**Panic Attack Prediction - Approach 2**

**Analysis**

***Training Dataset*** Contains 100, 000 entries. 17 columns.

**Demographics:** Age, Gender, Demographics

**Health History:** Family History, Personal History, Medical History, Psychiatric History, Substance Use

**Symptoms & Severity:** Current Stressors, Symptoms, Severity, Impact on Life

**Lifestyle & Support:** Coping Mechanisms, Social Support, Lifestyle Factors

**Target Variable:** Panic Disorder Diagnosis (0 or 1)

***Testing Dataset*** Contains 20, 000 entries. 17 columns.

Testing dataset is used to evaluate the performance of the model after training.

Purpose of this,

**Generalization Check:** It ensures the model works well on new, unseen data, not just the training data.

**Prevents Overfitting:** If a model performs well on training data but poorly on testing data, it might be overfitting.

**Final Model Evaluation:** After training, metrics like accuracy, precision, recall, and F1-score are computed on the test set to measure real-world performance.

***EDA Analysis*** (Exploratory Data Analysis)

Analyzing Age feature  
A **box plot** is created to detect outliers.

A **histogram** with a KDE (Kernel Density Estimation) is used to show the distribution.

Bar Plots  
This removes non-categorical or irrelevant features (Participant ID, Age, Panic Disorder Diagnosis).

**sns.FacetGrid** to visualize categorical feature distributions more effectively.

Why this is better  
✅ **More Compact**: Instead of plotting individual countplot graphs, this arranges them neatly.

✅ **Better Readability**: col\_wrap=3 ensures they fit within screen width.

✅ **Handles Large Categories**: Instead of overlapping x-axis labels, it rotates them.

✅ **Easier Comparison**: Since all plots share the same style, comparing distributions is simpler.

***Data Preprocessing***

**Data preprocessing** step where,

-categorical variables are encoded.

-numerical features are retained.

First,

**Target variable** (Panic Disorder Diagnosis) is separated from the datasets for model training.

**One-hot Encoding for the training set**

**Age** (a numerical feature) is stored separately.

**Categorical columns** are identified.

Uses OneHotEncoder to **convert categorical variables into binary features**.

sparse\_output=False: Ensures output is a dense NumPy array.

handle\_unknown='ignore': Avoids errors if new categories appear in the test set.

Converts the encoded data into a DataFrame.

**Combines** the transformed categorical data with the Age column.

**One-Hot Encoding for the Test Set (Same Steps)**

The test set is encoded using the **same feature transformation**.

The encoded test set is combined with the Age column.

***Standardization of Age Column***

**Standardizes the "Age" feature**, converts one-hot encoded columns to integers, and merges the target variable (Panic Disorder Diagnosis) back into the dataset.

**Purpose:** Standardization ensures Age follows a **normal distribution** with:

* **Mean = 0**
* **Standard deviation = 1**

StandardScaler() scales Age to remove bias due to different numerical ranges.

.fit\_transform():

* **Fit:** Computes the mean and standard deviation from the training data.
* **Transform:** Applies scaling.

reshape(-1, 1): Ensures data is in the correct format for StandardScaler

If we use .transform(), it will maintain consistency with the training set.

**Converting One-Hot Encoded Features to Integers**

* Purpose: Convert one-hot encoded features from float64 to int64 for memory efficiency.
* Steps:
  1. Store the "Age" column separately (age = X\_train\_preprocessed.loc[:, 'Age']) because it should remain float64 after standardization.
  2. Convert all columns to integers using .astype(dtype='int64').
  3. Drop "Age" and re-add it (X\_train\_preprocessed['Age'] = age) to ensure it's unchanged.

💡 Why keep "Age" as float?  
Since it was standardized, converting it to an integer would remove the decimal values, undoing the standardization effect.

**Resets indexes** to avoid mismatches after transformations.

**Combines** the processed feature set (X\_train\_preprocessed) with the target variable (train\_PPD).

The final **ALL\_train dataset** is now fully preprocessed and ready for training.

**\*\*Correlation Matrix\*\***

Since the values are all smaller than the general standard, it is hard to find meaningful relationship in the following correlation matrix.

**Cross validation of Models**

### **What is Cross-Validation?**

**Cross-validation is a resampling technique used to evaluate the performance of machine learning models by splitting the dataset into multiple training and testing sets. Instead of training and testing on the same fixed split (which may lead to overfitting or misleading results), cross-validation ensures that every data point gets a chance to be in both training and testing sets.**

### **Why Use Cross-Validation?**

**✅ Prevents Overfitting – Ensures that the model generalizes well to new data rather than memorizing patterns from one specific training set.  
✅ Maximizes Data Usage – Uses all available data efficiently for both training and validation.  
✅ More Reliable Model Evaluation – Provides a better estimate of how the model will perform on unseen data.  
✅ Reduces Variance in Performance Estimation – Instead of relying on a single train-test split, cross-validation averages multiple evaluations.**

**Model evaluation and comparison** using **Stratified K-Fold Cross-Validation** on different classification algorithms. Best for **imbalanced datasets** (e.g., if panic disorder cases are much rarer than non-cases). Ensures that each fold contains an equal percentage of positive and negative cases.

