ML Project Report

Syed Naveed Mohammed (IMT2023119)
Saheem Reshi (IMT203051)
Satyam Dewangan (IMT2023545)

3rd December 2025

1 Introduction

This report presents two machine learning projects carried out to analyze structured real-world data and develop predictive models for practical decision-making scenarios.

The first dataset, **Financial Risk Profiling**, focuses on predicting whether a loan applicant is financially risky. Models such as Logistic Regression, Support Vector Machines, Naive Bayes, and a neural network–based multilayer perceptron were trained.

The second dataset, **Travel Behavior Insights**, aims to classify tourists into spend categories based on travel-related characteristics such as group composition, trip duration, accommodation choices, and activity preferences.

The report summarizes the methodology, hyperparameters, and performance of each model, providing clear conclusions for both classification tasks.

GitHub Repository: `https://github.com/saheemReshi/ML_Project_part2`

2 Binary Classification = Financial Risk Profiling

2.1 Exploratory Data Analysis (EDA)

- Dataset contains approximately 200,000 rows with a mix of numeric and categorical features.
- Target variable: `RiskFlag`.
- Visualizations:
  - **Boxplots** to detect outliers and determine clipping thresholds.
  - **Histograms** to examine distribution and skewness.
  - **Summary statistics** to verify ranges and detect anomalies.

