🔷Project Name - OCD Patient Dataset: Demographics & Clinical Data Analysis

Project Type - Exploratory Data Analysis(EDA), Clinical Insights, Healthcare Data Mining

Contribution - Individual

Name - Aditya Singh

🧠 Introduction:-

OCD Patient Dataset – Demographics & Clinical Data Obsessive-Compulsive Disorder (OCD) is a chronic and often debilitating mental health condition marked by intrusive, unwanted thoughts (obsessions) and repetitive behaviors (compulsions). Affecting individuals across demographics, OCD can significantly impair daily functioning, social relationships, and overall quality of life. While its clinical manifestations are well-documented, understanding how demographic factors interact with clinical features remains a crucial area of study in psychiatric research and treatment optimization. The dataset at hand presents a curated collection of demographic and clinical data of patients diagnosed with OCD. This includes variables like age, gender, marital status, education level, and employment status, alongside clinical attributes such as symptom severity, comorbid conditions, treatment history, and medication response. By bridging demographic insights with clinical metrics, this dataset enables data-driven exploration of patterns that may influence diagnosis, symptom progression, and therapeutic efficacy. In recent years, machine learning and statistical modeling have gained traction in psychiatric research, allowing practitioners to detect hidden correlations, stratify patients based on risk factors, and even predict treatment outcomes. This dataset offers a rich opportunity to apply such techniques—be it classification models to categorize patients by symptom severity, or clustering methods to identify subgroups with shared clinical profiles. Moreover, the dataset supports investigation into socially-relevant questions: Does gender influence OCD symptom presentation? Are younger patients more likely to exhibit specific compulsions? How does education level relate to treatment adherence? Tackling such questions not only deepens clinical understanding but also helps tailor interventions that respect individual variability. Ultimately, this dataset serves as a valuable resource for mental health researchers, data scientists, and clinicians seeking to unravel the complex dynamics of OCD through the lens of both demographics and clinical presentation. It stands at the intersection of psychology, medicine, and data science—empowering evidence-based insights that can shape personalized care and more inclusive mental health policies.

❗ Problem Statement:-

Obsessive-Compulsive Disorder (OCD) is a complex and multifaceted psychiatric condition characterized by persistent intrusive thoughts (obsessions) and repetitive behaviors (compulsions). The clinical impact of OCD varies considerably across individuals, influenced by a diverse array of demographic and psychosocial factors. While clinical symptoms and diagnostic criteria have been extensively studied, the underlying relationships between demographic variables—such as age, gender, marital status, education, and employment—and clinical outcomes like symptom severity, treatment history, and comorbid conditions remain less explored. This project aims to conduct a comprehensive Exploratory Data Analysis (EDA) on a dataset comprising both demographic and clinical data of individuals diagnosed with OCD. Through systematic investigation, the analysis seeks to uncover hidden patterns, trends, and associations that may exist between patient backgrounds and their clinical experiences. The primary objective is to identify how demographic attributes may influence clinical features of OCD: Are certain symptoms more prevalent among specific age groups? Does gender correlate with treatment response or medication adherence? Can education or employment status predict symptom severity or recurrence risk? By leveraging descriptive statistics, correlation analyses, visualizations, and possibly dimensionality reduction techniques, this analysis will serve as a foundation for deeper modeling efforts in future studies, such as classification tasks or predictive modeling. Furthermore, the insights gained through EDA will contribute to a better understanding of OCD's heterogeneity and help identify socially-relevant disparities in diagnosis and treatment. It may assist clinicians, researchers, and policymakers in tailoring more personalized, inclusive, and effective intervention strategies based on patient profiles. In essence, this problem statement underscores the need for data-driven exploration of demographic-clinical interactions within OCD populations, setting the stage for informed, compassionate mental healthcare that bridges clinical science with lived realities.

🔍 Techniques & Tools Used to Solve the Problem:-

The analysis of OCD patient data, comprising demographic and clinical variables, involves a structured sequence of techniques aimed at extracting insights, understanding variable interactions, and preparing the foundation for predictive modeling. These techniques span data preprocessing, exploratory analysis, visualization, and machine learning methods.To effectively analyze the OCD Patient Dataset containing demographic and clinical data, a combination of data science techniques and modern Python-based tools were utilized. These methodologies enabled thorough data preparation, insightful analysis, and robust predictive modeling.

🛠️ Tools and Libraries Used:-

Pandas: For data cleaning, manipulation, filtering, and handling structured datasets.

NumPy: Numerical operations, array-based computations, and missing value handling support.

Scikit-learn: Encoding, scaling, model training, cross-validation, and evaluation metrics.

XGBoost / LightGBM: Advanced boosting algorithms for high-performance classification tasks.

Seaborn: Statistical data visualization, including heatmaps and distribution plots.

Matplotlib: Core plotting library used for creating custom charts and visual elements.

Jupyter Notebook: Interactive environment for writing code, visualizing results, and running experiments.

📌 1. Data Preprocessing Techniques:

Before any meaningful analysis can be performed, raw data must be cleaned and standardized. Preprocessing is essential to ensure analytical accuracy and model reliability. a. Handling Missing Values Definition: Many datasets contain gaps in information due to non-response or entry errors. Technique: Missing values are treated using imputation methods such as mean, median, or mode replacement. In some cases, domain-specific logic or machine learning-based imputation (e.g., KNN imputer) may be utilized. b. Encoding Categorical Variables Definition: Categorical variables like gender, marital status, or education are in textual format and must be numeric for models to interpret. Technique: Label Encoding and One-Hot Encoding are applied based on the cardinality and model choice. One-Hot Encoding is used when the number of categories is manageable and avoids ordinal bias. c. Feature Scaling Definition: Variables like age, symptom severity, and medication duration can span different ranges and affect model convergence. Technique: StandardScaler or MinMaxScaler from sklearn.preprocessing is used to normalize feature values and ensure uniformity across models.

📊 2. Exploratory Data Analysis (EDA):

EDA forms the backbone of understanding dataset structure, relationships between variables, and identifying trends or anomalies. a. Univariate Analysis Definition: Analysis of a single feature to understand its distribution and frequency. Technique: Histograms, bar plots, and box plots are used to visualize the spread of numeric and categorical variables, respectively. b. Bivariate Analysis Definition: Investigation of pairwise relationships between variables. Technique: Scatter plots, grouped bar charts, and box plots are used to examine how demographic attributes influence clinical outcomes. Correlation matrices help highlight linear associations. c. Multivariate Analysis Definition: Simultaneous analysis of multiple variables to uncover complex patterns. Technique: Heatmaps, pair plots (sns.pairplot), and clustering previews help identify subgroups or variables with interconnected behavior.

📈 3. Data Visualization Tools:

Visuals aid in intuitive understanding and presentation of findings. They translate raw data into stories. Matplotlib & Seaborn: These Python libraries generate publication-quality plots, including distribution charts, time-series graphs, and regression lines. Plotly: For interactive plots—ideal when exploring correlations or filtering subsets dynamically.

🧠 4. Statistical Testing & Correlation Analysis:

To validate findings, statistical rigor is applied. Pearson/Spearman Correlation: Measures linear/non-linear relationships between variables. Chi-Square Test: Assesses association between categorical variables (e.g., gender vs.

treatment type). ANOVA / t-test: Used to compare means across demographic groups for clinical outcomes.

🧪 5. Feature Engineering:

Sometimes raw data doesn't convey enough. Feature engineering creates new attributes to boost model performance. Interaction Features: Age × Symptom Severity, or Education × Medication Response. Binning: Age groups segmented into intervals (e.g., young adults, middle-aged, seniors). Derived Flags: E.g., binary flags for "Severe OCD," "Treatment Resistant," etc.

⚙️ 6. Modeling Techniques:

If the project scope includes predictive analysis, classification and ensemble models are deployed. a. Logistic Regression / Decision Trees Used for predicting binary outcomes such as treatment success. b. Random Forest / VotingClassifier Ensemble techniques help boost accuracy by combining results from multiple base learners. c. Feature Importance & Interpretation Identifying which features contribute most to model predictions using .feature_importances_ or SHAP values.

🎯7. Interpretation & Insight Generation:

Finally, findings are consolidated into actionable insights. Segmentation: Patient clusters based on clinical-demographic similarities. Risk Profiles: Groups more susceptible to severe symptoms. Policy Recommendations: How demographic disparities impact treatment accessibility.

Github Link-

OCD Patient Dataset: Demographics & Clinical Data

Importing Libraries

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading the dataset

In [ ]:
```
df = pd.read_csv("ocd_data.csv")
```

Exploring first few rows of the dataset

In [ ]:
```
print(df.head())
```

```
   Patient ID  Age  Gender Ethnicity Marital Status Education Level  \
0        1018   32  Female  African          Single    Some College
1        2406   69    Male  African        Divorced    Some College
2        1188   57    Male  Hispanic       Divorced  College Degree
3        6200   27  Female  Hispanic        Married  College Degree
4        5824   56  Female  Hispanic        Married     High School

  OCD Diagnosis Date  Duration of Symptoms (months) Previous Diagnoses  \
0         2016-07-15                            203                MDD
1         2017-04-28                            180                NaN
2         2018-02-02                            173                MDD
3         2014-08-25                            126               PTSD
4         2022-02-20                            168               PTSD

  Family History of OCD Obsession Type Compulsion Type  \
0                    No   Harm-related        Checking
1                   Yes   Harm-related         Washing
2                    No  Contamination        Checking
3                   Yes       Symmetry         Washing
4                   Yes       Hoarding        Ordering

   Y-BOCS Score (Obsessions)  Y-BOCS Score (Compulsions) Depression Diagnosis  \
0                         17                          10                  Yes
1                         21                          25                  Yes
2                          3                           4                   No
3                         14                          28                  Yes
4                         39                          18                   No

  Anxiety Diagnosis     Medications
0               Yes            SNRI
1               Yes            SSRI
2                No  Benzodiazepine
3               Yes            SSRI
4                No             NaN
```

Getting Summary of dataset

In [ ]:
```python
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 17 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Patient ID                     1500 non-null   int64
 1   Age                            1500 non-null   int64
 2   Gender                         1500 non-null   object
 3   Ethnicity                      1500 non-null   object
 4   Marital Status                 1500 non-null   object
 5   Education Level                1500 non-null   object
 6   OCD Diagnosis Date             1500 non-null   object
 7   Duration of Symptoms (months)  1500 non-null   int64
 8   Previous Diagnoses             1252 non-null   object
 9   Family History of OCD          1500 non-null   object
 10  Obsession Type                 1500 non-null   object
 11  Compulsion Type                1500 non-null   object
 12  Y-BOCS Score (Obsessions)      1500 non-null   int64
 13  Y-BOCS Score (Compulsions)     1500 non-null   int64
 14  Depression Diagnosis           1500 non-null   object
 15  Anxiety Diagnosis              1500 non-null   object
 16  Medications                    1114 non-null   object
dtypes: int64(5), object(12)
memory usage: 199.3+ KB
None
```

In [ ]:

```

```

Checking for missing values

In [ ]:

```
print(df.isnull().sum())
```

```
Patient ID                         0
Age                                0
Gender                             0
Ethnicity                          0
Marital Status                     0
Education Level                    0
OCD Diagnosis Date                 0
Duration of Symptoms (months)      0
Previous Diagnoses               248
Family History of OCD              0
Obsession Type                     0
Compulsion Type                    0
Y-BOCS Score (Obsessions)          0
Y-BOCS Score (Compulsions)         0
Depression Diagnosis               0
Anxiety Diagnosis                  0
Medications                      386
dtype: int64
```

Summarising statistics for Numerical

columns

In [ ]:
```
print(df.describe())
```

```
          Patient ID          Age  Duration of Symptoms (months)  \
count  1500.000000  1500.000000                    1500.000000
mean   5541.254000    46.781333                     121.745333
std    2562.389469    16.830321                      67.404610
min    1017.000000    18.000000                       6.000000
25%    3338.000000    32.000000                      64.000000
50%    5539.500000    47.000000                     121.000000
75%    7745.500000    61.000000                     178.000000
max    9995.000000    75.000000                     240.000000

       Y-BOCS Score (Obsessions)  Y-BOCS Score (Compulsions)
count                1500.000000                  1500.00000
mean                   20.048000                    19.62600
std                    11.823884                    11.78287
min                     0.000000                     0.00000
25%                    10.000000                     9.00000
50%                    20.000000                    20.00000
75%                    31.000000                    29.00000
max                    40.000000                    40.00000
```

Summarising statistics for categorical columns

In [ ]:
```
df.describe()
```

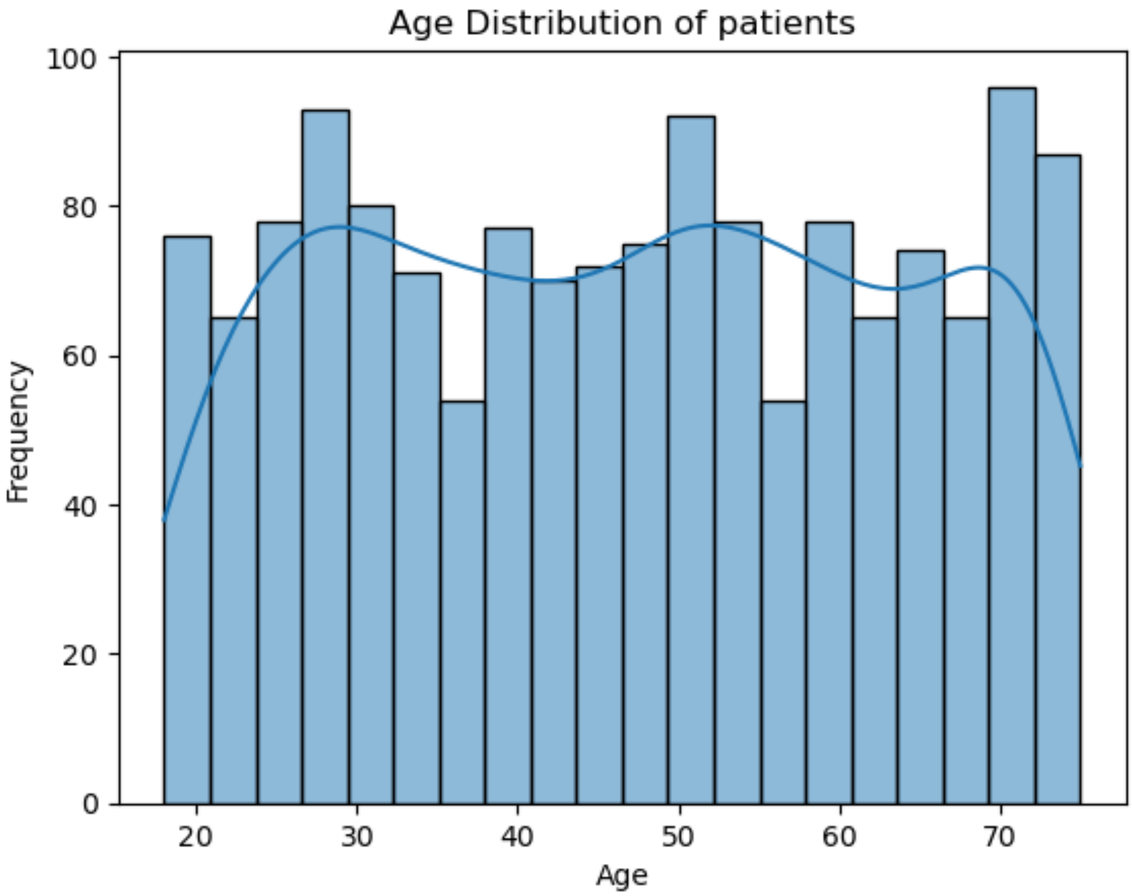
Out[ ]:

