ML Project Part 2

Course Name: AIT 511, Machine Learning

Submitted by:
Gathik Jindal (IMT2023089)
Sahil Kolte (IMT2023066)
Pratham Shetty (IMT2023534)

27 November 2025

Abstract

This project investigates a comprehensive suite of machine learning techniques to solve two real-world supervised learning tasks: a binary classification problem on *Financial Risk Profiling* and a multiclass classification problem on *Travel Behaviour Insights*. To develop robust predictive models, we evaluated both classical and modern algorithms across tree-based, probabilistic, distance-based, margin-based, and neural approaches. Tree and ensemble methods—including Decision Trees, Random Forests, XGBoost, and LightGBM—served as strong baselines due to their interpretability and ability to capture non-linear relationships. Probabilistic and distance-driven models such as Naive Bayes and K-Nearest Neighbors were assessed for their simplicity and effectiveness on high-dimensional data. Additionally, Logistic Regression and Support Vector Machines with multiple kernels were employed to examine linear and maximum-margin decision boundaries. Gaussian Mixture Models and unsupervised techniques like K-Means and hierarchical clustering were used to derive latent structure in the datasets and compare clustering patterns with supervised labels. A simple neural network was optionally implemented to benchmark performance against deep learning-based nonlinear classifiers.

GitHub Link: https://github.com/gathik-jindal/ML-Project-2

Contents

1 Project Objective

The primary objective of this project was to bridge the gap between theoretical understanding and practical application by evaluating the real-world performance of a broad range of machine learning models studied in this course. Using two diverse datasets—one for *Financial Risk Profiling* (binary classification) and another for *Travel Behaviour Insights* (multiclass classification)—our team aimed to assess how different algorithms perform across distinct problem types. By applying and comparing classical statistical models, modern ensemble techniques, and unsupervised learning methods, the project serves as a practical testbed to understand model behavior, strengths, and trade-offs in realistic prediction scenarios.

2 Binary Classification Problem

2.1 Unsupervised Classification Models

To explore latent structures in the data without relying on labeled supervision, an unsupervised learning pipeline was implemented. The target variable `RiskFlag` was removed from the training set, and to optimize computational efficiency and model performance, the feature space underwent rigorous preprocessing, including encoding and scaling, followed by Principal Component Analysis (PCA) to reduce dimensionality while retaining 95

2.1.1 K-Means Clustering

K-Means clustering was performed on the PCA-transformed data to identify distinct user segments. A grid search was conducted for $K \in [2, 10]$. The optimal number of clusters was determined by evaluating the Elbow Method (Inertia) and the Silhouette Score. While K-Means forces data into spherical clusters, it provided a baseline for separating high-risk and low-risk profiles.

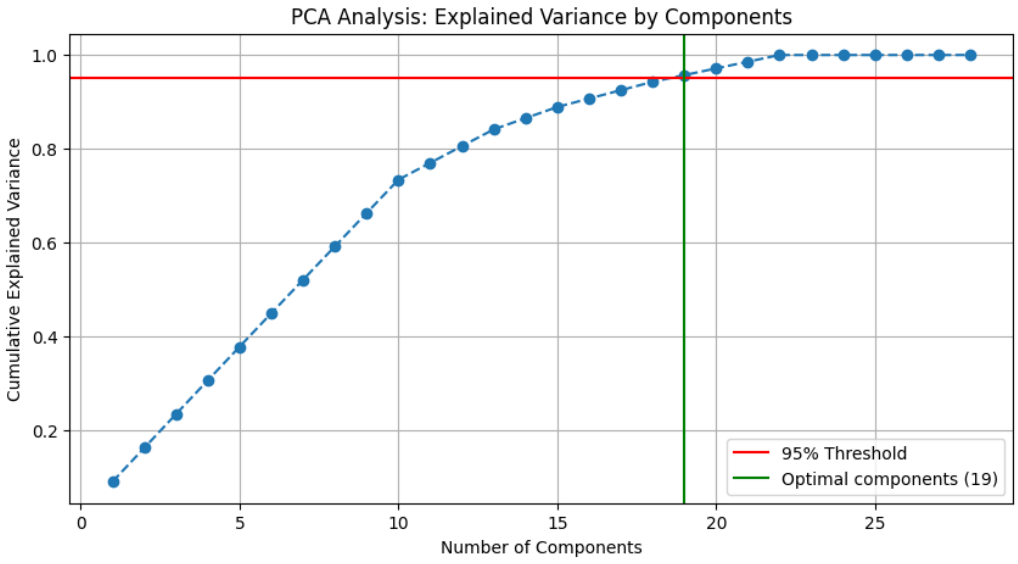

Figure 1: PCA Analysis: Explained Variance by Components

2.1.2 Hierarchical Clustering

To analyze the natural nesting of data points, Hierarchical Agglomerative Clustering was applied. Due to the computational complexity ($O(N^2)$), the algorithm was run on a stratified sample of 10,000 instances. A Dendrogram using Ward's linkage was generated to visualize the merge distances, revealing the optimal cut-off point for cluster formation.

For prediction on the test set (since Hierarchical Clustering lacks a `predict` method), a K-Nearest Neighbors (KNN) classifier was trained on the cluster labels to project test samples into the identified hierarchy.

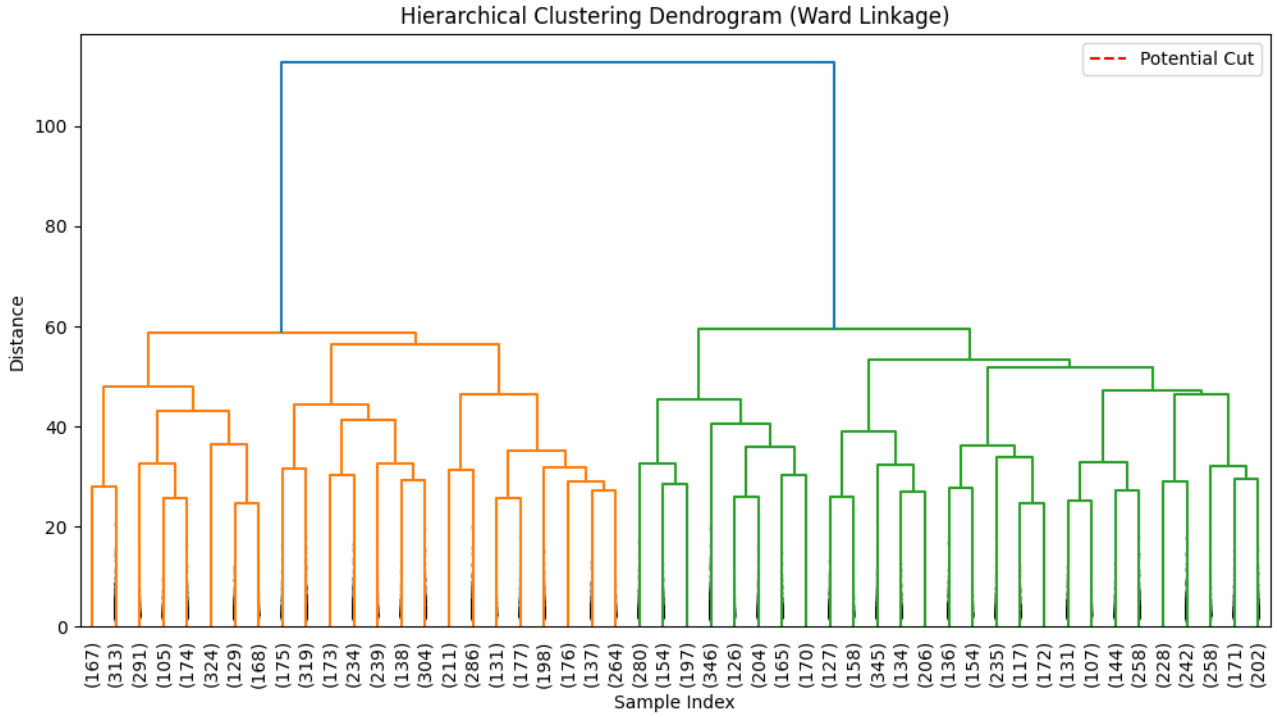

Figure 2: Hierarchical Clustering Dendrogram (Ward Linkage)

2.1.3 DBSCAN Clustering

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was utilized to identify clusters of arbitrary shapes and detect outliers (noise). A grid search was performed to optimize the `eps` (neighborhood radius) and `min_samples` parameters, maximizing the Silhouette Score while maintaining a reasonable noise ratio. The resulting noise points were treated as a separate "outlier" class or mapped to the majority class during profiling.

2.1.4 Cluster Profiling & Proxy Classification

The unsupervised clusters were evaluated for "Cluster Purity" by cross-referencing them with the actual `RiskFlag` (Ground Truth). A confusion matrix heatmap was generated to visualize how well the unsupervised groups aligned with the financial risk categories. Clusters with a high concentration of "Risk 1" were labeled as high-risk segments, effectively turning the clustering output into a classification mechanism.
Kaggle Score: 0.844

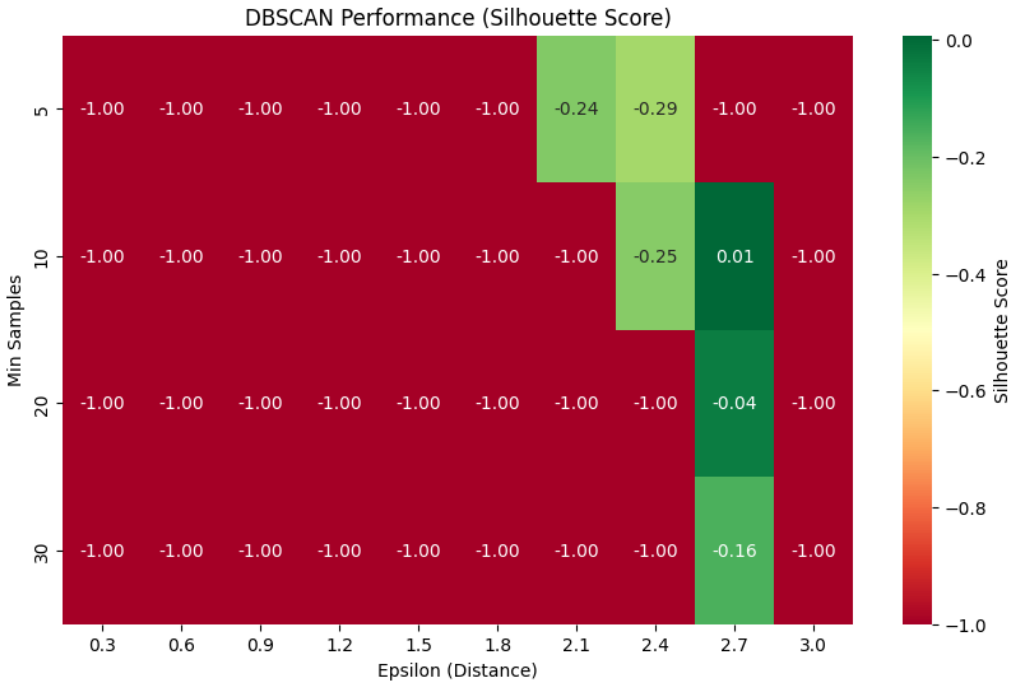

Figure 3: DBSCAN Performance (Silhouette Score)

