Data Science Fundamentals (90001) - Final Project

ASD (Autism Spectrum Disorder) - Prediction based on Phenotypic Data

Tali Aharon 034791236

Import

In [226…

```
# Basic libraries for data handeling
import pandas as pd
import numpy as np

# For exploring and ploting the data
import matplotlib.pyplot as plt
import seaborn as sns
import statistics as stat
import scipy as scip

# For preprocessing
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.decomposition import PCA

# For preprocessing and for train-test-predict-evaluate...
from sklearn.pipeline import Pipeline

# For train-test-predict
from sklearn.model_selection import train_test_split, GridSearchCV
```

```
# Classification Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans as KMeans

# For evaluating our models
from sklearn import datasets, metrics
from sklearn.metrics import confusion_matrix as confusion_matrix

# I removed warnings to get a clean notebook/pdf
import warnings
warnings.filterwarnings('ignore')
```

Exploratory Data Analysis

Load and Preview the data:

We start with 2 CSV files: "ABIDEII_Composite_Phenotypic" and "ABIDEII_Long_Composite_Phenotypic".

In [227…

```
# Load CSV data files
df = pd.read_csv('ABIDEII_Composite_Phenotypic.csv', encoding='windows-1252')
#df.columns = [x.lower() for x in df.columns]

df_long = pd.read_csv('ABIDEII_Long_Composite_Phenotypic.csv', encoding='windows-1252')
#df_long.columns = [x.lower() for x in df_long.columns]

# selecting rows from df_long where SESSION='Baseline'
df_long_baseline = df_long.loc[df_long['SESSION'] == 'Baseline']
df_long_baseline.drop('SESSION', inplace=True, axis=1)

df = pd.concat([df, df_long_baseline])

df.head(50)
```

Out[227]:

	SITE_ID	SUB_ID	NDAR_GUID	DX_GROUP	PDD_DSM_IV_TR	ASD_DSM_5	AGE_AT_SCAN	SEX	HANDEDNESS_CATEGORY	HANDED
0	ABIDEII-BNI_1	29006	NaN	1	NaN	NaN	48.0	1	1.0	
1	ABIDEII-BNI_1	29007	NaN	1	NaN	NaN	41.0	1	1.0	
2	ABIDEII-BNI_1	29008	NaN	1	NaN	NaN	59.0	1	1.0	
3	ABIDEII-BNI_1	29009	NaN	1	NaN	NaN	57.0	1	1.0	
4	ABIDEII-BNI_1	29010	NaN	1	NaN	NaN	45.0	1	1.0	
5	ABIDEII-BNI_1	29012	NaN	1	NaN	NaN	62.0	1	1.0	
6	ABIDEII-BNI_1	29013	NaN	1	NaN	NaN	51.0	1	1.0	
7	ABIDEII-BNI_1	29015	NaN	1	NaN	NaN	47.0	1	1.0	

8	ABIDEII-BNI_1	29017	NaN	1	NaN	NaN	55.0	1	1.0
9	ABIDEII-BNI_1	29025	NaN	1	NaN	NaN	57.0	1	1.0
10	ABIDEII-BNI_1	29026	NaN	1	NaN	NaN	54.0	1	1.0
11	ABIDEII-BNI_1	29027	NaN	1	NaN	NaN	45.0	1	1.0
12	ABIDEII-BNI_1	29028	NaN	1	NaN	NaN	21.0	1	1.0
13	ABIDEII-BNI_1	29029	NaN	1	NaN	NaN	20.0	1	1.0
14	ABIDEII-BNI_1	29030	NaN	1	NaN	NaN	18.0	1	1.0
15	ABIDEII-BNI_1	29031	NaN	1	NaN	NaN	21.0	1	1.0
16	ABIDEII-BNI_1	29037	NaN	1	NaN	NaN	19.0	1	1.0
17	ABIDEII-BNI_1	29039	NaN	1	NaN	NaN	19.0	1	1.0
18	ABIDEII-BNI_1	29041	NaN	1	NaN	NaN	19.0	1	1.0
19	ABIDEII-BNI_1	29042	NaN	1	NaN	NaN	25.0	1	1.0
20	ABIDEII-BNI_1	29043	NaN	1	NaN	NaN	53.0	1	1.0
21	ABIDEII-BNI_1	29046	NaN	1	NaN	NaN	18.0	1	1.0

22	ABIDEII-BNI_1	29052	NaN	1	NaN	NaN	40.0	1	1.0
23	ABIDEII-BNI_1	29053	NaN	1	NaN	NaN	24.0	1	1.0
24	ABIDEII-BNI_1	29055	NaN	1	NaN	NaN	22.0	1	1.0
25	ABIDEII-BNI_1	30144	NaN	1	NaN	NaN	22.0	1	1.0
26	ABIDEII-BNI_1	30148	NaN	1	NaN	NaN	55.0	1	1.0
27	ABIDEII-BNI_1	30150	NaN	1	NaN	NaN	47.0	1	1.0
28	ABIDEII-BNI_1	30151	NaN	1	NaN	NaN	22.0	1	1.0
29	ABIDEII-BNI_1	29011	NaN	2	NaN	NaN	48.0	1	1.0
30	ABIDEII-BNI_1	29014	NaN	2	NaN	NaN	64.0	1	1.0
31	ABIDEII-BNI_1	29016	NaN	2	NaN	NaN	41.0	1	1.0
32	ABIDEII-BNI_1	29018	NaN	2	NaN	NaN	51.0	1	1.0
33	ABIDEII-BNI_1	29019	NaN	2	NaN	NaN	62.0	1	1.0
34	ABIDEII-BNI_1	29020	NaN	2	NaN	NaN	46.0	1	1.0
35	ABIDEII-BNI_1	29021	NaN	2	NaN	NaN	43.0	1	1.0
	ABIDEII-								

36	BNI_1	29022	NaN	2	NaN	NaN	40.0	1	1.0
37	ABIDEII-BNI_1	29023	NaN	2	NaN	NaN	46.0	1	1.0
38	ABIDEII-BNI_1	29024	NaN	2	NaN	NaN	60.0	1	1.0
39	ABIDEII-BNI_1	29032	NaN	2	NaN	NaN	20.0	1	1.0
40	ABIDEII-BNI_1	29033	NaN	2	NaN	NaN	54.0	1	1.0
41	ABIDEII-BNI_1	29034	NaN	2	NaN	NaN	21.0	1	1.0
42	ABIDEII-BNI_1	29035	NaN	2	NaN	NaN	22.0	1	1.0
43	ABIDEII-BNI_1	29036	NaN	2	NaN	NaN	56.0	1	1.0
44	ABIDEII-BNI_1	29038	NaN	2	NaN	NaN	21.0	1	1.0
45	ABIDEII-BNI_1	29040	NaN	2	NaN	NaN	49.0	1	1.0
46	ABIDEII-BNI_1	29044	NaN	2	NaN	NaN	19.0	1	1.0
47	ABIDEII-BNI_1	29045	NaN	2	NaN	NaN	48.0	1	1.0
48	ABIDEII-BNI_1	29047	NaN	2	NaN	NaN	55.0	1	1.0
49	ABIDEII-BNI_1	29048	NaN	2	NaN	NaN	25.0	1	1.0

50 rows × 348 columns

Statisticaly describing the numeric columns

In [228…

```
df.describe()
```

Out[228]:

	SUB_ID	DX_GROUP	PDD_DSM_IV_TR	ASD_DSM_5	AGE_AT_SCAN	SEX	HANDEDNESS_CATEGORY	HANDEDNESS
count	1152.000000	1152.000000	633.000000	205.000000	1152.000000	1152.000000	1129.000000	6
mean	30035.146701	1.527778	0.829384	0.536585	14.797624	1.228299	1.181577	
std	3845.471750	0.499445	1.040002	0.499880	9.023974	0.419918	0.517236	
min	28675.000000	1.000000	0.000000	0.000000	5.128000	1.000000	1.000000	-1
25%	28991.750000	1.000000	0.000000	0.000000	9.388390	1.000000	1.000000	
50%	29311.500000	2.000000	0.000000	1.000000	11.645205	1.000000	1.000000	
75%	29622.250000	2.000000	2.000000	1.000000	17.757500	1.000000	1.000000	1
max	51315.000000	2.000000	3.000000	1.000000	64.000000	2.000000	3.000000	1

8 rows × 340 columns

1st phase of Feature Selection - using domain knowledge relevance to ASD

In [229…

```
df_ASD = df[['SUB_ID', 'NDAR_GUID', 'DX_GROUP', 'PDD_DSM_IV_TR', 'ASD_DSM_5', 'AGE_AT_SCAN ', 'SEX', 'HANDEDNESS_
df_ASD.head()
```

Out[229]:

	SUB_ID	NDAR_GUID	DX_GROUP	PDD_DSM_IV_TR	ASD_DSM_5	AGE_AT_SCAN	SEX	HANDEDNESS_CATEGORY	HANDEDNESS_SCC
0	29006	NaN	1	NaN	NaN	48.0	1	1.0	
1	29007	NaN	1	NaN	NaN	41.0	1	1.0	
2	29008	NaN	1	NaN	NaN	59.0	1	1.0	
3	29009	NaN	1	NaN	NaN	57.0	1	1.0	
4	29010	NaN	1	NaN	NaN	45.0	1	1.0	

5 rows × 135 columns

2nd phase of Feature Reduction - aggregating multiple columns into one (using mean)