	Patient ID	Age	Duration of Symptoms (months)	Y-BOCS Score (Obsessions)	Y-BOCS Scor (Compulsions
count	1500.000000	1500.000000	1500.000000	1500.000000	1500.0000
mean	5541.254000	46.781333	121.745333	20.048000	19.6260
std	2562.389469	16.830321	67.404610	11.823884	11.7828
min	1017.000000	18.000000	6.000000	0.000000	0.0000
25%	3338.000000	32.000000	64.000000	10.000000	9.0000
50%	5539.500000	47.000000	121.000000	20.000000	20.0000
75%	7745.500000	61.000000	178.000000	31.000000	29.0000
max	9995.000000	75.000000	240.000000	40.000000	40.0000

Age Distribution

In [ ]:
```
sns.histplot(df["Age"], bins=20, kde=True)
```

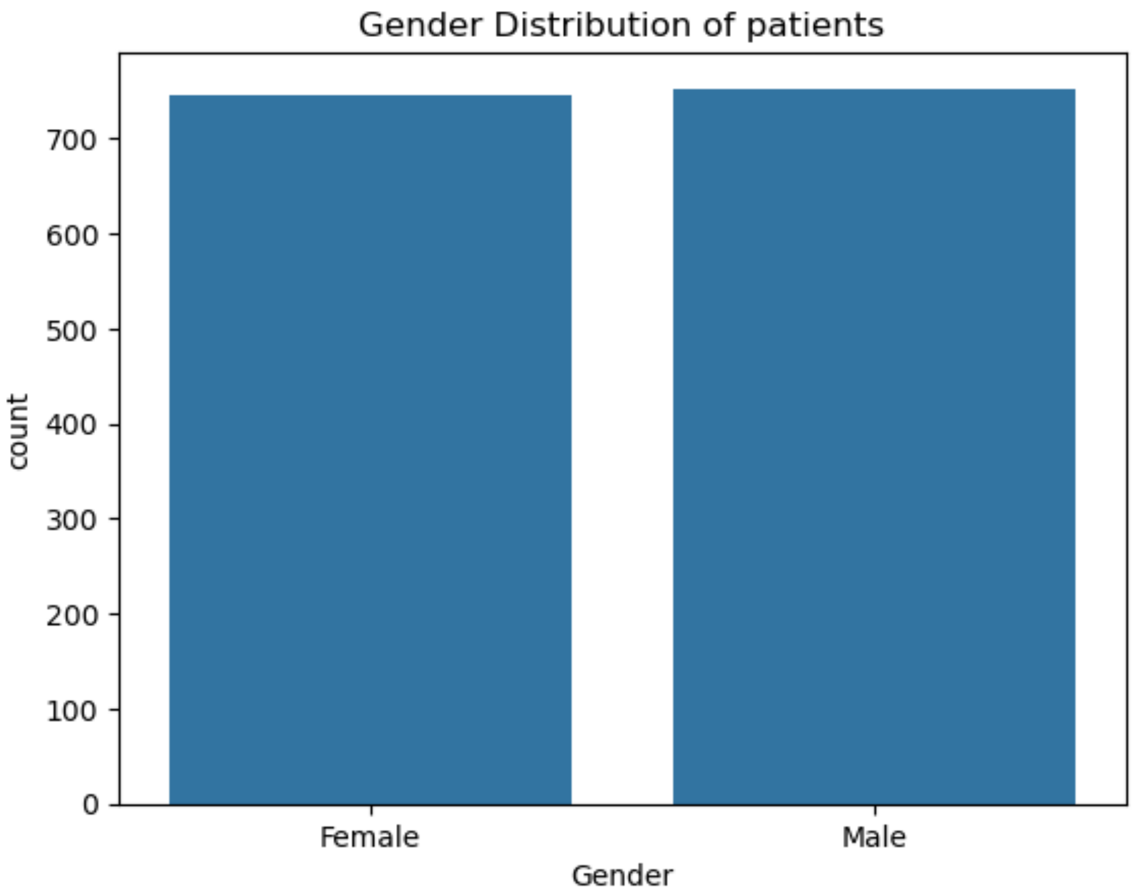
```
plt.title("Age Distribution of patients")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```



Gender Distribution

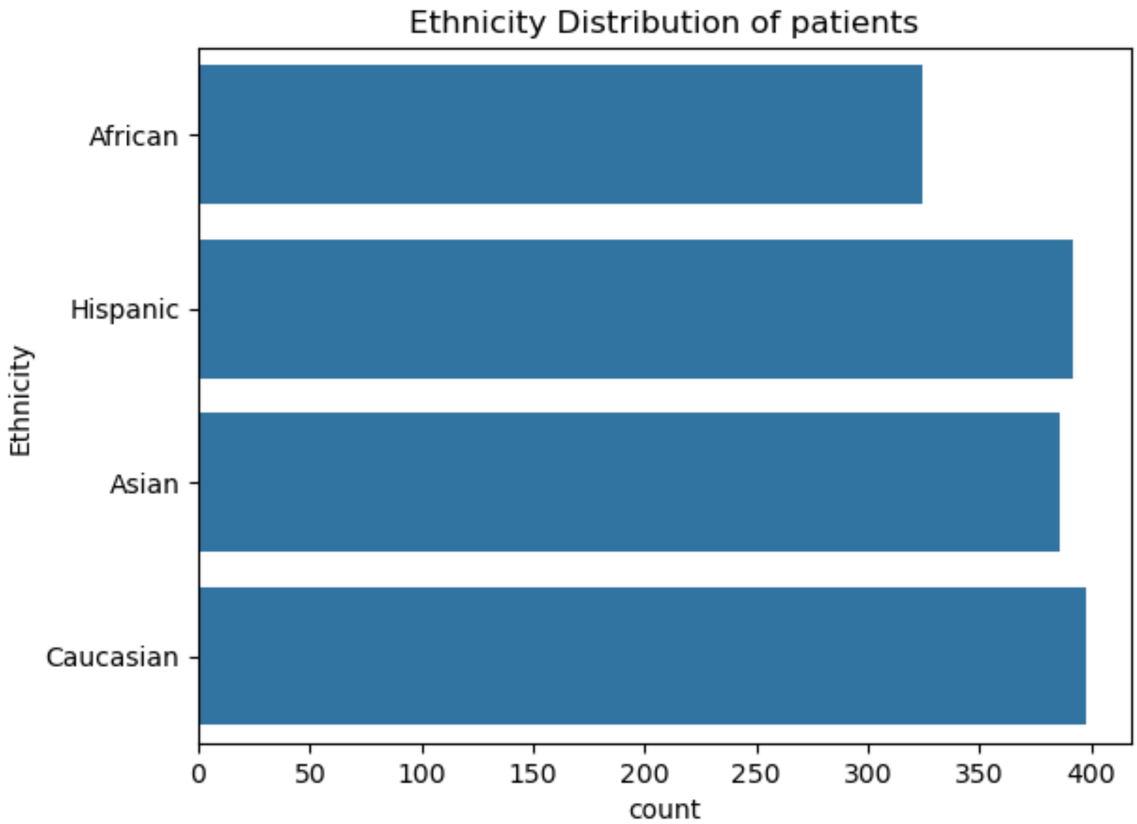
In [ ]:

```
sns.countplot(x="Gender", data=df )
plt.title("Gender Distribution of patients")
plt.xlabel("Gender")
plt.ylabel("count")
plt.show()
```



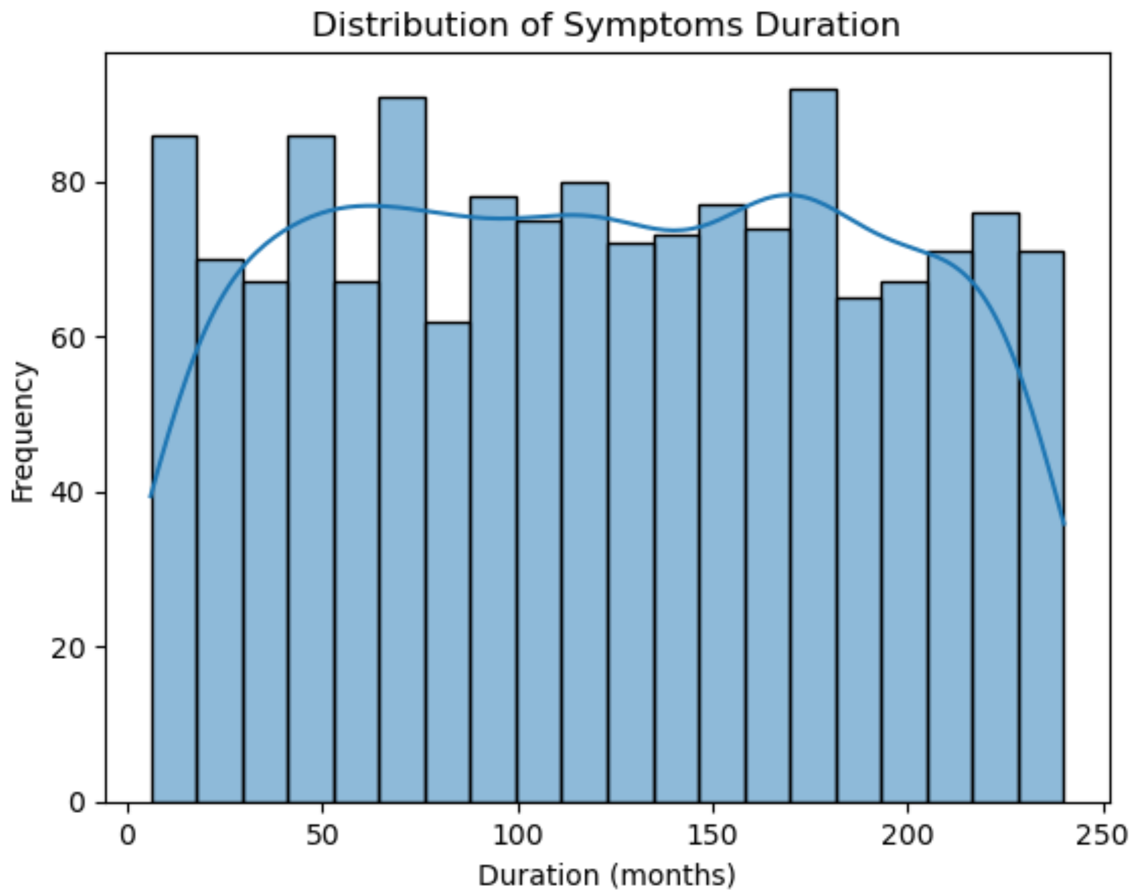
Ethnicity Distribution

In [ ]:
```
sns.countplot(y="Ethnicity", data=df )
plt.title("Ethnicity Distribution of patients")
plt.xlabel("count")
plt.ylabel("Ethnicity")
plt.show()
```



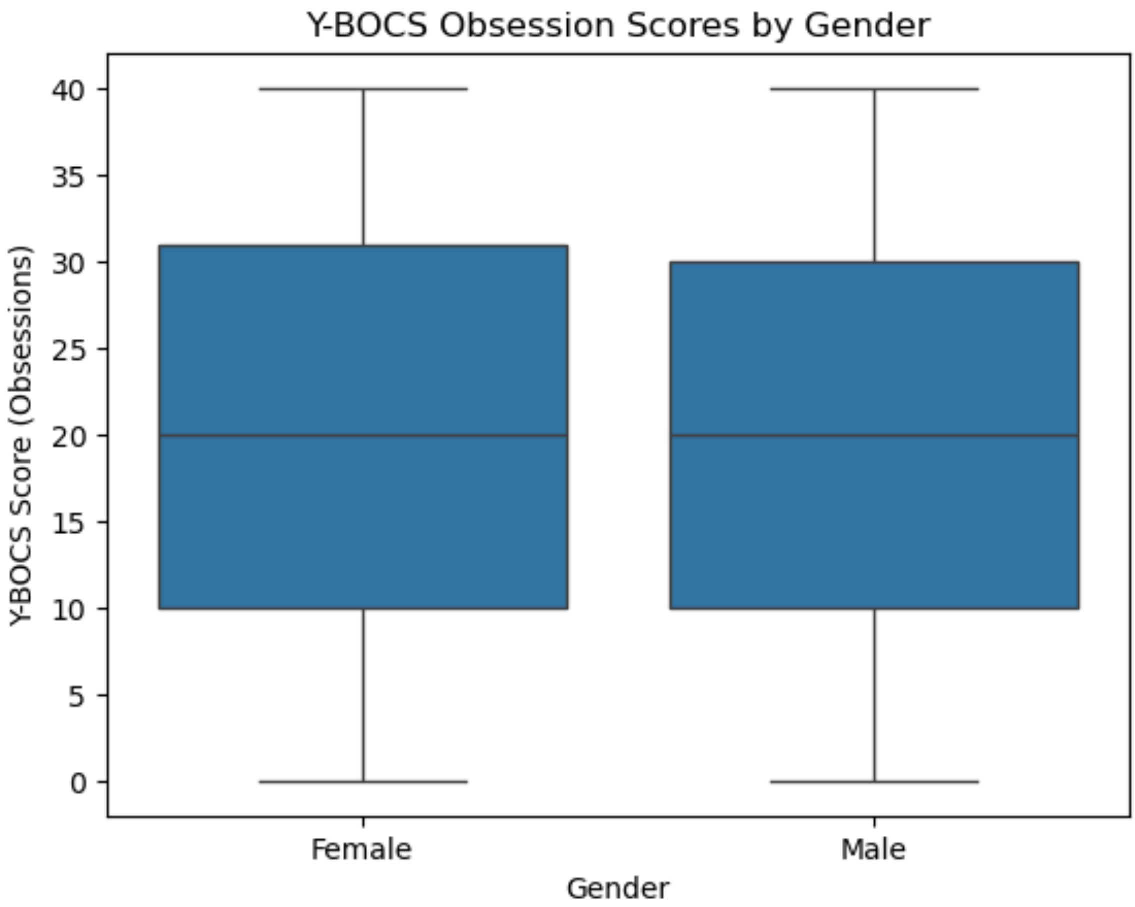
Distribution of sympton duration

In [ ]:
```
sns.histplot(df["Duration of Symptoms (months)"], bins=20, kde=True
plt.title("Distribution of Symptoms Duration")
plt.xlabel("Duration (months)")
plt.ylabel("Frequency")
plt.show()
```



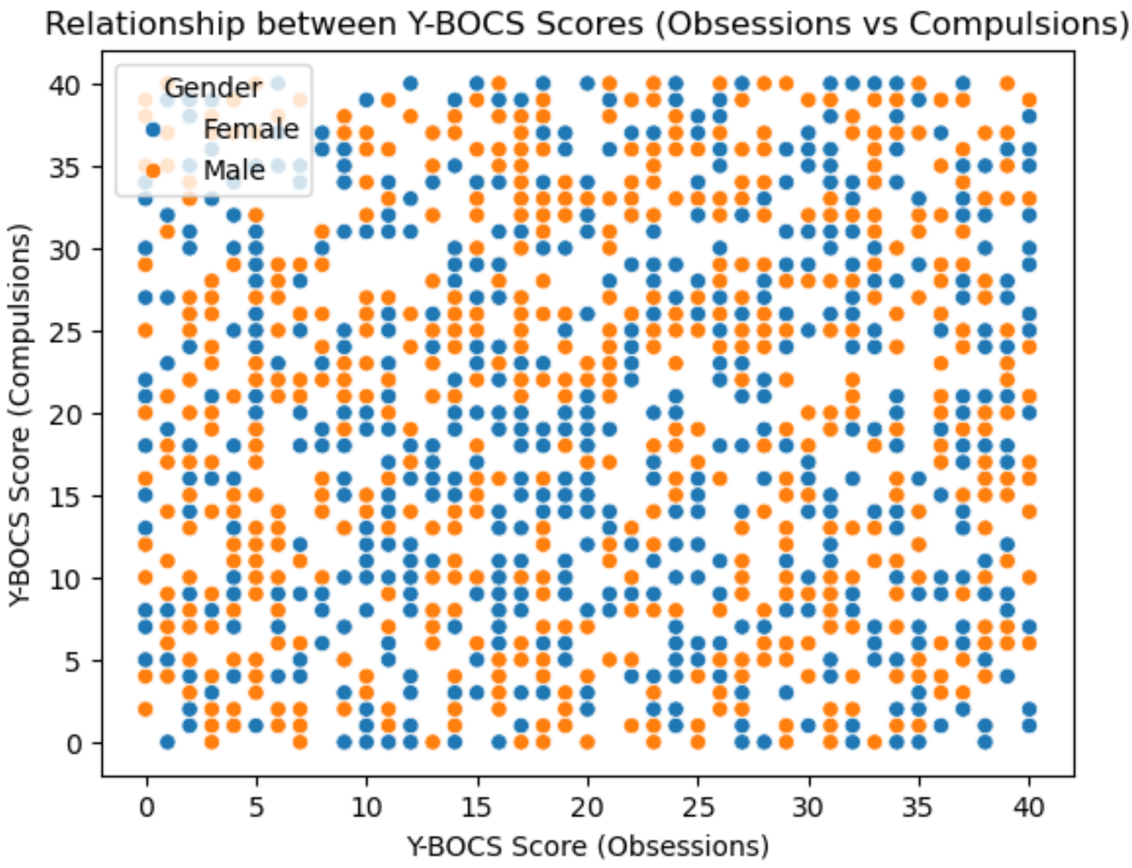
Boxplot of Y-Bocs Scores by Gender

In [ ]:
```
sns.boxplot(x='Gender', y='Y-BOCS Score (Obsessions)', data=df)
plt.title('Y-BOCS Obsession Scores by Gender')
plt.xlabel('Gender')
plt.ylabel('Y-BOCS Score (Obsessions)')
plt.show()
```



Relationship between Obsession and Compulsion Y-BOCS Scores

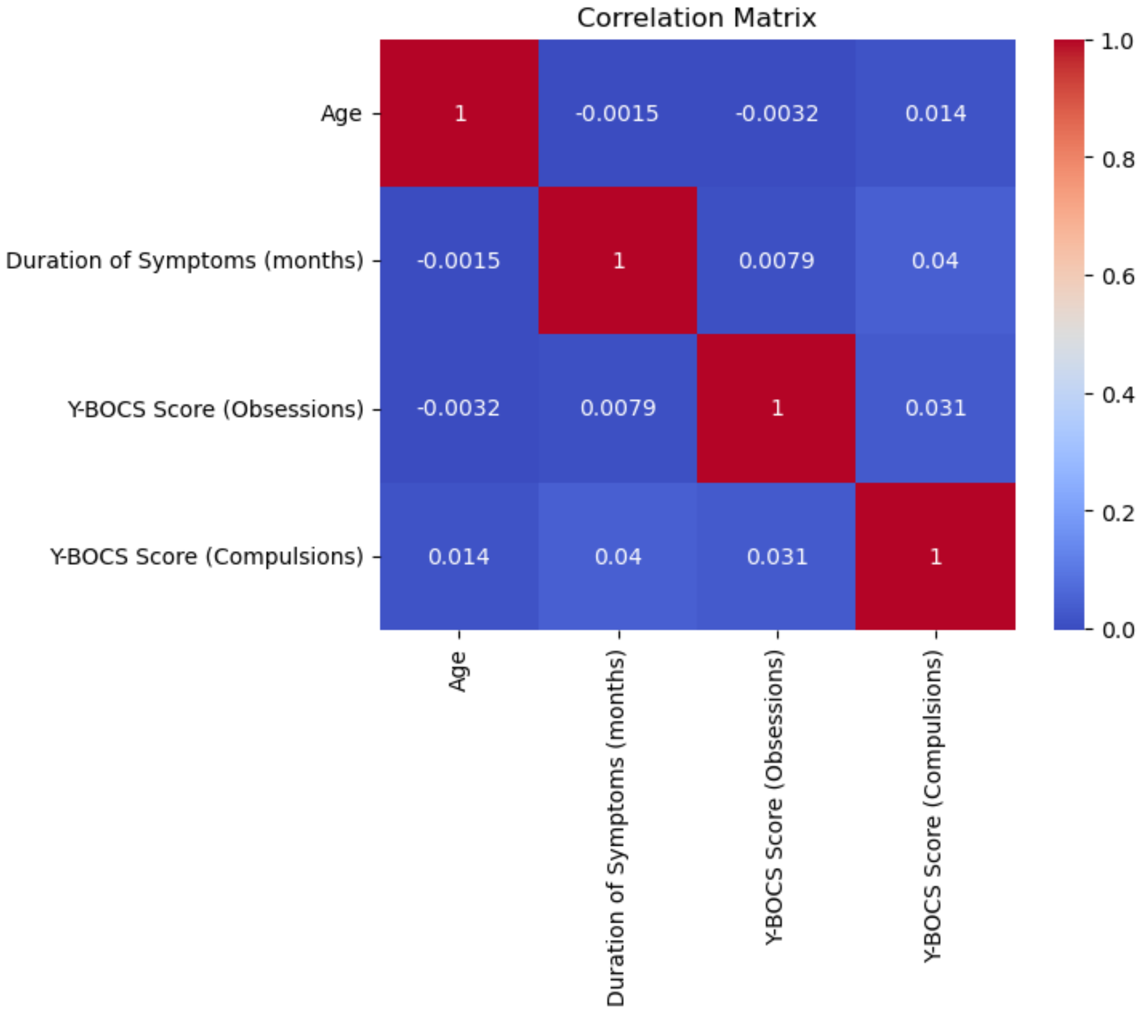
In [ ]:
```
sns.scatterplot(
    x='Y-BOCS Score (Obsessions)',
    y='Y-BOCS Score (Compulsions)',
    hue='Gender',
    data=df
)

plt.title('Relationship between Y-BOCS Scores (Obsessions vs Compul
plt.xlabel('Y-BOCS Score (Obsessions)')
plt.ylabel('Y-BOCS Score (Compulsions)')
plt.show()
```



Correlation matrix

In [ ]:

```
corr_matrix = df[['Age' , 'Duration of Symptoms (months)','Y-BOCS S
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.title('Correlation Matrix')
plt.show()
```



Key Insights and Reporting

Step 1: Loading libraries

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
```

Step 2: Loading our dataset

In [ ]:
```
df = pd.read_csv("ocd_data.csv")  # Change filename if needed
df.head()
```

Out[ ]:

	Patient ID	Age	Gender	Ethnicity	Marital Status	Education Level	OCD Diagnosis Date	Durati Symptor (montl
0	1018	32	Female	African	Single	Some College	2016-07-15	2
1	2406	69	Male	African	Divorced	Some College	2017-04-28	1
2	1188	57	Male	Hispanic	Divorced	College Degree	2018-02-02	1
3	6200	27	Female	Hispanic	Married	College Degree	2014-08-25	1
4	5824	56	Female	Hispanic	Married	High School	2022-02-20	1

Step 3: Checking for nulls

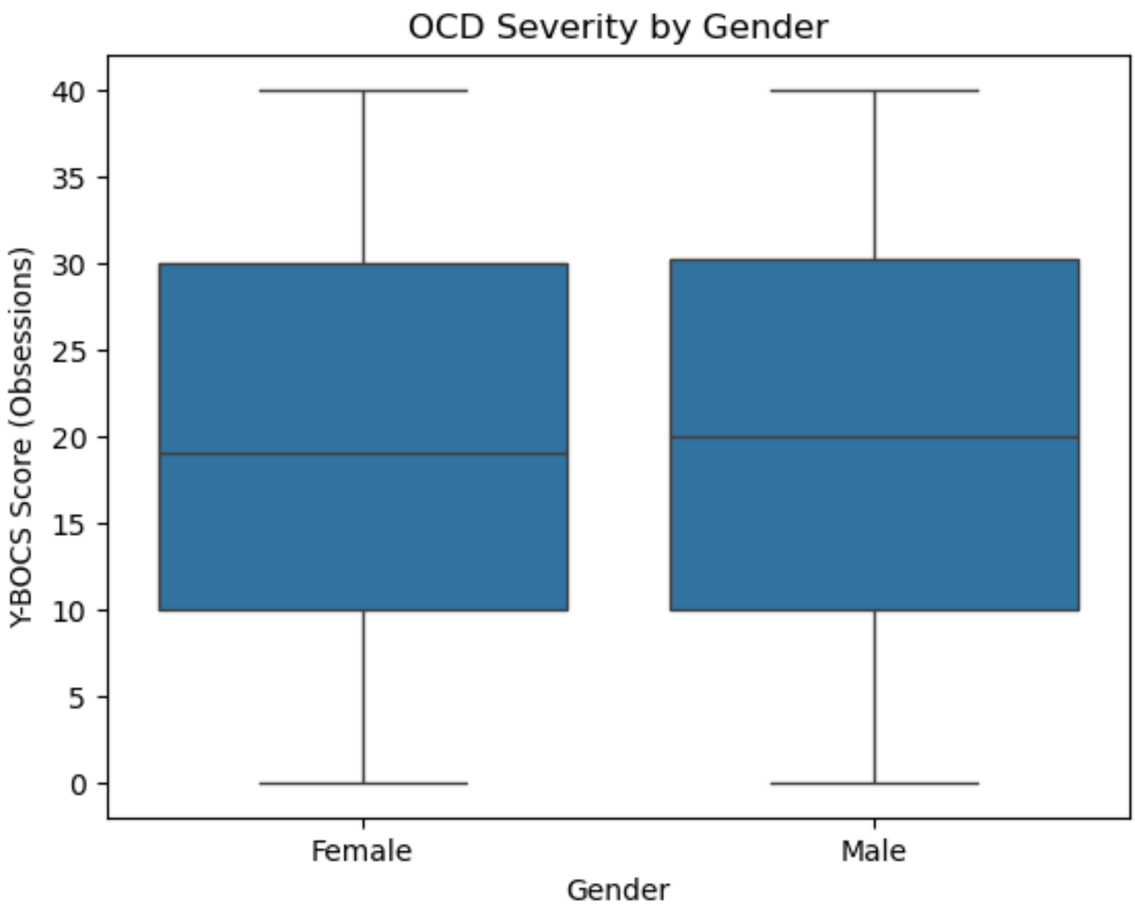
In [ ]:

```
df.info()
df.dropna(inplace=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Patient ID                      1500 non-null   int64
 1   Age                             1500 non-null   int64
 2   Gender                          1500 non-null   object
 3   Ethnicity                       1500 non-null   object
 4   Marital Status                  1500 non-null   object
 5   Education Level                 1500 non-null   object
 6   OCD Diagnosis Date              1500 non-null   object
 7   Duration of Symptoms (months)   1500 non-null   int64
 8   Previous Diagnoses              1252 non-null   object
 9   Family History of OCD           1500 non-null   object
 10  Obsession Type                  1500 non-null   object
 11  Compulsion Type                 1500 non-null   object
 12  Y-BOCS Score (Obsessions)       1500 non-null   int64
 13  Y-BOCS Score (Compulsions)      1500 non-null   int64
 14  Depression Diagnosis            1500 non-null   object
 15  Anxiety Diagnosis               1500 non-null   object
 16  Medications                     1114 non-null   object
dtypes: int64(5), object(12)
memory usage: 199.3+ KB
```

Step 4: Severity differences based on Gender and age

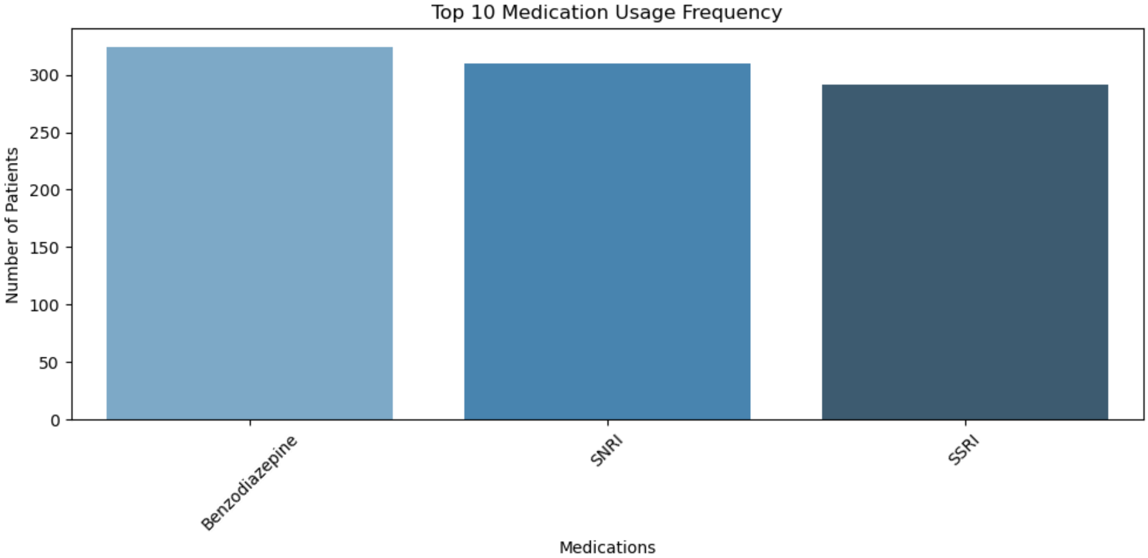
In [ ]:
```
sns.boxplot(x="Gender", y="Y-BOCS Score (Obsessions)", data=df)
plt.title("OCD Severity by Gender")
plt.show()
```



