													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Literature Survey	🟦	🟦												
Research Gap			🟥											
Problem				🟨										
Objectives					🟩									
Study of EEG Signal and dataset						🟦								
Study of Algorithms							🟪	🟪	🟪	🟪				
Coding											🟦			
Result												🟫		
Conclusion													⬛	
Document/Result														🟥

4.2 Implementation Plan for Next Semester

The future work outlined in the project focuses on several key areas for enhancing the current ADHD prediction model using machine learning:

- **Per channel minimum amount 10 feature (190 input)**
  Based on general guidelines and research, a minimum of 10 features per channel is a reasonable assumption for many applications.

- **PCA (Principal Component Analysis) for feature reduction**
  Reduce the number of features , Enhance model performance

- **ANN and NAIVE bayes (comparison)**
  ANNs are suitable for complex, high-dimensional datasets with non-linear relationships, while Naive Bayes Classifiers are effective for simpler datasets with independent features.

- **Frequency domain of signal**
  Frequency domain analysis is a fundamental step in processing and analyzing signals in ML and DL.

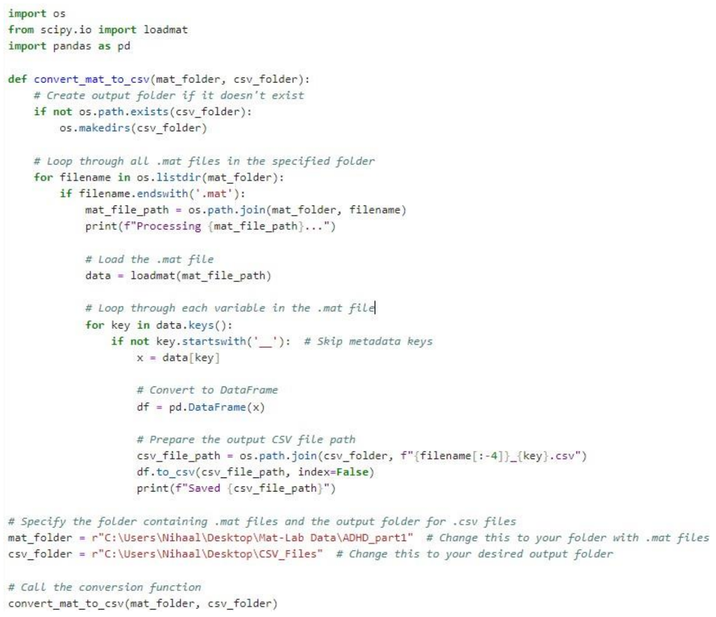
These steps aim to improve accuracy, reduce model complexity, and explore more sophisticated approaches for ADHD diagnosis.

CHAPTER NO. 5

IMPLEMENTATION RESULT AND ANALYSIS

5.1 Implementation Screenshots

1) File Conversion from .mat to .csv :



```python
import os
from scipy.io import loadmat
import pandas as pd

def convert_mat_to_csv(mat_folder, csv_folder):
    # Create output folder if it doesn't exist
    if not os.path.exists(csv_folder):
        os.makedirs(csv_folder)

    # Loop through all .mat files in the specified folder
    for filename in os.listdir(mat_folder):
        if filename.endswith('.mat'):
            mat_file_path = os.path.join(mat_folder, filename)
            print(f"Processing {mat_file_path}...")

            # Load the .mat file
            data = loadmat(mat_file_path)

            # Loop through each variable in the .mat file
            for key in data.keys():
                if not key.startswith('__'):  # Skip metadata keys
                    x = data[key]

                    # Convert to DataFrame
                    df = pd.DataFrame(x)

                    # Prepare the output CSV file path
                    csv_file_path = os.path.join(csv_folder, f"{filename[:-4]}_{key}.csv")
                    df.to_csv(csv_file_path, index=False)
                    print(f"Saved {csv_file_path}")

# Specify the folder containing .mat files and the output folder for .csv files
mat_folder = r"C:\Users\Nihaal\Desktop\Mat-Lab Data\ADHD_part1"  # Change this to your folder with .mat files
csv_folder = r"C:\Users\Nihaal\Desktop\CSV_Files"  # Change this to your desired output folder

# Call the conversion function
convert_mat_to_csv(mat_folder, csv_folder)
```