In [230…

```
# Add columns using aggregation functions applyed on existing columns
ADI_R_df = df_ASD[['ADI_R_SOCIAL_TOTAL_A', 'ADI_R_VERBAL_TOTAL_BV', 'ADI_R_NONVERBAL_TOTAL_BV', 'ADI_R_RRB_TOTAL_
df_ASD['ADI_R_MEAN'] = np.nan
for index, row in ADI_R_df.iterrows():
    df_ASD.at[index, 'ADI_R_MEAN'] = np.nanmean(np.array(row))

BRIEF_df = df_ASD[['BRIEF_INHIBIT_T', 'BRIEF_SHIFT_T', 'BRIEF_EMOTIONAL_T', 'BRIEF_BRI_T', 'BRIEF_INITIATE_T', 'B
df_ASD['BRIEF_MEAN'] = np.nan
for index, row in BRIEF_df.iterrows():
    df_ASD.at[index, 'BRIEF_MEAN'] = np.nanmean(np.array(row))

CELF_df = df_ASD[['CELF_5-8_CORE_S', 'CELF_5-8_RECEPTIVE_S', 'CELF_5-8_EXPRESSIVE_S', 'CELF_9-21_CORE_S', 'CELF_9
df_ASD['CELF_MEAN'] = np.nan
for index, row in CELF_df.iterrows():
    df_ASD.at[index, 'CELF_MEAN'] = np.nanmean(np.array(row))

BASC2_df = df_ASD[['BASC2_PRS_ANGER_T', 'BASC2_PRS_HYPERACTIVITY_T', 'BASC2_PRS_AGGRESSION_T', 'BASC2_PRS_CONDUCT
df_ASD['BASC2_MEAN'] = np.nan
for index, row in BASC2_df.iterrows():
```

```
    df_ASD.at[index, 'BASC2_MEAN'] = np.nanmean(np.array(row))

CPRS_df = df_ASD[['CPRS_OPP', 'CPRS_COG-INATT', 'CPRS_HYPERACT', 'CPRS_ANX_SHY', 'CPRS_PERFECT', 'CPRS_SOCIAL_PRO
df_ASD['CPRS_MEAN'] = np.nan
for index, row in CPRS_df.iterrows():
    df_ASD.at[index, 'CPRS_MEAN'] = np.nanmean(np.array(row))

CASI_df = df_ASD[['CASI_ADHD-I_CUTOFF', 'CASI_ADHD-H_CUTOFF', 'CASI_ADHD-C_CUTOFF', 'CASI_ODD_CUTOFF', 'CASI_CD_C
df_ASD['CASI_MEAN'] = np.nan
for index, row in CASI_df.iterrows():
    df_ASD.at[index, 'CASI_MEAN'] = np.nanmean(np.array(row))

CSI_df = df_ASD[['CSI_ADHD-I_CUTOFF', 'CSI_ADHD-H_CUTOFF', 'CSI_ADHD-C_CUTOFF', 'CSI_ODD_ CUTOFF', 'CSI_CD_CUTOFF
df_ASD['CSI_MEAN'] = np.nan
for index, row in CSI_df.iterrows():
    df_ASD.at[index, 'CSI_MEAN'] = np.nanmean(np.array(row))

# Drop columns which were already aggregated into a new column
df_ASD = df_ASD.drop(['ADI_R_SOCIAL_TOTAL_A', 'ADI_R_VERBAL_TOTAL_BV', 'ADI_R_NONVERBAL_TOTAL_BV', 'ADI_R_RRB_TOT
df_ASD.head(200)
```

Out[230]:

	SUB_ID	NDAR_GUID	DX_GROUP	PDD_DSM_IV_TR	ASD_DSM_5	AGE_AT_SCAN	SEX	HANDEDNESS_CATEGORY	HANDEDNESS_S
0	29006	NaN	1	NaN	NaN	48.000000	1	1.0	
1	29007	NaN	1	NaN	NaN	41.000000	1	1.0	
2	29008	NaN	1	NaN	NaN	59.000000	1	1.0	
3	29009	NaN	1	NaN	NaN	57.000000	1	1.0	
4	29010	NaN	1	NaN	NaN	45.000000	1	1.0	
...	...	...	...	...	...	...	...	...	
195	28838	NaN	1	1.0	NaN	11.484932	1	1.0	
196	28839	NaN	1	1.0	NaN	11.353425	2	1.0	
197	28840	NaN	1	1.0	NaN	11.652055	1	1.0	
198	28843	NaN	1	1.0	NaN	11.123288	1	1.0	
199	28844	NaN	1	3.0	NaN	11.912329	1	1.0	

200 rows × 31 columns

In [231…

```
df_ASD.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1152 entries, 0 to 74
Data columns (total 31 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   SUB_ID                       1152 non-null   int64  
 1   NDAR_GUID                    310 non-null    object 
 2   DX_GROUP                     1152 non-null   int64  
 3   PDD_DSM_IV_TR                633 non-null    float64
 4   ASD_DSM_5                    205 non-null    float64
 5   AGE_AT_SCAN                  1152 non-null   float64
 6   SEX                          1152 non-null   int64  
 7   HANDEDNESS_CATEGORY          1129 non-null   float64
 8   HANDEDNESS_SCORES            641 non-null    float64
 9   ADOS_G_TOTAL                 369 non-null    float64
 10  ADOS_2_TOTAL                 283 non-null    float64
 11  ADOS_2_SEVERITY_TOTAL        278 non-null    float64
 12  SRS_TOTAL_RAW                785 non-null    float64
 13  SRS_TOTAL_T                  756 non-null    float64
 14  SCQ_TOTAL                    293 non-null    float64
 15  AQ_TOTAL                     44 non-null     float64
 16  VINELAND_SUM_SCORES          103 non-null    float64
 17  RBSR_6SUBSCALE_TOTAL         451 non-null    float64
 18  RBSR_5SUBSCALE_TOTAL         261 non-null    float64
 19  MASC_TOTAL_T                 216 non-null    float64
 20  CBCL_6-18_TOTAL_COMPETENCE_T  101 non-null    float64
 21  CBCL_6-18_TOTAL_PROBLEM_T    400 non-null    float64
 22  CBCL_1.5-5_TOTAL_T           17 non-null     float64
 23  BDI_TOTAL                    135 non-null    float64
 24  ADI_R_MEAN                   346 non-null    float64
 25  BRIEF_MEAN                   487 non-null    float64
 26  CELF_MEAN                    56 non-null     float64
 27  BASC2_MEAN                   149 non-null    float64
 28  CPRS_MEAN                    100 non-null    float64
 29  CASI_MEAN                    90 non-null     float64
 30  CSI_MEAN                     17 non-null     float64
dtypes: float64(27), int64(3), object(1)
memory usage: 320.3+ KB
```

3rd phase of Feature Selection - removing features where 90% or more of the data is missing

In [232…

```
df_ASD = df_ASD.drop(['AQ_TOTAL', 'VINELAND_SUM_SCORES', 'CBCL_6-18_TOTAL_COMPETENCE_T', 'CBCL_1.5-5_TOTAL_T', 'C
df_ASD.head(200)
```

Out[232]:

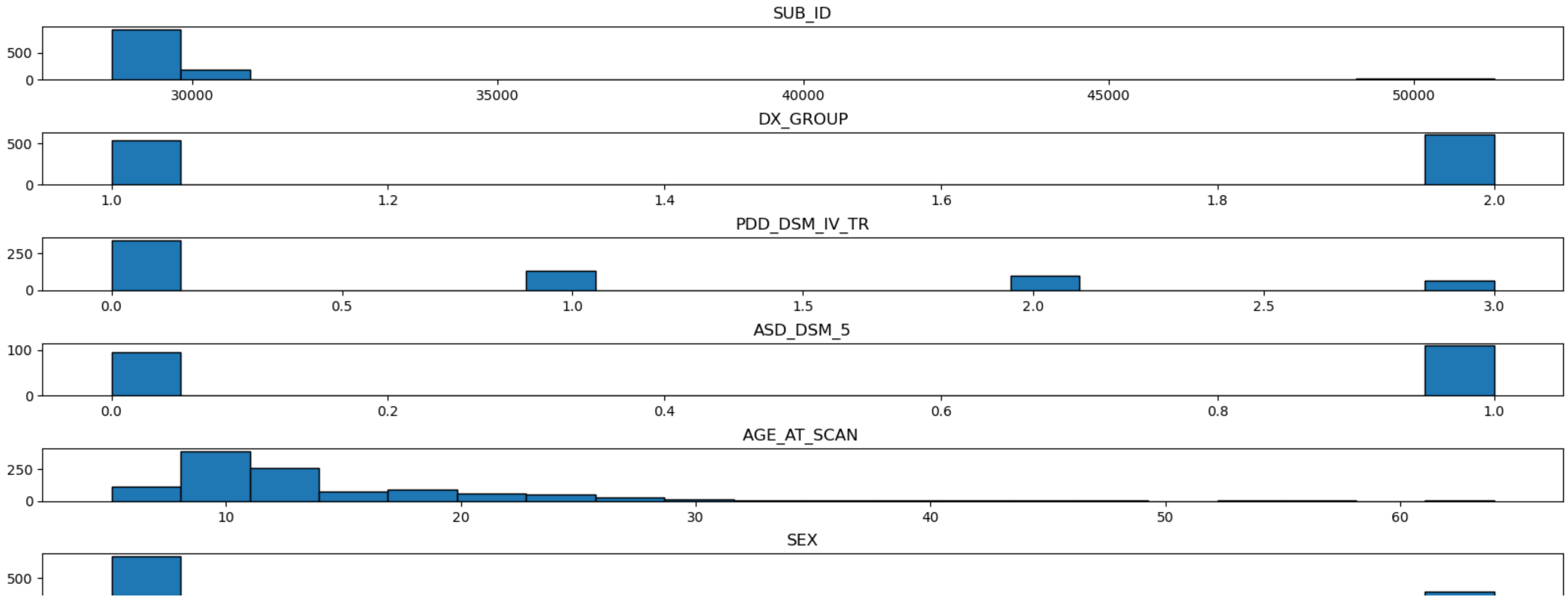
	SUB_ID	NDAR_GUID	DX_GROUP	PDD_DSM_IV_TR	ASD_DSM_5	AGE_AT_SCAN	SEX	HANDEDNESS_CATEGORY	HANDEDNESS_S
0	29006	NaN	1	NaN	NaN	48.000000	1	1.0	
1	29007	NaN	1	NaN	NaN	41.000000	1	1.0	
2	29008	NaN	1	NaN	NaN	59.000000	1	1.0	
3	29009	NaN	1	NaN	NaN	57.000000	1	1.0	
4	29010	NaN	1	NaN	NaN	45.000000	1	1.0	
...	...	...	...	...	...	...	...	...	
195	28838	NaN	1	1.0	NaN	11.484932	1	1.0	
196	28839	NaN	1	1.0	NaN	11.353425	2	1.0	
197	28840	NaN	1	1.0	NaN	11.652055	1	1.0	
198	28843	NaN	1	1.0	NaN	11.123288	1	1.0	
199	28844	NaN	1	3.0	NaN	11.912329	1	1.0	