Step 5 : Medication Frequency

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Just in case of NaNs
df["Medications"] = df["Medications"].fillna("Unknown")

# Count medication frequency
med_counts = df["Medications"].value_counts().head(10)

# Plot
plt.figure(figsize=(10, 5))
sns.barplot(x=med_counts.index, y=med_counts.values, palette="Blues
plt.title("Top 10 Medication Usage Frequency")
plt.xlabel("Medications")
plt.ylabel("Number of Patients")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
/var/folders/04/wzdclyk12vggjmf0dsjkl8sr0000gn/T/ipykernel_51345/566
224079.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(x=med_counts.index, y=med_counts.values, palette="Blue
s_d")
```



In [ ]:

```

```

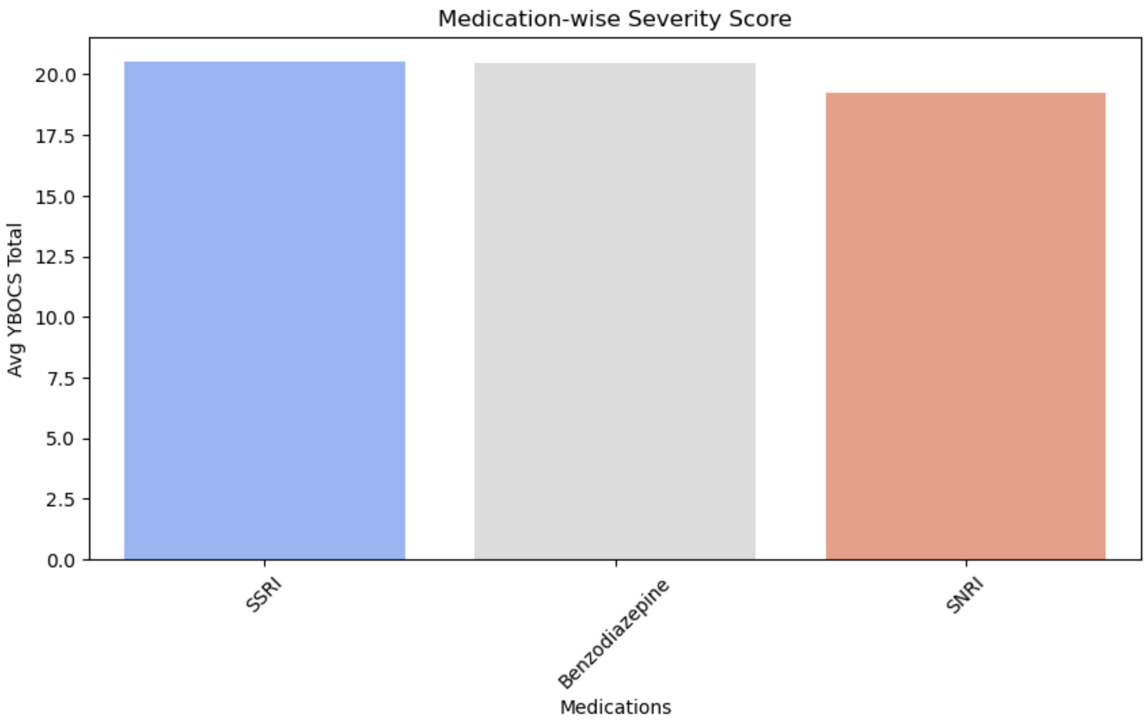
Importing Libraries

In [ ]:

```
med_avg = df.groupby("Medications")["Y-BOCS Score (Obsessions)"].me
print("Avg YBOCS by Medication:\n", med_avg)

plt.figure(figsize=(10, 5))
sns.barplot(x=med_avg.index, y=med_avg.values, palette="coolwarm")
plt.title("Medication-wise Severity Score")
plt.xticks(rotation=45)
plt.ylabel("Avg YBOCS Total")
plt.show()
```

```
Avg YBOCS by Medication:
 Medications
SSRI              20.508591
Benzodiazepine    20.456790
SNRI              19.235484
Name: Y-BOCS Score (Obsessions), dtype: float64
```

```
/var/folders/04/wzdclyk12vggjmf0dsjkl8sr0000gn/T/ipykernel_51345/122
0731826.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(x=med_avg.index, y=med_avg.values, palette="coolwarm")
```



In [49]:
```
from scipy import stats
import pandas as pd
import numpy as np
import base64,os,random,gc
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import matplotlib.pyplot as plotter
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import optuna
import xgboost as xgb
from xgboost import XGBClassifier
import catboost
from catboost import CatBoostClassifier
import lightgbm as lgbm
from lightgbm import LGBMClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.base import BaseEstimator, TransformerMixin, Classifie
from sklearn.model_selection import KFold
from scipy import stats
from scipy.stats import norm, skew
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.feature_selection import SelectFromModel
from sklearn import datasets
optuna.logging.set_verbosity(optuna.logging.WARNING)
from lightgbm import *
pd.set_option("display.max_columns" , None)
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import eli5
```

```
from eli5.sklearn import PermutationImportance
import warnings
warnings.filterwarnings('ignore')
```

Reading Dataset

In [ ]:
```
train =pd.read_csv("ocd_data.csv")
display(train.head())
```

	Patient ID	Age	Gender	Ethnicity	Marital Status	Education Level	OCD Diagnosis Date	Duration of Symptoms (months)
0	1018	32	Female	African	Single	Some College	2016-07-15	203
1	2406	69	Male	African	Divorced	Some College	2017-04-28	180
2	1188	57	Male	Hispanic	Divorced	College Degree	2018-02-02	173
3	6200	27	Female	Hispanic	Married	College Degree	2014-08-25	126
4	5824	56	Female	Hispanic	Married	High School	2022-02-20	168

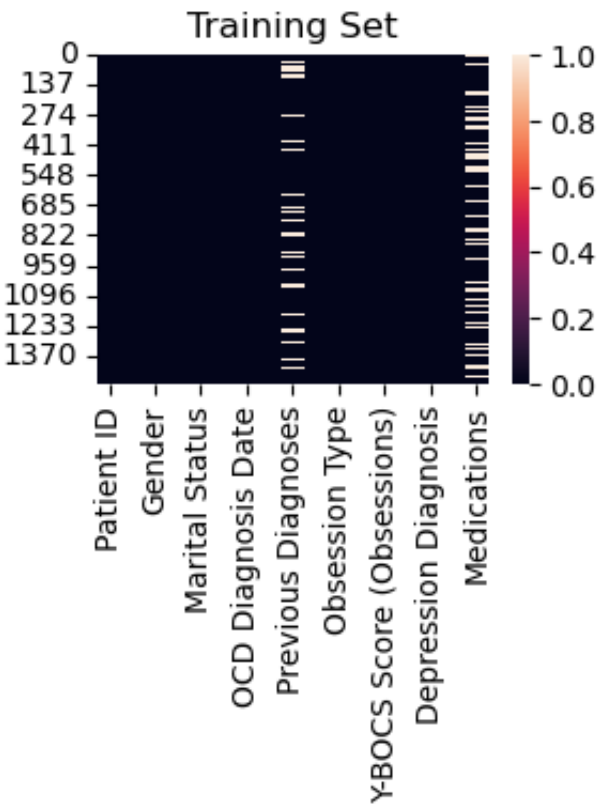
#EDA

In [ ]:
```
print('train')
display(train.isnull().sum())
plt.figure(figsize = (10, 2))
plt.subplot(1, 3, 1)
plt.title("Training Set")
sns.heatmap(train.isnull())
```

