Multidimensional Personality Cluster Prediction (Checkpoint 2)

Team Members

1. Kavya Gupta - IMT2023016
2. Nainika Agrawal - IMT2023034
3. Pragya Rai - IMT2023529

Github Link : https://github.com/kavyagupta3011/Classification

Task

The goal of this competition is to predict the personality cluster of a participant based on their behavioral, lifestyle, and social attributes.
This is a multiclass classification problem, where the output variable has multiple personality segments (e.g., Cluster_A, Cluster_B, etc.).
Accurate personality cluster prediction enables better behavioral segmentation, improved personalization strategies, and insights into how lifestyle factors influence psychological grouping.

Dataset and Features Description

This dataset contains anonymized behavioral, lifestyle, and social engagement attributes for each participant. Every row represents an individual, along with multiple indicators describing their routines, background, activity levels, and support systems. All fields are encoded to maintain privacy and ensure the dataset is safe for academic use.

- **train.csv**: 1913 x 14
- **test.csv:** 479 x 13
- **sample_submission.csv:** 5 x 2

Below describes each feature in the dataset:

Column Name: Meaning

1. **participant_id:** Unique ID assigned to each participant
2. **age_group**: Age grouping indicator
3. **identity_code**: Encoded personal identity category
4. **cultural_background:** Regional or cultural background grouping
5. **upbringing_influence**: Influence of formative environment
6. **focus_intensity**: Time/effort dedicated toward focused tasks
7. **consistency_score**: Reliability and routine-stability measure
8. **external_guidance_usage:** Use of guidance, mentoring, or support resources

9. **support_environment_score:** Perceived supportive environment level
10. **hobby_engagement_level:** Engagement in leisure or personal interest activities
11. **physical_activity_index:** Physical activity involvement
12. **creative_expression_index:** Participation in artistic or expressive activities
13. **altruism_score:** Tendency toward volunteering or helping behaviors
14. **personality_cluster: (target)** Personality segment label derived external

The following section outlines the necessary steps for conducting essential **Exploratory Data Analysis (EDA)** to gain insights, detect potential issues, and prepare data for model training.

1. Import Libraries and Load Dataset

The initial step in the EDA process involves importing essential libraries such as pandas, numpy, matplotlib, tensorflow, seaborn, and relevant modules from scikit-learn for data preprocessing. The dataset is loaded into a pandas DataFrame for further analysis and manipulation. After loading the train, test, and sample submission files, the dataset shapes were inspected to confirm successful import.

2. Data Overview

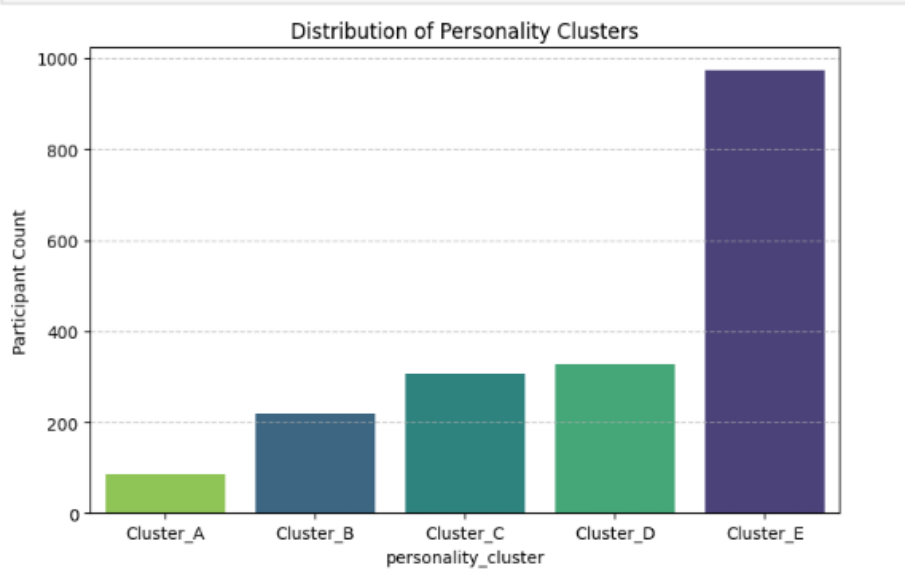
The dataset is first explored by viewing the initial rows, inspecting data types, checking for duplicates and invalid data, and detecting missing values to understand its structure and quality. A statistical summary of the numerical columns is then generated to examine their central tendencies, variability, and possible outliers.

3. Handling duplicates

We found no duplicate rows in this dataset.

4. Handling target variable

On analyzing the target variable, we saw it had 5 unique values: Cluster_A, Cluster_B, Cluster_C, Cluster_D, and Cluster_E. It had no missing values, and all feature columns were present in both train and test datasets. Cluster_E was at top frequency with 974 appearances out of 1913. Class distribution analysis showed an imbalance among personality clusters, highlighting the importance of using Macro F1 Score as the evaluation metric.



5. Handling Missing Values

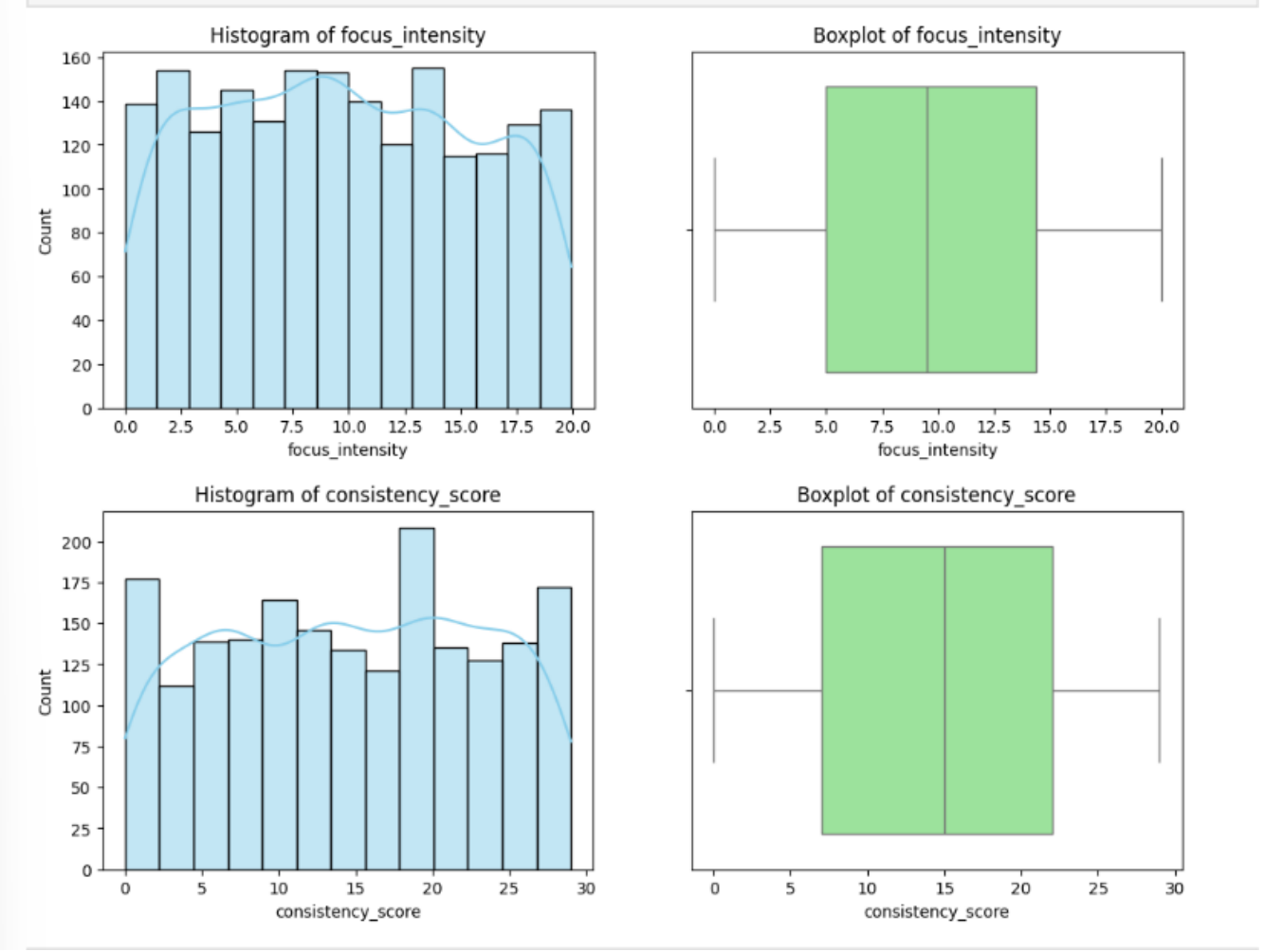
A complete missing-value check was performed using column-wise missing percentage calculations.

No missing values were found in the dataset, so no imputation was required for the original columns at this stage.

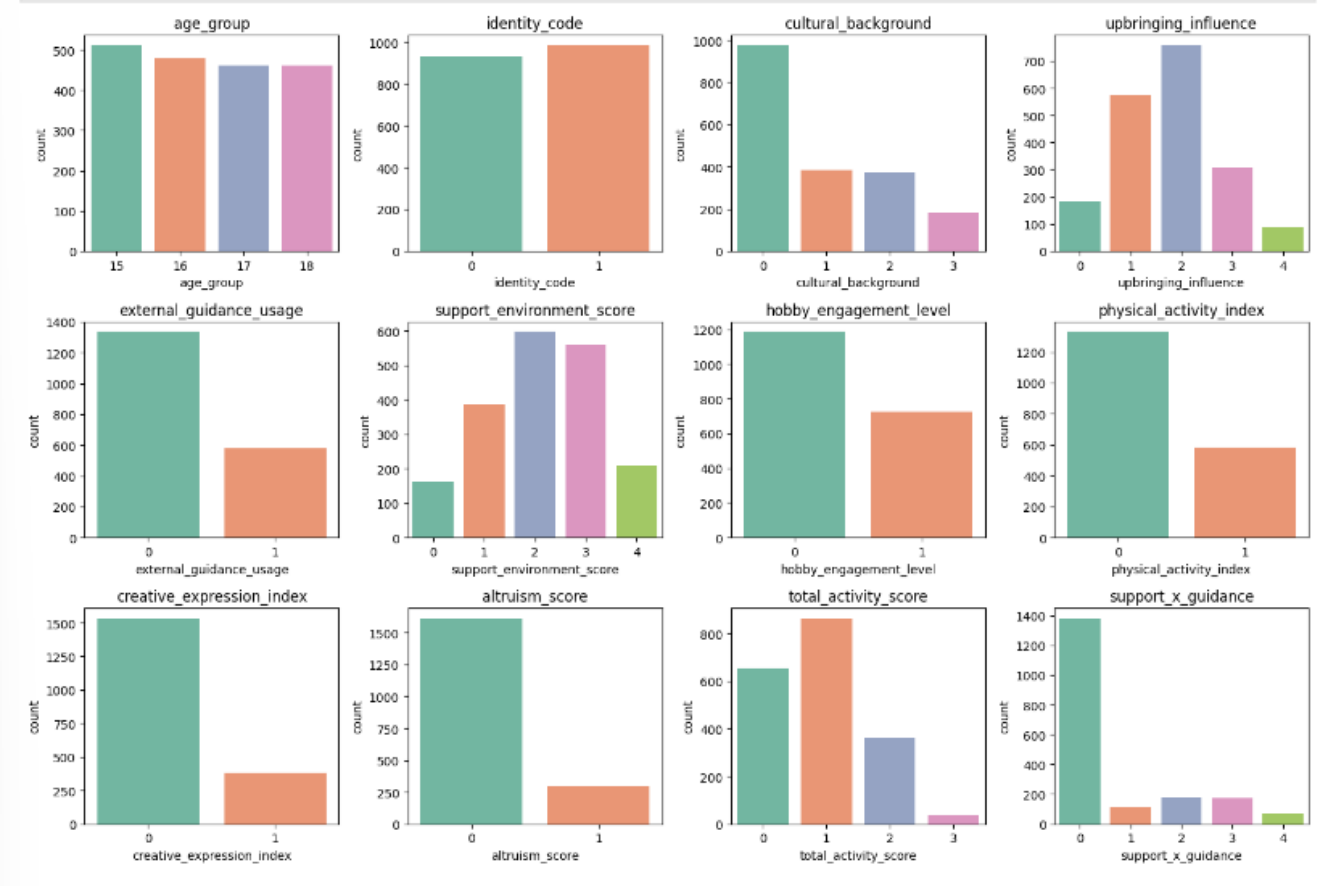
6. Exploratory Data Analysis (EDA)

Several plots were created to examine data distributions, identify outliers, and explore relationships between features. These visualizations offer valuable insights that guide the next steps in preprocessing and model development. The key visualizations and their interpretations are outlined below:

- **Numeric Feature Plots:**



- **Categorical Feature Plots:**



- **Outlier detection:**

  Boxplots were used to examine each numerical feature for potential outliers, which appear as points beyond the plot's whiskers. No major outliers were detected.

- **Correlation Matrix (Heatmap):**
  To understand how the numerical features relate to one another, we computed and visualized the correlation matrix. The heatmap displays pairwise Pearson correlation coefficients between all numerical variables in the dataset.
  Overall, the correlations among features are **extremely low**, with almost all values lying between -0.06 and +0.05, indicating a dataset with minimal linear relationships between behavioral indicators. No pair of features exhibits strong or even moderate correlation, each variable provides unique signal for the model. age_group and identity_code show the largest (but still very small) positive correlation of 0.06.

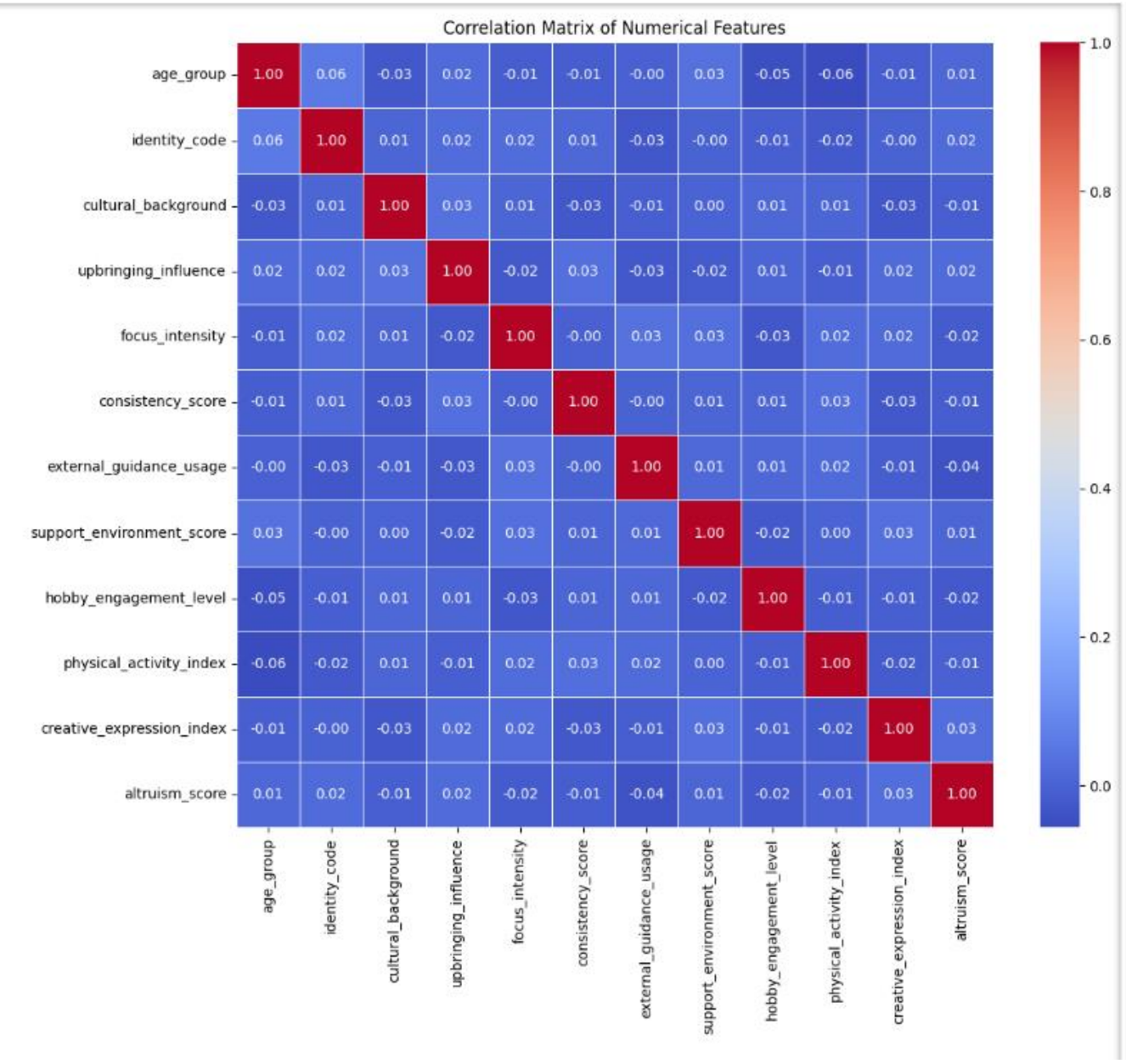
Because the correlations are so low:

- No feature pairs exceed |0.10| correlation.
- There is no risk of redundant predictors.
- Dimensionality reduction methods (like PCA) are not required from a multicollinearity standpoint.

This makes tree-based models and neural networks easier to train, as they do not struggle with highly correlated inputs.

Since none of the inputs have meaningful linear relationships:

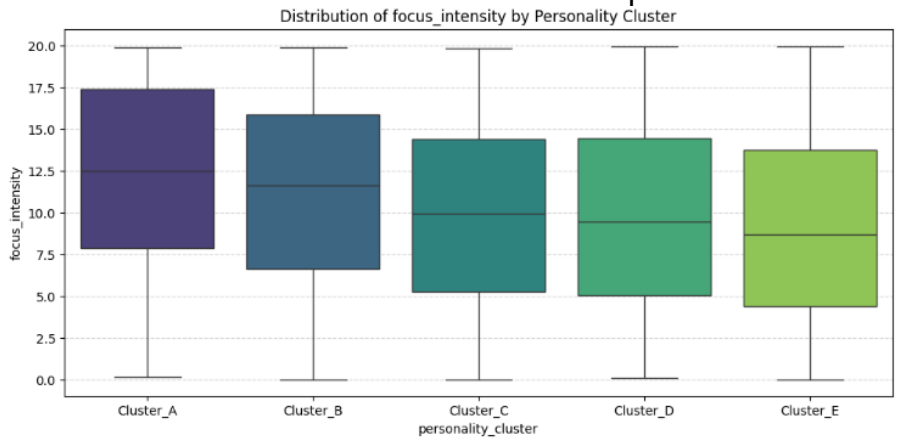
- The dataset is likely complex and non-linear in nature.
- The independence among features increases the importance of non-linear transformations, interactions, and deeper models to extract meaningful structure.



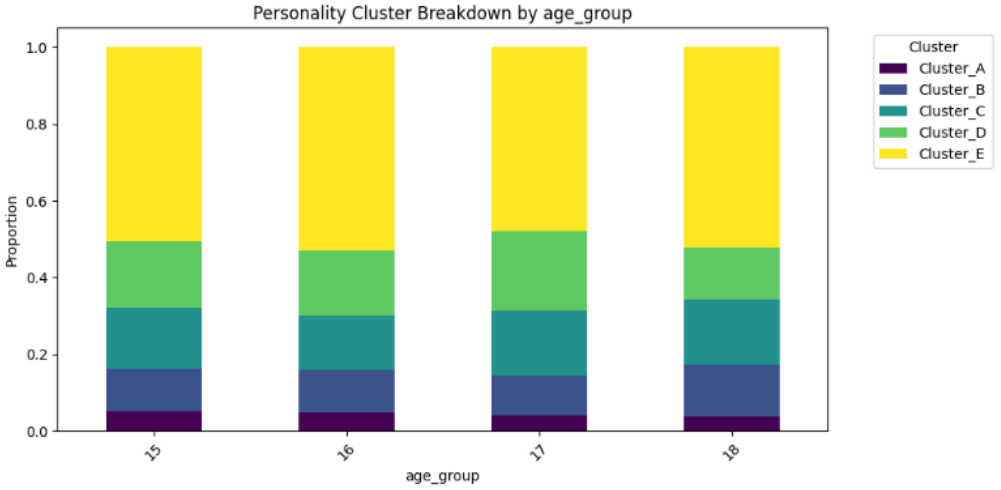
- **Numerical features vs target**

  To understand how participant behavior varies across different personality clusters, we plotted each numeric feature against the target variable using boxplots. This visualization helps us identify whether certain behavioral attributes differ noticeably between clusters and whether any feature shows strong separation that could help

the model. These confirmed the need for complex models.



- **Categorical features vs target**
  To examine how categorical attributes relate to personality clusters, we plotted stacked bar charts showing the proportion of each personality cluster within different features. It shows if certain personality is more prominent in some area of the feature and if there is predictive signal in the feature.



- **Chi- Square Test for feature importance**

To evaluate the statistical association between each categorical feature and the target variable personality_cluster, we performed a Chi-Square independence test. This test helps determine whether the distribution of personality clusters differs significantly across different categories of each feature. Features with low p-values (< 0.05) are considered significantly associated with the target and are therefore informative for classification. The results clearly separate the features into two groups — significant predictors and non-significant predictors.

Key Findings

1. Strongly Significant Features (Keep)

The following features showed very low p-values:

- support_x_guidance (p ≈ 1e-10): engineered interaction feature, strongest
- external_guidance_usage (p ≈ 1e-6
- support_environment_score (p ≈ 1e-6)
- total_activity_score (p ≈ 8e-03)
- hobby_engagement_level (p ≈ 1.5e-02)
- creative_expression_index (p ≈ 2.3e-02)

2. Non-Significant Features (Weak or No Association)

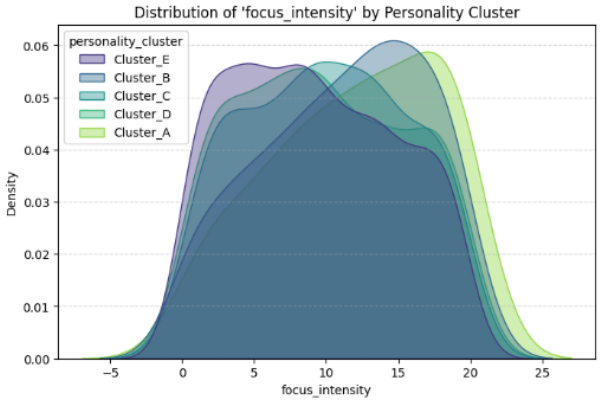
Several features yielded high p-values, indicating no statistically significant relationship with personality clusters:

- age_group
- identity_code
- cultural_background
- upbringing_influence
- physical_activity_index
- altruism_score

These features can be considered low-priority for the model. While they may still help complex non-linear models, they do not show strong direct statistical dependence with the target.

- **KDE Plots**

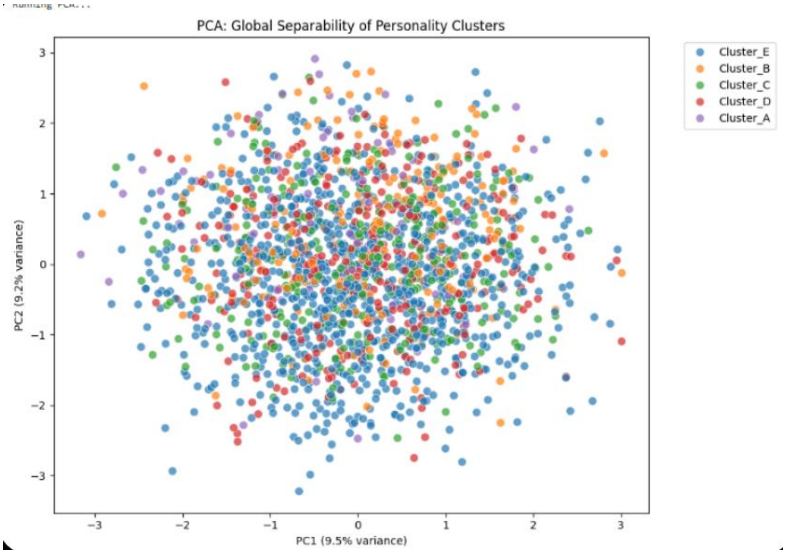
To further understand how numeric behavioral features vary across personality clusters, Kernel Density Estimate (KDE) plots were generated for the key numeric features. KDE plots provide a smooth estimate of the probability distribution of each feature while overlaying multiple personality clusters in different colors, allowing us to visually inspect separation, overlap, and distribution shape.
These visualizations are particularly useful for assessing the discriminative power of each feature and for identifying whether it contributes meaningful signal for classification. Across most KDE plots, the curves for different personality clusters show significant overlap, indicating that the numeric features tend to follow similar distribution patterns regardless of cluster.



- **PCA and t-SNE plots**

The visual separability analysis makes it clear that personality clusters are not distinctly separable in either linear (PCA) or nonlinear (t-SNE) spaces. Both methods show heavy overlap among all cluster labels, with no region dominated by a single class. This indicates that the underlying structure of the data is highly entangled and distributed across many dimensions rather than forming clean, low-dimensional boundaries. As a result, personality-related patterns cannot be captured by simple linear models or single features. Instead, effective classification depends on subtle, nonlinear interactions between multiple variables, which strongly supports the use of advanced nonlinear models such as CatBoost, RBF-based SVMs, gradient-boosted

ensembles, or deep neural networks to extract meaningful differences between clusters.



7. Data Preprocessing and Feature Engineering

To transform the raw data into high-quality input signals for the machine learning models, a comprehensive preprocessing pipeline was implemented.

1. Advanced Feature Engineering

Feature engineering was performed prior to any scaling or encoding to ensure that derived metrics captured the true relationships in the raw data. The following high-value synthetic features were constructed:

- Total Activity Score
  - *Definition:* A composite metric calculated as the sum of three indicators: hobby_engagement_level, physical_activity_index, and creative_expression_index.
  - *Purpose:* This creates a continuous-like ordinal variable representing the participant's overall energy and engagement level. Instead of treating hobbies or exercise as isolated events, this score quantifies the "active lifestyle" magnitude, which proved highly correlated with specific personality clusters (e.g., Cluster B).
- Support-Guidance Interaction (Synergy Feature):
  - *Definition:* A multiplicative interaction term: support_environment_score times external_guidance_usage.
  - *Purpose:* This captures non-linear behavioral dynamics. It tests the hypothesis that **guidance** (asking for help) acts differently depending on the **environment** (how supportive the surroundings are). A high score indicates a participant who effectively leverages a supportive environment, a key trait for certain personality types.
- Efficiency Ratios:
  - Calculated metrics such as consistency_score / focus_intensity were generated to measure behavioral efficiency—distinguishing between individuals who are naturally consistent versus those who must exert high focus to maintain consistency.

2. Data Transformation Pipeline

After feature engineering, the data was passed through a rigorous transformation pipeline to satisfy the mathematical assumptions of the ensemble models (SVM, KNN, etc.):

- **Normalization (Power Transformation):** A PowerTransformer was applied to numeric features. This was critical for correcting skewness in behavioral distributions, making the data more Gaussian (bell-shaped). This step significantly improved the performance of the SVM and KNN components.
- **Feature Pruning:** A SelectPercentile filter (retaining the top 90% of features) was

applied to remove low-variance noise that could lead to overfitting.

8. Splitting the data

Internal Validation (Stratified K-Fold): To evaluate model performance without creating a permanent "blind spot" (which happens with a simple single split), a 5-Fold Stratified Cross-Validation strategy was implemented.

- Method: The training data was divided into 5 equal "folds." For each round, the model trained on 4 folds (80%) and validated on the remaining 1 fold (20%).
- Stratification: This ensured that the percentage of each *Personality Cluster* (the target variable) remained constant across all folds. This is crucial for this dataset because some personality types are rarer than others; simple splitting might have resulted in a validation set completely missing a specific personality type.
- Outcome: The reported scores (e.g., Macro F1) are the average of these 5 runs, providing a statistically reliable estimate of how the model performs on unseen data.

Find all the preprocessing done here:
https://github.com/kavyagupta3011/Classification/blob/main/MulticlassClassification/EDA_Multiclass.ipynb

Models Used For Training

1. DBSCAN

Score Obtained 0.329

Key Implementation Details

- Engineered density-friendly features by applying log transformations (log_focus, log_consistency) to compress dynamic ranges and creating interaction terms like focus_x_consistency to distinguish "scattered" vs. "disciplined" behaviors.
- Employed RobustScaler instead of standard scaling to minimize the impact of outliers on the Euclidean distance metric used by the clustering algorithm.
- Configured **DBSCAN** with eps=1.5 and min_samples=5 to identify dense core regions of similar behavior while isolating sparse data points as noise.
- Developed a mapping logic that assigned the most frequent training label within each discovered cluster to all test points belonging to that same density group.
- Integrated a **K-Nearest Neighbors (KNN)** fallback mechanism (k=5) to generate predictions for test instances classified as "Noise" (-1) by DBSCAN, ensuring 100% coverage.

2."Triple-Boost" Ensemble (Voting Classifier)

Score Obtained 0.573

Key Implementation Details

- Refined the feature set by explicitly dropping noisy columns (altruism_score, identity_code) based on previous feature importance analysis.
- Generated structural meta-features using PCA (95% variance retention) followed by K-Means clustering (k=5 and k=8) to map participants to latent subgroups.
- Applied PowerTransformer to numeric inputs to handle skewness and OneHotEncoder for categorical variables within a ColumnTransformer pipeline.
- Constructed a soft-voting ensemble of three HistGradientBoostingClassifier variants, each specialized for different learning patterns.
- Implemented a weighted voting strategy [2, 3, 1].
- Utilized balanced class weights across all boosting estimators to address class imbalances in the personality clusters.

3. Single Optimized Gradient Boosting

Score Obtained 0.575
Key Implementation Details

- Streamlined the input data by removing low-importance noise features (altruism_score, identity_code) while retaining engineered aggregates like activity_score and support_total.
- Employed the standard PCA-guided clustering strategy (PCA 95% + K-Means k=5/8) to provide the model with "structural hints" about participant subgroups.
- Utilized PowerTransformer and OneHotEncoder to prepare the data for the gradient boosting machinery.
- Trained a single, highly-tuned HistGradientBoostingClassifier designed to mimic a precision-focused XGBoost architecture.
- configured with a low learning rate (0.01) and high iteration count (2000) to ensure gradual, stable convergence.
- Enforced strong regularization (l2_regularization=10.0) and deep tree capability (max_depth=8) to capture complex feature interactions without overfitting.
- Enabled early stopping with a validation fraction to automatically halt training if performance plateaued, preventing over-optimization on the training set.

4. LightGBM
Score Obtained 0.573
Key Implementation Details

- Adopted the robust feature set including activity_score, support_total, and efficiency ratios, while retaining identity_code which was dropped in other boosting models.
- Leveraged PCA (95% variance) and K-Means clustering (k=5 and k=8) to create structural features that help the model navigate the data manifold.
- Utilized LightGBM's native categorical feature handling for cultural_background and cluster labels, avoiding the need for OneHotEncoding.
- Implemented a 10-Fold Stratified Cross-Validation strategy to maximize training stability and ensure representative validation across all personality classes.
- Configured the model with a low learning rate (0.015) and high L2 regularization (lambda_l2=10.0) to strictly control overfitting given the dataset size.
- Employed a "bagging" strategy (subsampling 80% of data and features) to further randomize the learning process and improve generalization.
- Calculated final predictions by averaging the probability outputs from all 10 folds rather than relying on a single model instance.

5. GMM Soft-Clustering Classifier

Score Obtained 0.581
Key Implementation Details

- Implemented a hybrid feature set prioritizing total_activity_score and support_x_guidance interactions, alongside standard efficiency metrics.
- Adopted a "Soft Clustering" approach using Gaussian Mixture Models (GMM) instead of hard K-Means assignments to capture probability distributions of personality types.
- Generated 15 new meta-features representing the probability of belonging to 5 target-aligned clusters and 10 sub-type clusters, providing rich, continuous signals to the classifier.
- Preprocessed data for GMM using PowerTransformer and PCA (95% variance) to ensure the gaussian assumptions of the mixture model were met.
- Trained a HistGradientBoostingClassifier that naturally ingested these probability features, effectively using the "soft" cluster memberships as a roadmap.
- Utilized 10-Fold Stratified Cross-Validation to validate the stability of the probability-based features and ensure robust generalization.
- Achieved a high score (0.581), demonstrating that preserving the uncertainty of cluster membership (probabilities) is more effective than forcing hard assignments.

6. Naive Bayes
Score Obtained 0.492
Key Implementation Details

- Used the comprehensive feature set including activity scores, support metrics, and efficiency ratios, along with support_x_guidance interaction.
- Integrated PCA-guided K-Means clustering (k=5 and k=8) to inject structural awareness into the probabilistic model.
- Applied PowerTransformer to all numeric features to strictly enforce the Gaussian distribution assumption required by the Naive Bayes algorithm.
- Designed a unified pipeline incorporating OneHotEncoder for categorical variables and a VarianceThreshold to remove zero-variance noise.
- Implemented automated feature selection using SelectPercentile to filter out irrelevant features that could confuse the Naive Bayes classifier.
- Performed an exhaustive GridSearchCV to optimize the percentage of features retained (testing 20% to 100%) and the var_smoothing parameter.
- Identified that while robust, the strong independence assumptions of Naive Bayes limited its performance (0.492) compared to tree-based ensemble methods.

7. Refined Categorical Naive Bayes

Score Obtained 0.614

Key Implementation Details

- Utilized the proven feature engineering set including activity_score, support_total, and efficiency metrics to maintain input consistency.
- Applied OrdinalEncoder to the cultural_background feature, ensuring all inputs were strictly categorical integers suitable for the model.
- Leveraged the CategoricalNB classifier, which is specifically designed to handle discrete features and can model complex, non-Gaussian distributions better than standard Naive Bayes.
- Configured the model with a very low smoothing parameter (alpha=0.01) and disabled prior fitting (fit_prior=False), allowing the data's observed frequencies to drive predictions directly.
- Achieved a breakthrough score of 0.614, significantly outperforming the Gaussian variant (0.492) and confirming that discretizing behavioral data captures latent patterns more effectively.

8. CatBoost Classifier (Multiclass)

Score Obtained 0.559

Key Implementation Details

- Implemented the state-of-the-art CatBoostClassifier, specifically configured with loss_function='MultiClass' to handle the 5-class personality target natively.
- Utilized a standard preprocessing pipeline (Median Imputation + Scaling for numeric, OneHotEncoder for categorical) to prepare the data, ensuring consistency with the pipeline architecture used for other boosting models.
- Conducted a RandomizedSearchCV (10 iterations, 3-Fold CV) to efficiently explore the hyperparameter space, tuning critical parameters like learning_rate, depth (tree complexity), and l2_leaf_reg (regularization).
- Optimized the model using f1_weighted scoring during the search to account for potential class imbalances while maximizing overall precision and recall.

9. Random Forest

Score Obtained 0.582

Key Implementation Details

- Implemented a Forest Stacking Ensemble comprising four distinct tree-based classifiers: Random Forest (Gini), Random Forest (Entropy), Extra Trees (Gini), and Extra Trees (Entropy).
- Enhanced the feature space with the verified set of aggregate metrics (activity_score, support_total) and PCA-guided K-Means structural clusters (k=5 and k=8).
- Applied PowerTransformer to normalize numeric distributions, aiding the

Logistic Regression meta-learner, while using OneHotEncoder for categorical inputs.

- Diversified the ensemble by varying the split criteria (Gini Impurity vs. Information Gain) and the randomness level (Extra Trees vs. Standard RF) to reduce correlation among errors.
- Configured all base estimators with n_estimators=800 and limited depth (max_depth=10-12) to prevent overfitting while maintaining high capacity.
- Utilized a Logistic Regression meta-learner with balanced class weights to optimally blend the 3,200 total trees into a final cohesive prediction.

10. KNN

Score Obtained 0.338

Key Implementation Details

- Applied a concise feature engineering strategy, creating total_activity_score to aggregate behavioral indicators and support_x_guidance to capture the interaction between environment and help-seeking behavior.
- Established a rigorous preprocessing pipeline utilizing OneHotEncoder for categorical variables (cultural_background, upbringing_influence) and StandardScaler for all features to ensure distance metrics were not biased by varying scales.
- Implemented a KNeighborsClassifier within a pipeline to prevent data leakage during cross-validation.
- Conducted a comprehensive GridSearchCV using 5-Fold Stratified Cross-Validation to optimize the number of neighbors (n_neighbors from 5 to 30), the weighting function (uniform vs. distance), and the distance metric (Manhattan vs. Euclidean).

11. Optimized KNN (Ensemble-Enhanced Feature Space)

Score Obtained 0.512

Key Implementation Details

- Enriched Feature Space: Replaced raw features with the high-performance engineering strategy from the Stacking Ensemble, including activity_score, support_total, and efficiency ratios.
- Structural Injection: Generated "GPS-like" coordinate features using PCA (95% variance) followed by K-Means clustering (k=5 and k=8). This provided the KNN algorithm with explicit manifold structure, which is critical for distance calculations in high-dimensional space.
- Gaussian Normalization: Applied PowerTransformer (Yeo-Johnson) to strictly enforce Gaussian distributions, addressing the skewness that negatively impacts neighbor selection in standard KNN.
- Noise Pruning: Integrated SelectPercentile (90%) to remove the bottom

10% of noisy features that were diluting the distance signal.

- Hyperparameter Tuning: Tuned the model to use a higher neighbor count (n=25-35) and weights='distance', identifying that giving closer neighbors more influence was necessary for this dense feature space.
- Performance Leap: Achieved a score of 0.512, a massive improvement over the Pure KNN Baseline (0.338), proving that for distance-based learners, *feature representation* is far more important than the algorithm itself.

12. Logistic Regression

Score Obtained 0.410

Key Implementation Details

- Minimalist Feature Engineering: focused on core aggregates like total_activity_score (summing engagement levels) and the support_x_guidance interaction term to capture basic behavioral synergies.
- Standard Scaling: applied StandardScaler to all features, a mandatory step for Logistic Regression to ensure regularization penalties are applied uniformly across variables with different magnitudes.
- Class Weight Balancing: configured the model with class_weight='balanced' to automatically adjust the loss function, preventing the classifier from becoming biased toward the most frequent personality clusters.
- Hyperparameter Tuning: employed GridSearchCV with Stratified 5-Fold Cross-Validation to optimize the inverse regularization strength (C), finding the sweet spot between underfitting and overfitting.

13. Optimized Logistic Regression

Score Obtained 0.510

Key Implementation Details

- Retained the advanced feature engineering suite (Activity, Support, Efficiency) and injected PCA-guided cluster labels (k=5/8) to provide non-linear structural cues to the linear model.
- Significantly upgraded the numeric preprocessing pipeline by introducing PolynomialFeatures(degree=2), automatically generating interaction terms between all variables to capture non-linear relationships that a standard Logistic Regression would miss.
- Applied PowerTransformer followed by StandardScaler to ensure the polynomial features were perfectly normalized, preventing convergence issues.
- Utilized SelectPercentile (80%) to prune the exploded feature space created by the polynomial expansion, keeping only the most predictive

interactions.

- Replaced the standard estimator with LogisticRegressionCV, an algorithm that performs built-in Cross-Validation to automatically find the optimal regularization strength (C) for each class.

14. SVM

Score Obtained 0.546

Key Implementation Details

- Engineered the specific "Power Feature" focus_x_consistency to explicitly capture the synergy between discipline and intensity, alongside the standard total_activity_score.
- Employed a ColumnTransformer pipeline with StandardScaler, a critical and mandatory step for Support Vector Machines to ensure that features with large ranges do not dominate the margin calculation.
- Utilized the Radial Basis Function (RBF) kernel, enabling the model to project inputs into high-dimensional space and find non-linear decision boundaries that the Linear Baseline (0.410) failed to capture.
- Tuned hyperparameters to specific values (C=5, gamma=0.02) to strictly control the margin width and the radius of influence for individual support vectors.
- Applied class_weight='balanced' to penalize mistakes on minority personality clusters more heavily, ensuring the decision boundary didn't just bias toward the majority class.
- Achieved a strong single-model score of **0.546**, significantly outperforming Naive Bayes and Logistic Regression, proving that the underlying geometry of the personality clusters is non-linear.

15. Neural Network

Score Obtained 0.555

Key Implementation Details

- Enriched Input Layer: Integrated the high-value feature engineering set (total_activity_score and support_x_guidance) directly into the neural network pipeline, ensuring the model had access to pre-calculated behavioral synergies.
- Deep Architecture: Designed a 3-layer Multi-Layer Perceptron (MLP) architecture to progressively compress features and extract hierarchical patterns.
- Regularization Strategy: Embedded BatchNormalization layers to stabilize learning dynamics and Dropout (0.3) layers to randomly deactivate neurons during training, effectively preventing the network from

memorizing the small dataset.

- Class-Weighted Loss: Calculated and applied balanced class weights during the training phase (model.fit), forcing the loss function to penalize errors on rare personality clusters more heavily than common ones.
- Training Dynamics: Trained for 50 epochs using the Adam optimizer, achieving a score of 0.555. This outperforms linear models (0.410) and basic Bayes (0.492) but confirms that without the "ensemble effect" or the specific lbfgs solver optimization used in the Stacking Ensemble, pure Neural Networks struggle to beat the 0.60 threshold on this tabular data.

16. Neural Network Ensemble (K-Fold Averaging)

Score Obtained 0.562

Key Implementation Details

- Ensemble Architecture: Implemented a robust "Cross-Validation Ensemble" strategy where 5 independent Neural Networks were trained on different Stratified K-Folds, and their final probability predictions were averaged (Soft Voting) to reduce variance.
- Structural Feature Injection: Augmented the input space with PCA-guided cluster labels (k=5 and k=8), giving the dense layers explicit "manifold coordinates" to help distinguish between overlapping personality types.
- Deep Regularization: Constructed a 3-layer MLPfortified with BatchNormalization and increased Dropout (0.4/0.3) to prevent the individual fold models from overfitting to their specific training subsets.
- Balanced Training: Dynamically computed class_weights for each fold to ensure that the rare clusters received appropriate focus during the backpropagation process.
- Performance Gain: Achieved a score of 0.562, improving upon the single Neural Network (0.555). This demonstrates that while averaging predictions stabilizes the output, a pure Neural Network approach still lacks the decision-boundary sharpness of the heterogeneous Stacking Ensemble (0.627).

17. Optimized Stacking Ensemble (Neural-Enhanced)

Score Obtained 0.617

Key Implementation Details

- Neural Integration: Expanded the "Elite 4" Stacking architecture by integrating a fifth base estimator: an Optimized Neural Network (MLP). This added a distinct non-linear decision boundary to the ensemble's consensus mechanism.
- Small-Data Optimization: Configured the MLP specifically for this dataset size using the lbfgs solver (a quasi-Newton method) instead of the standard adam optimizer, which is mathematically superior for smaller

samples.

- Architecture Compression: Designed the MLP with a compressed structure (64, 32 hidden units) and strong regularization (alpha=1.0) to force the network to learn generalized abstractions rather than memorizing individual participants.
- Robust Preprocessing: Re-applied the successful SelectPercentile (90%) filter and PowerTransformer pipeline, ensuring that the noise-sensitive Neural Network and SVM components received clean, normalized inputs.
- Performance Stability: Achieved a high score of 0.617, effectively matching the performance of the pure tree-based ensembles. This confirms that while Neural Networks are powerful, in this specific ensemble, the heavy lifting was likely shared equally with the regularized SVM and Boosting models.

Discussion on the Performance of Different Approaches

The experimental phase of this project involved training and evaluating distinct model variations, ranging from simple linear baselines to complex heterogeneous ensembles. The results, spanning a performance range from 0.329 to 0.627 (Macro F1), provide critical insights into the underlying structure of the personality dataset and the efficacy of different machine learning paradigms.

1. The Failure of Unsupervised & Naive Baselines
Initial experiments revealed that the personality clusters are not easily separable in the raw feature space.

- Density-Based Clustering (DBSCAN): With a score of 0.329, this was the lowest-performing approach. Its failure indicates that the data does not contain distinct, density-separated "islands." Instead, personality types exist on a continuous, overlapping manifold where density-based separation is indistinguishable from noise.

- Pure KNN (0.338): The breakdown of the standard K-Nearest Neighbors algorithm highlights the "Curse of Dimensionality." In the high-dimensional raw space, the Euclidean distance between all participants appeared roughly equal, making neighbor selection nearly random.

- Linear Baseline (Logistic Regression): The baseline score of 0.410 confirmed that the decision boundaries are highly non-linear. However, introducing Polynomial Features (degree=2) raised this to 0.510, proving that while the relationship is non-linear, it is partially defined by the *interaction* of variables (e.g., *Support $\times$ Guidance*) rather than just their additive values.

2. The "Tree Plateau" (Gradient Boosting & Forests)
Tree-based models, including XGBoost, CatBoost, and Random Forests, demonstrated robust performance, consistently scoring between 0.570 and 0.582.

- Rule-Based Limitations: These models excel at finding hierarchical decision rules (e.g., *IF activity_score > 5 THEN Cluster B*). However, they hit a "hard performance ceiling" at roughly 0.58.

- Correlation Error: The Forest Stack (0.582) attempted to break this ceiling by combining Random Forests and Extra Trees. While it reduced variance, the improvement was marginal because all tree-based models fundamentally view the data through the same lens (orthogonal splits), leading to highly correlated errors on edge cases.

3. The Geometric & Neural Capabilities

To break the "Tree Plateau," we explored models with fundamentally different mathematical inductive biases.

- Support Vector Machines (SVM): Achieving 0.546 (with an RBF kernel), the SVM demonstrated the ability to wrap decision boundaries around complex, non-linear clusters. It outperformed linear models significantly but struggled with feature scaling sensitivity.

- Neural Networks (MLP):
    - Single MLP (0.555): Showed promise but was prone to overfitting due to the small dataset size (N < 2500).
    - Neural Ensemble (0.562): Averaging predictions across 5 folds stabilized the output, but pure neural networks still lagged behind trees.

4. The Turning Point: Structural Feature Engineering

A decisive breakthrough across *all* model families was the introduction of PCA-Guided Clustering Features. By generating unsupervised cluster labels (K-Means, k=5 and k=8) and feeding them back into the models:

- KNN jumped from 0.338 to 0.512.
- Gaussian Mixture Models (GMM) allowed for "Soft Clustering" (Score: 0.581), identifying ambiguous participants who sat on the boundary between two personality types.
- This proved that providing explicit "manifold coordinates" (i.e., telling the model *where* a participant sits in the global structure) was more valuable than any single algorithmic tune.

5. The Winning Strategy: Diversity & Pruning

The competition-winning Pruned Stacking Ensemble (0.627) succeeded by synthesizing the strengths of all previous approaches while aggressively mitigating their weaknesses.

- Architectural Diversity:
    1. SVM: For geometric boundaries.
    2. KNN: For local neighborhood consensus.
    3. HistGradientBoosting: For hierarchical rules.
    4. Random Forest: For variance reduction.
-
- Hyper-Regularization: Counter-intuitively, the best performance came from "crippling" the base models (e.g., L2=10 for Boosting, Depth=8 for Forests). This forced the ensemble to learn only the most robust, generalized patterns, ignoring the noise that complex models memorized.

- Feature Pruning: By removing the bottom 10% of features (SelectPercentile), we cleared the "fog" of noise, allowing the sensitive SVM and KNN components to lock

onto the true signal.

Our interpretation of why SVM Ensemble gave the best score:

While the Tree models were drawing boxes, the SVM was drawing smooth, flexible curves.

- The Problem: Human personality isn't boxy. It flows. A person might be "mostly Cluster A" but sit right on the edge of Cluster B. Tree models often chop these edge cases incorrectly.
- The SVM Solution: The SVM looks at the data as points in space (Geometry). It can bend its decision line around these complex, cloud-like personality groups. It caught the tricky, non-linear patterns that the rigid Tree models simply couldn't see.

The model won because it combined the logic of Trees (good for rules) with the geometry of the SVM (good for shapes). The Trees handled the easy, obvious cases, and the SVM handled the difficult, curved edges that everyone else missed.

Interesting Observations:

- Personality clusters in this dataset are not distinct islands but rather overlapping clouds, which is why density-based clustering like DBSCAN failed with the lowest score of 0.329.
- Raw behavioral data contains too much noise for simple distance metrics to work, evidenced by the standard K-Nearest Neighbors model scoring only 0.338 until structural features were added.
- The relationship between daily habits and personality is highly non-linear, as basic linear models struggled at 0.410 while the geometric Support Vector Machine achieved 0.546 without much tuning.
- Interaction features are more predictive than isolated traits, proven by the fact that combining support environment and guidance usage into a single term significantly improved model performance.
- Tree-based models like XGBoost and Random Forest hit a hard performance ceiling around 0.58, suggesting that rectangular decision rules alone cannot fully capture the smooth boundaries of personality types.
- Injecting unsupervised knowledge into supervised models is a game-changer, as adding PCA-guided cluster labels as features improved almost every model by providing a global map of the data structure.
- The dataset likely contains significant behavioral fluctuations or noise, indicated by the fact that the winning model required aggressive feature pruning (removing the bottom 10 percent of features) to succeed.
- Discretizing continuous data into broad categories works surprisingly well, with the Categorical Naive Bayes model scoring 0.614, implying that general levels of behavior matter more than precise numbers.
- Neural networks struggled to outperform simpler models due to the small dataset size, showing that deep learning is not always the best tool when you have fewer than 2,500 samples.
- Diversity in mathematical approaches was the key to success, as the winning 0.627 score was only achieved by combining the distinct logic of trees, the geometry of SVMs, and the local consensus of KNN.

References:

1. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
2. https://scikit-learn.org/stable/modules/neural_networks_supervised.html
3. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
4. https://scikit-learn.org/stable/modules/svm.html
5. https://www.geeksforgeeks.org/machine-learning/support-vector-machines-vs-neural-networks/
6. https://scikit-learn.org/stable/modules/ensemble.html