Data preprocessing task done by Neha Chaudhari and Partha Vemuri

```
# loading the required libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.impute import KNNImputer

# reading the train and test datasets
categorical_data =
pd.read_excel("train_new/TRAIN_CATEGORICAL_METADATA_new.xlsx")
quantitative_data =
pd.read_excel("train_new/TRAIN_QUANTITATIVE_METADATA_new.xlsx")
connectome_data =
pd.read_csv("train_new/TRAIN_FUNCTIONAL_CONNECTOME_MATRICES_new_36P_Pearson.c
sv")
train_data = pd.read_excel("train_new/TRAINING_SOLUTIONS.xlsx")

categorical_test = pd.read_excel("test/TEST_CATEGORICAL.xlsx")
quantitative_test = pd.read_excel("test/TEST_QUANTITATIVE_METADATA.xlsx")
connectome_test = pd.read_csv("test/TEST_FUNCTIONAL_CONNECTOME_MATRICES.csv")

# print(connectome_data)

# get summary statistics
print("Summary statistics for quantitatve metadata")
print(quantitative_data.describe().T)


# imputing quantitative data using median
# quantitative_data =
quantitative_data.fillna(quantitative_data.median(numeric_only=True))

# Separate participant_id
participant_id = quantitative_data['participant_id']

# Drop participant_id before imputation
quant_data_num = quantitative_data.drop(columns=['participant_id'])

# Applying KNN Imputer to handle missing values in the quantitative data
imputer = KNNImputer(n_neighbors=5)
quantitative_data = pd.DataFrame(imputer.fit_transform(quant_data_num),
columns=quant_data_num.columns)

# Adding participant_id back to succesfully merge the quantitative data later
based on participant id
```

```
quantitative_data['participant_id'] = participant_id.values

print(quantitative_data.isnull().sum())

# imputing categorial data using mode
for col in categorical_data.columns:
    if categorical_data[col].isnull().any():
        categorical_data[col].fillna(categorical_data[col].mode()[0],
inplace=True)

# handling connectcome data
print(connectome_data.isnull().sum())
print(connectome_data.info())

# summary statistics for connectcome data
print("Summary statistics for connectcome data")
print(connectome_data.describe().T)

# check for null values in the train data
print(train_data.isnull().sum())

# merging the 4 train datasets
merged_data = categorical_data.merge(quantitative_data, on='participant_id',
how='inner')
merged_data = merged_data.merge(connectome_data, on='participant_id',
how='inner')
merged_data = merged_data.merge(train_data, on='participant_id', how='inner')

# print(merged_data)

# check for missing values in the merged dataset
merged_data.isnull().sum()

merged_data_clean = merged_data.dropna() # drop any missing values present in
the dataset

# merged_data_clean.isnull().sum()
print("Shape after dropping missing values:", merged_data_clean.shape) #
check the no. of rows and columns in the final dataset

# categorical_cols =
merged_data_clean.select_dtypes(include='object').columns
# print(categorical_cols)
```

Exploratory Data Analysis

1. EDA for Quantitative Data

```
# plots for quant eda
```

```
# Creating histogram to understand the distribution of values in each
variable individually
quantitative_data.hist(bins=20, color='#69b3a2', edgecolor='black',
figsize=(15, 12))
plt.suptitle("Distribution of Coulmns in Quantitative data", fontsize=16)
plt.tight_layout()
plt.show()

# # corrrelation heatmap for quaantitative data
numeric_cols = quantitative_data.select_dtypes(include='number')
plt.figure(figsize=(10,12))
sns.heatmap(numeric_cols.corr(), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()



# Create a box plot for all numeric columns(except year)
categorical_clean = categorical_data.select_dtypes(include=['float64',
'int64'])
categorical_clean = categorical_clean.drop('Basic_Demos_Enroll_Year', axis =
1) # keeping year skews the data highly since the year value is in 2000's

# Create the boxplot
plt.figure(figsize=(12, 6))
sns.boxplot(data = categorical_clean)
plt.xticks(rotation=45)
plt.title("Box Plots based on Categorical factors")
plt.ylabel("Values")
plt.tight_layout()
plt.show()

# Separating the required numeric columns from the categorical daa
numerical_columns = [
    'Barratt_Barratt_P1_Edu',
    'Barratt_Barratt_P1_Occ',
    'Barratt_Barratt_P2_Edu',
    'Barratt_Barratt_P2_Occ'
]

# Boxplots by Study Site for Scores
for col in numerical_columns:
    plt.figure(figsize=(8, 5))
    sns.boxplot(x='Basic_Demos_Study_Site', y=col, data=categorical_data) #
creating boxplot using seaborn library
    plt.title(f"{col} by Study Site")
    plt.tight_layout()
    plt.show()

# Correlation Heatmap for Score Features
```

```
plt.figure(figsize=(8, 6))
corr = categorical_data[numerical_columns].corr() # using only the numeric
columns
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f") # Seaborn to create
a heatmap of feature correlation
plt.title("Correlation Heatmap (Barratt Scores)")
plt.tight_layout()
plt.show()

# Boxplot: ADHD score vs. Ethnicity
plt.figure(figsize=(12, 6))
sns.boxplot(data=merged_data, x='PreInt_Demos_Fam_Child_Ethnicity',
y='SDQ_SDQ_Hyperactivity')
plt.title('Hyperactivity Score by Ethnicity') # give appropriate plot title
plt.xticks(rotation=45) # rotate xlabels for better readability
plt.tight_layout()
plt.show()

# Boxplot: ADHD score vs. Study Site
plt.figure(figsize=(12, 6)) # set the figure size
sns.boxplot(data=merged_data, x='Basic_Demos_Study_Site',
y='SDQ_SDQ_Hyperactivity') # give appropriate plot title
plt.title('Hyperactivity Score by Study Site')
plt.xticks(rotation=45) # rotate xlabels for better readability
plt.tight_layout()
plt.show()


  sns.boxplot(data=merged_data, x='Basic_Demos_Study_Site',
y='SDQ_SDQ_Hyperactivity', palette='Set2')
```

3. EDA on Train data

```
 # print(train_data)
print(train_data.info()) # Check for object types

print(train_data.describe().T) # To get summary statistics on train data to
understand it

# histogram to see distribution of ADHD Outcomes by Gender
train_data_grouped = train_data.groupby(['Sex_F',
'ADHD_Outcome']).size().unstack(fill_value=0)
train_data_grouped.plot(kind='bar', stacked=False, color=['lightblue',
'pink'])
plt.xlabel('Gender')
plt.ylabel('No. of Participants')
plt.title('Distribution of ADHD Outcomes by Gender')
plt.xticks([0, 1], ['Male', 'Female'], rotation=0)
plt.legend(['No ADHD', 'ADHD'], title='ADHD Outcome')
```