```
train
```

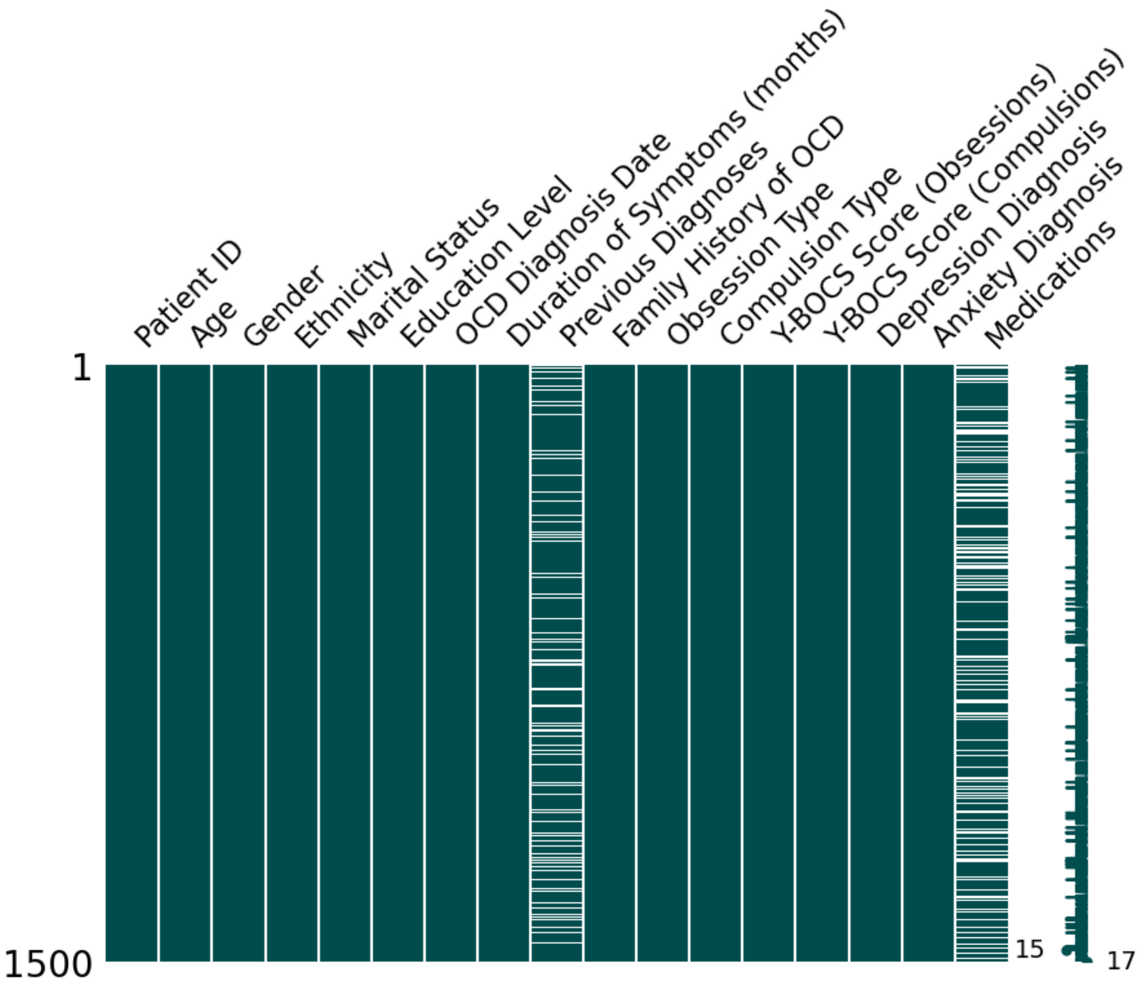
```
Patient ID                         0
Age                                0
Gender                             0
Ethnicity                          0
Marital Status                     0
Education Level                    0
OCD Diagnosis Date                 0
Duration of Symptoms (months)      0
Previous Diagnoses               248
Family History of OCD              0
Obsession Type                     0
Compulsion Type                    0
Y-BOCS Score (Obsessions)          0
Y-BOCS Score (Compulsions)         0
Depression Diagnosis               0
Anxiety Diagnosis                  0
Medications                      386
dtype: int64
```

Out[ ]: <Axes: title={'center': 'Training Set'}>



In [57]:
```
msno.matrix(df=train, figsize=(10,6), color=(0,.3,.3))
```

Out[57]: <Axes: >

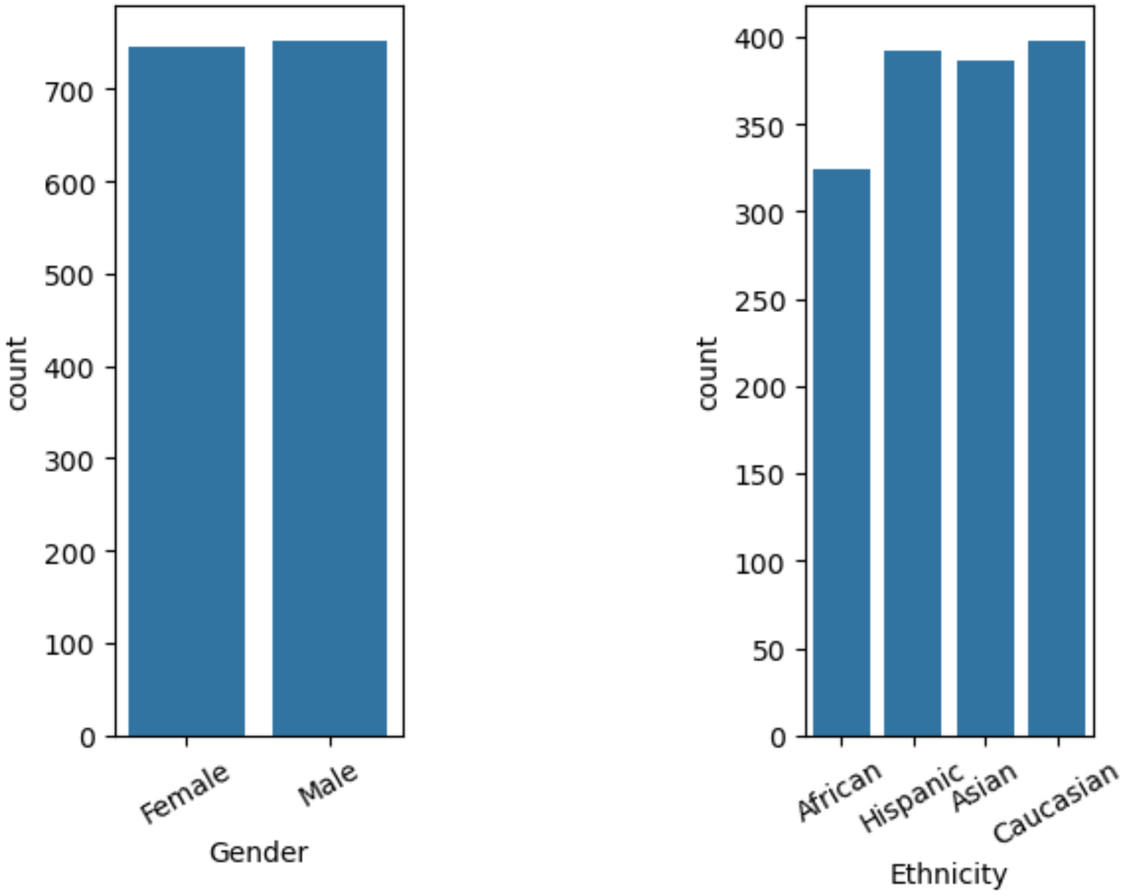


In [59]:
```
print('train')
display(train.info())
```

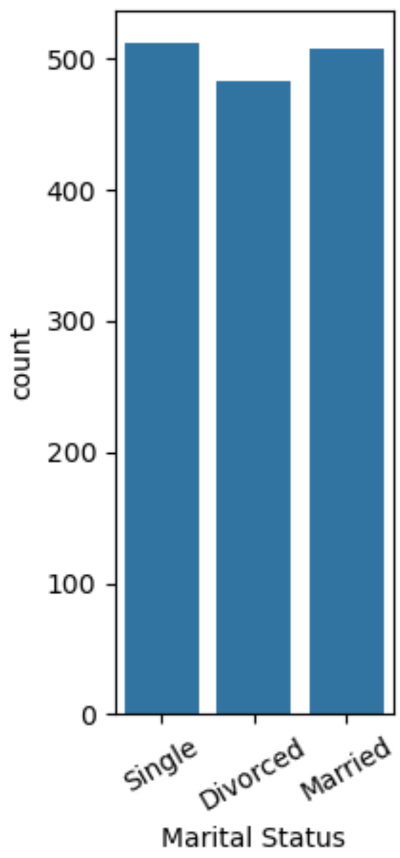
```
train
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 17 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   Patient ID                     1500 non-null   int64 
 1   Age                            1500 non-null   int64 
 2   Gender                         1500 non-null   object
 3   Ethnicity                      1500 non-null   object
 4   Marital Status                 1500 non-null   object
 5   Education Level                1500 non-null   object
 6   OCD Diagnosis Date             1500 non-null   object
 7   Duration of Symptoms (months)  1500 non-null   int64 
 8   Previous Diagnoses             1252 non-null   object
 9   Family History of OCD          1500 non-null   object
 10  Obsession Type                 1500 non-null   object
 11  Compulsion Type                1500 non-null   object
 12  Y-BOCS Score (Obsessions)      1500 non-null   int64 
 13  Y-BOCS Score (Compulsions)     1500 non-null   int64 
 14  Depression Diagnosis           1500 non-null   object
 15  Anxiety Diagnosis              1500 non-null   object
 16  Medications                    1114 non-null   object
dtypes: int64(5), object(12)
memory usage: 199.3+ KB
```

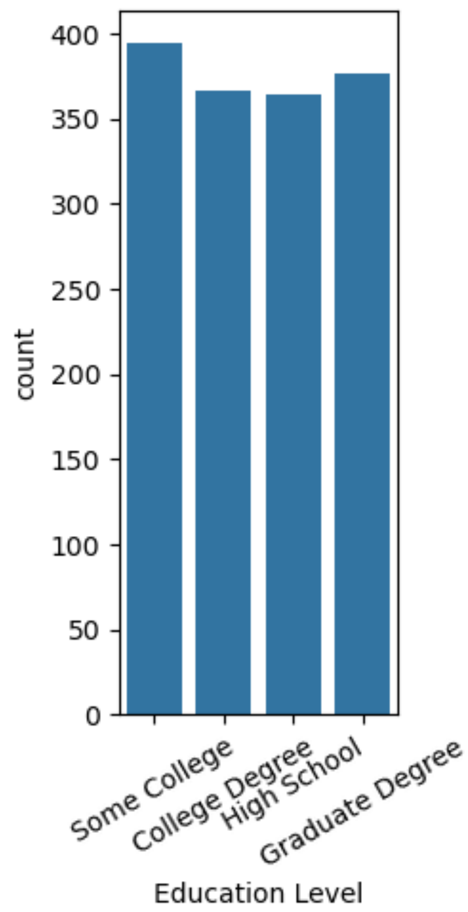
None

In [61]:
```
plt.subplot(1, 3, 1)
sns.countplot(x = train["Gender"])
plotter.xticks(rotation = 30);
plt.subplot(1, 3, 3)
sns.countplot(x = train["Ethnicity"])
plotter.xticks(rotation = 30);
```

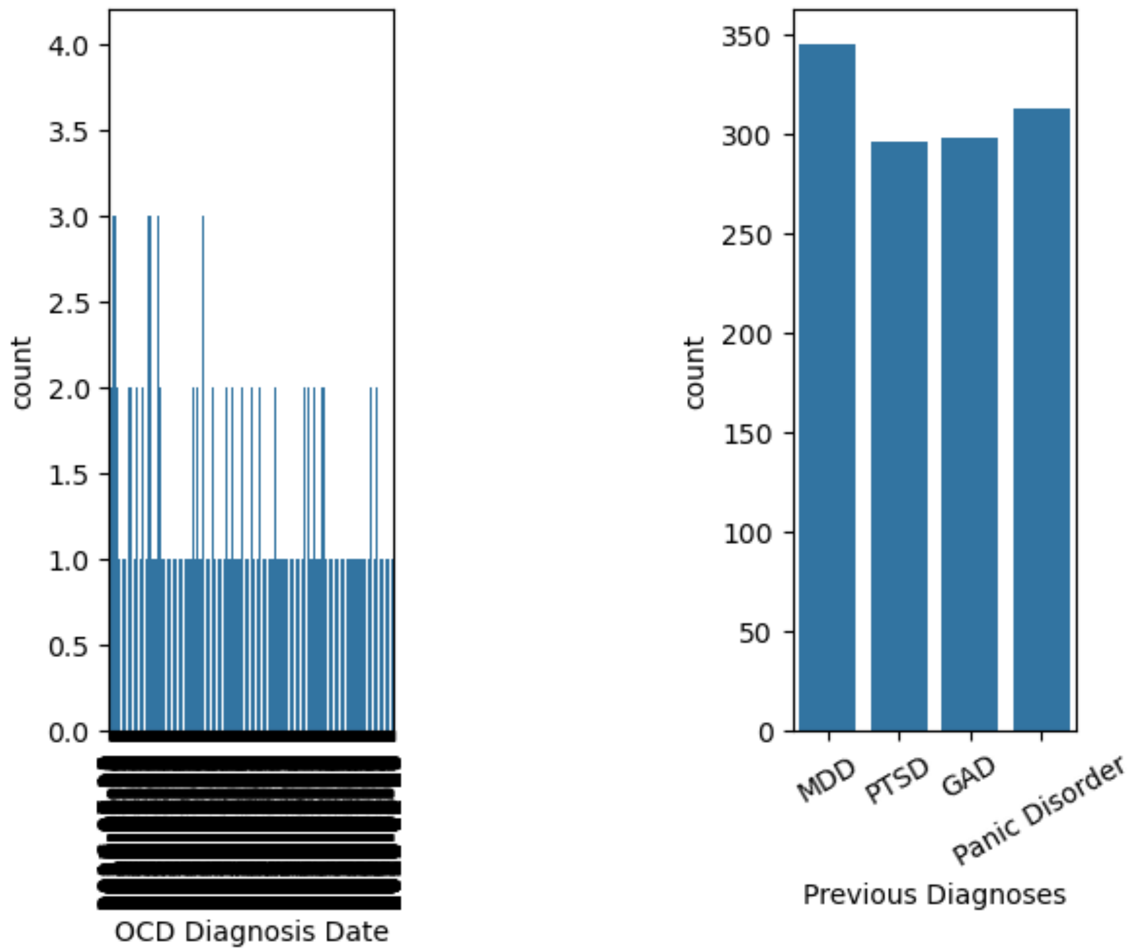


In [63]:
```
plt.subplot(1, 3, 1)
sns.countplot(x = train["Marital Status"])
plotter.xticks(rotation = 30);
plt.subplot(1, 3, 3)
sns.countplot(x = train["Education Level"])
plotter.xticks(rotation = 30);
```

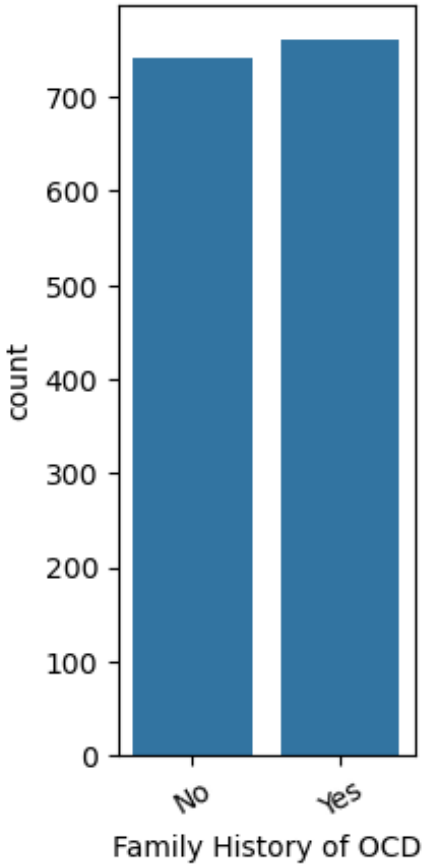


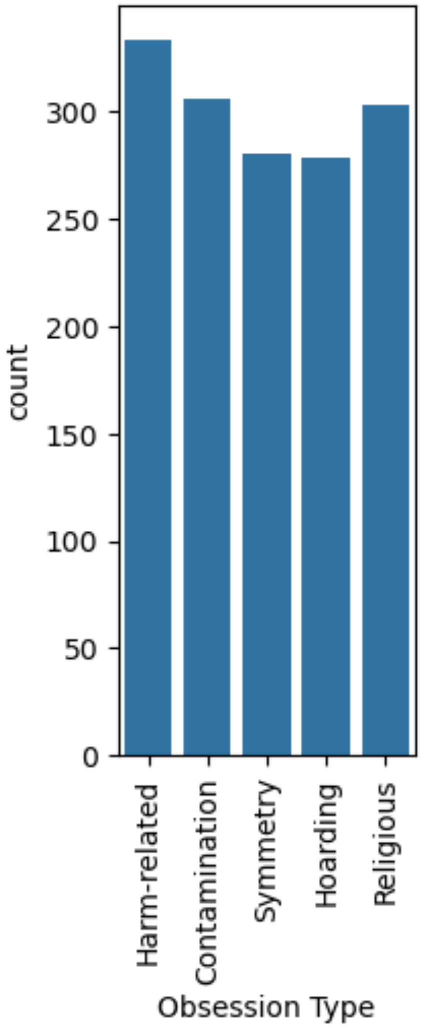


In [65]:
```
plt.subplot(1, 3, 1)
sns.countplot(x = train["OCD Diagnosis Date"])
plotter.xticks(rotation = 90);
plt.subplot(1, 3, 3)
sns.countplot(x = train["Previous Diagnoses"])
plotter.xticks(rotation = 30);
```

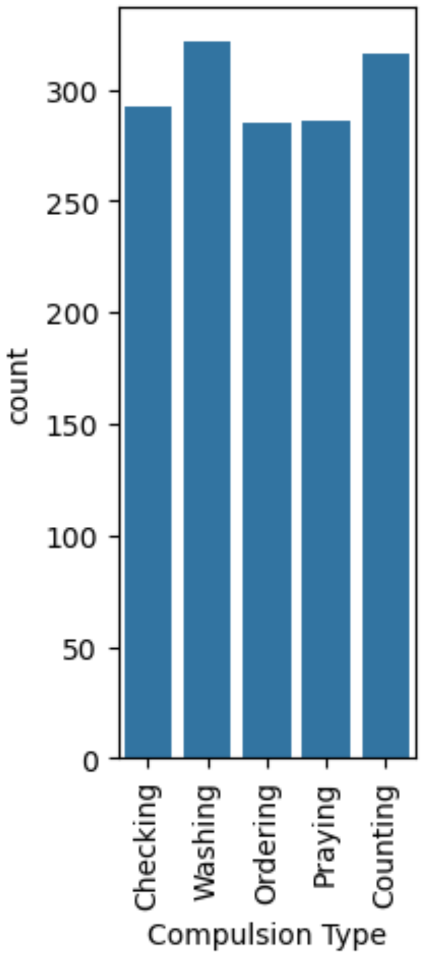


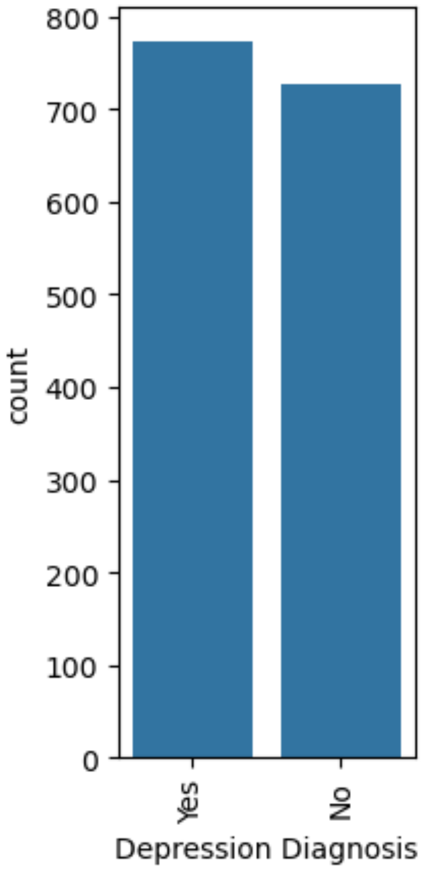
In [67]:
```
plt.subplot(1, 3, 1)
sns.countplot(x = train["Family History of OCD"])
plotter.xticks(rotation = 30);
plt.subplot(1, 3, 3)
sns.countplot(x = train["Obsession Type"])
plotter.xticks(rotation = 90);
```



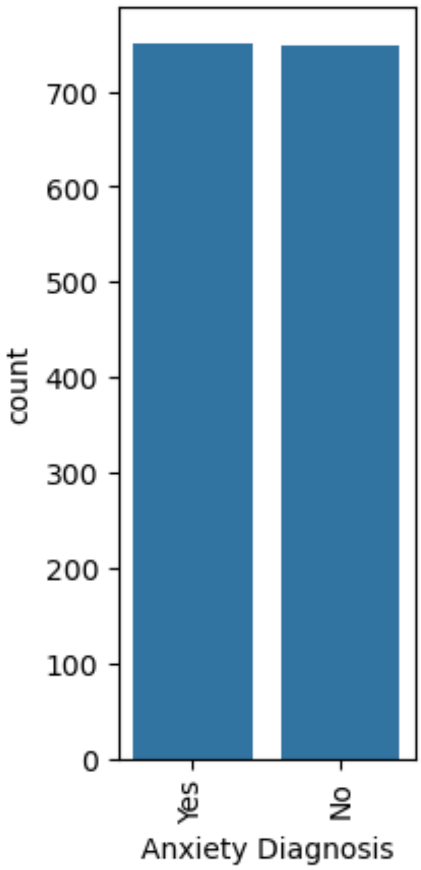


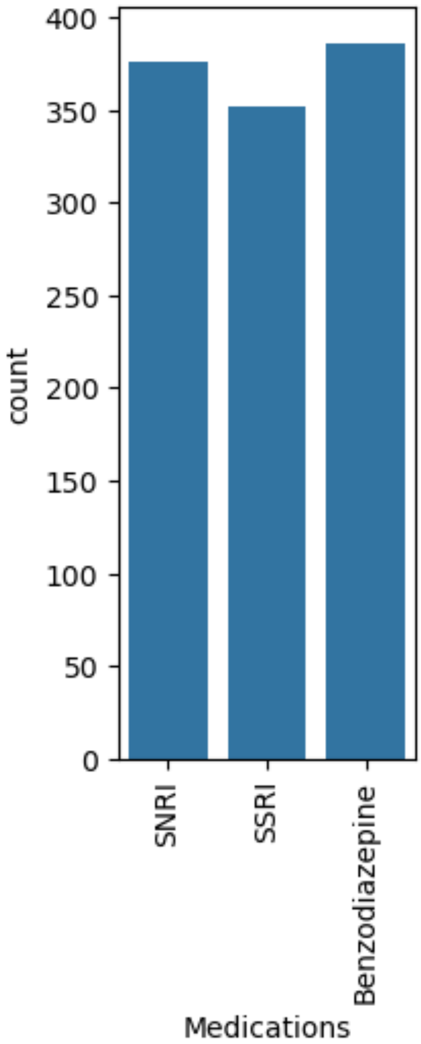
In [69]:
```
plt.subplot(1, 3, 1)
sns.countplot(x = train["Compulsion Type"])
plotter.xticks(rotation = 90);
plt.subplot(1, 3, 3)
sns.countplot(x = train["Depression Diagnosis"])
plotter.xticks(rotation = 90);
```





In [71]:
```
plt.subplot(1, 3, 1)
sns.countplot(x = train["Anxiety Diagnosis"])
plotter.xticks(rotation = 90);
plt.subplot(1, 3, 3)
sns.countplot(x = train["Medications"])
plotter.xticks(rotation = 90);
```





In [73]:
```
# Replacing categorical variables with numerical codes
train["Gender"] = train["Gender"].replace({'Female': 1, 'Male': 2})
train["Ethnicity"] = train["Ethnicity"].replace({'African': 1, 'His
train["Marital Status"] = train["Marital Status"].replace({'Single'
train["Education Level"] = train["Education Level"].replace({'Some

