Machine Learning Project Report

Signal Cluster Classification

GitHub Repo: `https://github.com/BallaVishnu/Machine-Learning-Classification`

November 27, 2025

Team Members:
B. Vishnu Sai Prashanth (IMT2023097)
M. Praneeth (IMT2023555)

1 Task

The primary objective of this project is to train a machine learning model that can accurately determine the *personality_ cluster* for each participant based on their signal measurements. This is a multiclass classification problem, where the target variable represents multiple cluster categories rather than only two outcomes. Using the features *signal_ strength* and *response_ level*, the task involves learning the patterns that define each cluster and applying this knowledge to classify new participants in the test dataset. The performance of the model is evaluated using the Macro F1 Score, which gives equal importance to all clusters regardless of their frequency.

2 Dataset and Features Description

The dataset contains synthetic behavioural signal measurements designed to support experimentation with machine learning classification methods. Each participant is described by two numerical features: *signal_ strength* and *response_ level*. These features capture different types of measured signal behaviours. The target label, *personality_ cluster*, indicates the group the participant belongs to based on the relationship between the two features. The test dataset contains the same features but does not include the cluster label, which must be predicted.

3 Initial Data Health Check

We first checked the basic quality of the dataset to make sure it was ready for analysis.

- There were no missing values in any of the columns.
- No duplicate rows were found.
- A few negative values were present in both features. These were kept because they are normal signal readings.

- Using the IQR method, we found zero outliers in both signal_strength and response_level.

The dataset was already clean and did not require any correction or removal of samples.

4 Exploratory Data Analysis (EDA)

Different visualizations were used to understand how the features behave and how the clusters are formed.

4.1 Category Distribution

A bar plot showed that:

- Group B had the highest number of samples.
- Group C had a moderate number of samples.
- Group A had the lowest number of samples.

This shows slight class imbalance, which is why Macro F1 score is a fair metric.

4.2 Feature Distributions

Histograms for both features showed smooth shapes with multiple peaks. This means the features naturally form different cluster patterns and are not concentrated in one region.

4.3 Boxplots

The boxplots showed that both features stretch across a wide range. There were no strong unusual values, which means the data is consistent and stable.

4.4 Scatter Plot Analysis

The scatter plot of signal_strength vs response_level was the most useful. We saw three clear groups, but the boundaries were curved instead of straight.

This told us:

- Linear models will not work well.
- Non-linear models like SVM with RBF kernel, Neural Networks, and KNN are more suitable.

4.5 Pairplot Analysis

The pairplot again showed clear separation between the three groups when combining both features. This confirms that both features together help in identifying clusters.

4.6 Correlation Matrix

The correlation matrix showed a moderate negative relationship between the two features. This means both features provide different information instead of repeating the same pattern.

4.7 Overall EDA Understanding

The graphs helped us understand the structure of the dataset:

- The clusters follow curved and non-linear shapes.
- The dataset has no outliers and is evenly spread.
- Both features have good variety and are useful for classification.
- Non-linear models are needed for best accuracy.

5 Data Preprocessing

Before training models, we prepared the dataset to make it suitable for machine learning.

5.1 Target Encoding

The labels Group_A, Group_B, and Group_C were converted into numbers:

- Group_A $\rightarrow$ 0
- Group_B $\rightarrow$ 1
- Group_C $\rightarrow$ 2

This is required because machine learning models work with numerical targets.

5.2 Feature Scaling

The two features had very different ranges. To make them equal and avoid bias, we applied Standard Scaling:

- Mean becomes 0
- Standard deviation becomes 1

This helped models like SVM, KNN, and Neural Networks learn better and faster.
We also tested Min-Max scaling, but StandardScaler gave better model performance.

5.3 Train-Validation Split

We split the dataset into:

- 80% training data
- 20% validation data

The split was stratified so that all groups kept the same proportion in both sets.