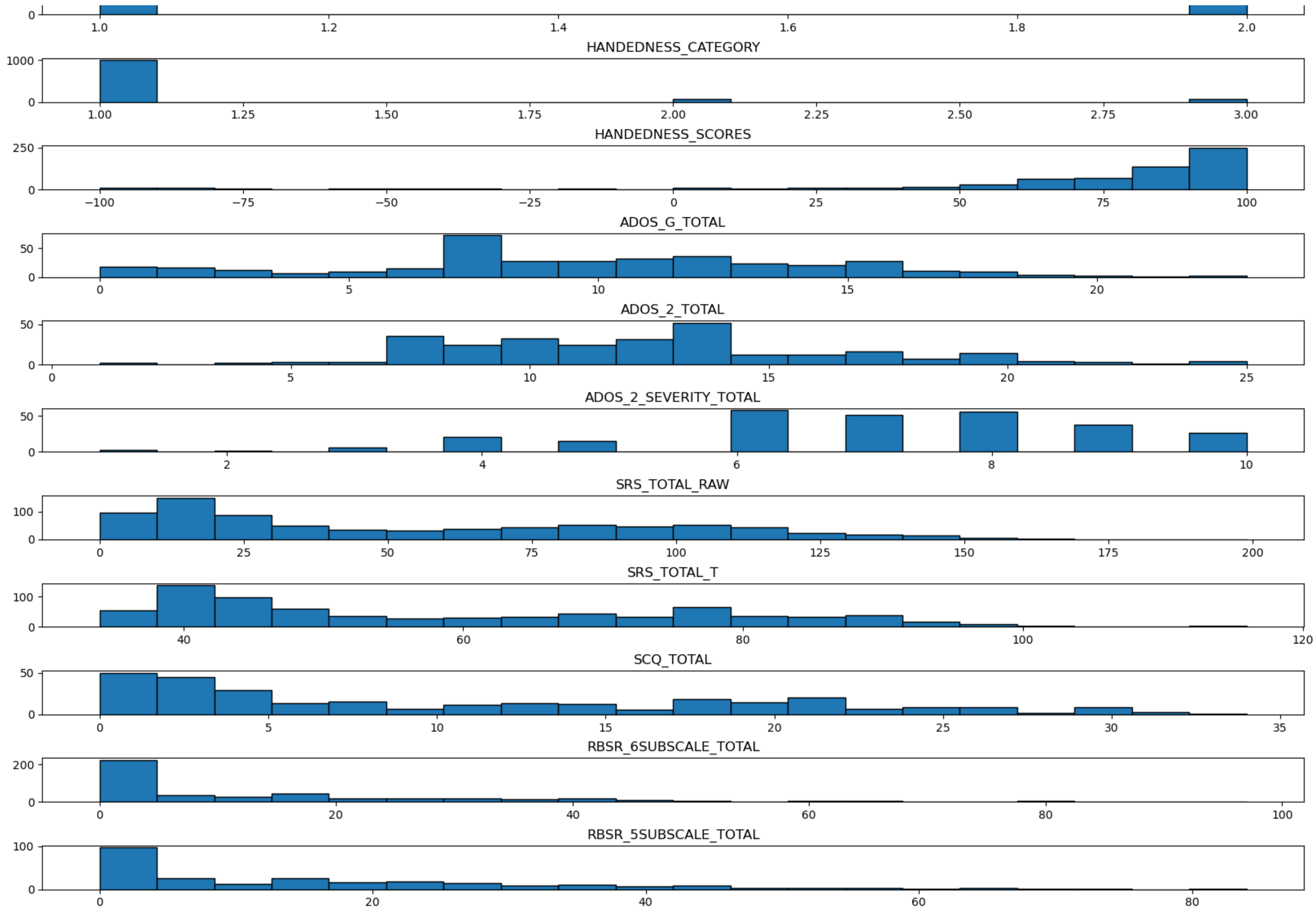
200 rows × 23 columns

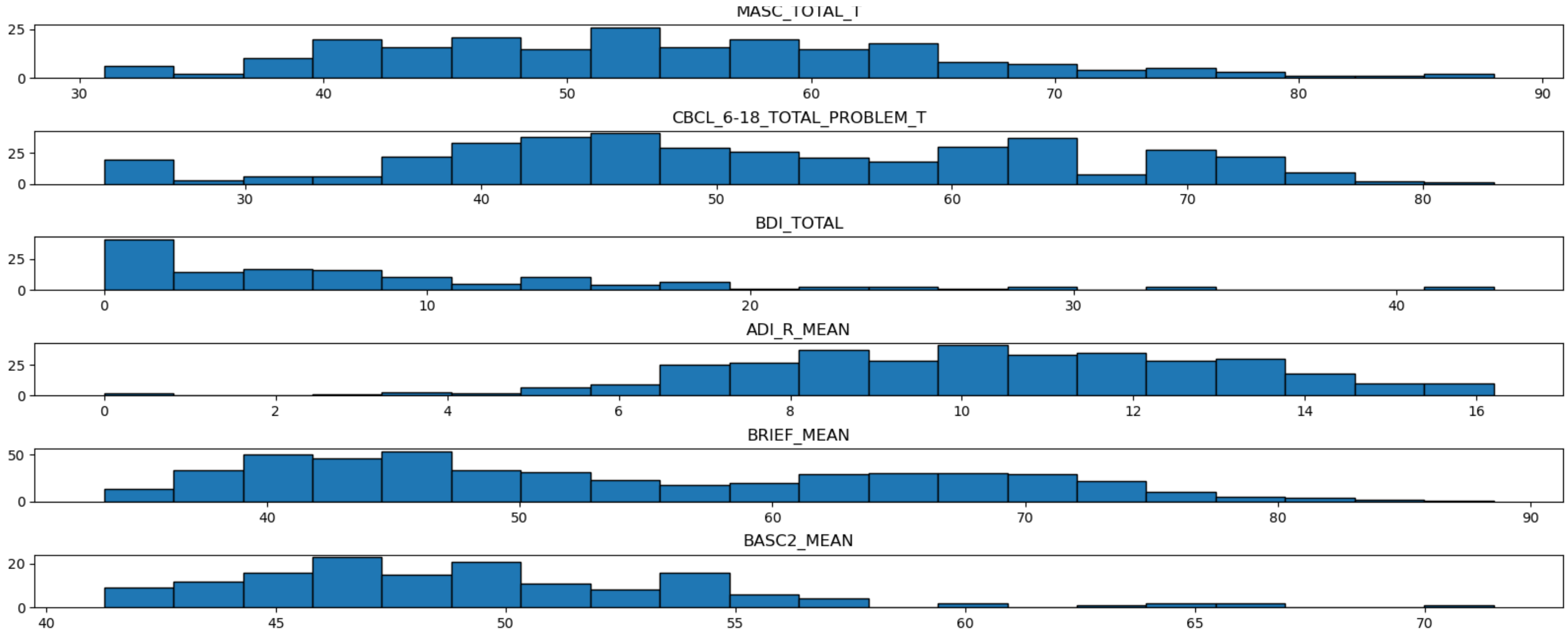
Plots Showing the distribution of values in each numeric column

In [233…

```
# Select the numeric columns
numeric_columns = df_ASD.select_dtypes(include=['int64', 'float64']).columns

# Create a figure with a subplot for each numeric column
fig, axes = plt.subplots(nrows=len(numeric_columns), figsize=(20, 30))

# Set the figure's margins
fig.subplots_adjust(hspace=1, wspace=1)

# Create a histogram for each numeric column
for ax, column in zip(axes, numeric_columns):
    ax.hist(df_ASD[column], bins=20, edgecolor='k')
    ax.set_title(column)

# Show the plot
plt.show()
```