Fig 1.1

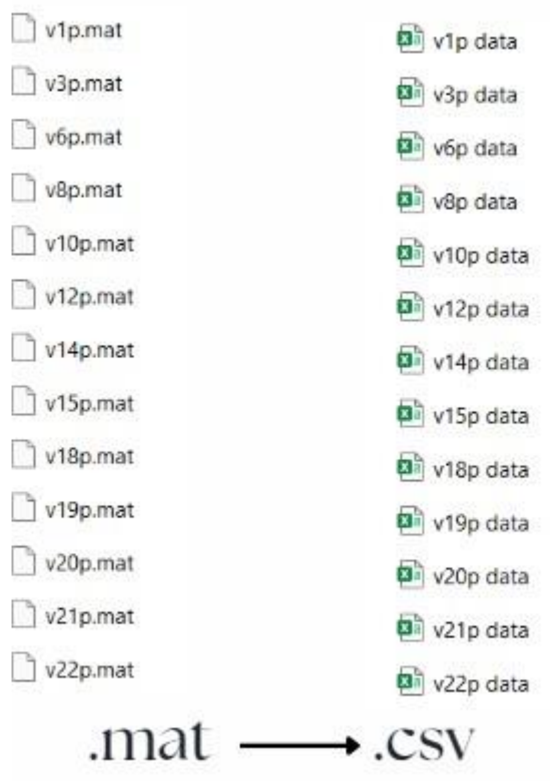
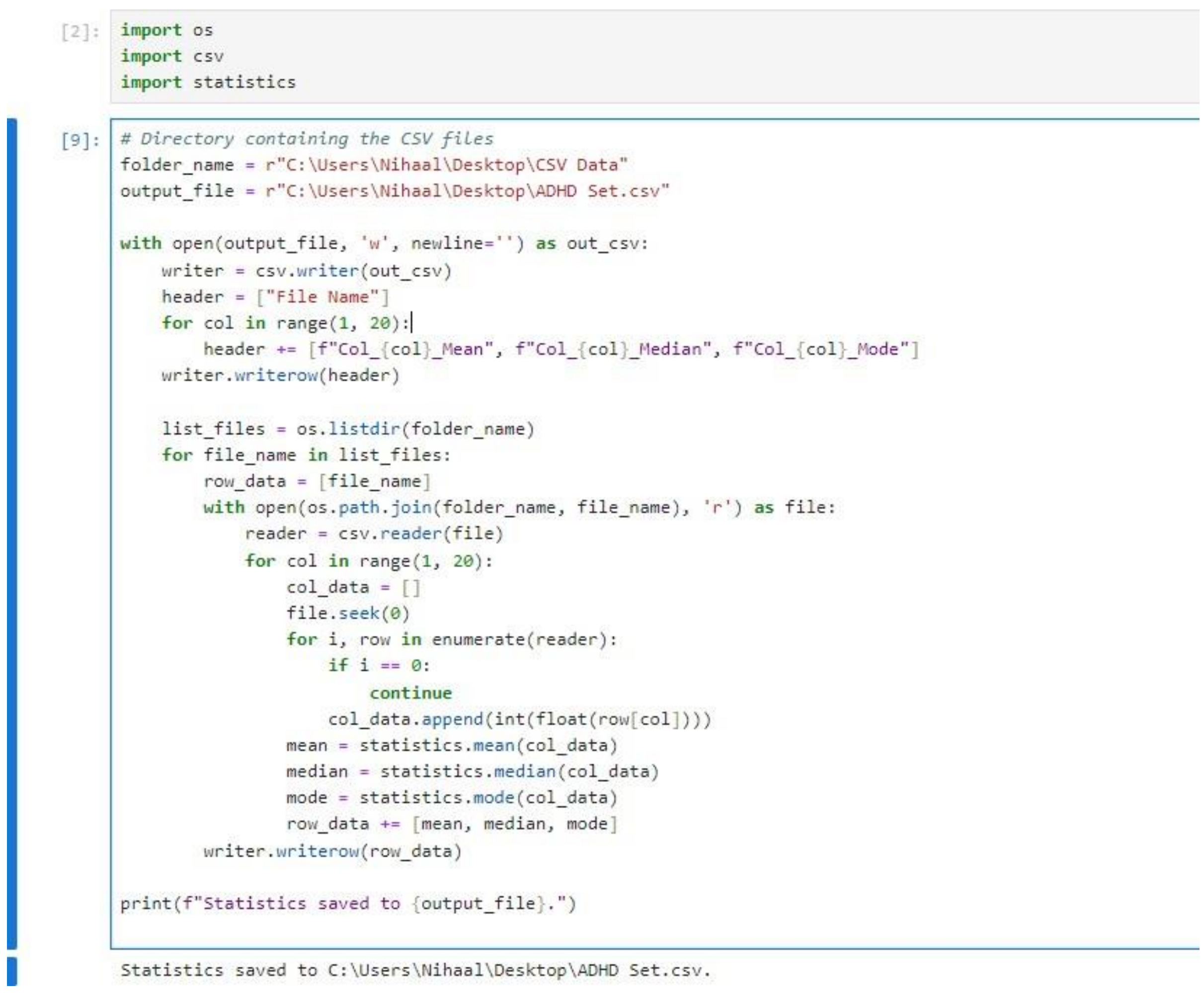


Fig 1.2

2) Data extracting using features :