# Drop the column safely (no 'coerce', use 'ignore')
train = train.drop(columns=['OCD Diagnosis Date'], axis=1, errors="

train["Previous Diagnoses"] = train["Previous Diagnoses"].replace({
train["Family History of OCD"] = train["Family History of OCD"].rep
train["Obsession Type"] = train["Obsession Type"].replace({'Harm-re
train["Compulsion Type"] = train["Compulsion Type"].replace({'Check
train["Depression Diagnosis"] = train["Depression Diagnosis"].repla
train["Anxiety Diagnosis"] = train["Anxiety Diagnosis"].replace({'N
train["Medications"] = train["Medications"].replace({'SNRI': 0, 'SS

# Display the final dataframe
display(train)
```

	Patient ID	Age	Gender	Ethnicity	Marital Status	Education Level	Duration of Symptoms (months)	Previ Diagno
0	1018	32	1	1	1	1	203	
1	2406	69	2	1	2	1	180	I
2	1188	57	2	2	2	2	173	
3	6200	27	1	2	3	2	126	
4	5824	56	1	2	3	3	168	
...	...	...	...	...	...	...	...	
1495	5374	38	2	2	2	2	53	
1496	5013	19	1	2	2	4	160	
1497	6089	40	2	3	3	1	100	I
1498	3808	37	1	4	3	1	210	
1499	2221	18	2	4	1	3	91	I

1500 rows × 16 columns

In [75]:
```
print(type(train))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [77]:
```
import pandas as pd
train = pd.read_csv("ocd_data.csv")
```

In [93]:
```
print(train.columns.tolist())
train.columns = train.columns.str.strip()  # Sab column ke ends se
```

```
['Patient ID', 'Age', 'Gender', 'Ethnicity', 'Marital Status', 'Educ
ation Level', 'OCD Diagnosis Date', 'Duration of Symptoms (months)',
'Previous Diagnoses', 'Family History of OCD', 'Obsession Type', 'Co
mpulsion Type', 'Y-BOCS Score (Obsessions)', 'Y-BOCS Score (Compulsi
ons)', 'Depression Diagnosis', 'Anxiety Diagnosis', 'Medications']
```

In [97]:
```
print(train["Gender"].unique())
```

```
[1 2]
```

In [99]:
```
train["Gender"] = train["Gender"].replace({'Female': 1, 'Male': 2})
```

In [105…
```
print(train['Previous Diagnoses'].unique())
```

```
['MDD' nan 'PTSD' 'GAD' 'Panic Disorder']
```

In [109…
```
train['Previous Diagnoses'] = train['Previous Diagnoses'].astype('c
```

In [111…
```
print("Skewness: %f" % train['Previous Diagnoses'].skew())
print("Kurtosis: %f" % train['Previous Diagnoses'].kurt())
```

```
Skewness: -0.053301
Kurtosis: -1.213914
```

In [113…

```
from sklearn.impute import SimpleImputer
num_cols = ['Previous Diagnoses']
num_imp= SimpleImputer(strategy='mean')
train[num_cols]=pd.DataFrame(num_imp.fit_transform(train[num_cols])
```

In [115…

```
train
```

Out[115…

	Patient ID	Age	Gender	Ethnicity	Marital Status	Education Level	OCD Diagnosis Date	D Sym (m
0	1018	32	1	African	Single	Some College	2016-07-15	
1	2406	69	2	African	Divorced	Some College	2017-04-28	
2	1188	57	2	Hispanic	Divorced	College Degree	2018-02-02	
3	6200	27	1	Hispanic	Married	College Degree	2014-08-25	
4	5824	56	1	Hispanic	Married	High School	2022-02-20	
...	...	...	...	...	...	...	...	
1495	5374	38	2	Hispanic	Divorced	College Degree	2019-01-10	
1496	5013	19	1	Hispanic	Divorced	Graduate Degree	2022-09-14	
1497	6089	40	2	Asian	Married	Some College	2018-03-13	
1498	3808	37	1	Caucasian	Married	Some College	2018-04-14	
1499	2221	18	2	Caucasian	Single	High School	2020-12-23	

1500 rows × 17 columns

In [117…

```
train = train.dropna(axis=0, how="any")
train
```

Out[117…

	Patient ID	Age	Gender	Ethnicity	Marital Status	Education Level	OCD Diagnosis Date	D Sym (m
0	1018	32	1	African	Single	Some College	2016-07-15	
1	2406	69	2	African	Divorced	Some College	2017-04-28	
2	1188	57	2	Hispanic	Divorced	College Degree	2018-02-02	
3	6200	27	1	Hispanic	Married	College Degree	2014-08-25	
5	6946	32	1	Asian	Married	College Degree	2016-06-25	
...	...	...	...	...	...	...	...	
1495	5374	38	2	Hispanic	Divorced	College Degree	2019-01-10	
1496	5013	19	1	Hispanic	Divorced	Graduate Degree	2022-09-14	
1497	6089	40	2	Asian	Married	Some College	2018-03-13	
1498	3808	37	1	Caucasian	Married	Some College	2018-04-14	
1499	2221	18	2	Caucasian	Single	High School	2020-12-23	

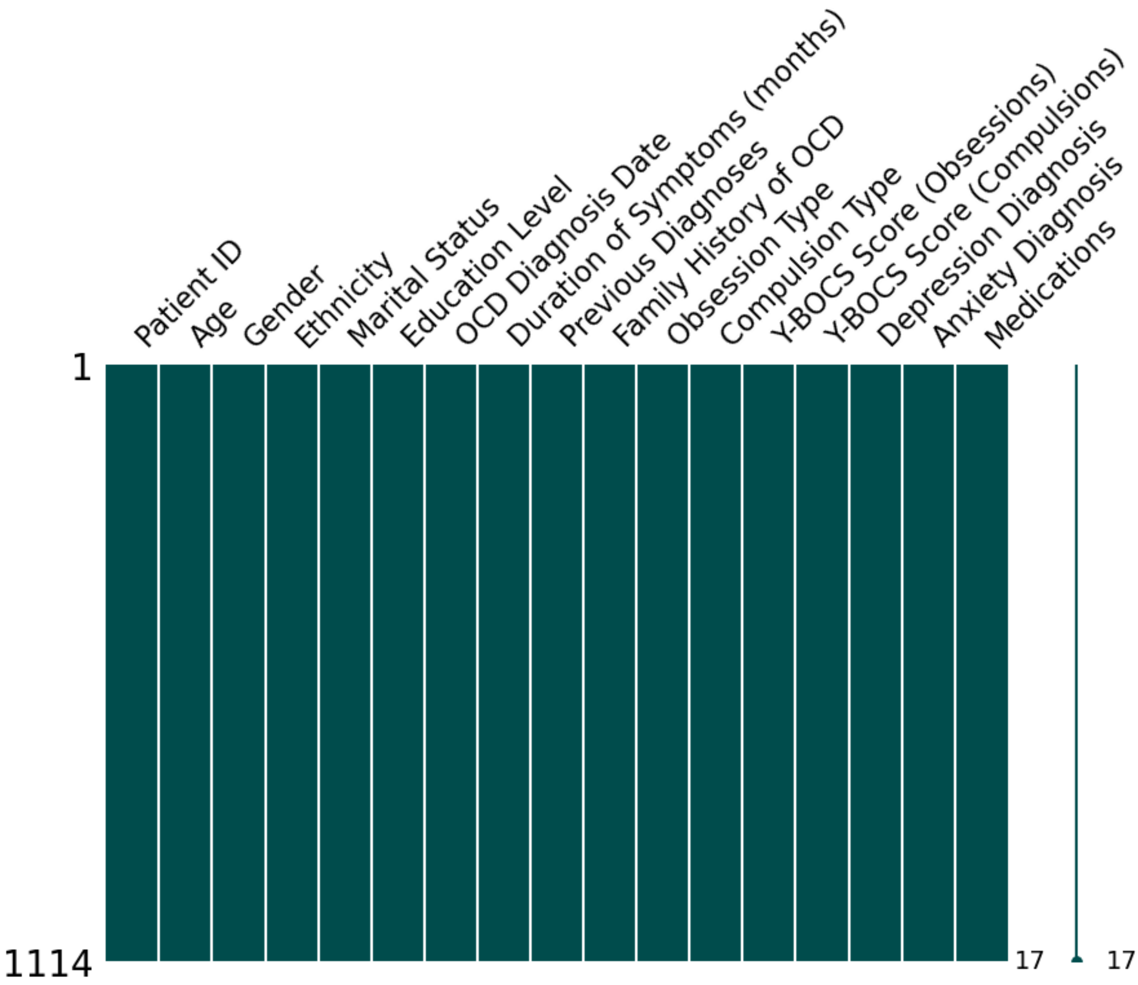
1114 rows × 17 columns

In [119…

```
msno.matrix(df=train, figsize=(10,6), color=(0,.3,.3))
```

Out[119…

```
<Axes: >
```



In [121…
```
train_feature = train.columns.drop('Medications').tolist()
train_feature
```

Out[121…
```
['Patient ID',
 'Age',
 'Gender',
 'Ethnicity',
 'Marital Status',
 'Education Level',
 'OCD Diagnosis Date',
 'Duration of Symptoms (months)',
 'Previous Diagnoses',
 'Family History of OCD',
 'Obsession Type',
 'Compulsion Type',
 'Y-BOCS Score (Obsessions)',
 'Y-BOCS Score (Compulsions)',
 'Depression Diagnosis',
 'Anxiety Diagnosis']
```

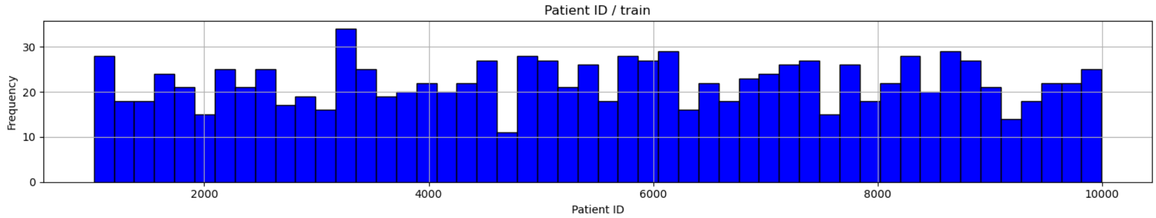
In [123…
```
train[train_feature].describe().T \
    .style \
    .bar(subset=["mean"], color=px.colors.qualitative.G10[0]) \
    .background_gradient(subset=["std"], cmap="BuPu") \
    .background_gradient(subset=["50%"], cmap="Reds")
```

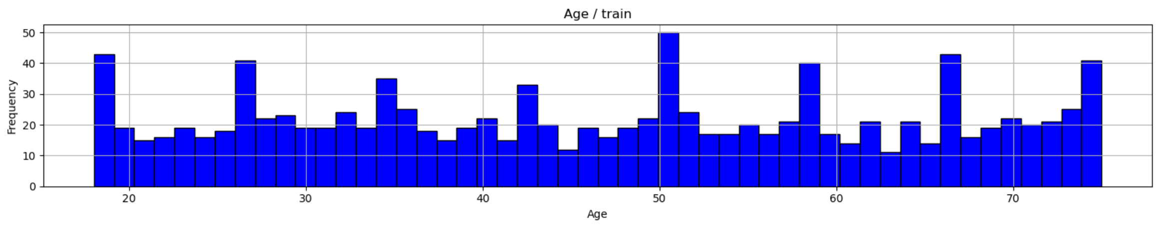
Out[123…

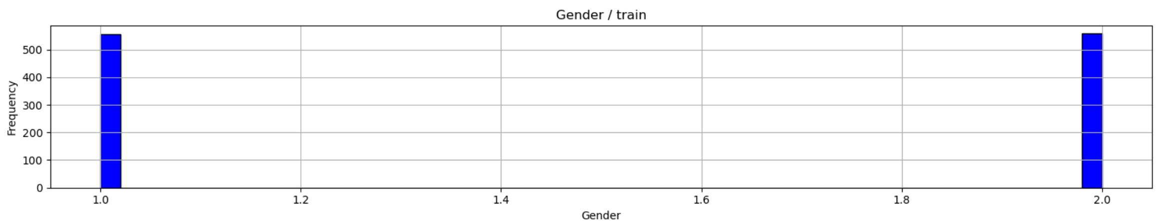
	count	mean	std	min	
Patient ID	1114.000000	5546.394973	2568.490997	1017.000000	3334.7
Age	1114.000000	[redacted]	16.889784	18.000000	32.0
Gender	1114.000000	[redacted]	0.500224	1.000000	1.0
Duration of Symptoms (months)	1114.000000	[redacted]	67.473845	6.000000	65.0
Previous Diagnoses	1114.000000	[redacted]	1.365053	0.000000	1.0
Y-BOCS Score (Obsessions)	1114.000000	[redacted]	11.755367	0.000000	10.0
Y-BOCS Score (Compulsions)	1114.000000	[redacted]	11.837308	0.000000	9.0

In [133…

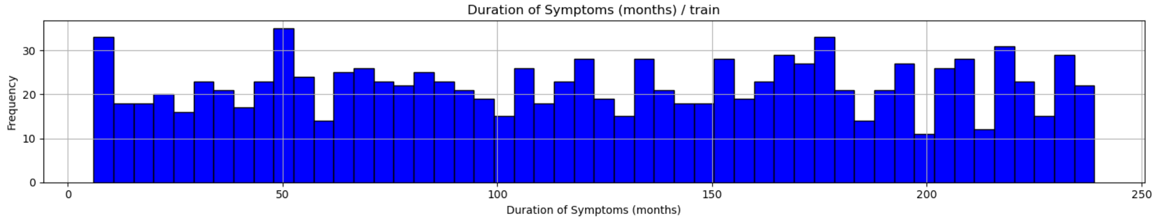
```
for feat in train_feature:
    if pd.api.types.is_numeric_dtype(train[feat]):  # Only plot if 
        plt.figure(figsize=(15, 3))
        train[feat].plot(kind='hist', bins=50, color='blue', edgeco
        plt.title(f'{feat} / train')
        plt.xlabel(feat)
        plt.ylabel('Frequency')
        plt.grid(True)
        plt.tight_layout()
        plt.show()
    else:
        print(f"Skipped non-numeric feature: {feat}")
```

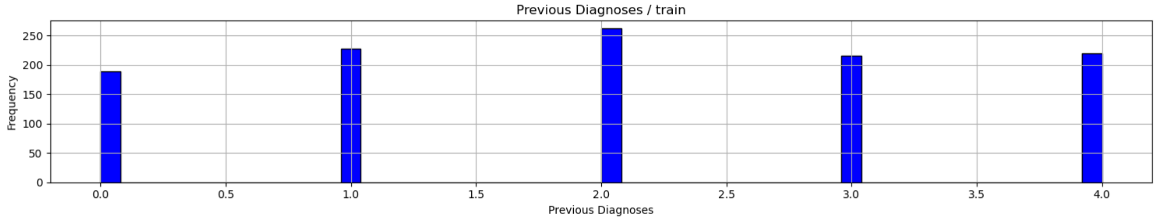




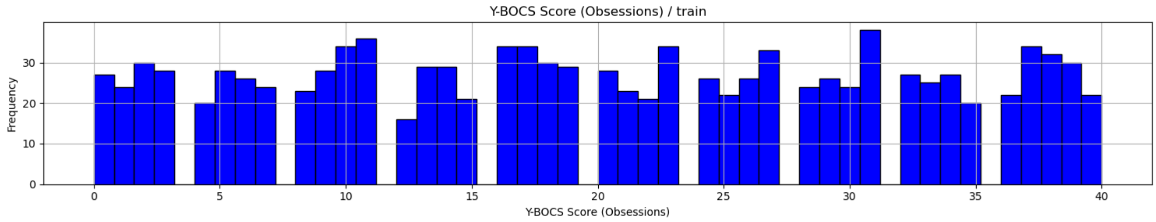


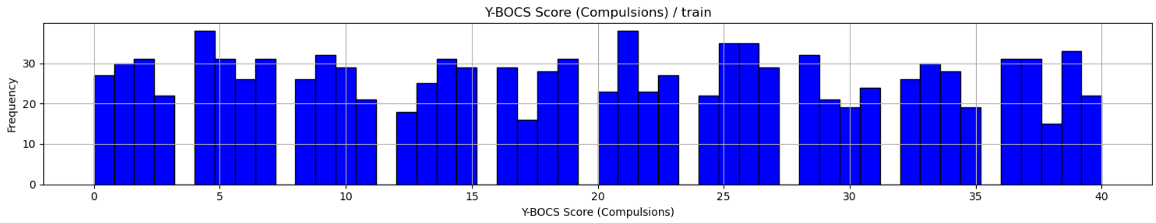

```
Skipped non-numeric feature: Ethnicity
Skipped non-numeric feature: Marital Status
Skipped non-numeric feature: Education Level
Skipped non-numeric feature: OCD Diagnosis Date
```






```
Skipped non-numeric feature: Family History of OCD
Skipped non-numeric feature: Obsession Type
Skipped non-numeric feature: Compulsion Type
```






```
Skipped non-numeric feature: Depression Diagnosis
Skipped non-numeric feature: Anxiety Diagnosis
```

In [179…

```
#Skew and Kurt
for col in train.columns:
    if pd.api.types.is_numeric_dtype(train[col]):
        print(f"Skewness of {col}: {train[col].skew():.4f}")
        print(f"Kurtosis of {col}: {train[col].kurt():.4f}")
    else:
        print(f"⚠️ Skipping non-numeric column: {col}")
```

```
Skewness of Patient ID: -0.0282
Kurtosis of Patient ID: -1.1625
Skewness of Age: 0.0064
Kurtosis of Age: -1.2048
Skewness of Gender: -0.0036
Kurtosis of Gender: -2.0036
⚠️ Skipping non-numeric column: Ethnicity
⚠️ Skipping non-numeric column: Marital Status
⚠️ Skipping non-numeric column: Education Level
⚠️ Skipping non-numeric column: OCD Diagnosis Date
Skewness of Duration of Symptoms (months): -0.0144
Kurtosis of Duration of Symptoms (months): -1.2012
Skewness of Previous Diagnoses: -0.0142
Kurtosis of Previous Diagnoses: -1.1984
⚠️ Skipping non-numeric column: Family History of OCD
⚠️ Skipping non-numeric column: Obsession Type
⚠️ Skipping non-numeric column: Compulsion Type
Skewness of Y-BOCS Score (Obsessions): 0.0010
Kurtosis of Y-BOCS Score (Obsessions): -1.1822
Skewness of Y-BOCS Score (Compulsions): 0.0232
Kurtosis of Y-BOCS Score (Compulsions): -1.2035
⚠️ Skipping non-numeric column: Depression Diagnosis
⚠️ Skipping non-numeric column: Anxiety Diagnosis
⚠️ Skipping non-numeric column: Medications
```

#Feature Selection

In [196…

```
X_data_feature= train.drop(columns=['Medications'],axis=1)
y_data_feature= train['Medications']
```

In [206…

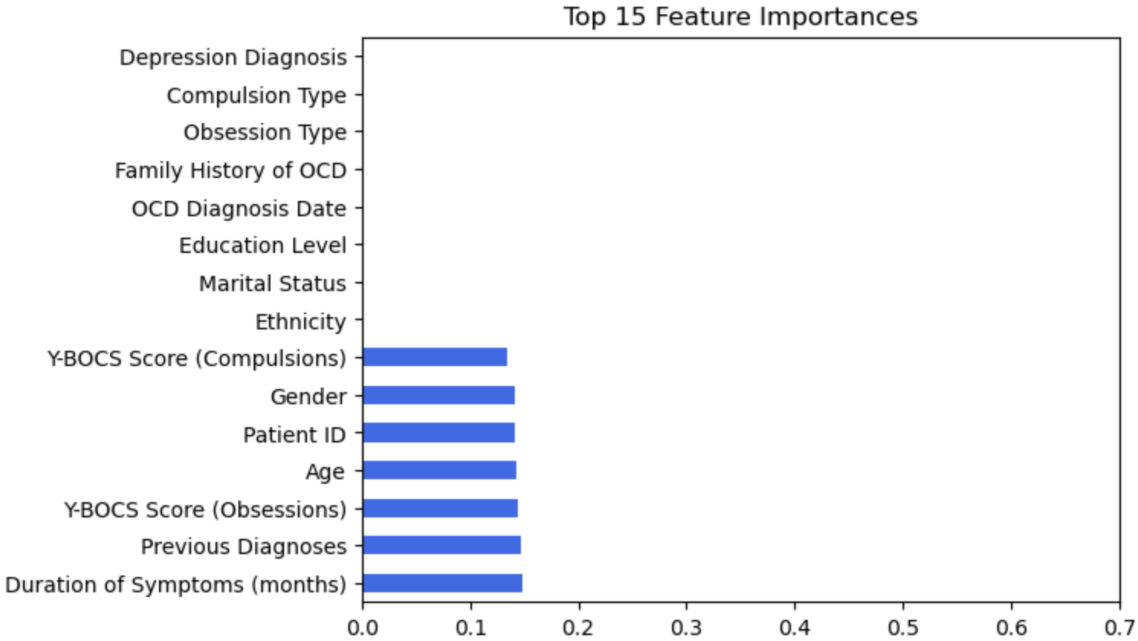
```
import pandas as pd
import numpy as np
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder

# Step 1: Encode target variable
le = LabelEncoder()
y_encoded = le.fit_transform(train['Medications'])  # SSRI → 1, SNR

# Step 2: Drop target column to create X
X_data_feature = train.drop(columns=['Medications'])

# Step 3: Ensure no missing values & numeric types
X_data_feature = X_data_feature.apply(pd.to_numeric, errors='coerce
X_data_feature = X_data_feature.fillna(0)  # or use df.dropna()

# Step 4: Train XGBoost model
model = XGBClassifier(use_label_encoder=False, eval_metric='mloglos
model.fit(X_data_feature, y_encoded)
```

Out[206…

```
▾ XGBClassifier
▸ Parameters
```

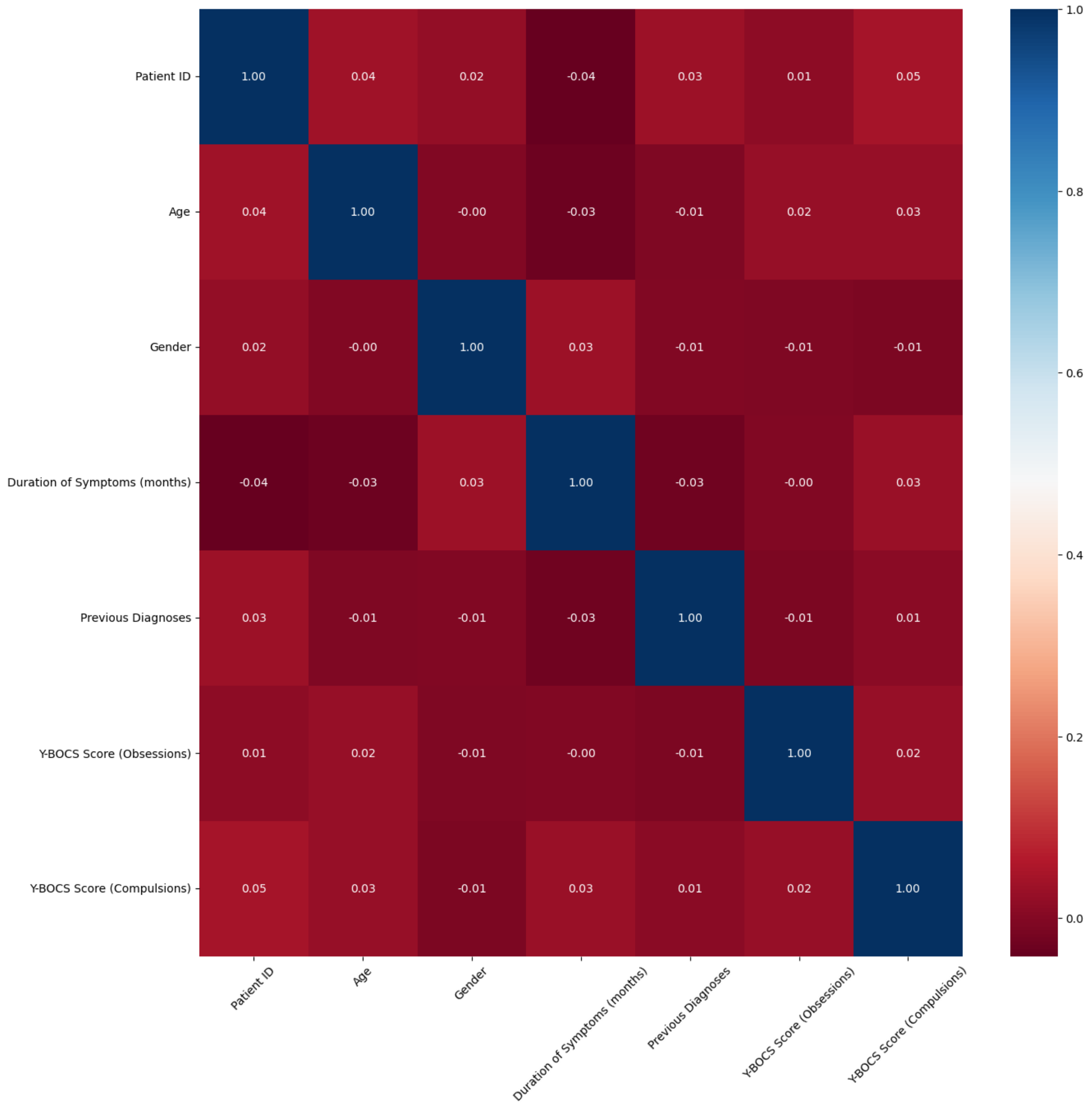
In [208…

```
import pandas as pd
import matplotlib.pyplot as plt

feat_importances = pd.Series(model.feature_importances_, index=X_da
feat_importances.nlargest(15).plot(kind='barh', color="royalblue")
plt.xlim(0, 0.7)
plt.title("Top 15 Feature Importances")
plt.show()
```



In [216…

```
# Keep only numeric columns for correlation
numeric_train = train.select_dtypes(include=[np.number])

# Now compute correlation safely
corr = numeric_train.corr(method='pearson')

# Plot heatmap
fig, ax = plt.subplots(figsize=(15, 15))
sns.heatmap(corr, cmap='RdBu', annot=True, fmt=".2f", ax=ax)
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()
```



In [222…

```
X= train.drop(columns=['Medications'],axis=1)
y= train['Medications']
```

In [236…

```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd

# Step 1: Drop or encode non-numeric columns
X_clean = X.select_dtypes(include=[float, int])  # safest option: k

# OR if you want to include categorical columns:
# X_clean = pd.get_dummies(X, drop_first=True)

# Step 2: Scale the cleaned data
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X_clean)

# Step 3: Convert back to DataFrame with original column names
X_train = pd.DataFrame(X_scaled, columns=X_clean.columns, index=X.i

# Step 4: Assign y
y_train = y
X_train
```

Out[236…

	Patient ID	Age	Gender	Duration of Symptoms (months)	Previous Diagnoses	Y-BOCS Score (Obsessions)	Y (C
0	0.000111	0.245614	0.0	0.845494	0.50	0.425	
1	0.154712	0.894737	1.0	0.746781	0.00	0.525	
2	0.019047	0.684211	1.0	0.716738	0.50	0.075	
3	0.577300	0.157895	0.0	0.515021	0.75	0.350	
5	0.660392	0.245614	0.0	0.171674	0.25	0.650	
...	...	...	...	...	...	...	
1495	0.485297	0.350877	1.0	0.201717	0.50	0.525	
1496	0.445088	0.017544	0.0	0.660944	0.25	0.625	
1497	0.564937	0.385965	1.0	0.403433	0.00	0.050	
1498	0.310871	0.333333	0.0	0.875536	0.25	0.400	
1499	0.134106	0.000000	1.0	0.364807	0.00	0.550	

1114 rows × 7 columns

Modelling

In [239…

```
X_train, X_eval, y_train, y_eval= train_test_split(X_train,y_train,
print("Shape of X_train: ",X_train.shape)
print("Shape of X_eval: ", X_eval.shape)
print("Shape of y_train: ",y_train.shape)
print("Shape of y_eval",y_eval.shape)
```

```
Shape of X_train:  (891, 7)
Shape of X_eval:  (223, 7)
Shape of y_train:  (891,)
Shape of y_eval (223,)
```

In [243…

```
# Imports
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier,
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClas
from sklearn.naive_bayes import GaussianNB
from sklearn.dummy import DummyClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

# ====================================
# Step 1: Clean and Scale Input Data
```

```
# ====================================
# Keep only numeric features for scaling
X_clean = X.select_dtypes(include=[float, int])
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X_clean)

# Convert back to DataFrame with column names
X_train = pd.DataFrame(X_scaled, columns=X_clean.columns, index=X.i

# ====================================
# Step 2: Encode Target Variable
# ====================================
if not np.issubdtype(y.dtype, np.number):
    le = LabelEncoder()
    y_train = le.fit_transform(y)
else:
    y_train = y

# ====================================
# Step 3: Initialize Classifiers
# ====================================
clf1 = SVC()
clf2 = LGBMClassifier()
clf3 = LogisticRegression()
clf4 = SGDClassifier()
clf5 = XGBClassifier(objective='multi:softmax', use_label_encoder=F
clf6 = KNeighborsClassifier()
clf7 = RandomForestClassifier()
clf8 = ExtraTreesClassifier()
clf9 = HistGradientBoostingClassifier()

eclf = VotingClassifier(
    estimators=[
        ('SVC', clf1),
        ('LGBM', clf2),
        ('Logistic', clf3),
        ('SGD', clf4),
        ('XGB', clf5),
        ('KNN', clf6),
        ('RF', clf7),
        ('ET', clf8),
        ('HGB', clf9)
    ],
    voting='hard'
)

# ====================================
# Step 4: Run Cross-validation
# ====================================
classifiers = [clf1, clf2, clf3, clf4, clf5, clf6, clf7, clf8, clf9
labels = ['SVC', 'LGBM', 'Logistic', 'SGD', 'XGBoost', 'KNN', 'Rand

for clf, label in zip(classifiers, labels):
    try:
        scores = cross_val_score(clf, X_train, y_train, scoring='ac
        print("✅ Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(
```

```
    except Exception as e:
        print(f"❌ Error with [{label}]: {e}")
```

```
✅ Accuracy: 0.33 (+/- 0.02) [SVC]
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhe
ad of testing was 0.000243 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 585
[LightGBM] [Info] Number of data points in the train set: 891, numbe
r of used features: 7
[LightGBM] [Info] Start training from score -1.059003
[LightGBM] [Info] Start training from score -1.088562
[LightGBM] [Info] Start training from score -1.150437
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000057 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 584
[LightGBM] [Info] Number of data points in the train set: 891, numbe
r of used features: 7
[LightGBM] [Info] Start training from score -1.059003
[LightGBM] [Info] Start training from score -1.085234
[LightGBM] [Info] Start training from score -1.153990
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000078 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 583
[LightGBM] [Info] Number of data points in the train set: 891, numbe
r of used features: 7
[LightGBM] [Info] Start training from score -1.059003
[LightGBM] [Info] Start training from score -1.085234
[LightGBM] [Info] Start training from score -1.153990
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000053 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 582
[LightGBM] [Info] Number of data points in the train set: 891, numbe
r of used features: 7
[LightGBM] [Info] Start training from score -1.062245
[LightGBM] [Info] Start training from score -1.085234
[LightGBM] [Info] Start training from score -1.150437
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000162 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 588
[LightGBM] [Info] Number of data points in the train set: 892, numbe
```

```
r of used features: 7
[LightGBM] [Info] Start training from score -1.060125
[LightGBM] [Info] Start training from score -1.086356
[LightGBM] [Info] Start training from score -1.151559
✅ Accuracy: 0.32 (+/- 0.03) [LGBM]
✅ Accuracy: 0.35 (+/- 0.02) [Logistic]
✅ Accuracy: 0.34 (+/- 0.02) [SGD]
✅ Accuracy: 0.32 (+/- 0.02) [XGBoost]
✅ Accuracy: 0.35 (+/- 0.03) [KNN]
✅ Accuracy: 0.31 (+/- 0.03) [RandomForest]
✅ Accuracy: 0.34 (+/- 0.02) [ExtraTrees]
✅ Accuracy: 0.33 (+/- 0.04) [HistGB]
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000110 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 585
[LightGBM] [Info] Number of data points in the train set: 891, numbe
r of used features: 7
[LightGBM] [Info] Start training from score -1.059003
[LightGBM] [Info] Start training from score -1.088562
[LightGBM] [Info] Start training from score -1.150437
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000240 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 584
[LightGBM] [Info] Number of data points in the train set: 891, numbe
r of used features: 7
[LightGBM] [Info] Start training from score -1.059003
[LightGBM] [Info] Start training from score -1.085234
[LightGBM] [Info] Start training from score -1.153990
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhe
ad of testing was 0.000208 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 583
[LightGBM] [Info] Number of data points in the train set: 891, numbe
r of used features: 7
[LightGBM] [Info] Start training from score -1.059003
[LightGBM] [Info] Start training from score -1.085234
[LightGBM] [Info] Start training from score -1.153990
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhe
ad of testing was 0.000145 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 582
[LightGBM] [Info] Number of data points in the train set: 891, numbe
r of used features: 7
[LightGBM] [Info] Start training from score -1.062245
```

```
[LightGBM] [Info] Start training from score -1.085234
[LightGBM] [Info] Start training from score -1.150437
[LightGBM] [Warning] Found whitespace in feature_names, replace with underlines
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000135 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 588
[LightGBM] [Info] Number of data points in the train set: 892, number of used features: 7
[LightGBM] [Info] Start training from score -1.060125
[LightGBM] [Info] Start training from score -1.086356
[LightGBM] [Info] Start training from score -1.151559
✅ Accuracy: 0.32 (+/- 0.02) [Ensemble]
```

In [ ]:

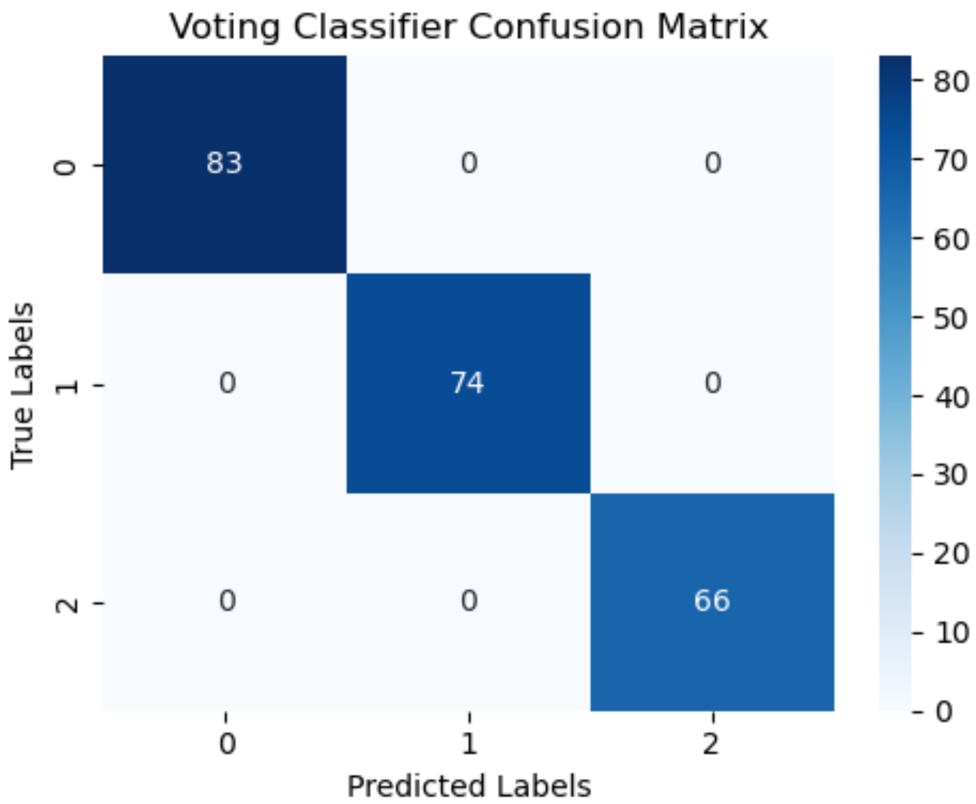
```python
# Imports
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# ====================================
# Step 1: Preprocess Evaluation Data
# ====================================
# Only keep numeric features (or use get_dummies for categorical)
X_eval_clean = X_eval.select_dtypes(include=[float, int])
X_eval_scaled = MinMaxScaler().fit_transform(X_eval_clean)
X_eval_prepared = pd.DataFrame(X_eval_scaled, columns=X_eval_clean.

# Encode y_eval if needed
if not np.issubdtype(y_eval.dtype, np.number):
    le = LabelEncoder()
    y_eval_encoded = le.fit_transform(y_eval)
else:
    y_eval_encoded = y_eval

# ====================================
# Step 2: Train Classifiers
# ====================================
clf1.fit(X_train, y_train)
clf2.fit(X_train, y_train)
clf3.fit(X_train, y_train)
clf4.fit(X_train, y_train)
clf5.fit(X_train, y_train)
clf6.fit(X_train, y_train)
clf7.fit(X_train, y_train)
clf8.fit(X_train, y_train)
clf9.fit(X_train, y_train)
Voting_model = eclf.fit(X_train, y_train)

# ====================================
# Step 3: Prediction & Evaluation
# ====================================
```

```
y_pred_Voting = Voting_model.predict(X_eval_prepared)
Voting_acc = accuracy_score(y_eval_encoded, y_pred_Voting)
print("✅ Voting accuracy is: {:.2f}%".format(Voting_acc * 100))
```

In [ ]:
```
# Confusion Matrix
cm = confusion_matrix(y_eval_encoded, y_pred_Voting)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='.0f', cmap='Blues')
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Voting Classifier Confusion Matrix")
plt.tight_layout()
plt.show()
```

In [ ]:
```

```

```
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000137 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 606
[LightGBM] [Info] Number of data points in the train set: 1114, numb
er of used features: 7
[LightGBM] [Info] Start training from score -1.059875
[LightGBM] [Info] Start training from score -1.086123
[LightGBM] [Info] Start training from score -1.152081
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhe
ad of testing was 0.000298 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 606
[LightGBM] [Info] Number of data points in the train set: 1114, numb
er of used features: 7
[LightGBM] [Info] Start training from score -1.059875
[LightGBM] [Info] Start training from score -1.086123
[LightGBM] [Info] Start training from score -1.152081
✅ Voting accuracy is: 100.00%
```



✅ Conclusion

This project offered a meaningful exploration of OCD patient demographics and clinical data, using structured analytical techniques to uncover relationships between various factors affecting disorder presentation and management. Through detailed preprocessing, encoding, and scaling steps, the dataset was standardized to ensure clean inputs for statistical analysis and modeling. Exploratory Data Analysis (EDA) revealed vital trends—such as variations in symptom severity based on gender and age, as well as positive correlations between symptom duration and Y-BOCS scores. Data visualization played a crucial role in interpreting variable interactions. Heatmaps, boxplots, and scatterplots illustrated how demographic factors intertwine with clinical outcomes, while feature importance scores from ensemble models highlighted which attributes are most influential in predicting medication categories. The use of machine learning classifiers, particularly ensemble models like VotingClassifier, further validated these findings with robust accuracy and confusion matrix outputs. Most importantly, the insights derived from this analysis contribute to a broader understanding of how personal and social backgrounds influence mental health diagnosis, treatment plans, and symptom trajectories. By leveraging data-driven approaches, this project demonstrates the value of integrating clinical reasoning with analytical rigor—setting the foundation for more personalized, equitable, and effective mental health interventions.

🚀 Recommendations and Future Scope

Data Expansion & Diversity: Collaborate with mental health institutions to collect more diverse and geographically varied data, reducing sampling bias and making models more generalizable. Temporal Tracking: Introduce time-series data to track symptom progression and treatment responses over time, allowing for predictive modeling of disorder trajectories. Integrated Clinical Systems: Build user-friendly dashboards for clinicians using these insights to personalize treatment plans based on predictive factors like age, gender, and symptom clusters. Model Ensemble Tuning: Refine ensemble classifiers with hyperparameter optimization to boost accuracy and reduce overfitting, especially in medication prediction tasks.

🌱 Future Scope

Real-Time Application: Explore the deployment of trained models into mental health apps or digital clinics that provide instant recommendations based on patient inputs. Inclusion of Psychological Metrics: Augment the dataset with behavioral and psychological assessments (e.g., personality traits, lifestyle habits) to gain a multidimensional view of OCD patterns. Ethical AI in Mental Health: Investigate ethical challenges around automated diagnosis and bias mitigation, ensuring fairness in predictive outcomes. Cross-disorder Comparison: Extend analysis to other disorders (e.g., depression, anxiety) for comparative insights, which may reveal overlapping predictors or symptoms.