Personality Cluster Prediction using Machine Learning

A Comprehensive Analysis and Model Comparison

Team Name: B120
Team Members: Aaryan Antala, Ramya Parsania, Krishna Sai Velidanda

November 27, 2025

1 Team Details and Problem Statement

1.1 Team Information

- **Team Name:** B120
- **Team Members:**
  - Aaryan Antala - IMT2023548
  - Ramya Parsania - IMT2023015
  - Krishna Sai Velidanda - IMT2023094

1.2 Problem Statement

The objective of this project is to develop a machine learning model capable of predicting personality clusters based on behavioral and lifestyle indicators. This is a **multiclass classification problem** where participants are categorized into one of five distinct personality clusters (Cluster_A through Cluster_E) based on various observable behavioral patterns.

1.2.1 Problem Characteristics

- **Type:** Multiclass Classification
- **Number of Classes:** 5 (Cluster_A, Cluster_B, Cluster_C, Cluster_D, Cluster_E)
- **Evaluation Metric:** Macro F1 Score (to ensure fairness across imbalanced classes)
- **Challenge:** Significant class imbalance with Cluster_E representing over 50% of the data

2 Project Overview and Dataset Details

2.1 Project Overview

This project involves building predictive models to classify individuals into personality clusters based on anonymized behavioral and lifestyle data. The dataset contains various features related to daily habits, personal background, activity involvement, and social engagement. The personality clusters represent different groups of behavioral tendencies derived through prior segmentation analysis and are not labeled as "good" or "bad."

2.2 Dataset Description

2.2.1 Dataset Statistics

- **Training Samples:** 1,913 participants
- **Test Samples:** 479 participants
- **Number of Features:** 13 (excluding participant_id and target)
- **Target Variable:** personality_cluster (5 classes)

2.2.2 Feature Descriptions

The dataset contains the following features:

Table 1: Dataset Features and Descriptions

Feature	Description
participant_id	Unique ID assigned to each participant
age_group	Age grouping indicator (15-18)
identity_code	Encoded personal identity category (binary)
cultural_background	Regional or cultural background grouping (0-3)
upbringing_influence	Influence of formative environment (0-4)
focus_intensity	Time/effort dedicated toward focused tasks (continuous)
consistency_score	Reliability and routine-stability measure (0-29)
external_guidance_usage	Use of guidance/mentoring resources (binary)
support_environment_score	Perceived supportive environment level (0-4)
hobby_engagement_level	Engagement in leisure activities (binary)
physical_activity_index	Physical activity involvement (binary)
creative_expression_index	Participation in artistic activities (binary)
altruism_score	Tendency toward helping behaviors (binary)

2.2.3 Target Distribution

The target variable shows significant class imbalance:

Table 2: Class Distribution in Training Data

Cluster	Count	Percentage
Cluster_A	85	4.44%
Cluster_B	220	11.50%
Cluster_C	306	16.00%
Cluster_D	328	17.15%
Cluster_E	974	50.91%
Total	**1,913**	**100.00%**

2.3 Data Quality

- **Missing Values:** None detected in either training or test datasets
- **Data Types:** Mix of continuous (focus_intensity) and categorical/binary features
- **Anonymization:** All personally identifiable information has been removed

3 Data Preprocessing and Feature Engineering

This section details all preprocessing steps and feature engineering techniques applied to prepare the data for model training.

3.1 Overview of Preprocessing Pipeline

The preprocessing pipeline consists of the following major steps:

1. Feature Engineering
2. Removal of Non-Predictive Features
3. Target Variable Encoding
4. One-Hot Encoding for Categorical Features
5. Feature Scaling
6. Train-Validation Split
7. Handling Class Imbalance with SMOTE

3.2 Feature Engineering

Feature engineering was performed to create meaningful derived features that capture complex relationships in the data. Seven new features were created:

3.2.1 Engineered Features

1. **total_engagement** = hobby_engagement + physical_activity + creative_expression + altruism
   - *Rationale:* Aggregates all activity indicators to measure overall engagement level.
   - *Purpose:* Individuals with higher overall engagement may exhibit different personality traits.
2. **structured_living** = consistency_score × focus_intensity
   - *Rationale:* Combines discipline-related metrics.
   - *Purpose:* Captures the interaction between routine stability and task dedication.
3. **social_support** = support_environment_score × (external_guidance_usage + 1)
   - *Rationale:* Measures the combined effect of environmental support and guidance seeking.
   - *Purpose:* Individuals who both seek guidance and have support may form distinct clusters.
4. **age_focus_interaction** = age_group × focus_intensity
   - *Rationale:* Captures how focus varies with age.
   - *Purpose:* Younger vs. older participants may show different focus patterns.

5. **high_focus** =

$$\text{high\_focus} = \begin{cases} 1, & \text{if focus\_intensity} > \text{median}, \\ 0, & \text{otherwise.} \end{cases}$$

- *Rationale:* Creates a binary indicator for above-median focus.
- *Purpose:* Simplifies non-linear relationships for linear models.

6. **high_consistency** =

$$\text{high\_consistency} = \begin{cases} 1, & \text{if consistency\_score} > 75\text{th percentile}, \\ 0, & \text{otherwise.} \end{cases}$$

- *Rationale:* Identifies highly consistent individuals.
- *Purpose:* High consistency may be a strong personality indicator.

7. **is_active** =

$$\text{is\_active} = \begin{cases} 1, & \text{if } (hobby + physical + creative) > 0, \\ 0, & \text{otherwise.} \end{cases}$$

- *Rationale:* Binary indicator for any activity participation.
- *Purpose:* Distinguishes active vs. inactive individuals.

Result: Feature count increased from 13 to 19 features (excluding `participant_id`).

3.3 Feature Removal

Removed Feature: `participant_id`

Rationale: The `participant_id` is merely an identifier and contains no predictive information about personality clusters. Including it would lead to overfitting, as the model might memorize IDs rather than learn generalizable patterns.

3.4 Target Variable Encoding

The categorical target variable (personality_cluster) was encoded into numerical labels:

Table 3: Target Encoding Mapping

Cluster Name	Encoded Value
Cluster_A	0
Cluster_B	1
Cluster_C	2
Cluster_D	3
Cluster_E	4

Rationale: Machine learning algorithms require numerical inputs. Label encoding preserves the distinct nature of each class while making them compatible with ML algorithms.

3.5 One-Hot Encoding

Features Encoded: cultural_background, upbringing_influence

Rationale: These categorical features have no ordinal relationship. One-hot encoding prevents the model from assuming false ordinal relationships (e.g., cultural_background=3 is not "greater than" cultural_background=1).

Result:

- For Logistic Regression and SVM: 19 features → 26 features after one-hot encoding
- For Neural Networks: Kept original 19 features (NNs can learn categorical relationships)

3.6 Feature Scaling

Method: StandardScaler (z-score normalization)

Formula: $x_{scaled} = \frac{x-\mu}{\sigma}$

Rationale:

- Features have different scales (e.g., focus_intensity: 0-20, consistency_score: 0-29, binary features: 0-1)
- Distance-based algorithms (SVM, Neural Networks) are sensitive to feature scales
- Standardization ensures all features contribute equally to the model
- Prevents features with larger scales from dominating the learning process

Implementation: Separate scalers were fitted on training data and applied to validation/test data to prevent data leakage.

3.7 Train-Validation Split

Split Ratio: 80% training, 20% validation

Method: Stratified split

Rationale:

- Stratification maintains class distribution in both sets
- Critical for imbalanced datasets to ensure all classes are represented
- Validation set used for hyperparameter tuning and model selection

Result:

- Training set: 1,530 samples
- Validation set: 383 samples

3.8 Handling Class Imbalance - SMOTE

Method: Synthetic Minority Over-sampling Technique (SMOTE)
Parameters: k_neighbors=3
Rationale:

- Severe class imbalance (Cluster_E: 50.91%, Cluster_A: 4.44%)
- Models trained on imbalanced data tend to bias toward majority class
- SMOTE creates synthetic samples for minority classes by interpolating between existing samples
- Improves model's ability to learn patterns in minority classes

Result:

- Before SMOTE: 1,530 samples (imbalanced)
- After SMOTE: 3,895 samples (balanced - 779 samples per class)

3.9 Summary of Preprocessing

Table 4: Preprocessing Summary

Metric	Value
Original training samples	1,913
Test samples	479
Features after engineering	19
Features after one-hot (LR/SVM)	26
Training set (80%)	1,530
Validation set (20%)	383
Training set after SMOTE	3,895
Target classes	5

3.10 Diagrams Required

The following diagrams should be included in this section:

1. **Class Distribution Bar Chart** - Shows original imbalanced distribution
2. **Before/After SMOTE Comparison** - Side-by-side bar charts showing class balance improvement
3. **Feature Correlation Heatmap** - Shows relationships between engineered features
4. **Preprocessing Pipeline Flowchart** - Visual representation of the entire pipeline

4 Models Trained and Results

This section presents all models trained, their configurations, results, and justifications for their selection.

4.1 Experimental Setup

4.1.1 Evaluation Metrics

- **Primary Metric:** Macro F1 Score (competition metric)
- **Secondary Metrics:** Accuracy, Precision, Recall, Confusion Matrix

4.1.2 Hyperparameter Optimization

Method: Optuna (Bayesian Optimization)

- Automated hyperparameter search
- Objective: Maximize Macro F1 Score
- Cross-validation used during optimization

4.2 Models Overview

A total of 9 experiments were conducted with different algorithms:

Table 5: Experiments Conducted

Experiment	Model Type	Key Variant
Experiment 1	Neural Network	Optuna optimization
Experiment 2	Logistic Regression	Softmax + Optuna
Experiment 3	SVM	RBF Kernel
Experiment 4	SVM	Linear Kernel
Experiment 5	Neural Network	Optimized architecture
Experiment 6	K-Nearest Neighbors	Initial attempt
Experiment 7	CatBoost	Gradient boosting on decision trees
Experiment 8	XGBoost	Gradient boosting
Experiment 9	Bayesian	Probabilistic modeling

4.3 Model 1: Neural Network (Experiment 1)

4.3.1 Model Selection Rationale

- Neural networks can learn complex non-linear relationships
- Suitable for multiclass classification with multiple features
- Can automatically learn feature interactions
- Flexible architecture allows for optimization

4.3.2 Architecture

- **Input Layer:** 19 features
- **Hidden Layers:** Optimized through Optuna (typically 2-3 layers)
- **Activation:** ReLU for hidden layers
- **Output Layer:** 5 neurons with Softmax activation
- **Optimizer:** Adam
- **Loss Function:** Categorical Cross-Entropy

4.3.3 Results

- **Validation Macro F1:** 0.6229
- **Test Score:** 0.546
- **Training Time:** Moderate

4.3.4 Analysis

The neural network showed good performance due to its ability to learn complex patterns. However, it required careful tuning to avoid overfitting given the relatively small dataset size.

4.4 Model 2: Logistic Regression (Experiment 2)

4.4.1 Model Selection Rationale

- Baseline model for multiclass classification
- Interpretable coefficients
- Fast training and prediction
- Works well with one-hot encoded features
- Good for understanding feature importance

4.4.2 Configuration

- **Solver:** Optimized through Optuna (lbfgs/saga)
- **Regularization:** L2 (Ridge)
- **C parameter:** Tuned via Optuna
- **Multi-class:** Softmax (multinomial)
- **Max iterations:** 1000

4.4.3 Results

- **Validation Macro F1:** 0.5191 (Optuna), 0.5410 (Softmax Baseline)
- **Test Score:** 0.476
- **Training Time:** Very fast

4.4.4 Analysis

Logistic Regression provided a strong baseline. Its performance indicates that many relationships in the data are approximately linear, making it a reliable choice.

4.5 Model 3 & 4: Support Vector Machines (Experiments 3 & 4)

4.5.1 Model Selection Rationale

- Effective for high-dimensional data
- Can handle non-linear relationships (RBF kernel)
- Robust to outliers
- Two kernels tested: RBF (non-linear) and Linear

4.5.2 Configuration

Experiment 3 (RBF Kernel):

- **Kernel:** Radial Basis Function
- **C parameter:** Tuned
- **Gamma:** Tuned ('scale' or 'auto')

Experiment 4 (Linear Kernel):

- **Kernel:** Linear
- **C parameter:** Tuned

4.5.3 Results

RBF Kernel:

- **Validation Macro F1:** 0.5478
- **Test Score:** 0.474

Linear Kernel:

- **Validation Macro F1:** 0.4352
- **Test Score:** 0.412

4.5.4 Analysis

Comparison between RBF and Linear kernels helps understand whether non-linear decision boundaries are necessary. The significant drop in performance with the Linear kernel (0.412 vs 0.474 for RBF) confirms that the data has non-linear characteristics.

4.6 Model 5: Optimized Neural Network (Experiment 5)

4.6.1 Model Selection Rationale

- Refined version of Experiment 1
- Incorporated lessons learned from initial experiments
- More extensive hyperparameter search
- Optimized architecture for this specific problem

4.6.2 Improvements Over Experiment 1

- Extended Optuna search space
- Dropout layers for regularization
- Batch normalization
- Learning rate scheduling
- Early stopping to prevent overfitting

4.6.3 Results

- **Validation Macro F1:** 0.6349
- **Test Score:** 0.635
- **Improvement over Experiment 1:** +0.0120 (1.9%)

4.6.4 Analysis

The optimized neural network represents our best deep learning approach, incorporating regularization and architecture improvements. The minimal gap between validation and test scores indicates robust generalization.

4.7 Model 6 & 7: K-Nearest Neighbors (Exp 6) & CatBoost (Exp 7)

4.7.1 Model Selection Rationale

- Non-parametric algorithm
- No assumptions about data distribution
- Can capture local patterns
- Simple and interpretable

4.7.2 Configuration

- **n_neighbors:** Tuned (typically 3-15)
- **Weights:** Uniform vs. Distance-weighted
- **Metric:** Euclidean distance

4.7.3 Results

- **Validation Macro F1 (KNN):** 0.4829
- **Validation Macro F1 (CatBoost):** 0.6145
- **Test Score (CatBoost):** 0.563
- **Test Score (KNN):** 0.480

4.7.4 Analysis

KNN performance (0.480) was relatively poor, suggesting that simple distance metrics in feature space are insufficient for this classification task. CatBoost, however, performed strongly (0.563), acting as the second-best model after Neural Networks.

4.8 Model 8: XGBoost (Experiment 8)

4.8.1 Model Selection Rationale

- State-of-the-art gradient boosting algorithm
- Excellent performance on tabular data
- Handles feature interactions automatically
- Built-in regularization
- Robust to overfitting

4.8.2 Configuration

- **Objective:** multi:softmax
- **n_estimators:** Tuned
- **max_depth:** Tuned
- **learning_rate:** Tuned
- **subsample:** Tuned
- **colsample_bytree:** Tuned

4.8.3 Results

- **Validation Macro F1:** 0.5784
- **Test Score:** 0.536
- **Training Time:** Moderate

4.8.4 Analysis

XGBoost performed respectably (0.536) but was outperformed by CatBoost and Neural Networks. This may be due to the specific nature of the engineered features favoring the other architectures.

4.9 Model 9: Bayesian (Experiment 9)

4.9.1 Model Selection Rationale

- Bayesian methods provide probabilistic predictions and can quantify uncertainty in classifications
- Suitable for multiclass problems with imbalanced data by incorporating prior knowledge
- Serves as a baseline for probabilistic modeling approaches in comparison to deterministic methods

4.9.2 Configuration

- **Method:** Naive Bayes classifier (GaussianNB from scikit-learn)
- **Priors:** Uniform priors for all classes
- **Features:** Used the preprocessed and scaled features
- **Assumptions:** Assumes feature independence given the class

4.9.3 Results

- **Test Score:** 0.380
- **Training Time:** Very Fast

4.9.4 Analysis

The Bayesian approach using Naive Bayes provided a simple probabilistic baseline but performed poorly (0.380) compared to other methods. This is likely due to the strong assumption of feature independence, which does not hold for the engineered features in this dataset. It serves as a useful comparison point, demonstrating the limitations of simple probabilistic models for complex behavioral data.

4.10 Model Comparison

Table 6 summarizes the performance of all trained models. The Optimized Neural Network (Experiment 5) achieved the highest performance on both validation and test sets.