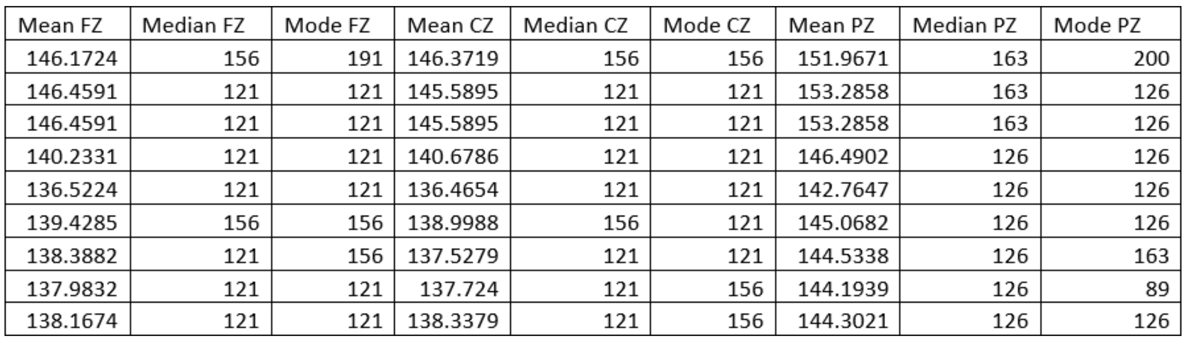


```
[2]: import os
     import csv
     import statistics
```

```
[9]: # Directory containing the CSV files
     folder_name = r"C:\Users\Nihaal\Desktop\CSV Data"
     output_file = r"C:\Users\Nihaal\Desktop\ADHD Set.csv"

     with open(output_file, 'w', newline='') as out_csv:
         writer = csv.writer(out_csv)
         header = ["File Name"]
         for col in range(1, 20):
             header += [f"Col_{col}_Mean", f"Col_{col}_Median", f"Col_{col}_Mode"]
         writer.writerow(header)

         list_files = os.listdir(folder_name)
         for file_name in list_files:
             row_data = [file_name]
             with open(os.path.join(folder_name, file_name), 'r') as file:
                 reader = csv.reader(file)
                 for col in range(1, 20):
                     col_data = []
                     file.seek(0)
                     for i, row in enumerate(reader):
                         if i == 0:
                             continue
                         col_data.append(int(float(row[col])))
                     mean = statistics.mean(col_data)
                     median = statistics.median(col_data)
                     mode = statistics.mode(col_data)
                     row_data += [mean, median, mode]
             writer.writerow(row_data)

     print(f"Statistics saved to {output_file}.")
```

```
Statistics saved to C:\Users\Nihaal\Desktop\ADHD Set.csv.
```

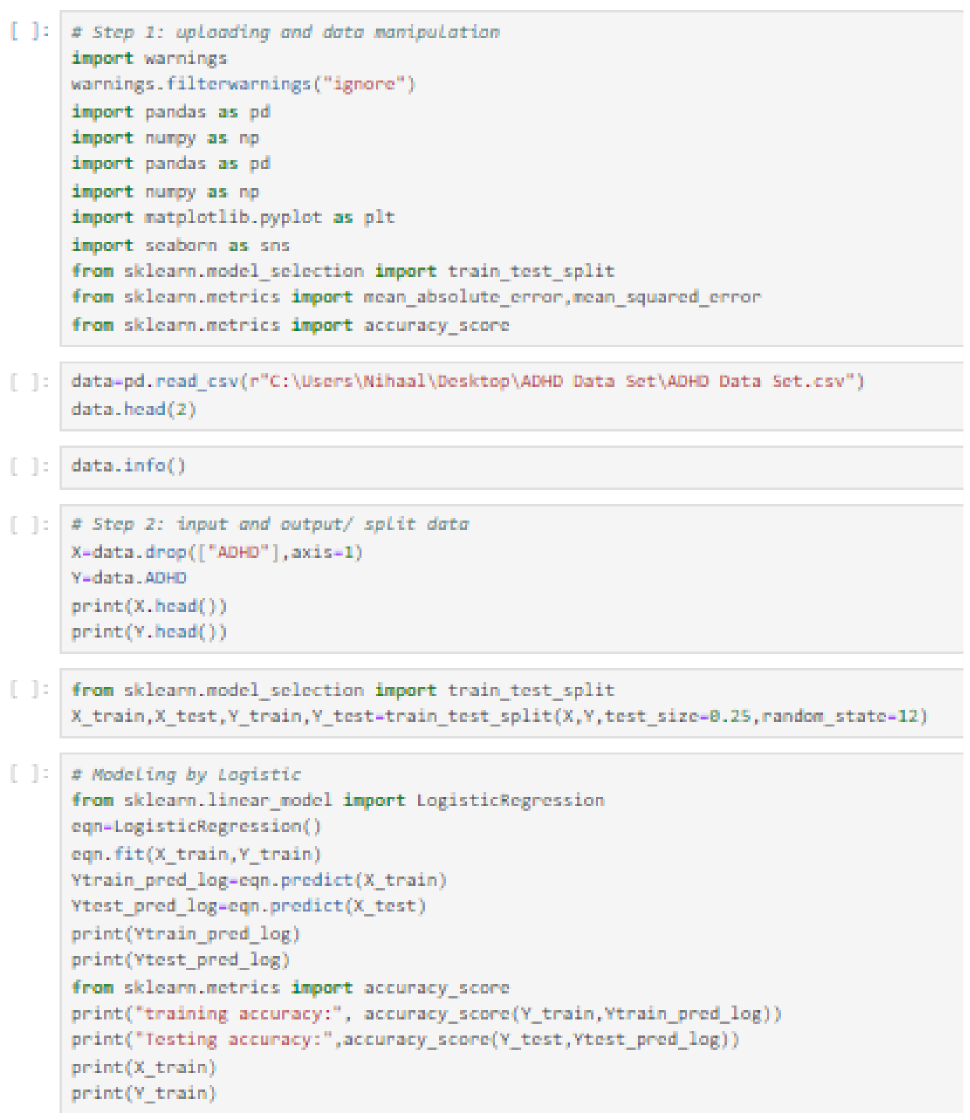
Fig 2.1



Mean FZ	Median FZ	Mode FZ	Mean CZ	Median CZ	Mode CZ	Mean PZ	Median PZ	Mode PZ
146.1724	156	191	146.3719	156	156	151.9671	163	200
146.4591	121	121	145.5895	121	121	153.2858	163	126
146.4591	121	121	145.5895	121	121	153.2858	163	126
140.2331	121	121	140.6786	121	121	146.4902	126	126
136.5224	121	121	136.4654	121	121	142.7647	126	126
139.4285	156	156	138.9988	156	121	145.0682	126	126
138.3882	121	156	137.5279	121	121	144.5338	126	163
137.9832	121	121	137.724	121	156	144.1939	126	89
138.1674	121	121	138.3379	121	156	144.3021	126	126