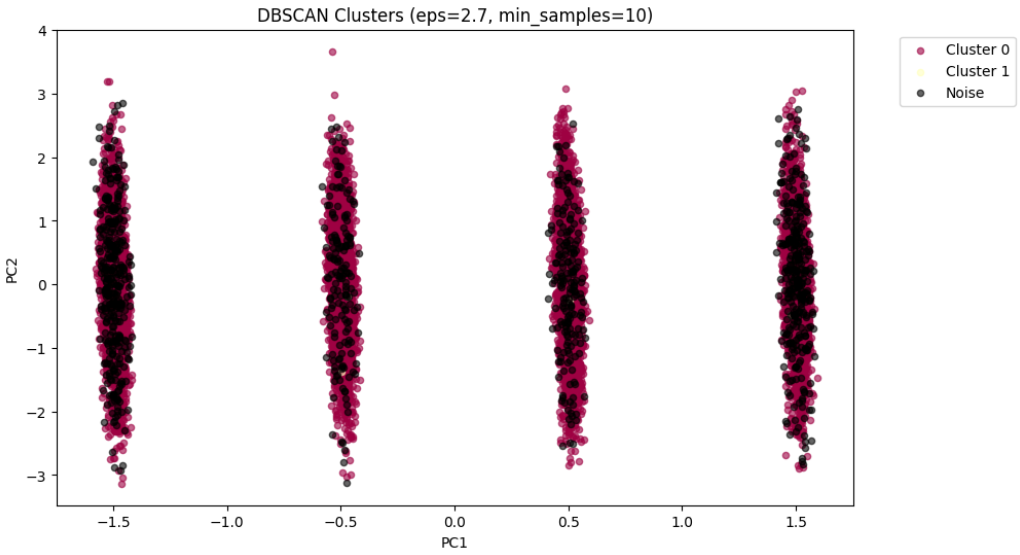

Figure 4: Cluster Purity Heatmap against Ground Truth

2.2 Support Vector Machine (SVM)

To effectively handle the high-dimensional feature space and capture non-linear decision boundaries, a Support Vector Machine (SVM) classifier was implemented. Given the computational intensity of SVMs with large datasets, a dimensionality reduction step using Principal Component Analysis (PCA) was applied first. PCA reduced the feature space to 20 components, retaining 95% of the explained variance, which improved training efficiency without significant information loss.

A grid search was performed to optimize the SVM hyperparameters, specifically comparing different kernel functions (Linear, Polynomial, RBF) and regularization parameters (C). The Radial Basis Function (RBF) kernel with optimized C and γ parameters yielded the best validation performance. The final model was evaluated using a confusion matrix to assess its precision and recall balance across risk classes.

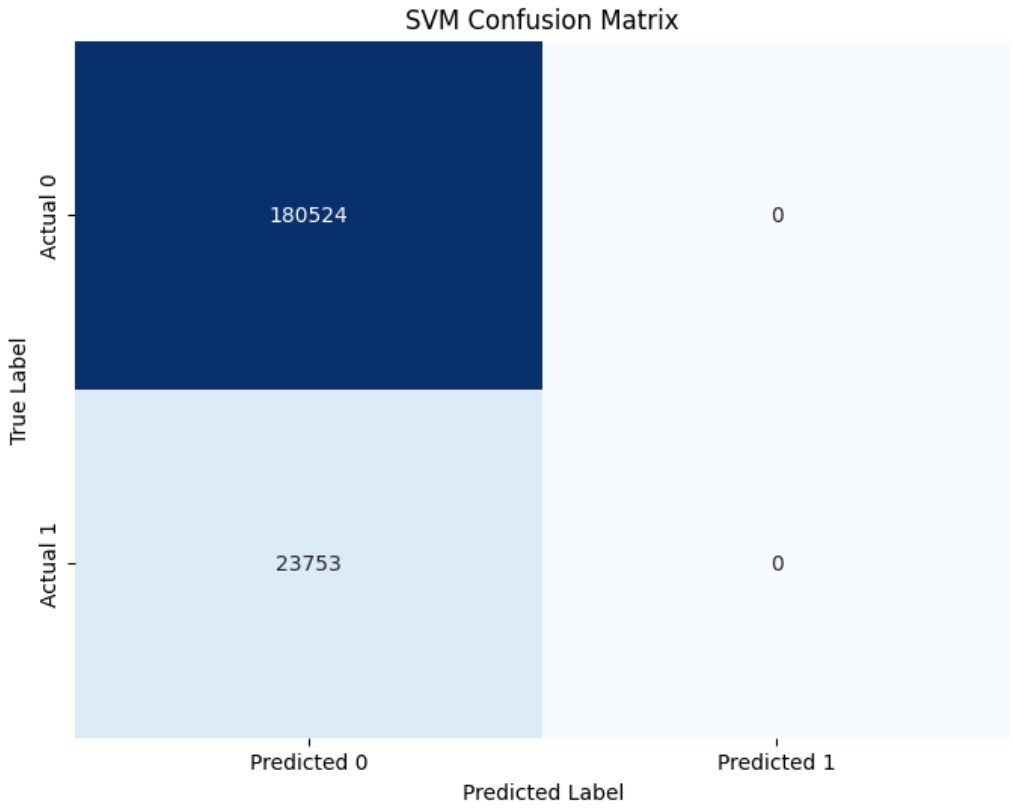

Figure 5: SVM Confusion Matrix

Best Validation Accuracy: 0.879
Kaggle Score: 0.844

2.3 Gaussian Mixture Models (GMM)

To capture more complex cluster structures and probabilistic assignments, a Gaussian Mixture Model (GMM) was implemented. Unlike K-Means, which assumes spherical clusters, GMM allows for flexible cluster shapes through full covariance matrices. Due to the "Curse of Dimensionality" affecting covariance estimation, the feature space was first reduced using Principal Component Analysis (PCA) to retain 95% of the variance (resulting in 19 components).

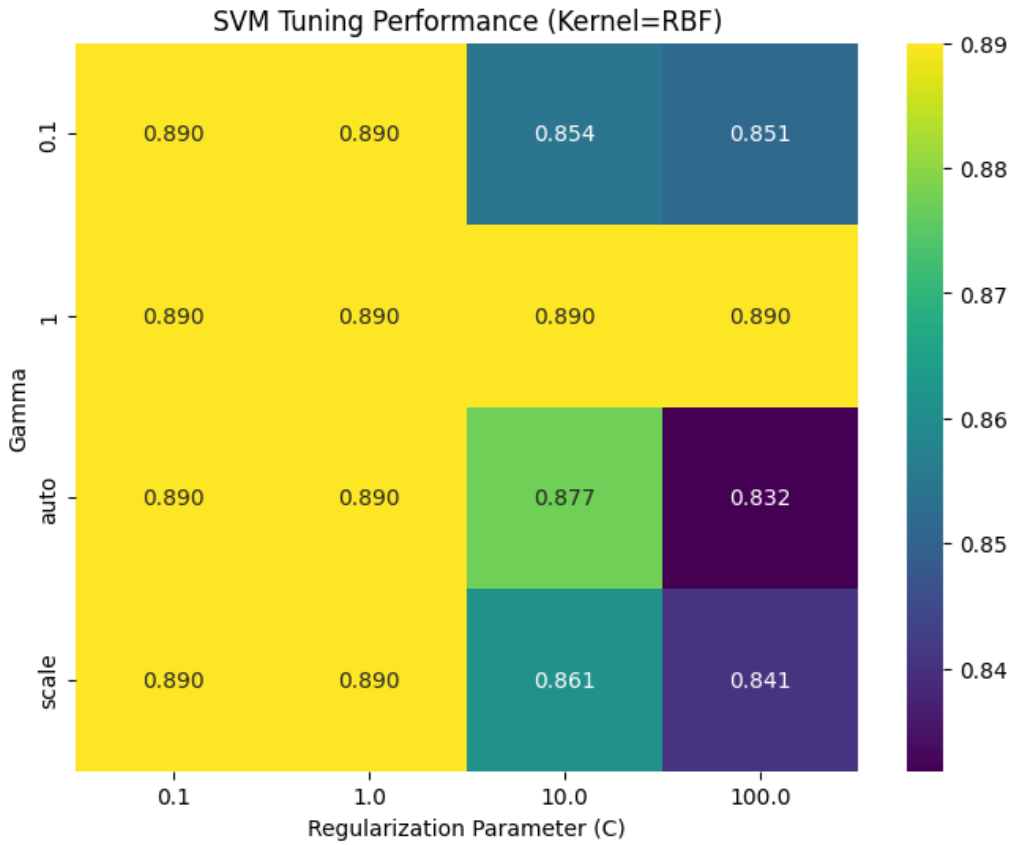

Figure 6: SVM Tuning Performance (Kernel=RBF)

A grid search was conducted to determine the optimal model configuration, testing component counts from 2 to 10 and various covariance types (*full*, *tied*, *diag*, *spherical*). The Bayesian Information Criterion (BIC) was used for model selection, penalizing complexity to avoid overfitting. The analysis identified a model with **10 components** and a **full** covariance type as the optimal solution.

The clusters were profiled against the `RiskFlag` target to determine risk purity. While the probabilistic nature of GMM provided soft clustering assignments, the separation between high-risk and low-risk groups remained challenging, as indicated by the cluster risk probabilities.