Table 6: Model Performance Comparison

Model	Val F1	Test F1	Training Time
NN (Exp 1)	0.6229	0.546	Moderate
Logistic Reg	0.5191	0.476	Very Fast
SVM (RBF)	0.5478	0.474	Slow
SVM (Linear)	0.4352	0.412	Moderate
NN Optimized (Exp 5)	**0.6349**	**0.635**	**Moderate**
KNN (Exp 6)	0.4829	0.480	Fast
CatBoost (Exp 7)	0.6145	0.563	Moderate
XGBoost (Exp 8)	0.5784	0.536	Moderate
Bayesian (Exp 9)	-	0.380	Very Fast

4.11 Best Model Selection

Selected Model: Optimized Neural Network (Experiment 5)

Justification: The Optimized Neural Network was selected as the final model for the following reasons:

- **Superior Performance:** It achieved the highest Validation F1 score (0.6349) and Test F1 score (0.635), outperforming the next best model (CatBoost) by a margin of 0.072 on the test set.
- **Generalization:** The gap between validation and test scores is minimal (< 0.001), indicating excellent generalization without overfitting.
- **Class Handling:** The architecture effectively handled the class imbalance, likely due to the inclusion of SMOTE and specific class-weight adjustments within the loss function.

4.12 Results Analysis

4.12.1 Key Findings

1. **Non-Linearity is Key:** The significant performance gap between Linear SVM (0.412) and Neural Networks (0.635) confirms that the boundaries between personality clusters are highly non-linear.
2. **Impact of Optimization:** Refining the Neural Network architecture (Exp 5 vs Exp 1) resulted in a massive performance jump on the test set (from 0.546 to 0.635), highlighting the importance of hyperparameter tuning and regularization (Dropout).
3. **Tree-Based vs. Neural:** While CatBoost performed well (0.563), the Neural Network was better suited for this feature set, likely because the continuous engineered features (like 'focus_intensity') benefit from the differentiable nature of NNs.

4. **Class Imbalance Impact:** SMOTE significantly improved minority class performance across all models, preventing the models from simply predicting the majority class (Cluster_E).

4.12.2 Per-Class Performance

The best model showed consistent performance across clusters, though Cluster_A (the smallest minority) remained the most challenging to classify correctly, often being confused with Cluster_C.

4.13 Final Test Predictions

The best model was used to generate predictions on the test set, which were submitted for evaluation. The final test score of **0.635** represents the model's performance on unseen data.

5 Conclusion

This project successfully developed and evaluated nine distinct machine learning approaches for personality cluster prediction. Through comprehensive preprocessing, feature engineering, and rigorous model optimization, we achieved a final **Test Macro F1 Score of 0.635** with our best model (Optimized Neural Network).

5.1 Key Takeaways

1. **Data Quality:** Feature engineering played a critical role; derived features like `structured_living` and `total_engagement` provided higher predictive power than raw demographic inputs.

2. **Model Complexity:** Simple linear models (Logistic Regression, Linear SVM) failed to capture the complexity of human behavior data, capping at an F1 score of roughly 0.47.

3. **Balancing Act:** The application of SMOTE combined with careful validation splitting was essential. Without these, models heavily favored Cluster_E (50% of data), ignoring the minority Cluster_A.

5.2 Future Work

- **Ensemble Methods:** A voting classifier combining the Neural Network and CatBoost could potentially smooth out errors, as their confusion matrices showed they struggled with different classes.

- **Feature Expansion:** Incorporating text data (if available in future iterations) from participant survey responses could boost accuracy using NLP techniques.

- **Advanced Sampling:** Exploring ADASYN or other oversampling techniques beyond SMOTE to better handle the decision boundaries of minority classes.

A Code Repository

All code for preprocessing, model training, and evaluation is available in the project repository here:
`https://github.com/AaryanAntala/personality_cluster_prediction`

B Hyperparameter Search Spaces

The Optuna search spaces used for the Optimized Neural Network were:

- **Hidden Layers:** 1 to 4
- **Neurons per Layer:** 32 to 256
- **Learning Rate:** 1e-4 to 1e-2 (log scale)
- **Dropout Rate:** 0.1 to 0.5
- **Batch Size:** 32, 64, 128