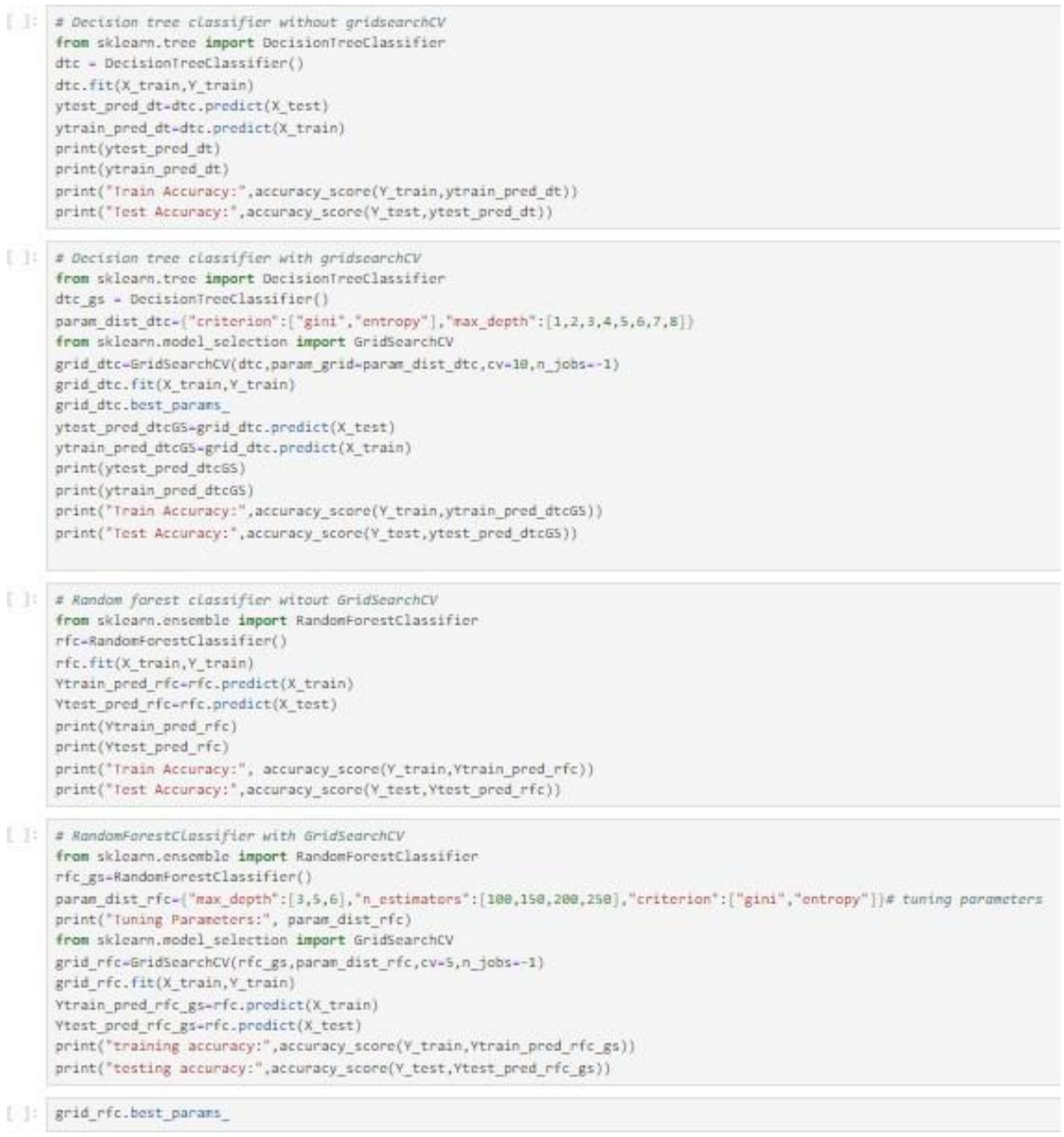
Fig 2.1

3)Applied Python code :



```
[ ]: # Step 1: uploading and data manipulation
     import warnings
     warnings.filterwarnings("ignore")
     import pandas as pd
     import numpy as np
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import mean_absolute_error,mean_squared_error
     from sklearn.metrics import accuracy_score

[ ]: data=pd.read_csv(r"C:\Users\Nihaal\Desktop\ADHD Data Set\ADHD Data Set.csv")
     data.head(2)

[ ]: data.info()

[ ]: # Step 2: input and output/ split data
     X=data.drop(["ADHD"],axis=1)
     Y=data.ADHD
     print(X.head())
     print(Y.head())

[ ]: from sklearn.model_selection import train_test_split
     X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,random_state=12)

[ ]: # Modeling by Logistic
     from sklearn.linear_model import LogisticRegression
     eqn=LogisticRegression()
     eqn.fit(X_train,Y_train)
     Ytrain_pred_log=eqn.predict(X_train)
     Ytest_pred_log=eqn.predict(X_test)
     print(Ytrain_pred_log)
     print(Ytest_pred_log)
     from sklearn.metrics import accuracy_score
     print("training accuracy:", accuracy_score(Y_train,Ytrain_pred_log))
     print("Testing accuracy:",accuracy_score(Y_test,Ytest_pred_log))
     print(X_train)
     print(Y_train)
```