0
1.0 1.2 1.4 1.6 1.8 2.0
HANDEDNESS_CATEGORY
1000
0
1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00
HANDEDNESS_SCORES
250
0
−100 −75 −50 −25 0 25 50 75 100
ADOS_G_TOTAL
50
0
0 5 10 15 20
ADOS_2_TOTAL
50
0
0 5 10 15 20 25
ADOS_2_SEVERITY_TOTAL
50
0
2 4 6 8 10
SRS_TOTAL_RAW
100
0
0 25 50 75 100 125 150 175 200
SRS_TOTAL_T
100
0
40 60 80 100 120
SCQ_TOTAL
50
0
0 5 10 15 20 25 30 35
RBSR_6SUBSCALE_TOTAL
200
0
0 20 40 60 80 100
RBSR_5SUBSCALE_TOTAL
100
0
0 20 40 60 80

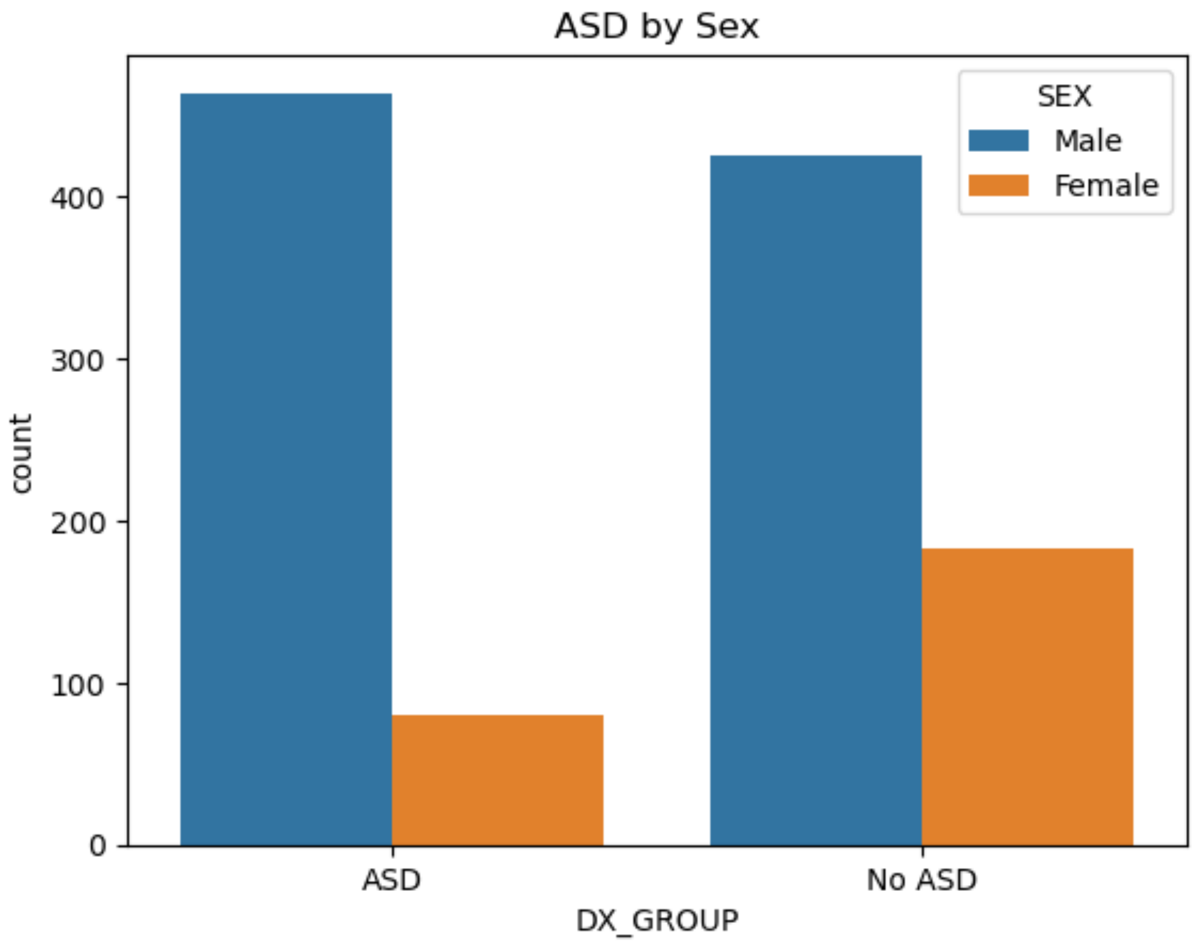


Plot Categorical Columns

I ploted each one of the categorical columns to find out their distribution along the ASD and No-ASD categories

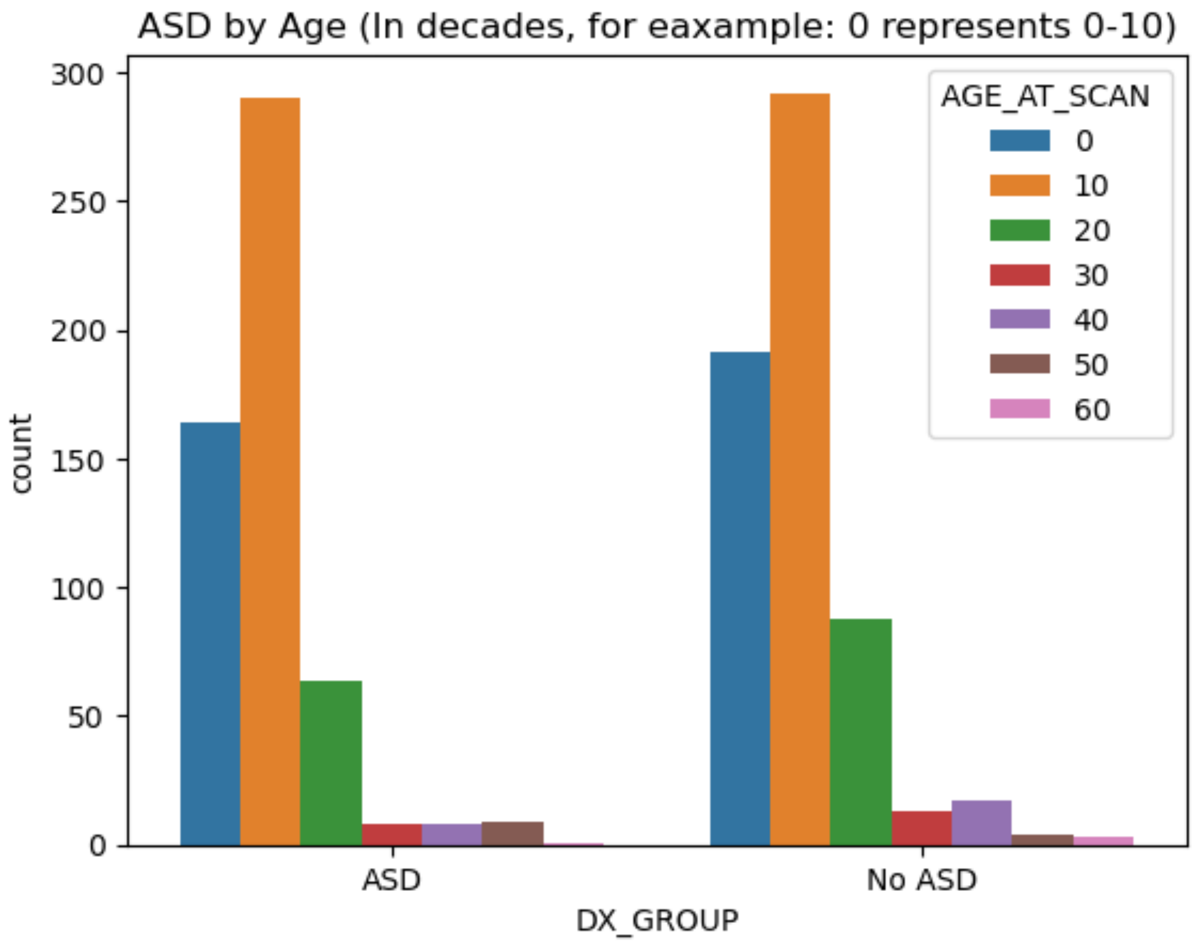
In [234…

```
# Create a bar plot for the sex column
sns.countplot(x='DX_GROUP', hue=df_ASD['SEX'].map({1:"Male", 2:"Female"}), data=df_ASD)

# Set actual diagnosis instead of DX_Group=1 or 2
plt.xticks([0,1], ["ASD", "No ASD"])

# Add a title
plt.title('ASD by Sex')

# Show the plot
plt.show()
```