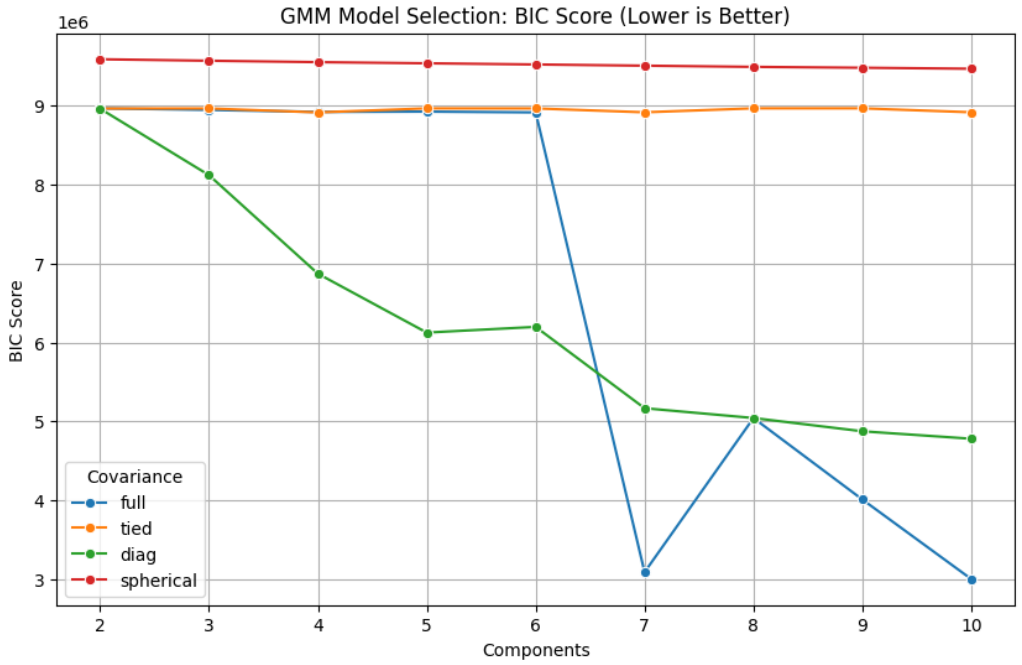

Figure 7: GMM Model Selection: BIC Score (Lower is Better)

Kaggle Score: 0.884

2.4 Neural Network (MLP)

To capture complex non-linear interactions in the data, a Multi-Layer Perceptron (MLP) Neural Network was implemented using the Keras Sequential API. The preprocessing pipeline standardized numerical features and one-hot encoded categorical variables to ensure compatibility with the neural architecture.

Architecture:

- **Input Layer:** Matched to the dimensionality of the preprocessed feature space.
- **Hidden Layers:** Two dense layers with 64 and 32 neurons, respectively, using **ReLU** activation functions.
- **Regularization: Dropout** layers (rate=0.3) were interleaved to prevent overfitting.
- **Output Layer:** A single neuron with a **Sigmoid** activation function for binary classification probability.

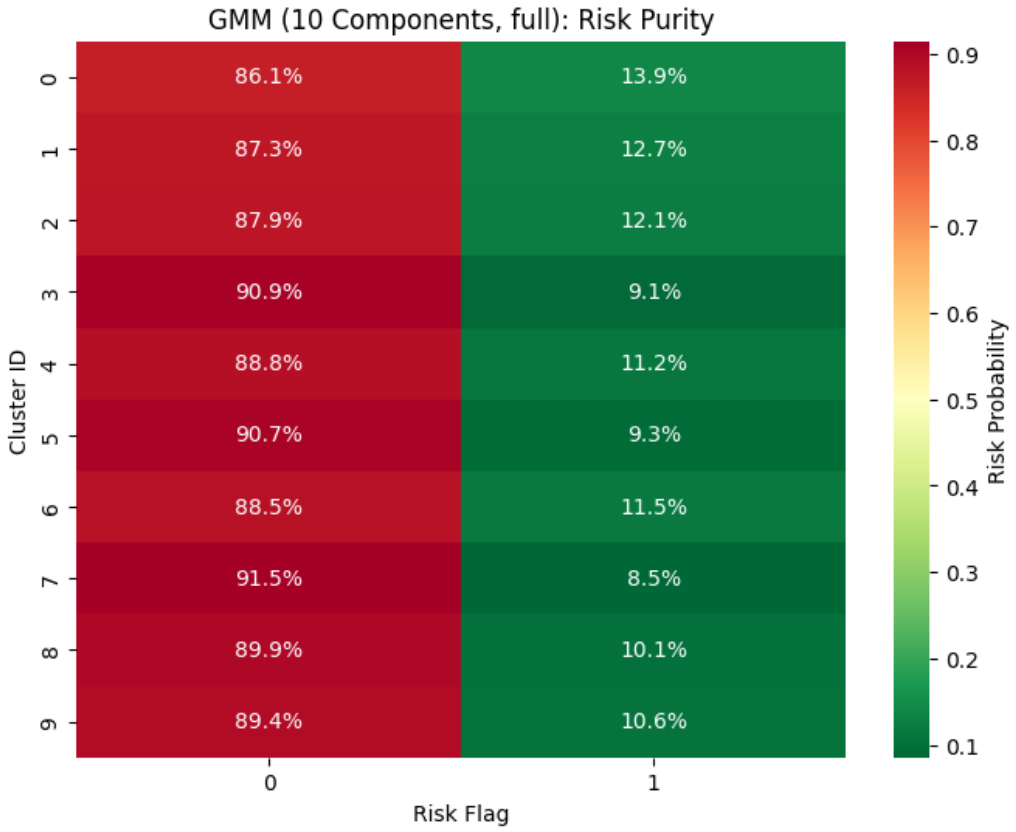

Figure 8: GMM Risk Purity Heatmap (10 Components)

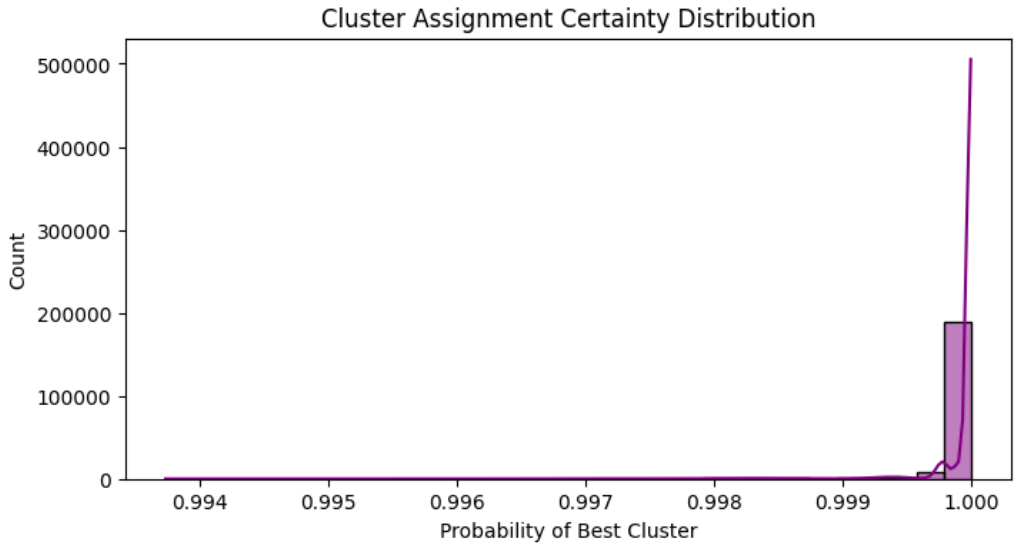

Figure 9: Cluster Assignment Certainty Distribution

The model was trained using the **Adam** optimizer and **Binary Cross-Entropy** loss. Early stopping was employed to halt training when validation loss ceased to improve. Performance was evaluated using accuracy, ROC-AUC, and Precision-Recall metrics.

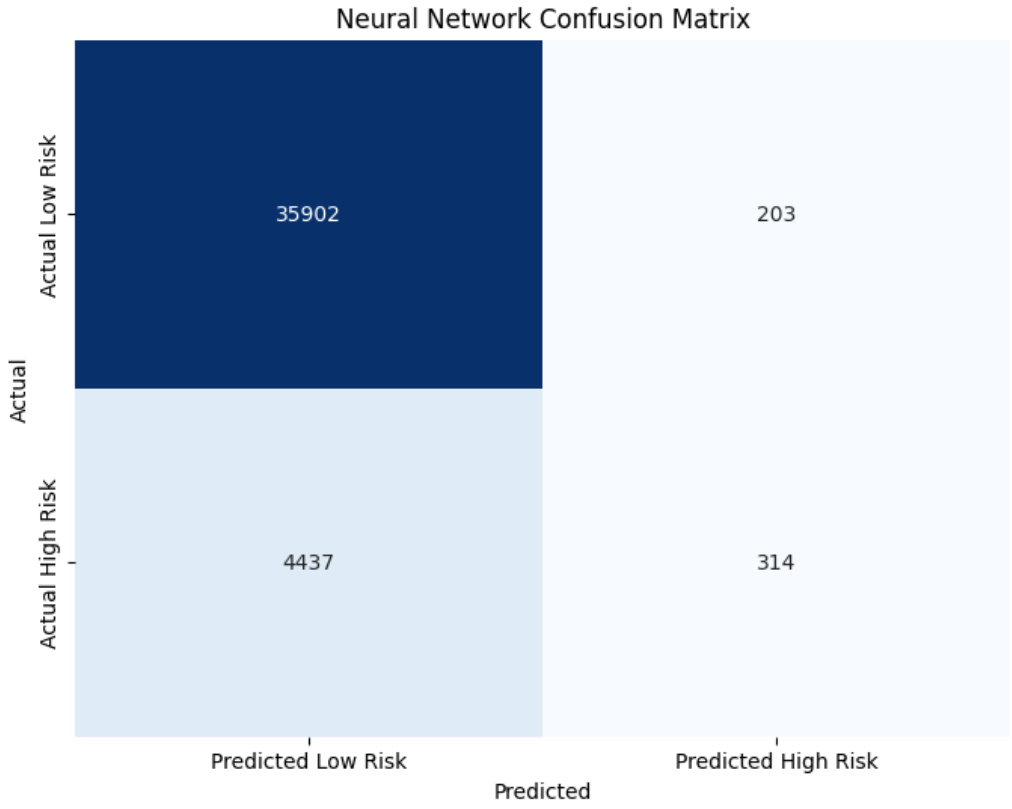

Figure 10: Neural Network Confusion Matrix

Validation Accuracy: 0.885 Kaggle Score: 0.887

2.5 Tree-Based and Ensemble Models

2.5.1 Decision Tree:

For this experiment, a Decision Tree Classifier was implemented to establish a baseline for the financial risk prediction task. The dataset was split into an 80–20 train–validation ratio, with `RiskFlag` as the target and `ProfileID` removed to avoid leakage. A preprocessing pipeline was used to automatically handle categorical variables through One-Hot Encoding while passing numerical features unchanged, ensuring consistent feature transformation during both training and testing.