Fig 3.1

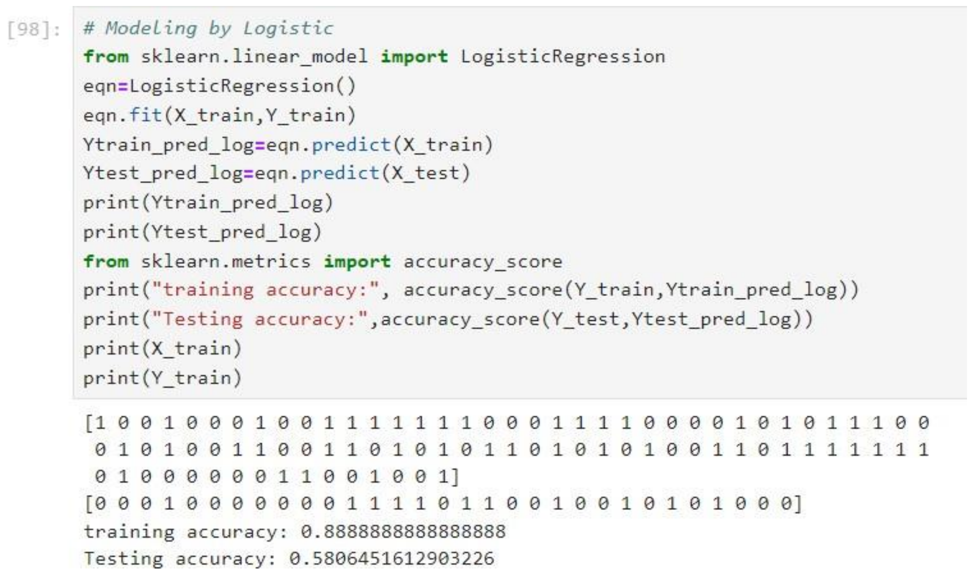


```
[ ]: # Decision tree classifier without gridsearchCV
     from sklearn.tree import DecisionTreeClassifier
     dtc = DecisionTreeClassifier()
     dtc.fit(X_train,Y_train)
     ytest_pred_dt=dtc.predict(X_test)
     ytrain_pred_dt=dtc.predict(X_train)
     print(ytest_pred_dt)
     print(ytrain_pred_dt)
     print("Train Accuracy:",accuracy_score(Y_train,ytrain_pred_dt))
     print("Test Accuracy:",accuracy_score(Y_test,ytest_pred_dt))

[ ]: # Decision tree classifier with gridsearchCV
     from sklearn.tree import DecisionTreeClassifier
     dtc_gs = DecisionTreeClassifier()
     param_dist_dtc={"criterion":["gini","entropy"],"max_depth":[1,2,3,4,5,6,7,8]}
     from sklearn.model_selection import GridSearchCV
     grid_dtc=GridSearchCV(dtc,param_grid=param_dist_dtc,cv=10,n_jobs=-1)
     grid_dtc.fit(X_train,Y_train)
     grid_dtc.best_params_
     ytest_pred_dtcGS=grid_dtc.predict(X_test)
     ytrain_pred_dtcGS=grid_dtc.predict(X_train)
     print(ytest_pred_dtcGS)
     print(ytrain_pred_dtcGS)
     print("Train Accuracy:",accuracy_score(Y_train,ytrain_pred_dtcGS))
     print("Test Accuracy:",accuracy_score(Y_test,ytest_pred_dtcGS))

[ ]: # Random forest classifier witout GridSearchCV
     from sklearn.ensemble import RandomForestClassifier
     rfc=RandomForestClassifier()
     rfc.fit(X_train,Y_train)
     Ytrain_pred_rfc=rfc.predict(X_train)
     Ytest_pred_rfc=rfc.predict(X_test)
     print(Ytrain_pred_rfc)
     print(Ytest_pred_rfc)
     print("Train Accuracy:", accuracy_score(Y_train,Ytrain_pred_rfc))
     print("Test Accuracy:",accuracy_score(Y_test,Ytest_pred_rfc))

[ ]: # RandomForestClassifier with GridSearchCV
     from sklearn.ensemble import RandomForestClassifier
     rfc_gs=RandomForestClassifier()
     param_dist_rfc={"max_depth":[3,5,6],"n_estimators":[100,150,200,250],"criterion":["gini","entropy"]}# tuning parameters
     print("Tuning Parameters:", param_dist_rfc)
     from sklearn.model_selection import GridSearchCV
     grid_rfc=GridSearchCV(rfc_gs,param_dist_rfc,cv=5,n_jobs=-1)
     grid_rfc.fit(X_train,Y_train)
     Ytrain_pred_rfc_gs=rfc.predict(X_train)
     Ytest_pred_rfc_gs=rfc.predict(X_test)
     print("training accuracy:",accuracy_score(Y_train,Ytrain_pred_rfc_gs))
     print("testing accuracy:",accuracy_score(Y_test,Ytest_pred_rfc_gs))

[ ]: grid_rfc.best_params_
```

Fig 3.2

3) Output :

I] Accuracy using Logistic Regresssion Algorithm:

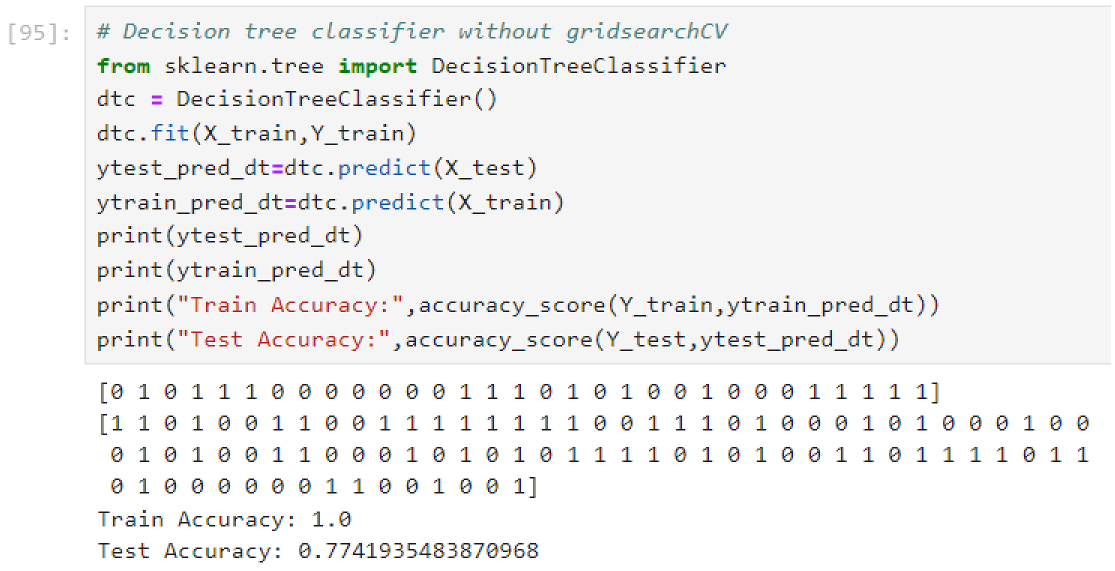


```
[98]: # Modeling by Logistic
from sklearn.linear_model import LogisticRegression
eqn=LogisticRegression()
eqn.fit(X_train,Y_train)
Ytrain_pred_log=eqn.predict(X_train)
Ytest_pred_log=eqn.predict(X_test)
print(Ytrain_pred_log)
print(Ytest_pred_log)
from sklearn.metrics import accuracy_score
print("training accuracy:", accuracy_score(Y_train,Ytrain_pred_log))
print("Testing accuracy:",accuracy_score(Y_test,Ytest_pred_log))
print(X_train)
print(Y_train)
```

```
[1 0 0 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 1 1 1 0 0
 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1
 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1]
[0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0]
training accuracy: 0.8888888888888888
Testing accuracy: 0.5806451612903226
```

Fig 4.1

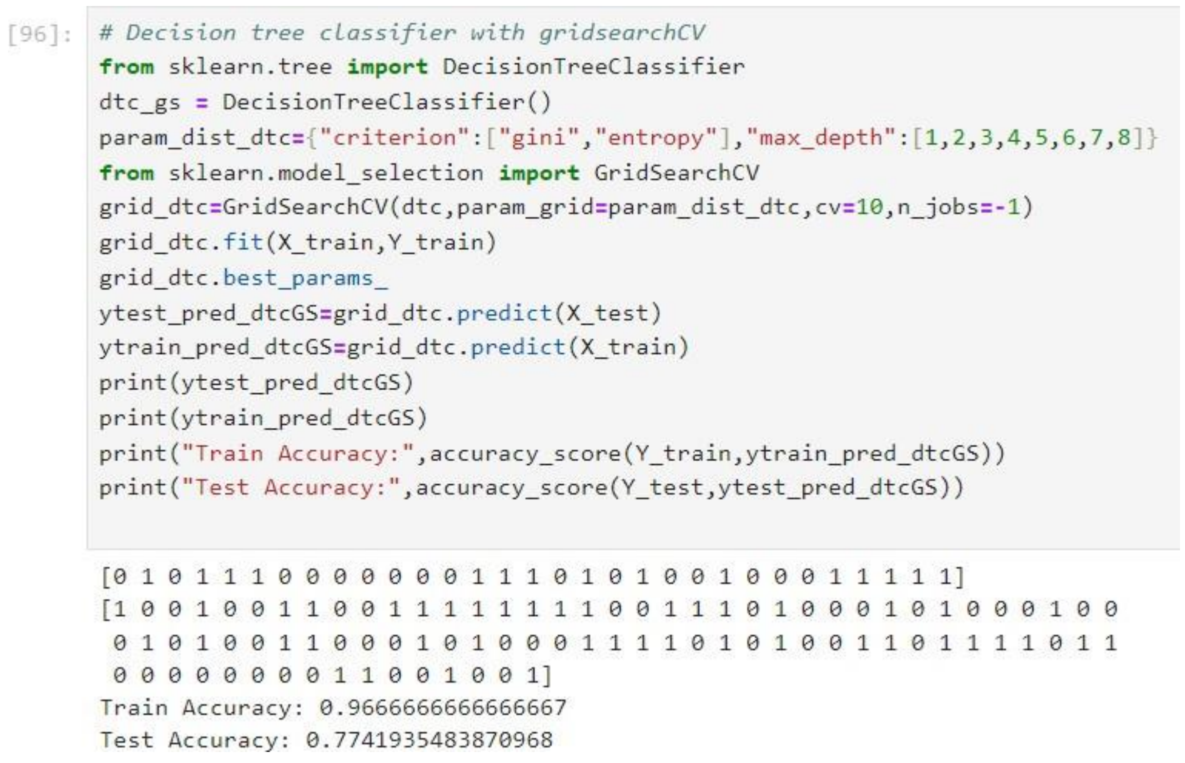
I] Accuracy using Decision Tree Algorithm:



```
[95]: # Decision tree classifier without gridsearchCV
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train,Y_train)
ytest_pred_dt=dtc.predict(X_test)
ytrain_pred_dt=dtc.predict(X_train)
print(ytest_pred_dt)
print(ytrain_pred_dt)
print("Train Accuracy:",accuracy_score(Y_train,ytrain_pred_dt))
print("Test Accuracy:",accuracy_score(Y_test,ytest_pred_dt))
```

```
[0 1 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 0 1 0 0 0 1 1 1 1 1]
[1 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0
 0 1 0 1 0 0 1 1 0 0 0 1 0 1 0 1 0 1 1 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1
 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1]
Train Accuracy: 1.0
Test Accuracy: 0.7741935483870968
```