ASD by Sex
SEX
Male
Female
count
0
100
200
300
400
ASD
No ASD
DX_GROUP

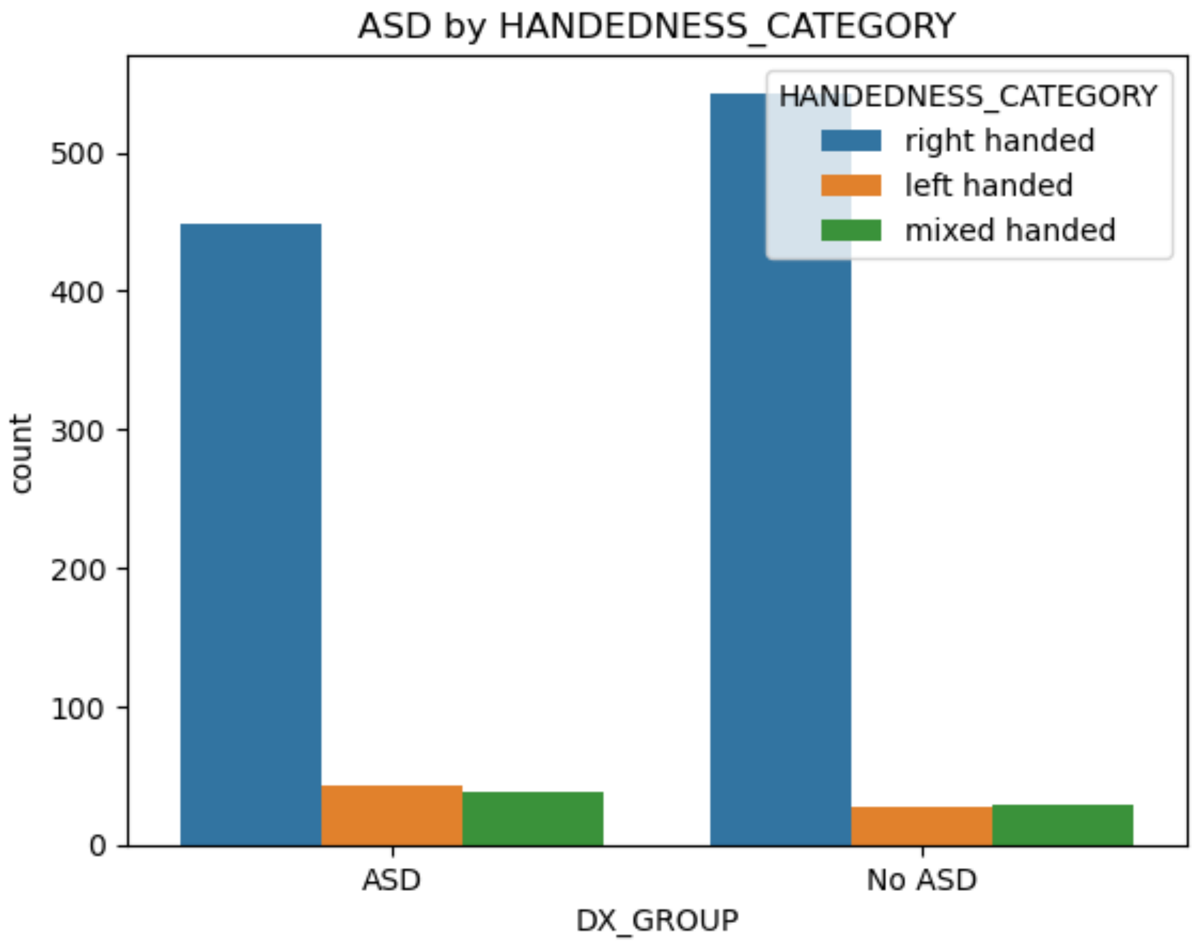
In [235...

```
# Create a bar plot for the AGE column
sns.countplot(x='DX_GROUP', hue=(((df_ASD['AGE_AT_SCAN '].astype(int))/10).astype(int))*10, data=df_ASD)

# Set actual diagnosis instead of DX_Group=1 or 2
plt.xticks([0,1], ["ASD", "No ASD"])

# Add a title
plt.title('ASD by Age (In decades, for eaxample: 0 represents 0-10)')

# Show the plot
plt.show()
```


ASD by Age (In decades, for eaxample: 0 represents 0-10)
count
0
50
100
150
200
250
300
AGE_AT_SCAN
0
10
20
30
40
50
60
ASD
No ASD
DX_GROUP

In [236…

```
# Create a bar plot for the HANDEDNESS_CATEGORY column
sns.countplot(x='DX_GROUP', hue=df_ASD['HANDEDNESS_CATEGORY'].map({1:"right handed", 2:"left handed", 3:"mixed ha

# Set actual diagnosis instead of DX_Group=1 or 2
plt.xticks([0,1], ["ASD", "No ASD"])

# Add a title
plt.title('ASD by HANDEDNESS_CATEGORY')

# Show the plot
plt.show()
```