**Stratified K-Fold ensures that each fold maintains the same class distribution** as the full dataset.

**Why?** This is important for **imbalanced datasets** (e.g., if panic disorder cases are much rarer than non-cases).

**n\_splits=5**: The dataset is split into **5 folds** (80% training, 20% validation in each iteration).

**A set of classifiers are chosen to compare their performance.**

**Why test multiple models?** Different algorithms handle **non-linearity, feature interactions, and feature scaling** differently.

Models included:

* **SVC (Support Vector Classifier)** – Works well for high-dimensional spaces.
* **Decision Tree** – Simple, interpretable, but prone to overfitting.
* **AdaBoost (Adaptive Boosting)** – An ensemble method that improves weak learners (Decision Trees).
* **Random Forest** – An ensemble of decision trees, reducing overfitting.
* **Extra Trees** – Similar to Random Forest but with more randomness.
* **Gradient Boosting** – Powerful ensemble learning method.
* **MLP (Multi-Layer Perceptron)** – A neural network model.
* **K-Nearest Neighbors (KNN)** – Non-parametric method based on distance.
* **Logistic Regression** – Works well for linearly separable data.
* **Linear Discriminant Analysis (LDA)** – Works well for class separability in lower dimension

**Calculates the mean and standard deviation of accuracy** across all cross-validation folds.

## **Key Takeaways**

✅ **Compares multiple ML models using Stratified K-Fold Cross-Validation.**✅ **Ensures balanced training/testing splits for imbalanced data.**✅ **Sorts and visualizes results using a bar plot with a broken axis.**✅ **Displays model performance numerically and visually.**

**## 6.1. Training models**

The top 5 models based on the accuracy graph above were trained by the train set and evaluated by the **\*\*validation set\*\***.

**#### \*\*Top 5 Classifiers based on accuracy:\*\***

1. Multiple Layer Perceptron

2. Support Vector Classifier (SVC)

3. GradientBoosting

4. ExtraTrees

5. AdaBoost

## **1. Multi-Layer Perceptron (MLP) - Neural Network Approach**

💡 **Why?**

* MLP is a type of **feedforward artificial neural network** that can learn **complex non-linear patterns**.
* Panic attack prediction likely involves **complex interactions** between multiple factors (age, symptoms, lifestyle, etc.), which MLP can model well.
* **Backpropagation + Activation Functions** help capture subtle relationships between input features.
* Can handle **high-dimensional and one-hot encoded categorical data** effectively.

✅ **Best for:** Learning hidden patterns that traditional models might miss.

## **2. Support Vector Classifier (SVC) - Margin-Based Learning**

💡 **Why?**

* SVC **finds the optimal hyperplane** that best separates classes (e.g., those who have panic disorder vs. those who don’t).
* It’s robust against **high-dimensional data** and can be used with **kernel tricks** to model non-linearity.
* Works well when the dataset is **not too large** (as training time scales with data size).

✅ **Best for:** Datasets where class separation is important, especially if features are not linearly separable.

## **3. Gradient Boosting Classifier (GBM) - Boosted Decision Trees**

💡 **Why?**

* **Handles imbalanced data well**, which is important if panic disorder cases are rarer than non-cases.
* Uses **sequential learning**: each tree corrects the errors of the previous ones, making it powerful for structured data.
* Can **detect subtle feature importance** in medical or psychological datasets.

✅ **Best for:** Highly accurate models where boosting helps refine predictions.

## **4. Extra Trees Classifier - Randomized Decision Trees**

💡 **Why?**

* Works similarly to **Random Forest**, but introduces extra randomness in feature selection and split points.
* **Less prone to overfitting** than traditional decision trees.
* **Fast training** and robust for **high-dimensional data** with many categorical features.

✅ **Best for:** Handling large feature spaces and reducing variance in predictions.

## **5. AdaBoost - Adaptive Boosting**

💡 **Why?**

* **Boosts weak classifiers** into a strong model, improving performance on structured data.
* Works well for **imbalanced datasets** by **focusing on misclassified samples**.
* **Less prone to overfitting** compared to standard decision trees.

✅ **Best for:** Datasets where boosting weak learners improves predictive accuracy.

### **Final Justification: Why These Models?**

✅ The **panic disorder dataset** likely contains complex feature interactions, so **MLP (neural network)** and **Gradient Boosting (ensemble method)** help capture deep patterns.  
✅ **SVC** is good for **finding clear class separations**, while **Extra Trees** is a fast, robust tree-based model.  
✅ **AdaBoost** works well for structured data and **helps balance class distributions** by focusing on misclassified cases.

🔹 **Best Overall Model?** Likely **MLP or Gradient Boosting**, but cross-validation will confirm the strongest performer! 🚀