Fig 4.2

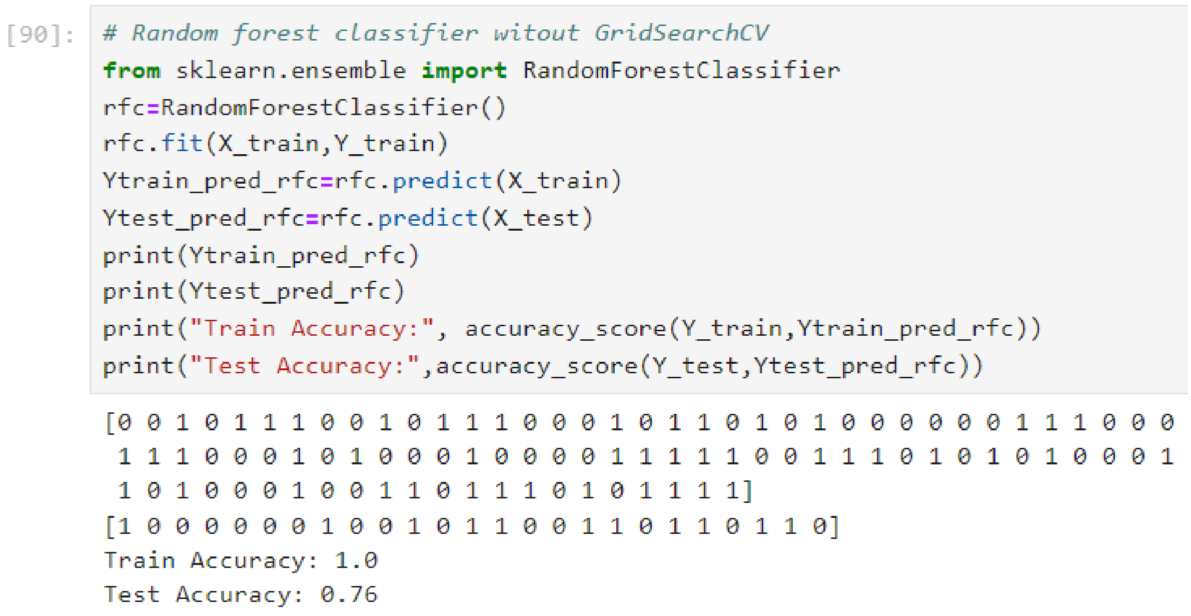


```
[96]: # Decision tree classifier with gridsearchCV
from sklearn.tree import DecisionTreeClassifier
dtc_gs = DecisionTreeClassifier()
param_dist_dtc={"criterion":["gini","entropy"],"max_depth":[1,2,3,4,5,6,7,8]}
from sklearn.model_selection import GridSearchCV
grid_dtc=GridSearchCV(dtc,param_grid=param_dist_dtc,cv=10,n_jobs=-1)
grid_dtc.fit(X_train,Y_train)
grid_dtc.best_params_
ytest_pred_dtcGS=grid_dtc.predict(X_test)
ytrain_pred_dtcGS=grid_dtc.predict(X_train)
print(ytest_pred_dtcGS)
print(ytrain_pred_dtcGS)
print("Train Accuracy:",accuracy_score(Y_train,ytrain_pred_dtcGS))
print("Test Accuracy:",accuracy_score(Y_test,ytest_pred_dtcGS))
```

```
[0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 1 0 0 1 0 0 0 1 1 1 1 1]
[1 0 0 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0
 0 1 0 1 0 0 1 1 0 0 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1
 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1]
Train Accuracy: 0.9666666666666667
Test Accuracy: 0.7741935483870968
```