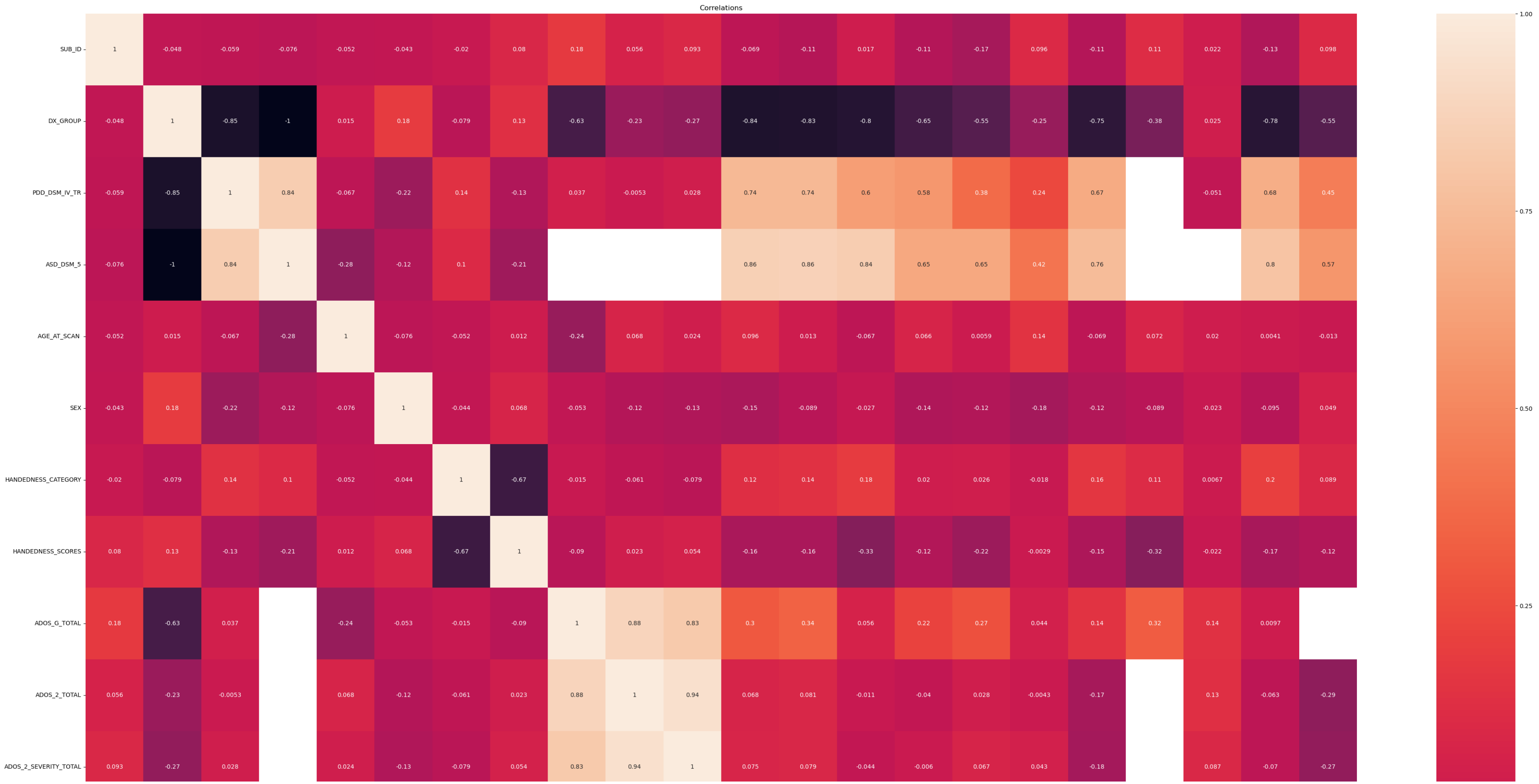


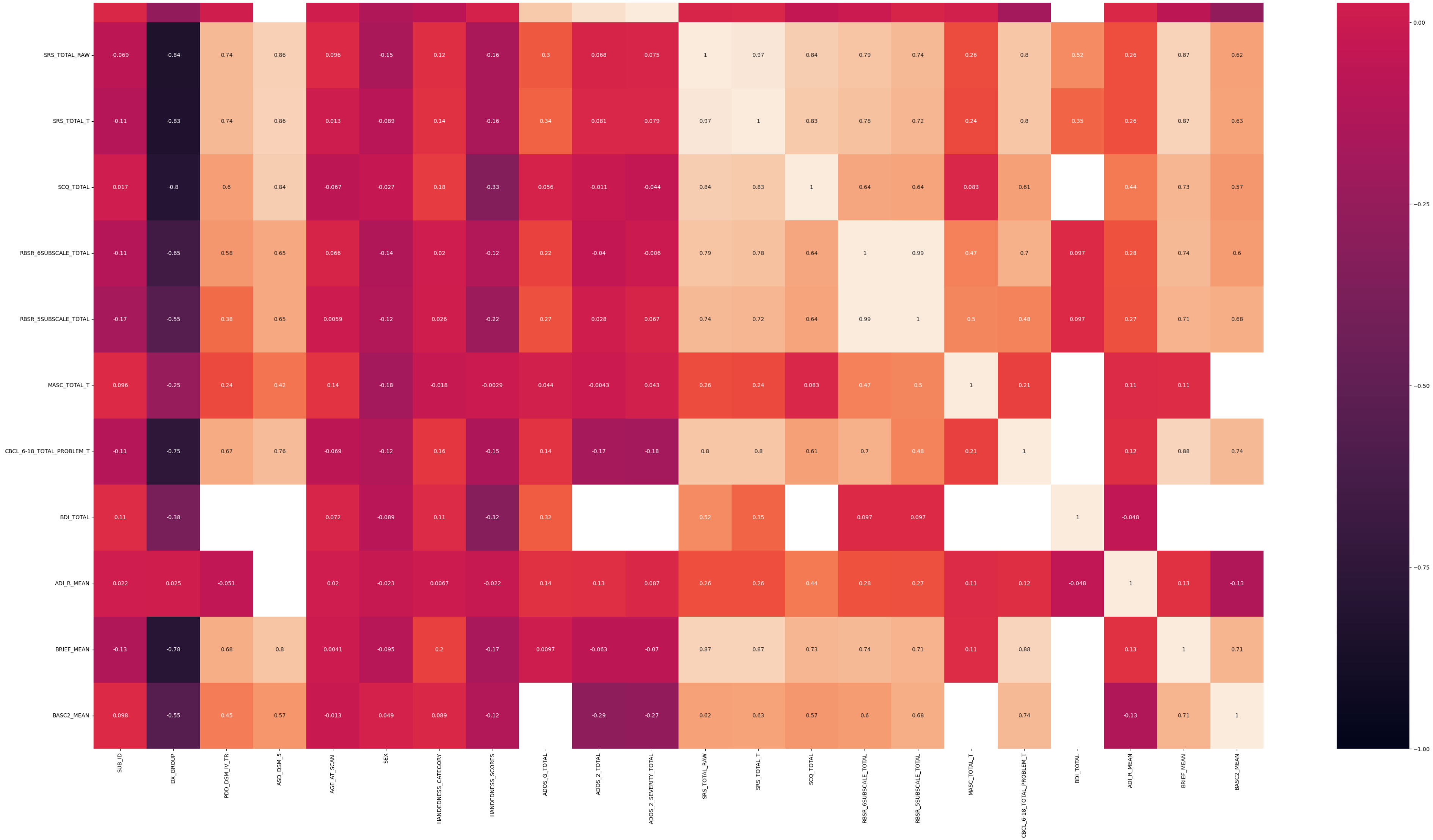
Correlations Analysis - using Heatmap

Showing the correlation of each numeric column with all other numeric columns (including our target column - DX_GROUP) We wish to keep the columns highly correlated to DX_GROUP Also, we want to discard of non-target columns which show high correlation between themselves (keep one of them and drop the other)

In [237…

```python
# Create a figure with a heatmap for the correlations
fig, ax = plt.subplots(figsize=(48, 48))
sns.heatmap(df_ASD.corr(), annot=True, ax=ax)

# Add a title
plt.title('Correlations')

# Show the plot
plt.show()
```





4th phase of Feature Selection - removing one feature from each pair of highly correlated non-target features

In [238…

```
# Remove one of highly correlated column-pair
df_ASD = df_ASD.drop(['ADOS_2_SEVERITY_TOTAL', 'HANDEDNESS_SCORES', 'SRS_TOTAL_RAW', 'RBSR_5SUBSCALE_TOTAL'],axis
df_ASD.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1152 entries, 0 to 74
Data columns (total 19 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   SUB_ID                     1152 non-null   int64  
 1   NDAR_GUID                  310 non-null    object 
 2   DX_GROUP                   1152 non-null   int64  
 3   PDD_DSM_IV_TR              633 non-null    float64
 4   ASD_DSM_5                  205 non-null    float64
 5   AGE_AT_SCAN                1152 non-null   float64
 6   SEX                        1152 non-null   int64  
 7   HANDEDNESS_CATEGORY        1129 non-null   float64
 8   ADOS_G_TOTAL               369 non-null    float64
 9   ADOS_2_TOTAL               283 non-null    float64
 10  SRS_TOTAL_T                756 non-null    float64
 11  SCQ_TOTAL                  293 non-null    float64
 12  RBSR_6SUBSCALE_TOTAL       451 non-null    float64
 13  MASC_TOTAL_T               216 non-null    float64
 14  CBCL_6-18_TOTAL_PROBLEM_T  400 non-null    float64
 15  BDI_TOTAL                  135 non-null    float64
 16  ADI_R_MEAN                 346 non-null    float64
 17  BRIEF_MEAN                 487 non-null    float64
 18  BASC2_MEAN                 149 non-null    float64
dtypes: float64(15), int64(3), object(1)
memory usage: 212.3+ KB
```

5th phase of Feature Selection - selecting the best correlated features to ASD (target=DX_GROUP)

In [239…

```
# Create a Dataframe containing the features showing the highest correlation to ASD (DX_GROUP)
df_ASD_Pred = df_ASD[['PDD_DSM_IV_TR', 'ASD_DSM_5', 'SRS_TOTAL_T', 'SCQ_TOTAL', 'RBSR_6SUBSCALE_TOTAL', 'CBCL_6-1

df_ASD_Pred.head(200)
```

Out[239]:

	PDD_DSM_IV_TR	ASD_DSM_5	SRS_TOTAL_T	SCQ_TOTAL	RBSR_6SUBSCALE_TOTAL	CBCL_6-18_TOTAL_PROBLEM_T	BRIEF_MEAN	ADOS_
0	NaN	NaN	79.0	NaN	NaN	NaN	NaN	
1	NaN	NaN	65.0	NaN	12.0	NaN	NaN	
2	NaN	NaN	57.0	NaN	15.0	NaN	NaN	
3	NaN	NaN	56.0	NaN	13.0	NaN	NaN	
4	NaN	NaN	87.0	NaN	20.0	NaN	NaN	
...	...	...	...	...	...	...	...	
195	1.0	NaN	68.0	NaN	NaN	65.0	61.090909	
196	1.0	NaN	67.0	NaN	NaN	57.0	59.272727	
197	1.0	NaN	83.0	NaN	NaN	57.0	56.090909	
198	1.0	NaN	65.0	NaN	NaN	63.0	53.818182	
199	3.0	NaN	66.0	NaN	NaN	60.0	50.636364	

200 rows × 8 columns

Preprocessing of the selected columns

In [240…

```
# We saw in the histograms that the RBSR, SCQ and SRS fields were skewed. I handeled it -

df_ASD_Pred['SRS_TOTAL_T'] = np.log1p(df_ASD_Pred['SRS_TOTAL_T'])
df_ASD_Pred['SCQ_TOTAL'] = np.log1p(df_ASD_Pred['SCQ_TOTAL'])
df_ASD_Pred['RBSR_6SUBSCALE_TOTAL'] = np.log1p(df_ASD_Pred['RBSR_6SUBSCALE_TOTAL'])

numeric_features = df_ASD_Pred.select_dtypes(include=['float', 'int']).columns
print (numeric_features)

# Standardize numeric values and filling NaN values (using KNN imputer as there are many missing values)
numeric_transformer = Pipeline(steps=[
    ('imputer', KNNImputer(n_neighbors=2)),
    ('scaler', StandardScaler())])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features)])

df_ASD_Pred.head(200)
```

```
Index(['PDD_DSM_IV_TR', 'ASD_DSM_5', 'SRS_TOTAL_T', 'SCQ_TOTAL',
       'RBSR_6SUBSCALE_TOTAL', 'CBCL_6-18_TOTAL_PROBLEM_T', 'BRIEF_MEAN',
       'ADOS_G_TOTAL'],
      dtype='object')
```

Out[240]:

	PDD_DSM_IV_TR	ASD_DSM_5	SRS_TOTAL_T	SCQ_TOTAL	RBSR_6SUBSCALE_TOTAL	CBCL_6-18_TOTAL_PROBLEM_T	BRIEF_MEAN	ADOS_
0	NaN	NaN	4.382027	NaN	NaN	NaN	NaN	
1	NaN	NaN	4.189655	NaN	2.564949	NaN	NaN	
2	NaN	NaN	4.060443	NaN	2.772589	NaN	NaN	
3	NaN	NaN	4.043051	NaN	2.639057	NaN	NaN	
4	NaN	NaN	4.477337	NaN	3.044522	NaN	NaN	
...	...	...	...	...	...	...	...	
195	1.0	NaN	4.234107	NaN	NaN	65.0	61.090909	
196	1.0	NaN	4.219508	NaN	NaN	57.0	59.272727	
197	1.0	NaN	4.430817	NaN	NaN	57.0	56.090909	
198	1.0	NaN	4.189655	NaN	NaN	63.0	53.818182	
199	3.0	NaN	4.204693	NaN	NaN	60.0	50.636364	

200 rows × 8 columns

Classification

Split into train-test

Keep a random 25% of the samples (rows) for test

In [241…

```
# Split data into training and testing sets
X = df_ASD_Pred
y = df_ASD['DX_GROUP']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=2)
```