The model was trained using a depth-limited Decision Tree (`max_depth=10`) to control overfitting and capture non-linear patterns in the financial attributes. After training, the model was evaluated on the validation set using accuracy, classification report, and confusion matrix, and final predictions for the test dataset were exported as a submission CSV.

Kaggle Score: 0.881

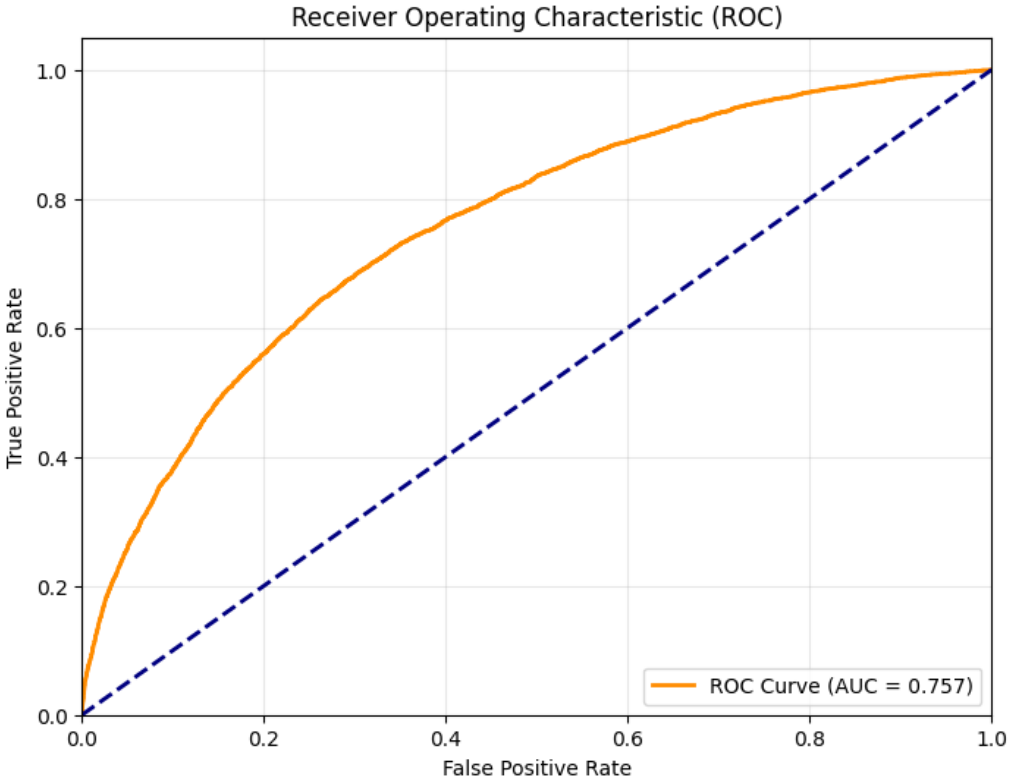

Figure 11: Receiver Operating Characteristic (ROC) Curve

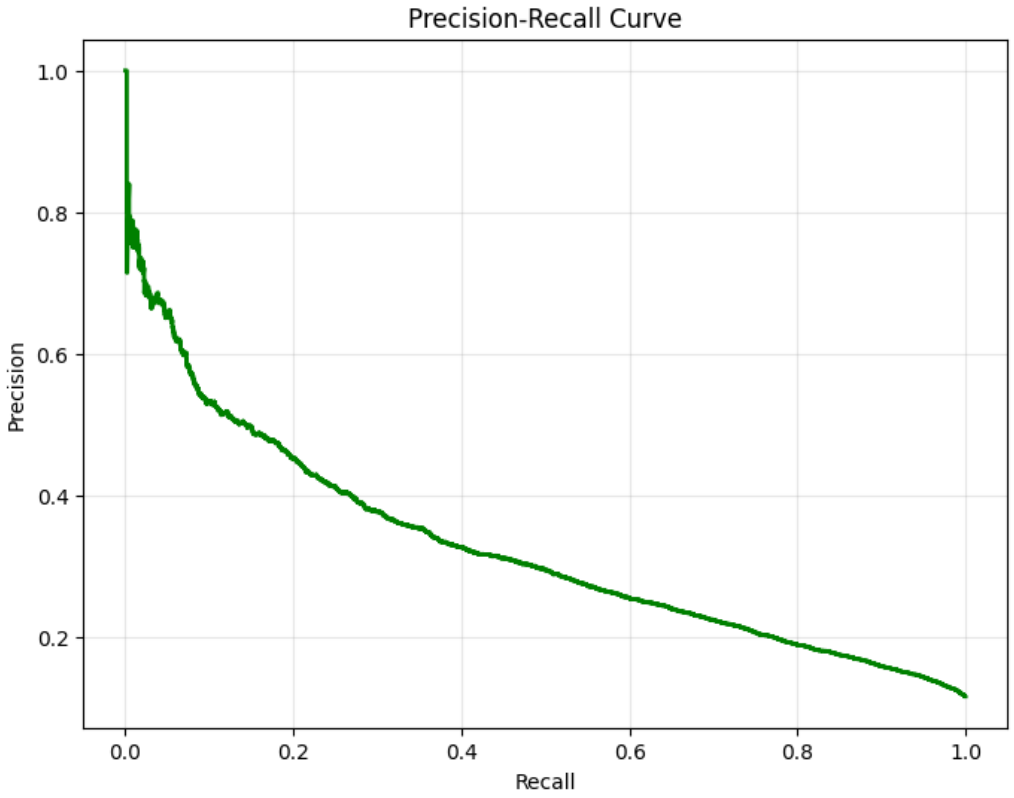

Figure 12: Precision-Recall Curve

```
Evaluating on Validation Set:
Accuracy: 0.8790

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     36073
           1       0.39      0.06      0.10      4783

    accuracy                           0.88     40856
   macro avg       0.64      0.52      0.52     40856
weighted avg       0.83      0.88      0.84     40856

Confusion Matrix:
[[35626   447]
 [ 4496   287]]
```

Figure 13: Decision Tree Model

2.5.2 Random Forest:

A Random Forest Classifier was implemented to improve predictive performance over the baseline models for financial risk classification. The dataset was split into an 80–20 stratified train–validation split to maintain class balance. A preprocessing pipeline was applied, where categorical features were transformed using One-Hot Encoding and numerical features were passed through directly. This ensured consistent feature handling across the training, validation, and test datasets.

The model used a Random Forest with 100 trees, a maximum depth of 10, and a minimum of 5 samples per leaf to control overfitting while capturing complex feature interactions. After training, the model was evaluated using accuracy, classification metrics, and a confusion matrix. Finally, predictions on the test data were generated and saved as a submission file for further analysis.

Kaggle Score: 0.884

2.5.3 XGBoost

To further improve predictive accuracy for the financial risk classification task, an XGBoost Classifier was implemented using a unified preprocessing and modeling pipeline. The dataset was split into an 80–20 training–validation set, with categorical variables transformed using One-Hot Encoding and numerical variables passed through unchanged. This ensured that both tree-based learning and preprocessing were consistently applied across training, validation, and test data.

The model used XGBoost with 500 trees, a learning rate of 0.05, and depth-controlled trees to balance

```
Evaluating on Validation Set:
Accuracy: 0.8841

Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     36105
           1       0.77      0.00      0.01      4751

    accuracy                           0.88     40856
   macro avg       0.83      0.50      0.47     40856
weighted avg       0.87      0.88      0.83     40856

Confusion Matrix:
[[36098     7]
 [ 4728    23]]
```

Figure 14: Random Forest Model

accuracy and generalization. After training, the model was evaluated using metrics such as accuracy, ROC-AUC, classification report, and confusion matrix to assess its ability to distinguish between risky and non-risky financial profiles. Final predictions for the test dataset were generated and saved in a submission-ready CSV file.

Kaggle Score: 0.887

2.5.4 LightGBM:

A LightGBM Classifier was implemented to leverage gradient boosting with optimized tree growth for the financial risk prediction task. As with other models, the dataset was split into an 80–20 stratified train–validation split, and preprocessing was handled using a `ColumnTransformer` where categorical variables were One-Hot Encoded and numerical variables were passed through directly. This ensured consistent feature transformation across all models used in the study.

LightGBM was configured with 100 boosting iterations, a learning rate of 0.1, and 31 leaves per tree—settings chosen to balance speed, accuracy, and generalization. The model was trained through a unified pipeline and evaluated using accuracy, classification metrics, and a confusion matrix to measure its performance in distinguishing risky versus non-risky financial profiles. Final predictions were generated for the test set and exported as a submission file.

Kaggle Score: 0.887

```
Evaluating Model on Validation Set...
Accuracy: 0.8844
ROC-AUC: 0.7455

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     36073
           1       0.55      0.07      0.12      4783

    accuracy                           0.88     40856
   macro avg       0.72      0.53      0.53     40856
weighted avg       0.85      0.88      0.84     40856

Confusion Matrix:
[[35796   277]
 [ 4446   337]]
```

