**Prediction of ADHD Using Machine Learning**

**Source Code:**

**1] .mat to .csv file:**

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)

**2] Data Extraction from .csv files code:**

import os

import csv

import statistics

# 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}.")

**3] Code for the three algorithms:**

# Step 1: uploading and data manipulation

import warnings

warnings.filterwarnings("ignore")

import pandas as pd

import numpy as np

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=37)

# 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)

# 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":[4],"n\_estimators":[300],"criterion":["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\_