International Institute of Information Technology Bangalore

ML Project Report: Multidimensional Personality Cluster Prediction

(Checkpoint 2)

Team Name: Predictify

Team Members:

1. Kartikeya Dimri - IMT2023126
2. Ayush Mishra - IMT2023129
3. Harsh Sinha - IMT2023571

Github Repo Link:
https://github.com/kartikeya-dimri/Personality-Cluster-Prediction

November 27, 2025

Contents

1 Task

The primary objective of this project is to construct a robust machine learning model capable of accurately classifying individuals into distinct "personality clusters." This is a multiclass classification problem where the inputs are behavioral and lifestyle attributes—ranging from daily routines to social engagement—and the target is the specific behavioral segment to which an individual belongs.

Given that certain personality types may be more prevalent than others within the population, the dataset is likely to exhibit class imbalance. Consequently, the performance of the models will be evaluated using the **Macro F1 Score**. This metric calculates the F1 score for each class independently and then takes the average, ensuring that the model's performance on minority classes is weighted equally to that of majority classes.

2 Background

Human behavior is not random; it is the result of a complex interaction of environment, experiences, habits, and personal interests. Understanding these behavioral patterns is crucial for domains ranging from psychological profiling to personalized recommendation systems.

In this competition, we are tasked with analyzing these underlying patterns. By examining how individuals allocate their time, their consistency in routines, and their engagement with their environment, we aim to algorithmically recreate the segmentation analysis that defines these personality clusters. This project serves as a practical application of predictive modeling on high-dimensional, mixed-type data (numerical and categorical) to solve a sociological classification problem.

3 Dataset Description

The dataset consists of anonymized behavioral and lifestyle indicators collected from a diverse group of participants. Each record represents a unique individual and contains various attributes related to their daily habits, upbringing, and social interactions.

The target variable is the `personality_cluster`, a label representing a specific behavioral tendency derived from prior segmentation analysis. It is important to note that these clusters are value-neutral; they simply represent different archetypes of behavior rather than a judgment of character.

3.1 Feature Descriptions

The dataset comprises the following features:

Column Name	Description
participant_id	Unique ID assigned to each participant.
record_code	Internal reference code (irrelevant to prediction).
age_group	Categorical indicator of the participant's age bracket.
identity_code	Encoded personal identity category.
cultural_background	Grouping based on regional or cultural background.
upbringing_influence	Score indicating the influence of the participant's formative environment.
focus_intensity	Measure of time and effort dedicated to focused tasks.
consistency_score	Metric reflecting reliability and stability in daily routines.
external_guidance_usage	Frequency of using mentoring or support resources.
support_environment_score	Rating of the perceived supportiveness of the participant's environment.
hobby_engagement_level	Level of engagement in leisure or personal interest activities.
physical_activity_index	Index representing involvement in physical activities.
creative_expression_index	Score for participation in artistic or expressive endeavors.
altruism_score	Tendency toward volunteering or helping behaviors.
personality_cluster	**Target Variable:** The assigned personality segment label.

Table 1: Dataset Column Descriptions

4 Exploratory Data Analysis (EDA)

In this phase, we performed an in-depth analysis to understand the data structure, detect patterns, and assess feature relationships with the target variable. We utilized `pandas` and `numpy` for data manipulation, and `seaborn` and `matplotlib` for visualizing distributions.

For detailed visualizations and plots, please refer to the EDA plots repository.

4.1 Target Variable Analysis (Class Imbalance)

The distribution of the target variable `personality_cluster` reveals a significant class imbalance:

- **Cluster E** is the dominant majority class, containing nearly 1,000 samples.
- **Cluster A** is a significant minority class, with very few samples compared to the rest.
- Clusters B, C, and D are moderately balanced, falling in the range of 200–350 samples.

Implication: This imbalance confirms the necessity of using the **Macro F1 Score** for evaluation, as accuracy alone would be biased toward Cluster E.

4.2 Correlation Analysis

We computed the correlation matrix for numerical features to identify multicollinearity.

- **Observation:** The correlation heatmap shows that the features are largely statistically independent. The correlation coefficients range strictly between -0.04 and +0.03 for almost all feature pairs.
- **Implication:** There is negligible redundancy among features. While this is beneficial for linear models (avoiding multicollinearity), it suggests that dimensionality reduction techniques like PCA may not be effective, as there is little shared variance to compress.

4.3 Feature-Specific Insights

By analyzing boxplots and distributions, we identified which features contribute most to cluster separation:

High Separability Features

- **Consistency Score:** This appears to be one of the strongest predictors. Cluster E exhibits a high median consistency (≈ 21), whereas Cluster A shows a very low median (≈ 3).
- **Focus Intensity:** This feature separates Cluster B and A (high median $\approx 12-13$) from Cluster E and D (lower median $\approx 8-9$).

Low Separability / Binary-Like Features

Several features, including `physical_activity_index`, `hobby_engagement_level`, and `external_guidance_usage`, display "saturated" distributions. The boxplots for these variables are identical across clusters, often maxing out at 1.0. This suggests these are likely binary or ordinal variables where the majority of participants scored the maximum value, offering less individual granular separation.

Sparse Features

Features such as `altruism_score` and `creative_expression_index` appear extremely compressed, mostly flatlining at 0 or 1 with outlier points. This indicates these are sparse features or highly skewed variables that might require specific transformation or tree-based models to handle effectively.

4.4 Demographic Patterns

- **Age Group:** The dataset covers age groups 15 through 18. Cluster E is the dominant personality type across all ages.

- **Cultural Background & Identity:** While Cluster E dominates all sub-categories, the relative proportions of Clusters B and C vary slightly across different cultural backgrounds (0–3), suggesting some interaction effects between demographics and personality type.

5 Preprocessing Strategies

Following the insights gained from the Exploratory Data Analysis (EDA), several preprocessing pipelines were developed and tested to prepare the data for various modeling approaches. Each pipeline focused on different feature selection, encoding, and imputation strategies. The goal was to evaluate how these different preprocessing choices impact model performance, particularly given the class imbalance and non-linearity observed.

Below is a summary of the key steps performed in each preprocessing script:

5.1 preprocessing_1.py (Baseline Full Pipeline)

This script represented the initial comprehensive preprocessing approach, establishing a baseline for model performance.

- **Target Handling:** Applied Label Encoding to the target variable `personality_cluster`, mapping the categorical labels (Cluster A-E) to integers (0-4).
- **Imputation:** Used median imputation for continuous numerical features (e.g., `consistency_score`) and mode imputation for categorical features to handle missing values.
- **Encoding:** Applied One-Hot Encoding (OHE) to nominal categorical features such as `cultural_background` and `age_group`.
- **Scaling:** Applied StandardScaler to all numerical features to ensure zero mean and unit variance, which is critical for Neural Networks and SVMs.
- **Output:** Saved processed data as CSV files (`train_preprocessing_1.csv`).

5.2 preprocessing_1_modified.py (Outlier Removal)

This script was a variation of the baseline pipeline designed to test the impact of noisy data points.

- **Outlier Detection:** Implemented the Interquartile Range (IQR) method on continuous features like `consistency_score` and `focus_intensity`.
- **Data Cleaning:** Removed rows where features fell outside the range $[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$.
- **Impact:** This process significantly reduced the training dataset size (from ≈454 KB to ≈256 KB), removing extreme behavior profiles that might confuse the model boundaries.
- **Remaining Steps:** Imputation, Encoding, and Scaling were identical to the baseline pipeline.
- **Output:** Saved processed data as CSV files (`train_preprocessing_1_modified.csv`).

5.3 preprocessing_2.py (Tree-Based Optimization)

This pipeline was designed specifically to test robustness for tree-based models (Random Forest, XGBoost), which do not necessarily require standard scaling.

- **Feature Engineering:** Treated binary-like features (e.g., `physical_activity_index`, `hobby_engagement_level`) strictly as categoricals rather than continuous variables.
- **Encoding:** Used Ordinal Encoding instead of One-Hot Encoding for certain high-cardinality features to reduce dimensionality.
- **Scaling:** MinMax Scaling was tested here instead of StandardScaler to preserve the original distribution shape.
- **Output:** Saved processed data as CSV files (`train_preprocessing_2.csv`).

5.4 preprocessing_3.py (Feature Selection)

Based on the EDA correlation analysis (which showed low correlation among features), this script tested a reduced feature set to prevent overfitting.

- **Feature Selection:** Dropped features with extremely low variance or those identified in EDA as "saturated" (e.g., `altruism_score` which was mostly static).
- **Target Handling:** Same as baseline (Label Encoding).
- **Scaling & Imputation:** Applied robust scaling to handle any remaining outliers without removing data points.
- **Output:** Saved processed data as CSV files (`train_preprocessing_3.csv`).

6 Model Training

Based on the various preprocessing pipelines developed, a comprehensive range of classification models were trained and evaluated. The primary goal was to compare the performance of different algorithms—ranging from simple linear baselines to complex deep learning architectures—and identify the impact of feature selection and preprocessing choices on the Macro F1 Score. Cross-validation and hyperparameter tuning were key components of this process.

6.1 Primary Models: NN, SVM, and Logistic Regression

These three families of models were the primary focus of our optimization efforts, representing Linear, Geometric, and Deep Learning approaches.

6.1.1 Neural Networks (Deep Learning)

Explanation: We designed Multi-Layer Perceptrons (MLP) capable of capturing complex, non-linear interactions between personality traits. Unlike simpler models, Neural Networks can learn high-level abstractions from the input features. We utilized techniques like Early Stopping to prevent overfitting and Ensemble methods to average out

errors.
Files:

- `neural_network_1.py` (Best Performer)
- `neural_network_1_early_stopping.py`
- `neural_network_1_ensemble.py`
- `neural_network_1_modified.py`
- `neural_network_2.py`
- `neural_network_3.py`
- `neural_network_a.py`

6.1.2 Support Vector Machines (SVM)

Explanation: SVMs work by finding the optimal hyperplane that separates the data points of different classes with the maximum margin. We employed kernel tricks (like RBF) to project the data into higher dimensions, making non-linearly separable personality clusters separable.
Files:

- `SVM_1.py`
- `SVM_2.py`
- `SVM_3.py`

6.1.3 Logistic Regression

Explanation: This served as our baseline linear classifier. It estimates the probability of a sample belonging to a specific personality cluster using the logistic function. While fast and interpretable, its performance helps determine if the data boundaries are linear.
Files:

- `logistic_regression_1.py`
- `logistic_regression_2.py`

6.2 Other Models

To ensure a robust comparison, we trained a variety of other algorithms to benchmark against our primary models.

6.2.1 Naive Bayes Variants

Explanation: These probabilistic classifiers are based on Bayes' Theorem with the "naive" assumption of feature independence. We tested three variants to handle different data distributions (Gaussian for continuous, Multinomial/Bernoulli for counts/binary).
Files:

- `bernoulli_naive_bayes_1.py`, `bernoulli_naive_bayes_2.py`
- `gaussian_naive_bayes_1.py`, `gaussian_naive_bayes_2.py`
- `multinomial_naive_bayes_1.py`, `multinomial_naive_bayes_2.py`

6.2.2 K-Nearest Neighbors (KNN)

Explanation: A non-parametric method that classifies a sample based on the majority class of its 'k' nearest neighbors in the feature space.
Files:

- `knn_classifier_1.py`
- `knn_classifier_2.py`

6.2.3 Decision Trees

Explanation: A tree-structured classifier where internal nodes represent tests on attributes and leaf nodes represent class labels. Prone to overfitting but highly interpretable.
Files:

- `decision_tree_1.py`
- `decision_tree_2.py`

6.2.4 Ensemble Methods (Forests & Boosting)

Explanation: These methods combine multiple weak learners (typically decision trees) to create a strong predictor. We tested **Bagging** (Random Forest) to reduce variance and **Boosting** (Gradient Boosting, XGBoost, LightGBM, AdaBoost) to reduce bias and correct errors sequentially.
Files:

- **Random Forest:** `random_forest_1.py`, `random_forest_2.py`
- **AdaBoost:** `adaboost_1.py`, `adaboost_2.py`
- **Gradient Boosting:** `gradient_boosting_classifier_1.py` to `...3.py`
- **XGBoost:** `xg_boost_1.py`, `xg_boost_2.py`, `xg_boost_3.py`
- **LightGBM:** `lightgbm_classifier_1.py`

7 Observations

- **Linearity vs. Non-Linearity:** A definitive trend emerged when comparing model architectures. **Logistic Regression** (0.477) performed significantly worse than non-linear models like **Neural Networks** (0.643). This strongly suggests that the decision boundaries between personality clusters are highly complex and non-linear. A simple linear combination of behavioral traits is insufficient to classify an individual; rather, it is the *interaction* between traits (captured by the hidden layers of a Neural Network) that defines the cluster.

- **Geometric vs. Axis-Aligned Splits: SVMs** (0.586) outperformed tree-based ensembles like **Random Forest** (0.564) and **XGBoost** (0.567). This is a noteworthy observation. Tree-based models create orthogonal (axis-aligned) splits. The superior performance of SVMs suggests that when projected into a higher-dimensional space (via the RBF kernel), the personality clusters become more separable than they are via simple rectangular cuts.

- **Signal in Independence:** As noted in the EDA, the feature correlation matrix showed near-zero correlation among features. This implies that each feature provides unique, non-redundant information. Consequently, models that can effectively aggregate high-dimensional, independent signals (like Neural Networks) held a distinct advantage over models that rely on feature importance hierarchies (like Random Forests), where correlated features often split the vote.

- **Impact of Outlier Handling:** We tested aggressive outlier removal in preprocessing_1_modified.py, which significantly reduced the dataset size. However, given the severe class imbalance (Cluster A is a minority), aggressive outlier removal likely purged valuable minority class examples that looked like "anomalies" compared to the dominant Cluster E. The robust performance of models on the fuller datasets suggests that "outliers" in this context were actually valid edge cases essential for defining the minority clusters.

- **Class Imbalance Macro F1:** The dominance of Cluster E (Age 15-18 majority) posed a risk of models biasing toward the majority class. Simple accuracy maximization would fail here. The **Macro F1** metric was crucial; it penalized models that ignored Cluster A, forcing the Neural Network to learn the subtle patterns distinguishing the minority groups, contributing to its top leaderboard score.

- **Model Comparison Summary:** Among all tested algorithms, the **Neural Network** trained on `preprocessing_1.py` (Baseline Full Pipeline) consistently yielded the best generalization on unseen data. While Gradient Boosting and Random Forests were competitive, they seemed to hit a "performance ceiling" around 0.56, likely due to their inability to fully capture the smooth, non-linear transitions between personality archetypes that the Neural Network could model.

8 Results

The best-performing model identified during the experimentation phase was the **Neural Network**, trained on the baseline preprocessing pipeline (`preprocessing_1.py`). This

model demonstrated superior ability to capture the non-linear relationships inherent in personality clustering, achieving a Macro F1 Score of **0.643** on the Kaggle Public Leaderboard.

The table below summarizes the best performance achieved by each primary model architecture:

Model Type	**Best Script / File**	**Kaggle Score (Macro F1)**
Neural Network	`neural_network_1.csv`	**0.643**
SVM	`SVM_3.csv`	0.586
XGBoost	`xg_boost_3.csv`	0.567
Random Forest	`random_forest_2.csv`	0.564
Logistic Regression	`logistic_regression_2.csv`	0.477

Table 2: Best Kaggle Public Leaderboard Scores by Model

Key Takeaway: The significant performance gap between the Neural Network (0.643) and the linear baseline (0.477) confirms that the underlying data structure is complex and non-linear. While tree-based ensembles (XGBoost, Random Forest) performed respectably, they could not match the generalization capability of the deep learning approach.

9 Interpretation: Why the Neural Network Model with Preprocessing-1 Performed Best

One of the most defining outcomes of our experiments was the superior performance of the **Neural Network** trained on the baseline dataset (`preprocessing_1.py`), achieving a Macro F1 Score of **0.643**. While tree-based ensembles like XGBoost are often considered the "gold standard" for tabular data, the nature of this specific dataset favored the Deep Learning approach.

We attribute this result to three primary factors:

9.1 Complex Feature Interactions (Non-Linearity)

Personality is rarely defined by a single threshold on a single trait (e.g., "If Consistency > 5, then Cluster B"). Instead, it is defined by the complex interplay between multiple traits (e.g., "High Consistency *combined with* Low Social Engagement").

- **Limit of Trees:** Decision Trees and Random Forests function by making orthogonal, axis-aligned cuts ($x > a$, $y < b$). They struggle to model smooth, curved decision boundaries or complex "XOR-like" interactions efficiently without growing extremely deep and overfitting.
- **Strength of NNs:** The hidden layers of the Neural Network act as feature extractors, learning non-linear combinations of the input variables. The significant gap between Logistic Regression (0.477) and the Neural Network (0.643) quantifies exactly how much "signal" resides in these non-linear interactions.

9.2 The "Manifold" Structure of Personality Clusters

The success of SVMs (Rank 2) over Random Forests (Rank 4) provides a clue to the data's geometry. SVMs with RBF kernels project data into higher dimensions to find separating hyperplanes. This suggests that the personality clusters exist as complex "blobs" or manifolds in the high-dimensional feature space, rather than hyper-rectangles. Neural Networks are universal function approximators capable of molding their decision boundary to fit these complex manifolds, whereas Random Forests attempt to approximate these curves using a series of jagged, rectangular steps.

9.3 Signal vs. Noise: The Importance of "Outliers"

A critical observation was that `preprocessing_1_modified.py`, which aggressively removed outliers using IQR, resulted in *worse* performance than the full dataset in `preprocessing_1.py`.

- **Minority Class Preservation:** Cluster A was a significant minority. In many cases, the data points that look like statistical "outliers" (extreme values in behavior) are actually the defining characteristic examples of these rare personality types.
- **Conclusion:** By keeping the "outliers" (in Preprocessing 1) and managing their scale via `StandardScaler`, the Neural Network was able to learn the edge cases. Removing them treated valuable minority signal as noise, degrading the model's ability to identify the rarer clusters.

9.4 Handling Feature Independence

Our EDA revealed that the features had near-zero correlation with each other. Neural Networks excel in this regime. When every input feature provides a unique, independent piece of information, the dense connections in the network allow it to aggregate these weak, disparate signals into a strong prediction more effectively than a greedy tree-based split, which might discard useful features early in the tree construction.

References

[1] *Lend or Lose: Loan Default Prediction Project*. GitHub Repository. https://github.com/standing-on-giants/Lend-or-lose-Loan-Default-prediction-project

[2] *Getting Started with Classification*. https://www.geeksforgeeks.org/getting-started-with-classification/