Figure 15: XGBoost Model

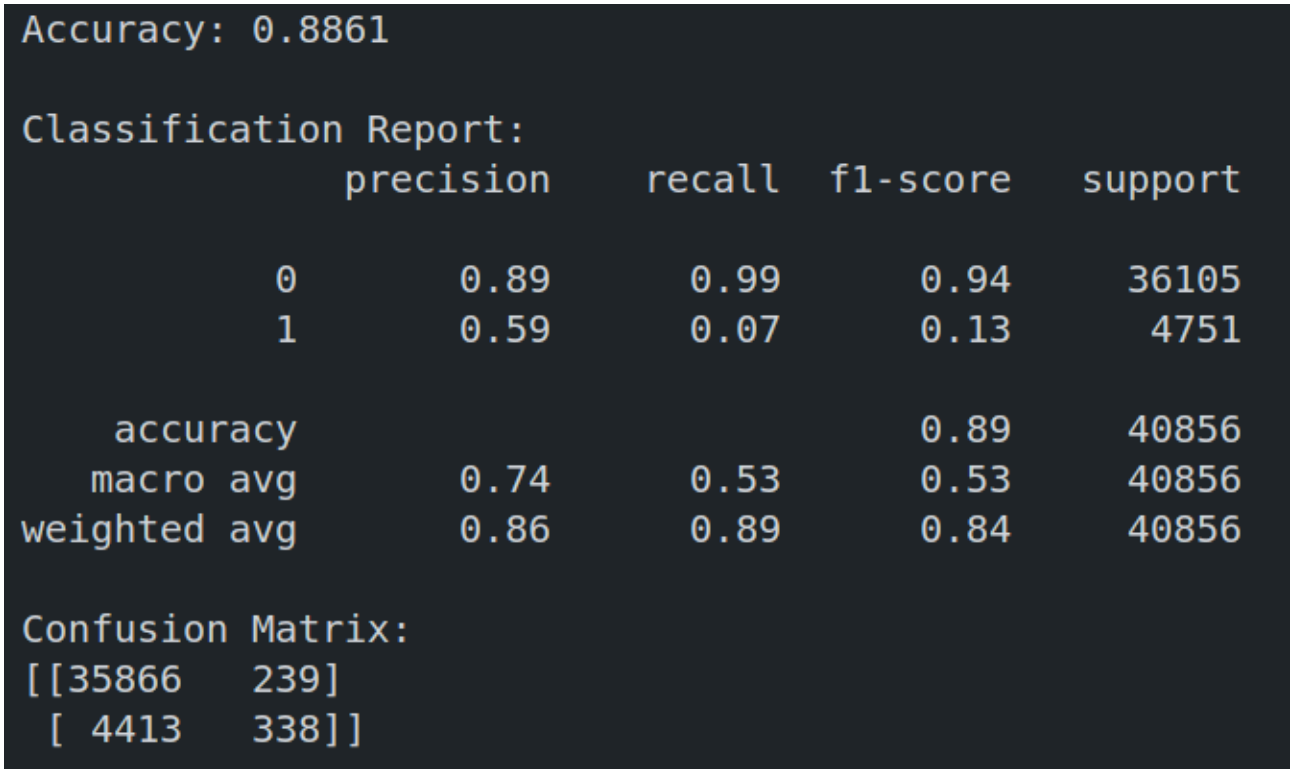

Figure 16: LigthGBM Model

2.6 Probabilistic and Distance-Based Models

For the binary classification task, we moved beyond tree-based ensembles to evaluate models that rely on linear separability and feature proximity. The pipeline was designed with strict preprocessing standards—Standard Scaling and ANOVA-based Feature Selection—to ensure these models performed optimally.

2.6.1 Logistic Regression

We implemented Logistic Regression as a parametric baseline to test the hypothesis that the decision boundary between risky and non-risky profiles is linear.

- **Configuration:** The model was instantiated with `max_iter=500` and a fixed random state. The increased iteration limit was crucial to ensure the optimization algorithm (lbfgs) converged fully given the high-dimensional sparse matrix resulting from One-Hot Encoding.
- **Mechanism:** This model computes the weighted sum of the standardized input features (IncomeToLoanRatio, Age, etc.) and passes the result through a sigmoid function to output a probability between 0 and 1.

2.6.2 Gaussian Naive Bayes

A Gaussian Naive Bayes classifier was employed as a probabilistic baseline.

- **Assumption:** This model assumes that the continuous features associated with each class are distributed according to a Gaussian (Normal) distribution. While this assumption is strong, the model is highly efficient and resistant to overfitting when feature independence holds.

- **Implementation:** We utilized the standard `GaussianNB` implementation from Scikit-Learn. It served as a high-bias, low-variance benchmark to contrast with the lower-bias ensemble methods.

2.6.3 K-Nearest Neighbors (KNN)

We implemented a K-Nearest Neighbors classifier to capture local non-linear patterns based on the similarity between financial profiles.

- **Configuration:** The model was set to use $k = 5$ neighbors (`n_neighbors=5`).
- **Optimization Strategy:** KNN is computationally expensive ($\mathcal{O}(N)$) during prediction. To address this, our Python code included a conditional optimization: if the training set exceeded 20,000 samples, the Grid Search phase was restricted to a subset of the data. This allowed for rapid prototyping without sacrificing the final model's performance, as the best-selected model was eventually retrained on the full dataset.

3 Multi-class Classification Problem

3.1 Unsupervised Clustering Models

To explore the underlying structure of spend behavior and identify user segments without relying solely on labeled data, several unsupervised clustering algorithms were implemented. Prior to clustering, the high-dimensional feature space (resulting from One-Hot Encoding) was reduced using Principal Component Analysis (PCA) to retain 95% of the explained variance, ensuring computationally efficient and robust distance calculations.

3.1.1 K-Means Clustering

K-Means clustering was applied to the PCA-transformed data to identify distinct user groups. A systematic grid search was conducted for cluster counts ranging from $K = 2$ to $K = 10$. The optimal number of clusters was determined by analyzing the Elbow Method (Inertia) and maximizing the Silhouette Score. The analysis indicated that $K = 2$ provided the best separation. The resulting clusters were profiled against the actual `spend_category` to assess their ability to differentiate high and low-value travelers.

For the test set submission, the K-Means model predicted cluster labels directly. These labels were then mapped to the most frequent spend category observed within each cluster in the training set.

Silhouette Score: 0.640
Kaggle Score: 0.479

3.1.2 Hierarchical Clustering

Hierarchical Agglomerative Clustering was implemented to visualize the nested structure of the data. Using Ward's linkage method to minimize variance within clusters, a Dendrogram was generated on a random sample of the dataset to identify natural groupings. This method served as a validation step for the cluster counts derived from K-Means.

Since Hierarchical Clustering does not support a direct `predict` method for new data, a K-Nearest Neighbors (KNN) classifier was trained on the cluster labels generated by the algorithm. This KNN model was then used to project cluster assignments onto the test dataset for submission.

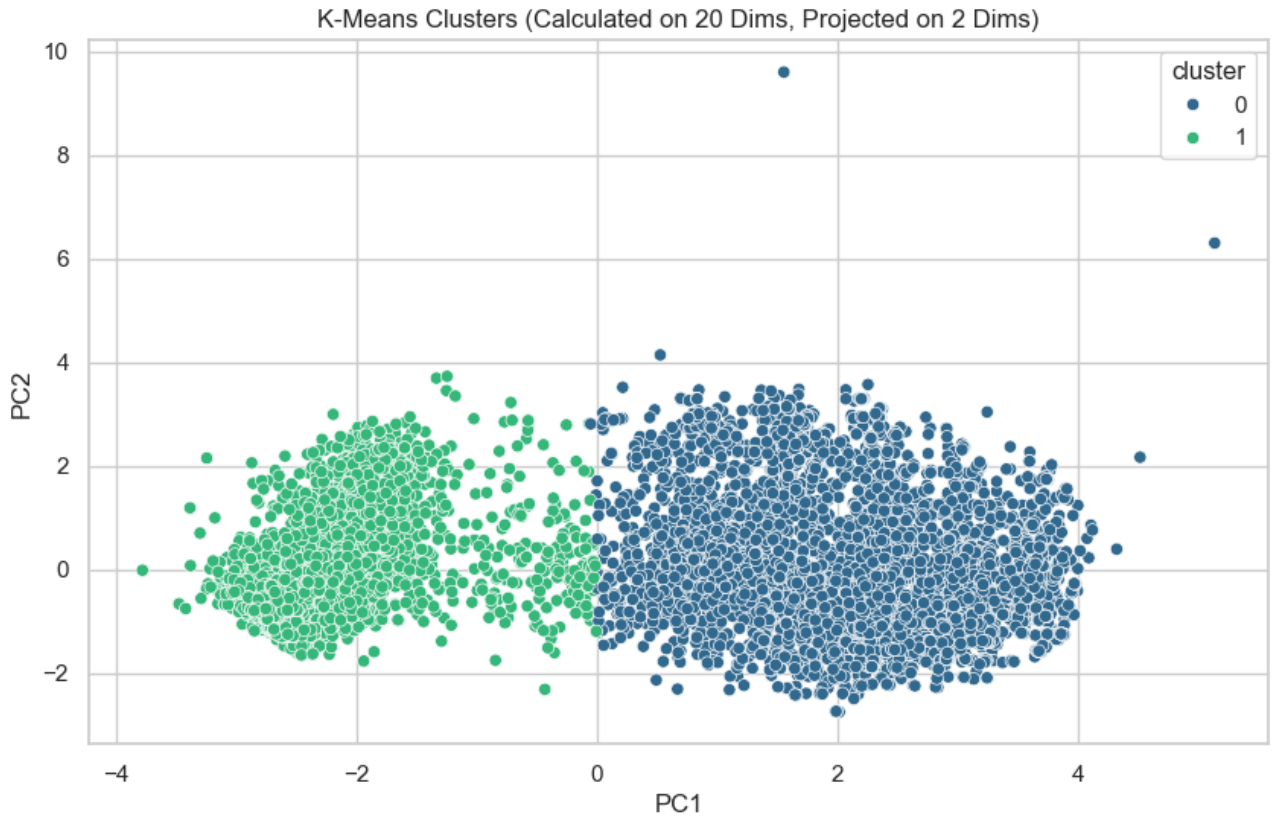

Figure 17: K-Means Clusters (PCA Projection)

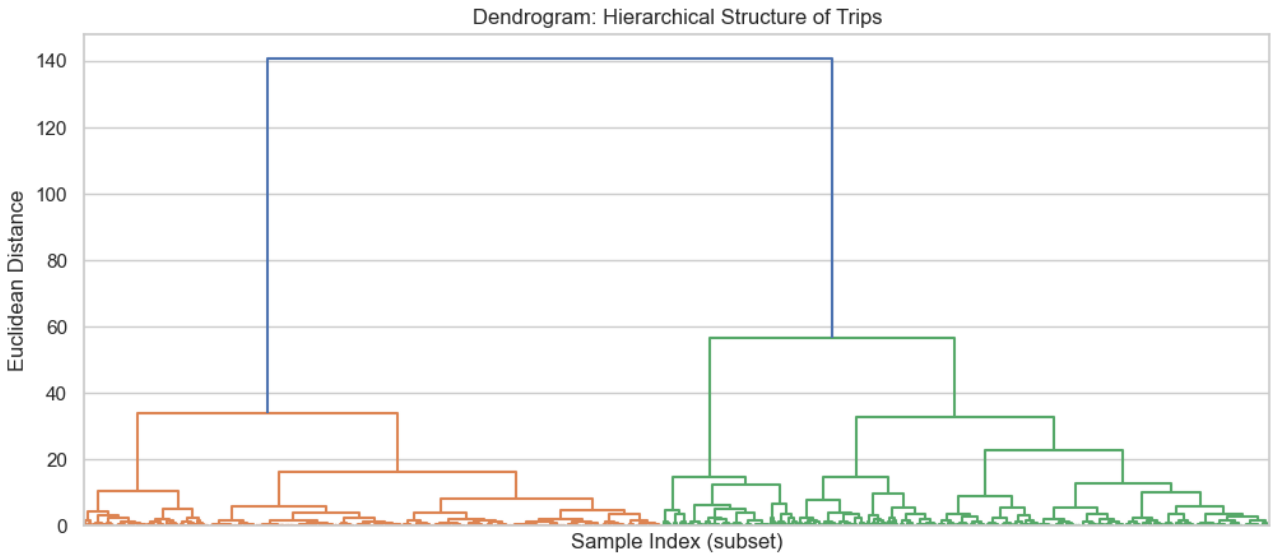

Figure 18: Hierarchical Clustering Dendrogram

Kaggle Score: 0.480

3.1.3 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was utilized to identify arbitrary shaped clusters and detecting outliers (noise). A hyperparameter tuning loop was executed to find the optimal `eps` (neighborhood distance) and `min_samples`. This approach allowed for the isolation of anomalous trip behaviors that did not fit into standard density regions.

