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_
```