In [242…

```
X_train = X_train[['PDD_DSM_IV_TR', 'ASD_DSM_5', 'SRS_TOTAL_T', 'SCQ_TOTAL', 'RBSR_6SUBSCALE_TOTAL', 'CBCL_6-18_T
X_train.head(200)
```

Out[242]:

	PDD_DSM_IV_TR	ASD_DSM_5	SRS_TOTAL_T	SCQ_TOTAL	RBSR_6SUBSCALE_TOTAL	CBCL_6-18_TOTAL_PROBLEM_T	BRIEF_MEAN	ADOS
962	2.0	NaN	4.262680	2.079442	1.945910	51.0	NaN	
658	0.0	NaN	3.737670	0.693147	NaN	52.0	67.000000	
556	0.0	NaN	3.806662	NaN	NaN	NaN	41.090909	
989	NaN	1.0	4.248495	3.555348	2.890372	NaN	NaN	
370	2.0	NaN	4.317488	NaN	2.944439	68.0	66.363636	
...	...	...	...	...	...	...	...	
177	2.0	NaN	4.488636	NaN	NaN	56.0	62.545455	
275	2.0	1.0	NaN	NaN	3.465736	NaN	NaN	
529	0.0	NaN	3.761200	NaN	0.000000	49.0	43.818182	
412	0.0	NaN	3.737670	NaN	0.693147	43.0	41.818182	
13	NaN	NaN	4.418841	NaN	3.737670	NaN	NaN	

200 rows × 8 columns

Fit-Test-Evaluate

I used 5 different classification models: Logistic Regression, KNN, Support Vector Machine, Decision Tree and Random Forest.

In [243...

```
# Create a pipeline for logistic regression
logistic_pipeline = Pipeline([
    ('column_transformer', preprocessor),
    ('classifier', LogisticRegression())
])

# Create a grid of hyperparameters for logistic regression
logistic_param_grid = {
    'classifier__C': [0.1, 1, 10],
    'classifier__max_iter': [500]
}


# Create a pipeline for KNN
KNN_pipeline = Pipeline([
    ('column_transformer', preprocessor),
    ('classifier', KNeighborsClassifier())
])


# Create a grid of hyperparameters for KNN
KNN_param_grid = {
    'classifier__n_neighbors': [8, 9, 10],
    'classifier__leaf_size': [2, 4, 6, 8, 10, 12]
}

# Create a pipeline for support vector machine
svm_pipeline = Pipeline([
    ('column_transformer', preprocessor),
    ('classifier', SVC())
])
```

```
# Create a grid of hyperparameters for support vector machine
svm_param_grid = {
    'classifier__C': [0.1, 1, 10],
    'classifier__kernel': ['linear', 'rbf']
}

# Create a pipeline for decision tree
tree_pipeline = Pipeline([
    ('column_transformer', preprocessor),
    ('classifier', DecisionTreeClassifier())
])

# Create a grid of hyperparameters for decision tree
tree_param_grid = {
    'classifier__max_depth': [3, 5, 7],
    'classifier__min_samples_leaf': [1, 2, 3]
}

# Create a pipeline for Random Forest
RF_pipeline = Pipeline([
    ('column_transformer', preprocessor),
    ('classifier', RandomForestClassifier())
])

# Create a grid of hyperparameters for random forest
RF_param_grid = {
    'classifier__max_depth': [3, 5, 7],
    'classifier__min_samples_leaf': [1, 2, 3]
}


# Create a list of pipelines and parameter grids
pipelines = [
    ('Logistic Regression', logistic_pipeline, logistic_param_grid),
    ('KNN', KNN_pipeline, KNN_param_grid),
    ('Support Vector Machine', svm_pipeline, svm_param_grid),
    ('Decision Tree', tree_pipeline, tree_param_grid),
```

```
    ('Random Forest', RF_pipeline, RF_param_grid)
]

# Loop through the pipelines and parameter grids
for name, pipeline, param_grid in pipelines:
    print(f'Fitting {name}...')

    # Create a GridSearchCV object for the pipeline
    gscv = GridSearchCV(pipeline, param_grid, cv=3, scoring='accuracy')

    # Fit the pipeline to the training data
    gscv.fit(X_train, y_train)#, classifier__sample_weight = sw_array)

    # Predict and print the best parameters and score
    y_pred = gscv.predict(X_test)

    # Evaluate the pipeline on the test data
    score = gscv.score(X_test, y_test)

    print(f'Classification report:\n {classification_report(y_test,y_pred)}')
    print(f'Best parameters: {gscv.best_params_}')
    print(f'Best score: {gscv.best_score_:.2f}')
    print(f'Test score: {score:.2f}')
    print()
```

```
Fitting Logistic Regression...
Classification report:
               precision    recall  f1-score   support

           1       0.96      0.84      0.90       141
           2       0.87      0.97      0.91       147

    accuracy                           0.91       288
   macro avg       0.91      0.90      0.91       288
weighted avg       0.91      0.91      0.91       288

Best parameters: {'classifier__C': 1, 'classifier__max_iter': 500}
Best score: 0.93
```

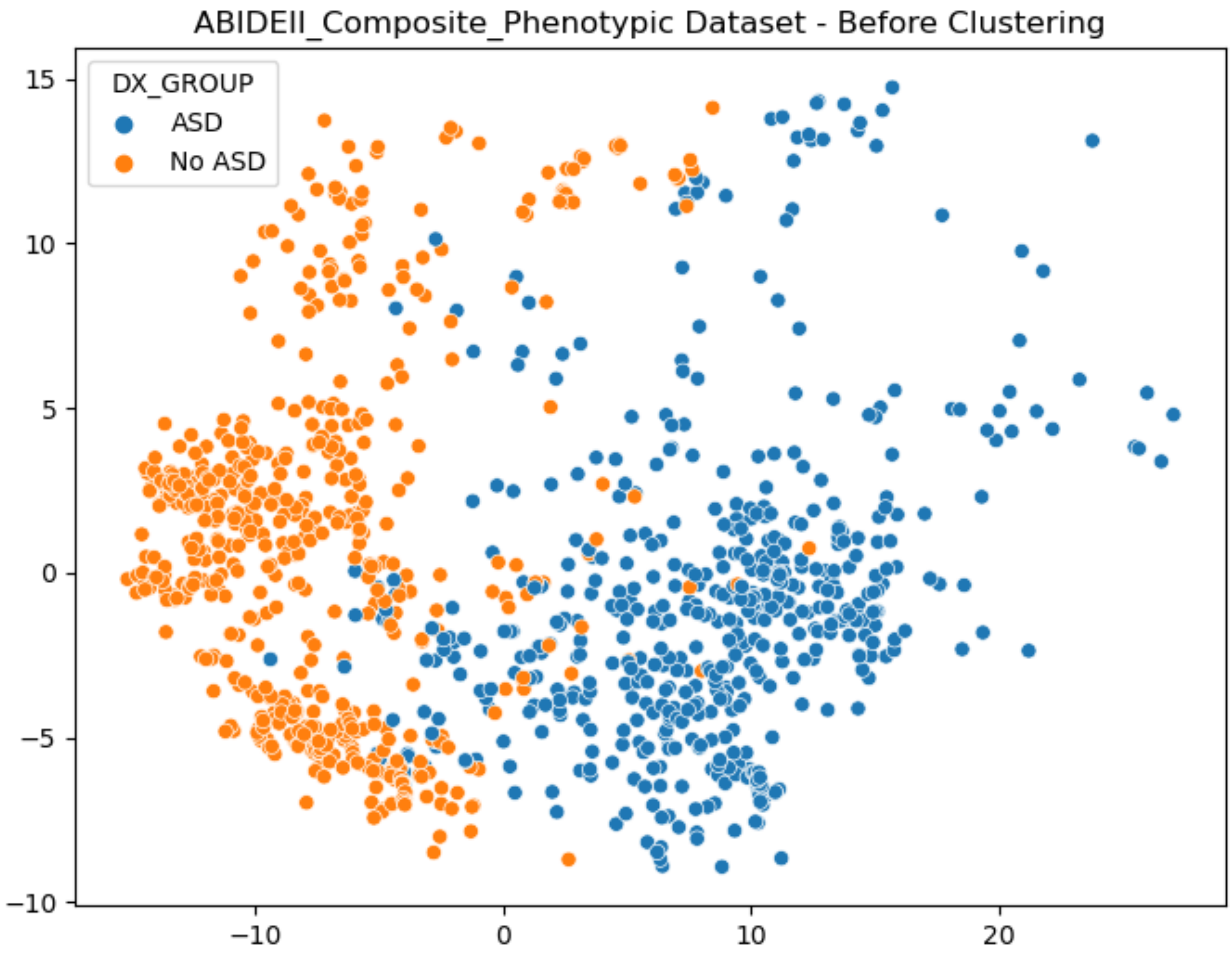
```
Test score: 0.91

Fitting KNN...
Classification report:
              precision    recall  f1-score   support

           1       0.95      0.88      0.92       141
           2       0.89      0.96      0.92       147

    accuracy                           0.92       288
   macro avg       0.92      0.92      0.92       288
weighted avg       0.92      0.92      0.92       288

Best parameters: {'classifier__leaf_size': 6, 'classifier__n_neighbors': 8}
Best score: 0.94
Test score: 0.92

Fitting Support Vector Machine...
Classification report:
              precision    recall  f1-score   support

           1       0.97      0.87      0.92       141
           2       0.89      0.97      0.93       147

    accuracy                           0.92       288
   macro avg       0.93      0.92      0.92       288
weighted avg       0.93      0.92      0.92       288

Best parameters: {'classifier__C': 10, 'classifier__kernel': 'rbf'}
Best score: 0.94
Test score: 0.92

Fitting Decision Tree...
Classification report:
              precision    recall  f1-score   support

           1       0.96      0.84      0.90       141
           2       0.87      0.97      0.91       147
```

```
    accuracy                           0.91       288
   macro avg       0.91      0.90      0.91       288
weighted avg       0.91      0.91      0.91       288

Best parameters: {'classifier__max_depth': 5, 'classifier__min_samples_leaf': 2}
Best score: 0.95
Test score: 0.91

Fitting Random Forest...
Classification report:
              precision    recall  f1-score   support

           1       0.98      0.87      0.92       141
           2       0.88      0.99      0.93       147

    accuracy                           0.93       288
   macro avg       0.93      0.93      0.93       288
weighted avg       0.93      0.93      0.93       288

Best parameters: {'classifier__max_depth': 7, 'classifier__min_samples_leaf': 1}
Best score: 0.95
Test score: 0.93
```