Similar to the Hierarchical approach, DBSCAN does not predict on unseen data. Therefore, a KNN classifier was trained on the resulting valid clusters (excluding noise) to predict labels for the test set, mapping low-density outliers to the baseline spend category.

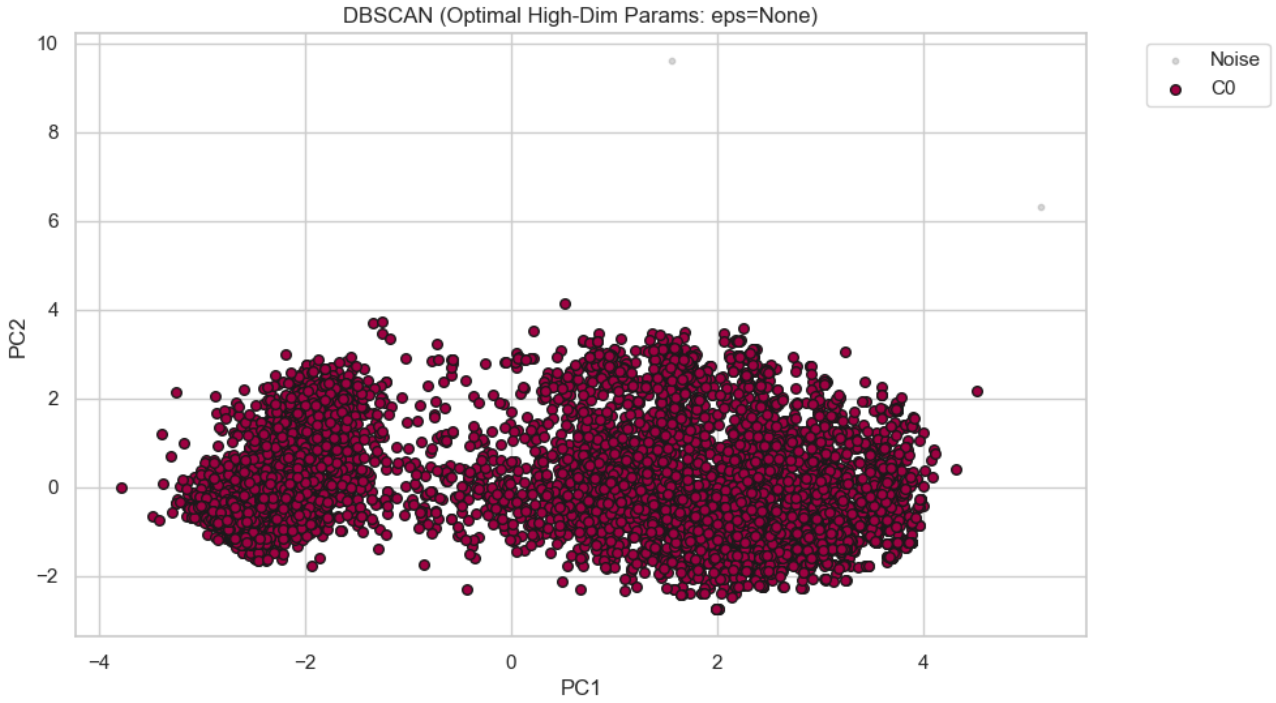

Figure 19: DBSCAN Clustering with Noise Detection

Kaggle Score: 0.207

3.2 Gaussian Mixture Models (GMM)

To capture more complex cluster geometries and provide probabilistic cluster assignments, a Gaussian Mixture Model (GMM) was implemented. Unlike K-Means, which assumes spherical clusters, GMM accommodates clusters of varying shapes and sizes via covariance matrices. To ensure stable estimation of these matrices and mitigate the curse of dimensionality, the feature space was reduced using Principal Component Analysis (PCA) to retain 95% of the explained variance (resulting in 20 components).

A grid search was conducted to optimize the model structure, testing component counts ranging from 2 to 10 and four covariance types (*full*, *tied*, *diag*, *spherical*). The Bayesian Information Criterion (BIC) was employed as the evaluation metric to select the model that best maximized likelihood

while penalizing complexity. The analysis determined that a model with **10 components** and a **full** covariance type provided the optimal balance.

For the final predictions, the model assigned test data points to clusters based on the highest posterior probability. These unsupervised cluster labels were then mapped to the `spend_category` by identifying the most frequent class (mode) within each cluster in the training data.

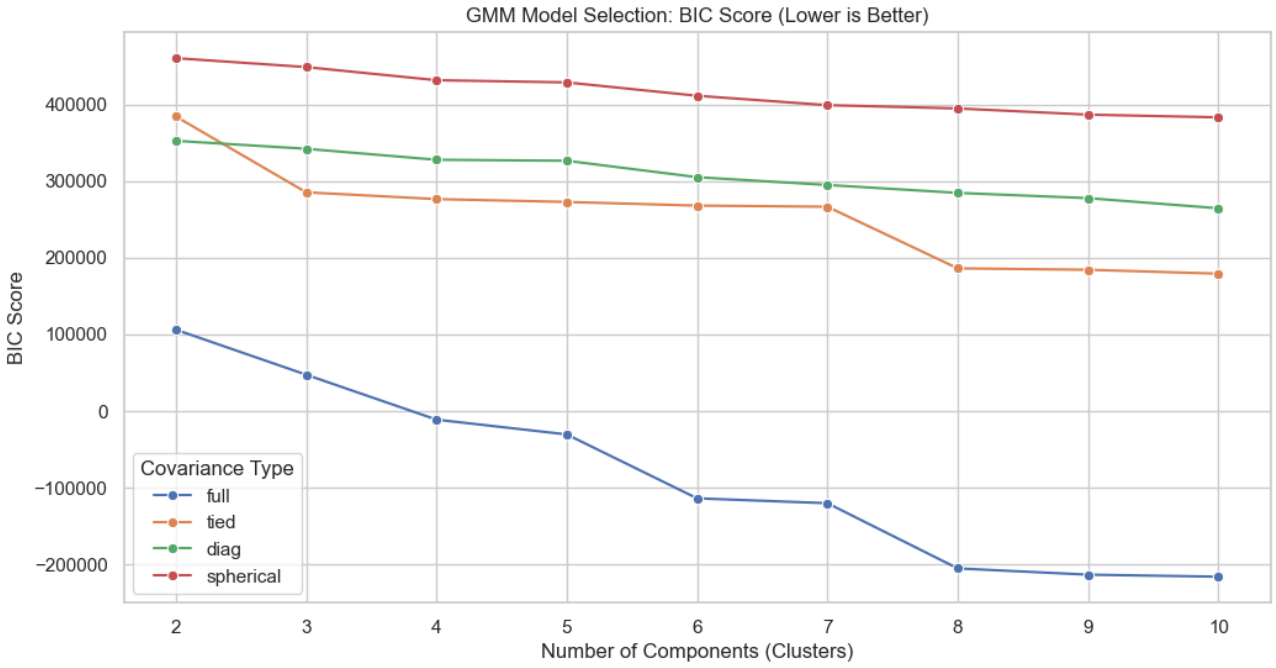

Figure 20: GMM Model Selection (BIC Score)

Kaggle Score: 0.480

3.3 Support Vector Machine (SVM)

To capture non-linear boundaries between spend categories, a Support Vector Machine (SVM) classifier was implemented. Given the high dimensionality resulting from the feature engineering process, a Principal Component Analysis (PCA) step was integrated into the pipeline. The PCA reduction retained 20 components, explaining 95% of the dataset's variance, which significantly optimized the SVM's convergence and computational speed.

The model underwent hyperparameter tuning via Grid Search to determine the optimal kernel function (comparing RBF, Linear, and Polynomial) and the regularization parameter C. The RBF kernel was selected for its superior ability to handle the complex data distribution. To interpret the model's behavior, a decision boundary map was generated by projecting the data onto the first two principal components, visualizing the separation regions for Low, Medium, and High spenders.

Validation Accuracy: 0.551

3.4 Neural Network (Multi-Input MLP)

To effectively capture complex interactions between categorical and numerical features, a Multi-Input Neural Network was developed using the TensorFlow/Keras Functional API. This architecture allows

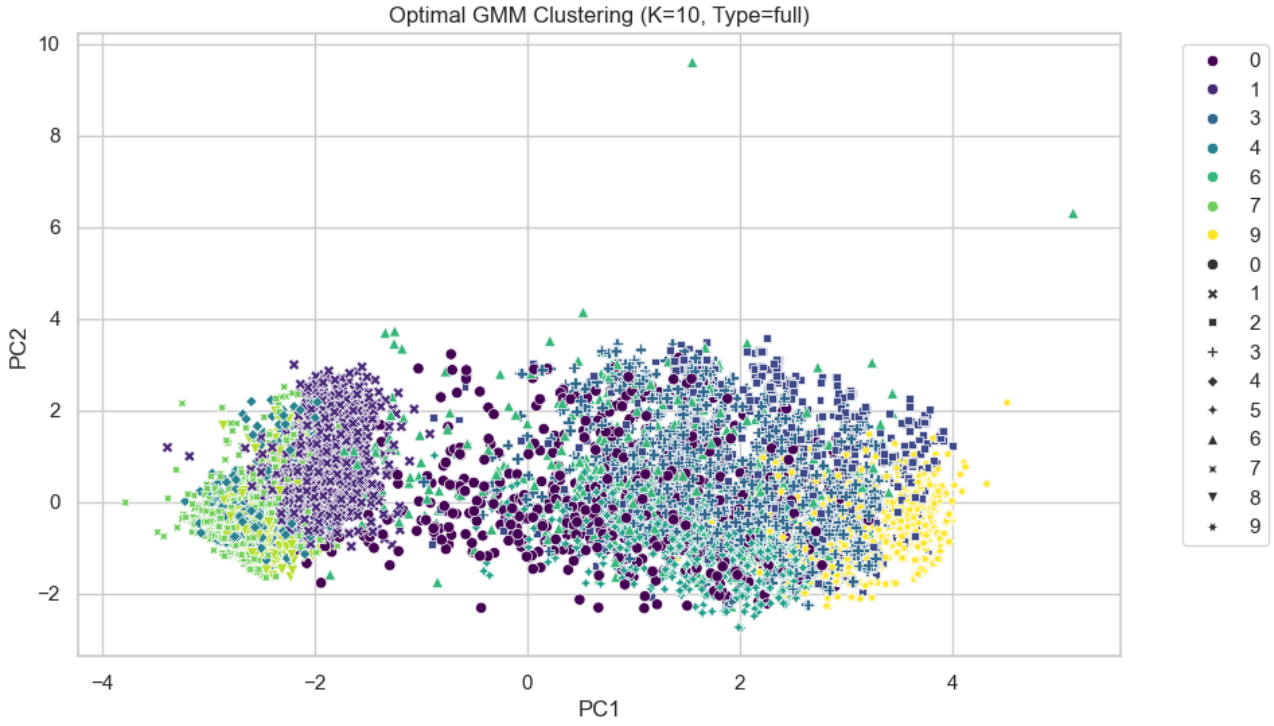

Figure 21: Optimal GMM Clustering Projection (K=10)