5.4 Preprocessing Insights

From preprocessing, we understood that:

- Scaling is necessary because the feature values vary a lot.
- Encoding removes category labels and makes training easier.
- Stratified splitting avoids bias in model evaluation.

Overall, preprocessing helped the models produce more stable and accurate results.

6 Models and Results

In this project, we tested several machine learning models to understand which one could predict the cluster labels most accurately. Because our EDA showed that the clusters have curved and non-linear boundaries, we focused mainly on non-linear models such as Neural Networks, SVM, and combinations of multiple models. This section explains each model we used, the scores we obtained, and what we learned from the results.

6.1 1. Basic Neural Network (MLP)

The first model we trained was a Multi-Layer Perceptron with two hidden layers. This was our starting point because neural networks naturally learn curved shapes and patterns.

- **Score:** 0.981
- **What we did:** Standardized the features, encoded the labels, and trained the model for 1000 iterations.
- **What we observed:** The network learned the general structure of the data, but it still misclassified some points near the borders between clusters. Increasing depth or neurons helped only slightly.

6.2 2. Improved Neural Networks

We tested deeper architectures with more hidden layers, different activation functions (ReLU, `tanh`), and slightly tuned learning rates.

- **Score:** 0.982–0.985
- **What we did:** Tried larger networks such as (120, 60) or (150, 75), changed learning rate and regularization values.
- **What we observed:** The model became smoother and more stable, but performance still did not exceed 0.986. Neural Networks alone were not enough to reach 0.99 with only two features.

6.3 3. Support Vector Machine (SVM)

Based on the scatter plot, the data clearly required curved boundaries. Therefore, we used an SVM with an RBF kernel, which is known for handling circular and curved shapes very well.

- **Score:** 0.984–0.987 (depending on tuning)
- **What we did:** Tuned the important parameters `C` and `gamma` using small grid searches.
- **What we observed:** SVM performed very strongly because the cluster shapes matched the RBF kernel. However, SVM sometimes struggled with points located exactly between two clusters, so it could not consistently go above 0.987.

6.4 4. K-Nearest Neighbors (KNN)

KNN is a very simple non-linear model that classifies a point based on nearby neighbors.

- **Score:** Around 0.97
- **What we did:** Used $k = 7$ to reduce noise and smooth the decision boundaries.
- **What we observed:** KNN worked fine in dense areas but struggled in regions where clusters were close to each other. Alone, it was not strong enough, but it became useful when combined with other models.

6.5 5. Logistic Regression

We used Logistic Regression mainly as a simple meta-model to combine predictions.

- **What we observed:** Since the data was clearly non-linear, Logistic Regression was not strong as a standalone model, but it helped when used on top of SVM and Neural Network outputs.

6.6 6. Combined Model (SVM + Neural Network + KNN)

This gave us the highest and most stable performance. Instead of relying on one model, we combined the strongest three models using soft voting.

- **Score: 0.988** (achieved 4–5 times in different runs)
- **What we did:** Allowed all three models to vote based on their prediction probabilities.
- **Why this worked so well:**
  - SVM captured the smooth curved boundaries
  - Neural Network learned deeper non-linear regions
  - KNN handled local clusters and dense areas

  When combined, their strengths covered each other's weaknesses. This made the model very stable, which is why we repeatedly got 0.988.

6.7 Final Best Model

After trying several models and combinations, the best-performing approach was the simple soft-voting combination of SVM, Neural Network, and KNN.

- **Final Best Score: 0.988**
- **Summary:** This model gave the most stable performance and consistently handled different types of patterns inside the dataset.

7 Conclusion

The dataset showed three clusters with clear non-linear boundaries, so non-linear models were needed. After testing several models such as Neural Networks, SVM, and KNN, we found that no single model could classify all points perfectly. The best performance came from combining these models using soft voting, which consistently achieved a Macro F1 score of 0.988. Reaching higher scores was difficult because some points lie very close to the borders between clusters and only two features are available. Overall, the combined model provided the most accurate and stable results for this classification task.