Fig 4.3

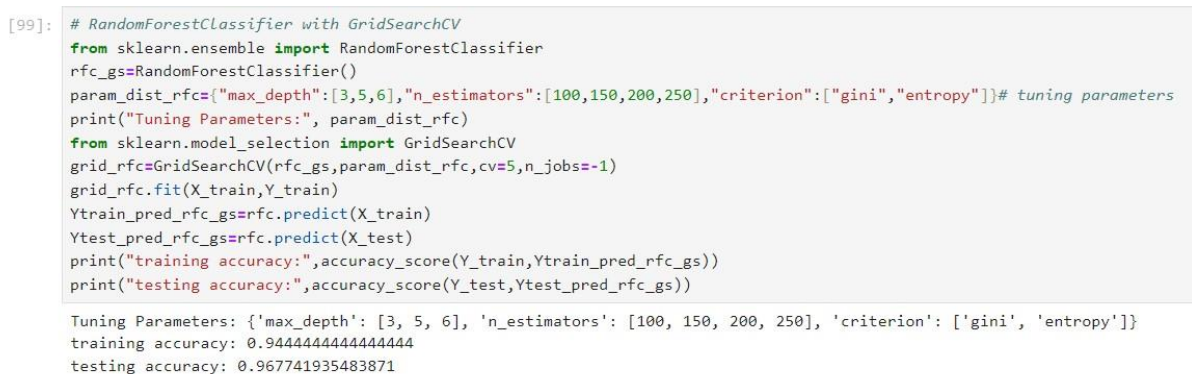
I] Accuracy using Random Forest Algorithm:



```
[90]: # Random forest classifier witout GridSearchCV
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(X_train,Y_train)
Ytrain_pred_rfc=rfc.predict(X_train)
Ytest_pred_rfc=rfc.predict(X_test)
print(Ytrain_pred_rfc)
print(Ytest_pred_rfc)
print("Train Accuracy:", accuracy_score(Y_train,Ytrain_pred_rfc))
print("Test Accuracy:",accuracy_score(Y_test,Ytest_pred_rfc))
```

```
[0 0 1 0 1 1 1 0 0 1 0 1 1 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 0
 1 1 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 1 0 1 0 0 0 1
 1 0 1 0 0 0 1 0 0 1 1 0 1 1 1 0 1 0 1 1 1 1]
[1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0]
Train Accuracy: 1.0
Test Accuracy: 0.76
```

Fig 4.4



```
[99]: # RandomForestClassifier with GridSearchCV
      from sklearn.ensemble import RandomForestClassifier
      rfc_gs=RandomForestClassifier()
      param_dist_rfc={"max_depth":[3,5,6],"n_estimators":[100,150,200,250],"criterion":["gini","entropy"]}# tuning parameters
      print("Tuning Parameters:", param_dist_rfc)
      from sklearn.model_selection import GridSearchCV
      grid_rfc=GridSearchCV(rfc_gs,param_dist_rfc,cv=5,n_jobs=-1)
      grid_rfc.fit(X_train,Y_train)
      Ytrain_pred_rfc_gs=rfc.predict(X_train)
      Ytest_pred_rfc_gs=rfc.predict(X_test)
      print("training accuracy:",accuracy_score(Y_train,Ytrain_pred_rfc_gs))
      print("testing accuracy:",accuracy_score(Y_test,Ytest_pred_rfc_gs))
```

```
Tuning Parameters: {'max_depth': [3, 5, 6], 'n_estimators': [100, 150, 200, 250], 'criterion': ['gini', 'entropy']}
training accuracy: 0.9444444444444444
testing accuracy: 0.967741935483871
```

Fig 4.5

CHAPTER NO. 6
CONCLUSION AND FUTURE WORK

The project titled "Prediction of ADHD Using Machine Learning" presents a mini-project aimed at developing a machine learning (ML) model to predict Attention Deficit Hyperactivity Disorder (ADHD) in children. ADHD is a common neurodevelopmental disorder characterized by symptoms like hyperactivity, inattention, and impulsivity. Traditional diagnostic methods rely on neuroimaging and manual assessments, which can be complex and costly. The project explores the potential of EEG (Electroencephalography) signals to aid in diagnosing ADHD, leveraging ML algorithms such as Logistic Regression, Decision Tree (DT), and Random Forest (RF). The key contributions include using EEG data from 19 channels, unlike prior studies that limited analysis to fewer channels. The model incorporates time-domain features and compares the performance of different algorithms. Hyperparameter tuning for DT and RF is applied to avoid overfitting. The dataset used consists of 121 children, with ADHD and control groups. The conclusions for the current work are as follows:

All the 19 channels signal was utilized to model the detection of ADHD or current work, three features namly median, mean and mode are used.

Hence the total inputs for the model is 57. The algorithms namely logistic regression, DT and RF are used for modeling study and RF gave best results without overfitting.

The future work outlined in the project focuses on several key areas for enhancing the current ADHD prediction model using machine learning:

Feature Expansion: The next phase will involve extracting additional features from each of the 19 EEG channels, with a goal of using at least 10 features per channel. This expansion will provide more data points for training the model and improve its accuracy.

Feature Reduction: To manage the increased complexity from more features, the project plans to implement Principal Component Analysis (PCA). This will help reduce the number of features while retaining the most important data, improving both performance and interpretability.

Algorithm Comparison: The study will compare more advanced machine learning algorithms, such as Artificial Neural Networks (ANN) and Naive Bayes, to evaluate their effectiveness in classifying ADHD. ANN, with its capability to model complex non-linear relationships, will be contrasted with simpler models like Naive Bayes to assess their performance on the dataset.

Frequency Domain Analysis: While the current model focuses on the time domain, future work will include frequency domain analysis of the EEG signals. This approach can uncover additional insights about brainwave patterns, potentially leading to more robust ADHD detection.

These steps aim to improve accuracy, reduce model complexity, and explore more sophisticated approaches for ADHD diagnosis.