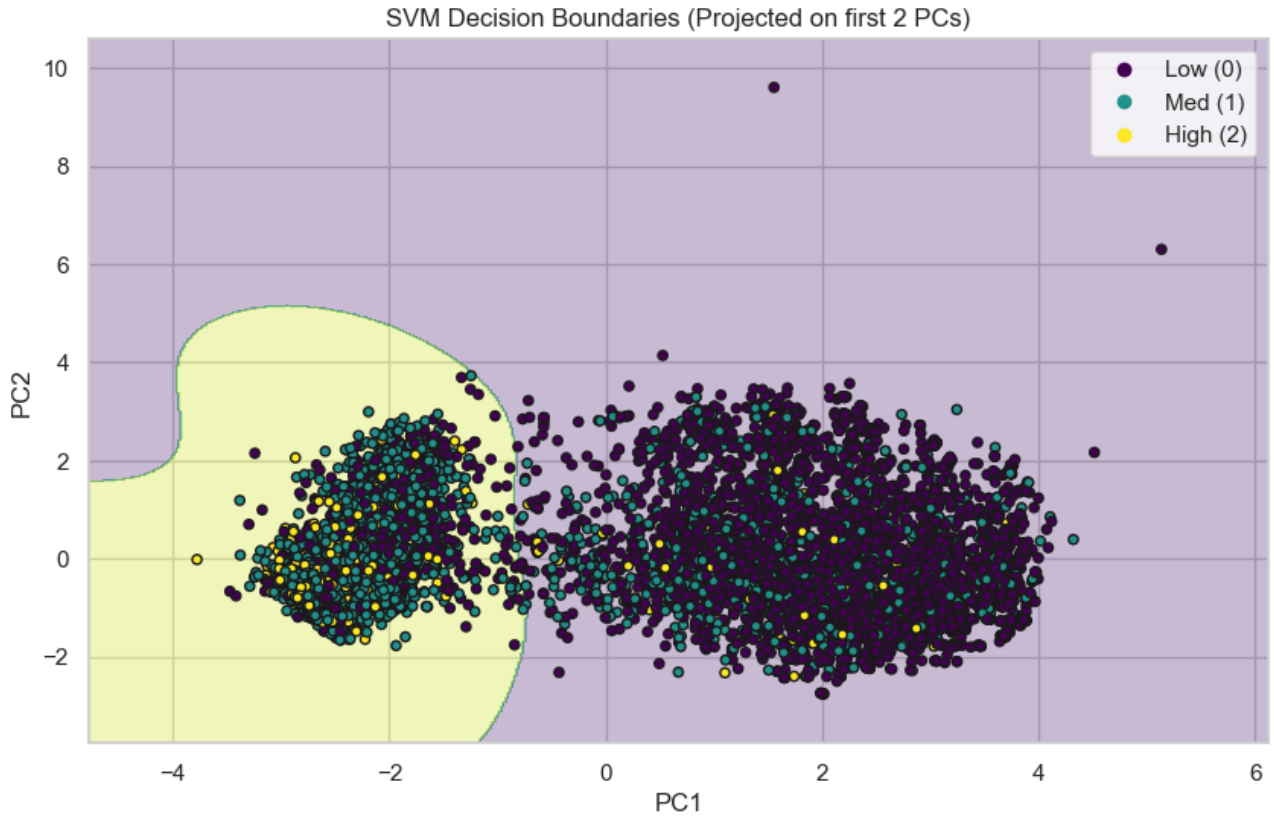

Figure 22: SVM Decision Boundaries (PCA Projection)

for distinct processing pipelines for different data types before fusion.

Architecture:

- **Categorical Features:** Processed via learned **Embedding Layers**, where the embedding dimension was dynamically scaled based on feature cardinality ($\min(50, \frac{n}{2}+1)$). This allows the model to learn dense representations for high-cardinality features like 'country' or 'main_activity'.
- **Numerical Features:** Standardized and fed directly into the concatenation layer.
- **Hidden Layers:** The inputs were concatenated and passed through a series of Dense layers equipped with **Batch Normalization** for stability and **Dropout** for regularization. The final output layer utilized a **Softmax** activation function for multi-class probability distribution.

A manual Grid Search was conducted to optimize hyperparameters, specifically the number of hidden units $(128, 256)$, dropout rates $(0.3, 0.5)$, and learning rates. The optimal configuration was identified as `{units: 256, dropout: 0.3, lr: 0.001}`.

The model was trained with **Early Stopping** to mitigate overfitting. Post-tuning, the model was retrained on the full training dataset to generate final predictions for the test set.

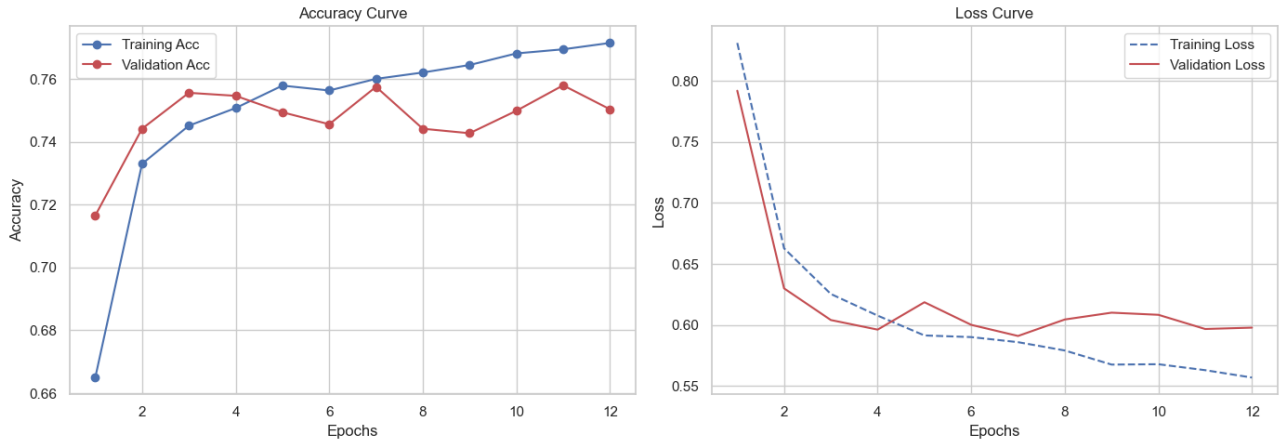

Figure 23: Neural Network Training History (Accuracy & Loss)

Best Validation Accuracy: 0.758
Kaggle Score: 0.633

3.5 Tree-Based and Ensemble Models

3.5.1 Decision Tree:

A Decision Tree Classifier was implemented as the baseline model for the multiclass travel behaviour prediction task. The dataset was split into an 80–20 stratified train–validation split to preserve class proportions across traveler spending categories. A unified preprocessing pipeline was applied using One-Hot Encoding for categorical variables and median imputation for numerical variables to address missing values. The model was trained with a maximum depth of 10 to balance interpretability with the ability to capture meaningful decision boundaries.

After training, the classifier was evaluated using accuracy, class-wise precision/recall, and a confusion matrix to assess its ability to differentiate between the various spending categories. The Decision

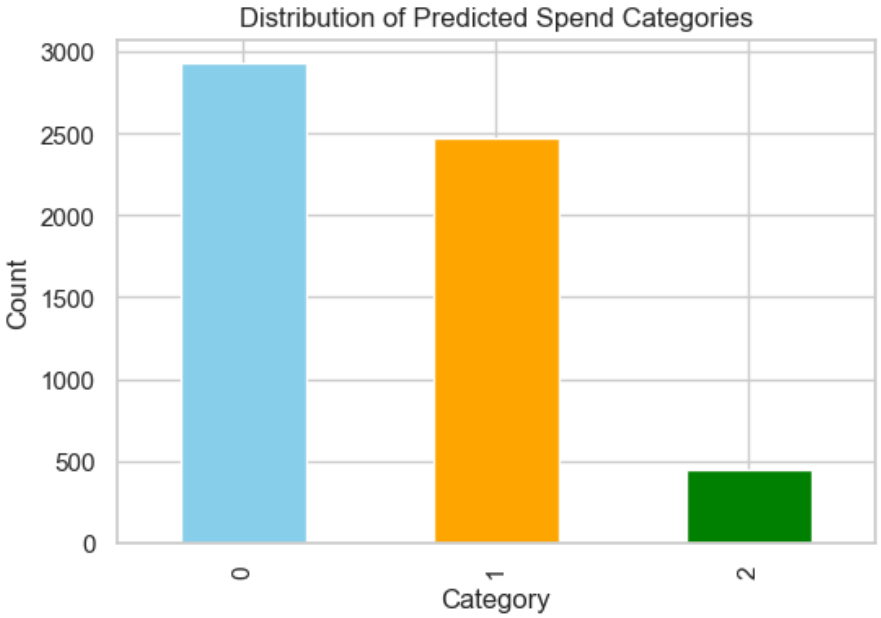

Figure 24: Distribution of Predicted Spend Categories

Tree provided a transparent, interpretable structure for understanding how demographic and trip-related attributes influence traveler behaviour. Final predictions on the test dataset were exported as a submission-ready CSV file.

Kaggle Score: 0.622

3.5.2 Random Forest:

A Random Forest Classifier was developed to model complex, non-linear relationships within the travel behaviour dataset. Using an 80–20 stratified split, the model was trained on demographic and trip-related predictors after preprocessing through a unified pipeline. Categorical features were imputed and encoded using One-Hot Encoding, while numerical variables were transformed using median imputation, ensuring consistent handling of missing values across the entire workflow. The classifier was integrated within a full pipeline and optimized using GridSearchCV, tuning parameters such as the number of estimators, maximum tree depth, and minimum split criteria.

The best-performing model obtained from hyperparameter tuning was evaluated on the reserved validation set using accuracy, class-wise classification metrics, and a confusion matrix to gauge its effectiveness in distinguishing between travel spending categories. Owing to its ensemble nature, the Random Forest model demonstrated robust performance and reduced variance compared to a single decision tree. Final predictions were generated for the test dataset using the optimized model and saved in submission-ready format.

Kaggle Score: 0.624

```
Evaluating on Validation Set:
Accuracy: 0.7508

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.83      0.84      1069
           1       0.66      0.77      0.71       801
           2       0.69      0.30      0.41       229

    accuracy                           0.75      2099
   macro avg       0.73      0.63      0.65      2099
weighted avg       0.75      0.75      0.74      2099

Confusion Matrix:
[[892 173   4]
 [158 616  27]
 [ 16 145  68]]
```

Figure 25: Decision Tree Model

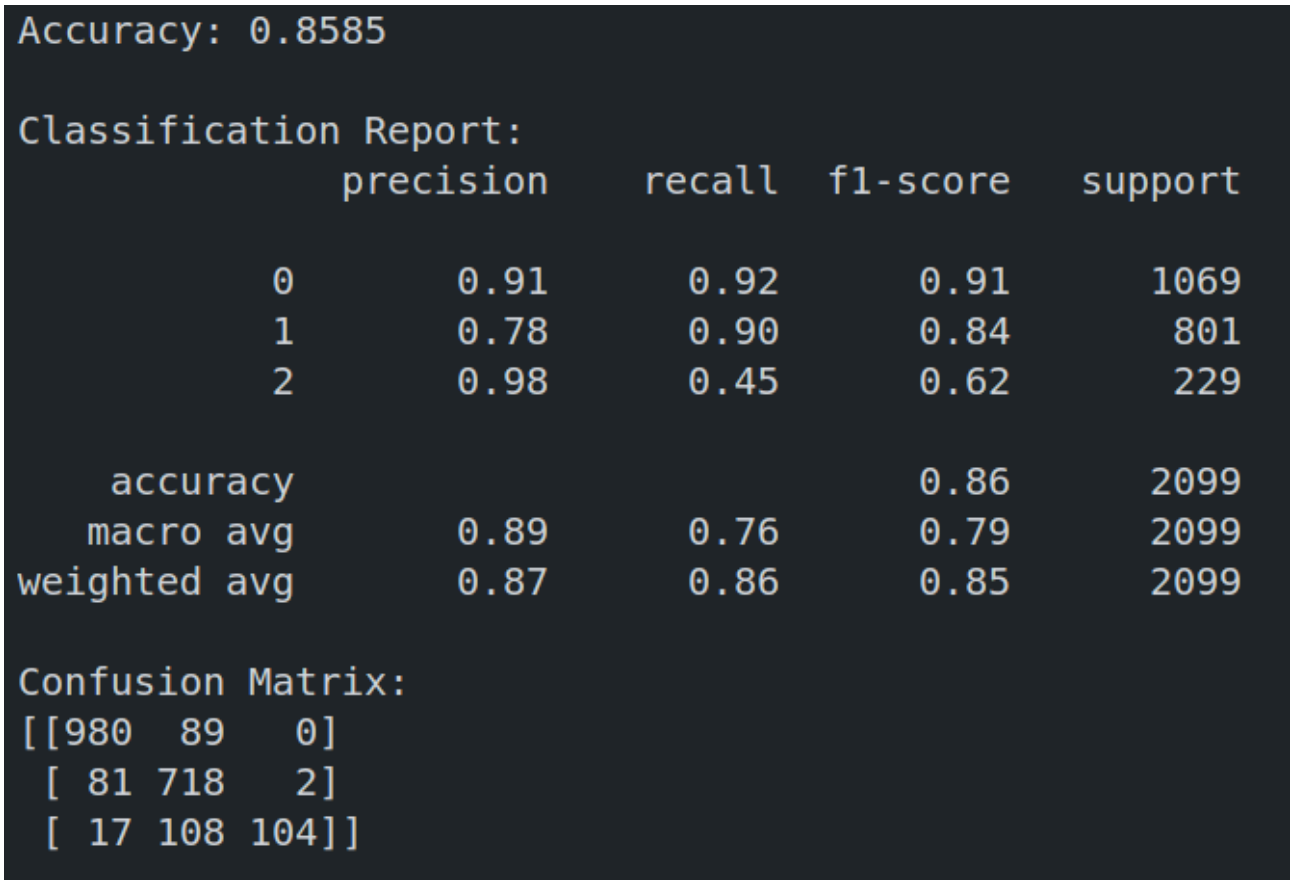

Figure 26: Random Forest Model

3.5.3 XGBoost:

To capture complex patterns within the travel behaviour dataset, an XGBoost classifier was implemented for the multiclass prediction of spending categories. The preprocessing pipeline applied median imputation for numerical variables and One-Hot Encoding for categorical attributes, ensuring consistent handling of missing values and unseen categories. Since XGBoost requires numerical targets, class labels were encoded using LabelEncoder, and an 80–20 stratified split was used to maintain class balance during training and evaluation. The model was configured with multi-class soft probability outputs and optimized hyperparameters such as 150 estimators, max depth of 6, and an 0.8 subsampling ratio to balance model expressiveness and generalization.

The trained model was assessed on the validation set using accuracy, macro-averaged ROC–AUC, a full classification report, and a confusion matrix. Its gradient-boosting framework allowed it to learn non-linear feature interactions more effectively than traditional tree-based models, producing competitive and stable multiclass performance. Finally, the validated pipeline was used to generate class-label predictions for the test dataset, forming the final submission file.

```
Evaluating Model on Validation Set (Multi-Class)...
Accuracy: 0.7608
Macro ROC-AUC (OvR): 0.8868

Classification Report:
              precision    recall  f1-score   support

         0.0       0.84      0.84      0.84      1069
         1.0       0.68      0.77      0.72       801
         2.0       0.72      0.35      0.47       229

    accuracy                           0.76      2099
   macro avg       0.74      0.65      0.68      2099
weighted avg       0.76      0.76      0.75      2099

Confusion Matrix:
[[903 166   0]
 [156 613  32]
 [ 19 129  81]]
```

Figure 27: XGBoost Model

Kaggle Score: 0.662

3.5.4 LightGBM:

To capture subtle, high-dimensional patterns in traveller behaviour, a LightGBM-based multiclass model was developed to predict spending categories. The pipeline incorporated robust preprocessing, including median imputation for numerical fields and One-Hot Encoding for categorical attributes, ensuring consistency across both training and test data. Since LightGBM requires numerical class labels, a LabelEncoder was applied to convert the raw target categories into integer form. The model was trained on an 80–20 stratified split to maintain class balance and configured with 300 estimators, a learning rate of 0.05, and a leaf size of 31—parameters chosen to balance performance with training stability.

The model's performance on the validation split was evaluated using accuracy, a detailed classification report, and a confusion matrix, allowing us to assess how well LightGBM captured the multi-class structure of the dataset. Thanks to its gradient boosting framework and optimized tree algorithm, the model demonstrated strong generalization and efficient handling of both numeric and high-cardinality categorical features. The trained pipeline was then used to generate final predictions for the test set, which were mapped back to human-readable class labels for submission.

```
Accuracy: 0.7570

Classification Report:
              precision    recall  f1-score   support

         0.0       0.84      0.85      0.84      1069
         1.0       0.68      0.74      0.71       801
         2.0       0.65      0.38      0.48       229

    accuracy                           0.76      2099
   macro avg       0.72      0.66      0.68      2099
weighted avg       0.75      0.76      0.75      2099

Confusion Matrix:
[[906 161   2]
 [161 595  45]
 [ 18 123  88]]
```

Figure 28: LightGBM Model

Kaggle Score: 0.657

3.6 Probabilistic and Distance-Based Models

For the multiclass travel behaviour problem, we implemented a sophisticated pipeline utilizing `GridSearchCV` to optimize probabilistic and distance-based models. A key component of this section was the use of **Robust Scaling** to handle outliers in travel duration and spending data.

3.6.1 Multinomial Logistic Regression

We deployed a Logistic Regression model specifically configured for multiclass classification.

- **Multinomial Loss:** The model was set to `multi_class='multinomial'`, which uses the Softmax function to predict the probability distribution across all possible spending categories, rather than training separate binary classifiers.
- **Hyperparameter Tuning:** We performed a Grid Search over the regularization parameter C ($[0.01, 0.1, 0.5, 1, 5]$) and the solvers (`lbfgs`, `newton-cg`). The tuning revealed that stronger regularization (lower C) helped prevent overfitting on the high-cardinality country data.

3.6.2 K-Nearest Neighbors (KNN)

KNN was implemented to classify travelers based on their similarity to existing profiles in the feature space.

- **Grid Search Optimization:** We systematically tested different values for k (`n_neighbors` $\in [10, 15, 20, 30]$) using 5-fold cross-validation.
- **Result:** The search identified the optimal balance between noise reduction (high k) and local sensitivity (low k). The final model selection logic automatically compared the best CV score of the KNN model against the Logistic Regression model to determine the superior approach for this specific dataset.

4 Conclusion

This project provided a comprehensive exploration of machine learning techniques across two distinct supervised learning tasks—binary financial risk prediction and multiclass travel behaviour classification. By systematically evaluating a wide spectrum of models, including tree-based ensembles, probabilistic methods, distance-based classifiers, margin-based models, neural networks, and multiple unsupervised clustering algorithms, we were able to observe how different approaches respond to varying data characteristics and problem complexities.

Across both datasets, ensemble methods such as Random Forest, XGBoost, and LightGBM consistently delivered strong performance due to their ability to model non-linear patterns and handle heterogeneous feature types. Linear and probabilistic models offered fast baselines, while KNN and SVM highlighted the strengths and limitations of distance- and margin-based learning in high-dimensional spaces. Unsupervised techniques—K-Means, Hierarchical Clustering, DBSCAN, and GMM—proved useful for understanding latent structures and validating supervised decision boundaries. Neural networks further demonstrated the advantages of learned representations, particularly for the multiclass problem.

Overall, the project successfully bridged theoretical concepts with practical implementation, offering hands-on insights into model selection, preprocessing strategies, hyperparameter tuning, and performance evaluation. The comparative analysis reinforces the importance of aligning algorithmic choices with data properties and task requirements, ultimately strengthening our ability to design robust prediction systems for real-world applications.