Now, let's try and do this faster - using PCA and Clustering

In [244…

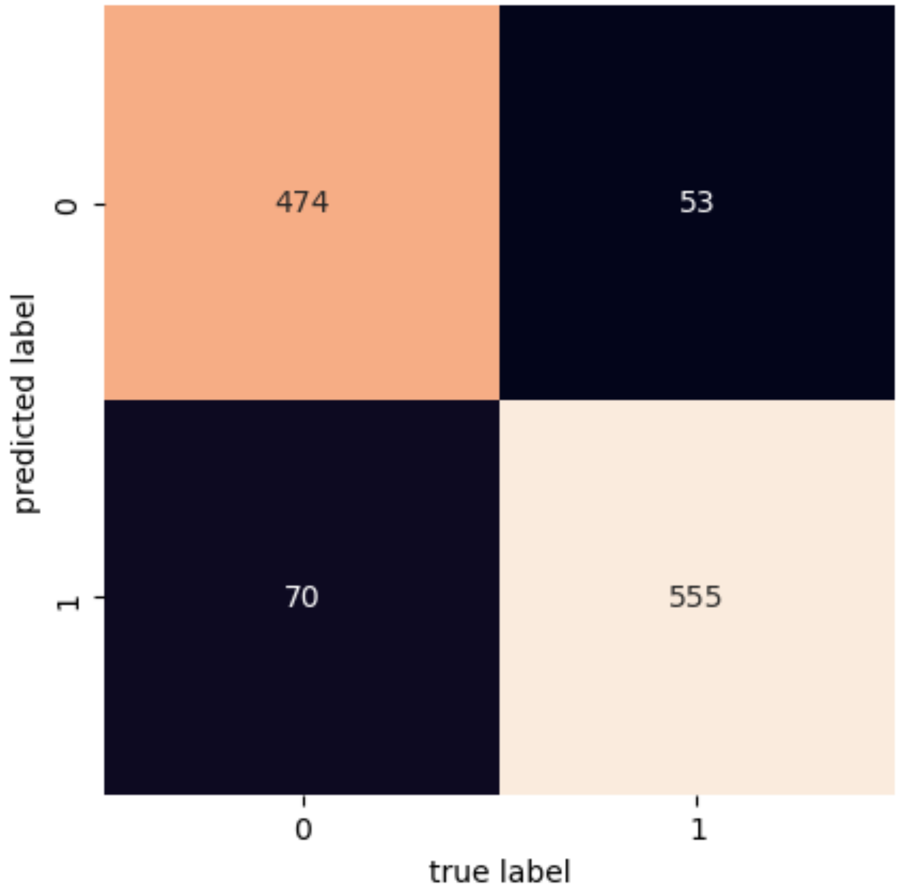
```
# Starting again from the original dataframe df (not using any of the previous data selection phases)
numeric_features = df.select_dtypes(include=['float', 'int']).columns

X = df[numeric_features.values].drop('DX_GROUP',axis=1)
y = df['DX_GROUP']

# Filling NaN values (using KNN imputer as there are many missing values)
imputer = KNNImputer(n_neighbors=2)
X_imputed = imputer.fit_transform(X)

# Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)

# Visualize the data with PCA
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(X_scaled)

# Create a figure with a scatter plot
fig, ax = plt.subplots(figsize=(8, 6))
sns.scatterplot(x=reduced_data[:, 0], y=reduced_data[:, 1], hue=y.map({1:"ASD", 2:"No ASD"}), ax=ax)

# Add a title
plt.title('ABIDEII_Composite_Phenotypic Dataset - Before Clustering')

# Show the plot
plt.show()
```



Fit-Predict and evaluate KMeans results

In [245…

```
# Define hyperparameters for kmeans: 2 clusters since we want to find ASD vs. No ASD groups
kmeans = KMeans(n_clusters=2, random_state=42)

# Generate predictions with the model using our X values (reduced by PCA)
y_pred = kmeans.fit_predict(reduced_data)

# Get the confusion matrix
y = y.map({1:0, 2:1}) # Mapping DX_GROUP into 0=ASD vs. 1=No ASD
mat = confusion_matrix(y,y_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=[0,1],
            yticklabels=[0,1])
plt.xlabel('true label')
plt.ylabel('predicted label');
```



Clusters Visualization

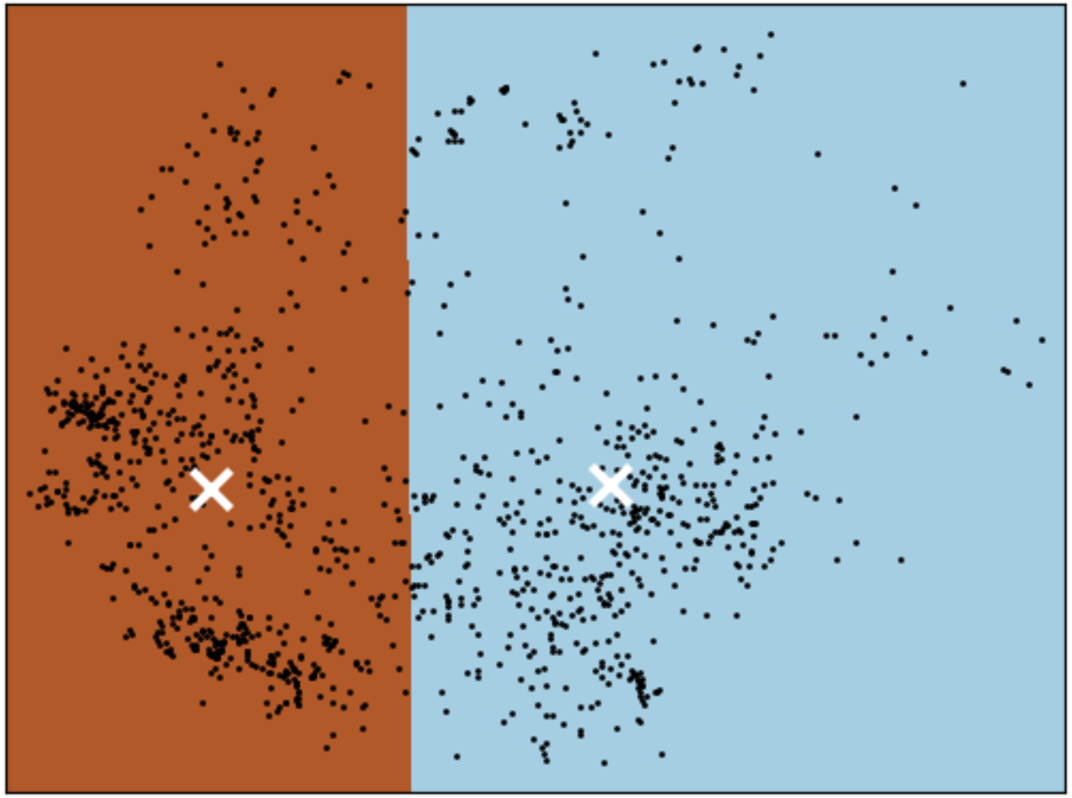
In [246…

```
# Step size of the mesh. Decrease to increase the quality of the VQ.
h = 0.02  # point in the mesh [x_min, x_max]x[y_min, y_max].

# Plot the decision boundary. For that, we will assign a color to each
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
```

```
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

# Obtain labels for each point in mesh. Use last applied kmeans model.
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(
    Z,
    interpolation="nearest",
    extent=(xx.min(), xx.max(), yy.min(), yy.max()),
    cmap=plt.cm.Paired,
    aspect="auto",
    origin="lower",
)

plt.plot(reduced_data[:, 0], reduced_data[:, 1], "k.", markersize=2)

# Plot the centroids as a white x
centroids = kmeans.cluster_centers_
plt.scatter(
    centroids[:, 0],
    centroids[:, 1],
    marker="x",
    s=169,
    linewidths=3,
    color="w",
    zorder=10,
)
plt.title(
    "K-means clustering on the ABIDEII_Composite_Phenotypic dataset (PCA-reduced data)\n"
    "Centroids are marked with white cross"
)
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
```

```
plt.yticks(())
plt.show()
```

K-means clustering on the ABIDEII_Composite_Phenotypic dataset (PCA-reduced data)
Centroids are marked with white cross



In [ ]: