

Data Collection and Preprocessing Phase

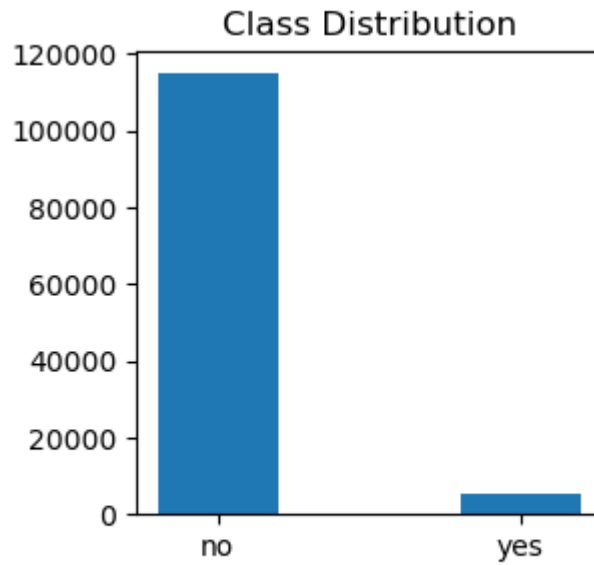
Date	6 th July 2024
Team ID	SWTID1720151909
Project Title	PANIC DISORDER DETECTION
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

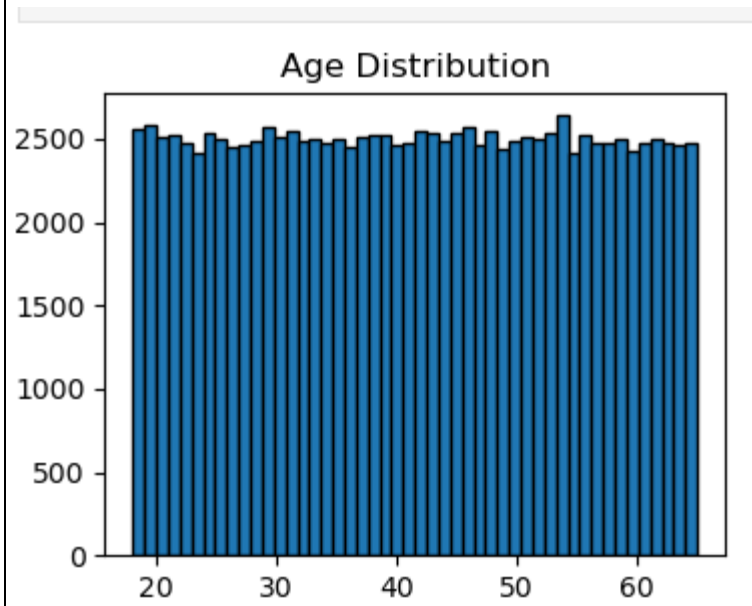
Section	Description
Data Overview	Dimensions: Rows - 5656458, columns - 6 Basic structure of the data.

Univariate Analysis

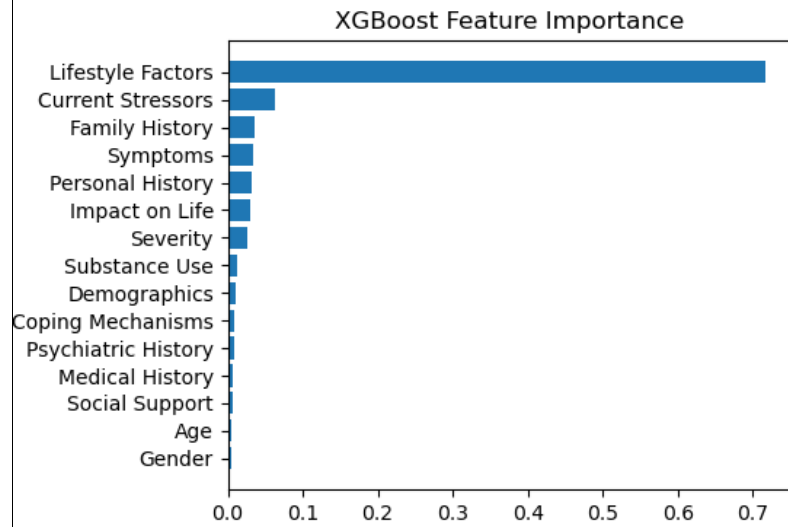


The target distribution is very imbalanced.

Bivariate Analysis



Multivariate Analysis



Data Preprocessing Code Screenshots

Loading Data

Handling Missing Data	<pre>def columns_summary(): col_list = df.columns.to_list() dtype_list = [] null_list = [] unique_list = [] for col in df.columns: dtype_list.append(df[col].dtype) null_list.append(df[col].isnull().sum()) unique_list.append(df[col].nunique()) df_sum = pd.DataFrame(list(zip(col_list, dtype_list, null_list, unique_list)), columns = ['Feature', 'Data type', 'Null values', 'Unique values']) return df_sum.style.hide(axis='index')</pre>
Data Transformation	<pre>def update_classification(): from sklearn.utils.class_weight import compute_sample_weight from sklearn.svm import SVC from sklearn.linear_model import SGDClassifier from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score from sklearn.metrics import roc_curve, auc, confusion_matrix, matthews_corrcoef def extract_class_weights(): classes = df['Panic Disorder Diagnosis'].unique() weights = compute_sample_weight(class_weight='balanced', y=df['Panic Disorder Diagnosis']) return dict(zip(classes, np.unique(weights))) class_weights = extract_class_weights()</pre>
Feature Engineering	Attached codes in the final folder
Save Processed Data	<pre>def ann_classification(): from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Dense from tensorflow.keras.metrics import BinaryAccuracy from tensorflow.keras.optimizers import Adam classifier = Sequential([Dense(units=9, kernel_initializer='he_uniform', activation='relu', input_dim=np.shape(X_train)[1]), Dense(units=9, kernel_initializer='he_uniform', activation='relu'), Dense(units=1, kernel_initializer='glorot_uniform', activation='sigmoid')]) adam_opt = Adam(learning_rate=0.001) classifier.compile(optimizer=adam_opt, loss='binary_crossentropy', metrics=[BinaryAccuracy(name='accuracy')]) code = 'ANN' classifiers.append(code) history = classifier.fit(X_train, y_train, batch_size=100, epochs=100, verbose=0) predictions = classifier.predict(X_test) y_pred = predictions > 0.5 from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score from sklearn.metrics import roc_curve, auc, confusion_matrix, matthews_corrcoef accuracies[code] = accuracy_score(y_test, y_pred) f1_scores[code] = f1_score(y_test, y_pred, zero_division=0) precision_scores[code] = precision_score(y_test, y_pred, zero_division=0)</pre>