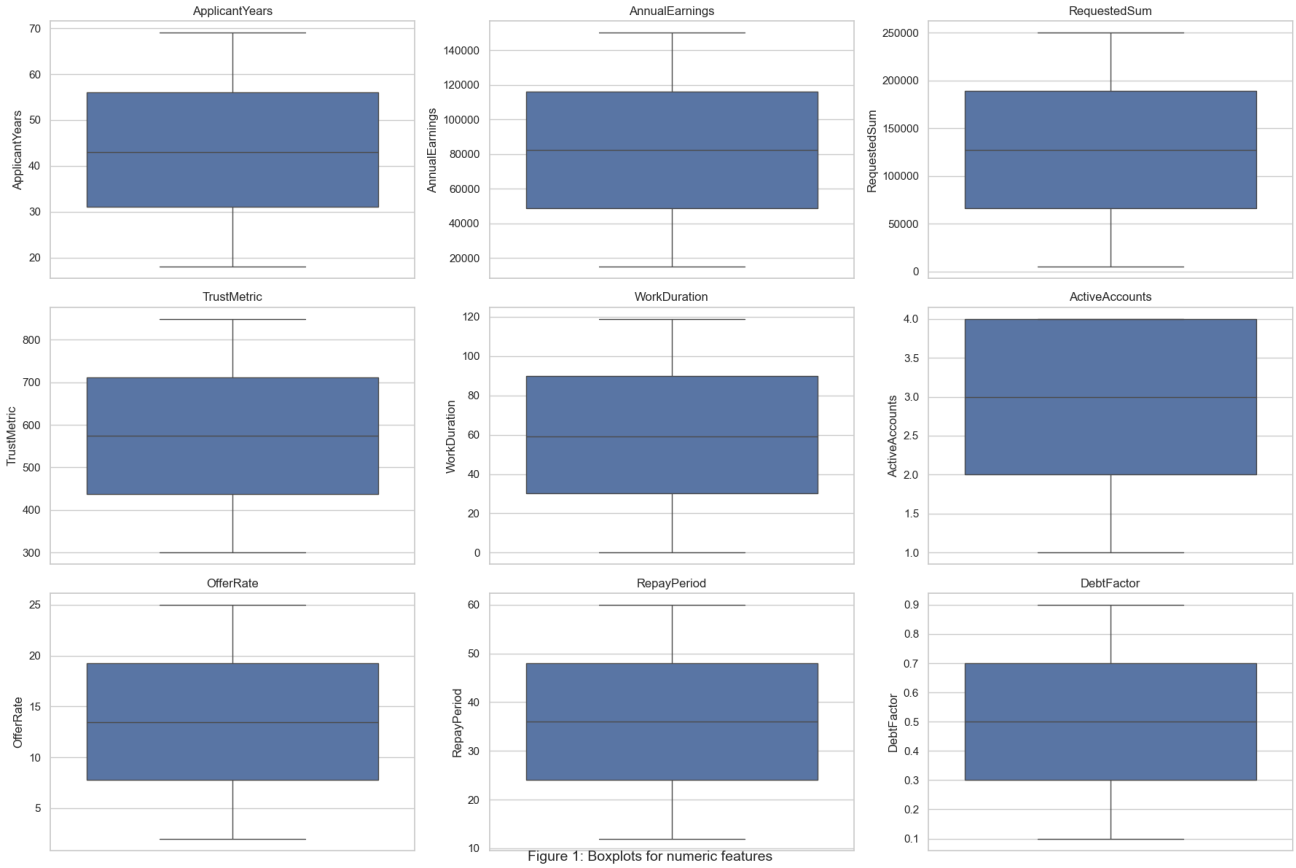

Figure 1: Boxplots for numeric features

Figure 1: Boxplots for numeric features

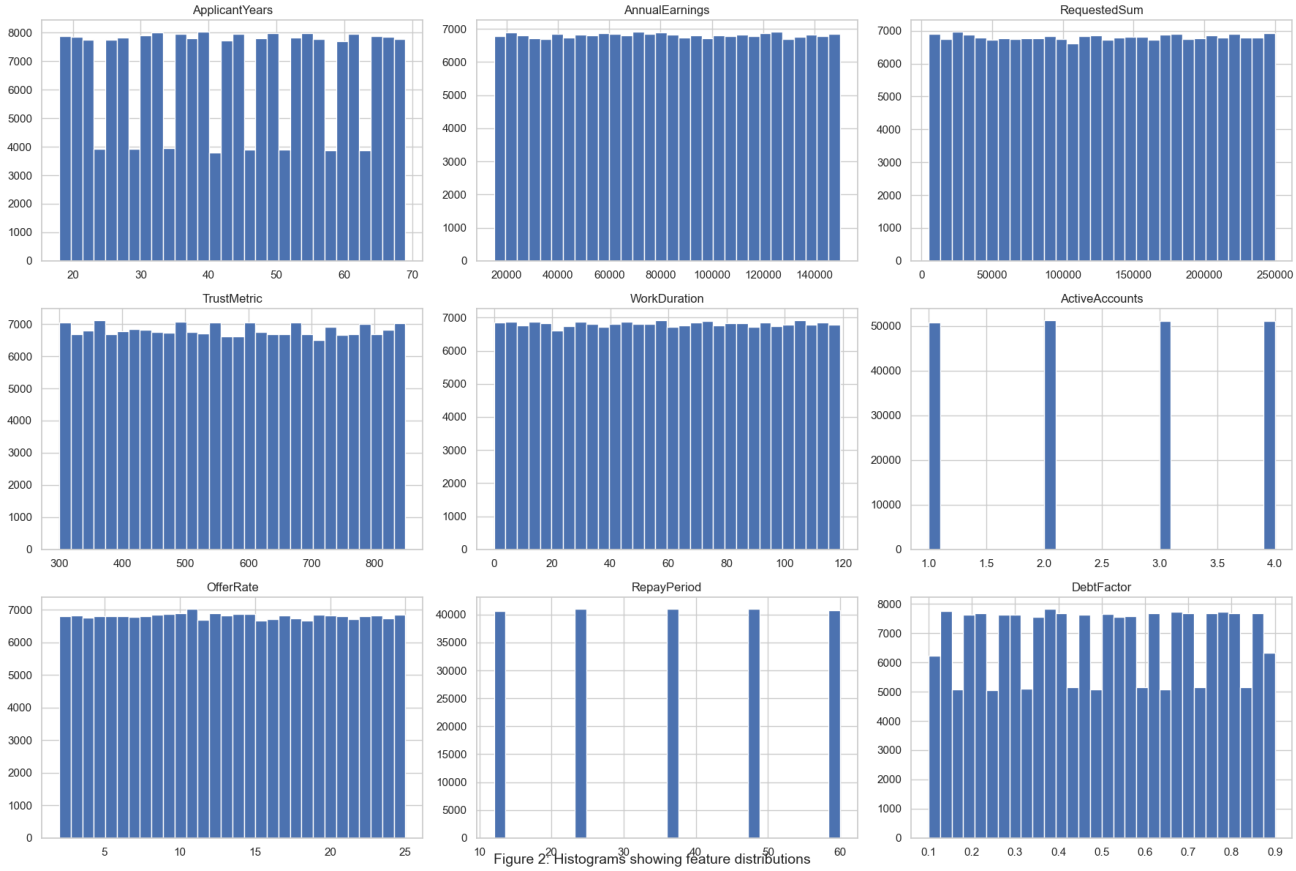

Figure 2: Histograms showing feature distributions

Figure 2: Histograms showing feature distributions

	count	mean	std	min	25%	50%	75%	max
ApplicantYears	204277.0	43.48934	14.995191	18.0	31.0	43.0	56.0	69.0
AnnualEarnings	204277.0	82506.22798	38952.103374	15000.0	48878.0	82400.0	116247.0	149999.0

Figure 3: Summary statistics for important features

	count	mean	std	min	25%	50%	75%	max
RequestedSum	204277.0	127547.496395	70855.064746	5001.0	66059.0	127603.0	188843.0	249999.0
TrustMetric	204277.0	574.075500	158.877098	300.0	437.0	574.0	712.0	849.0

Figure 4: Summary statistics for important features

	count	mean	std	min	25%	50%	75%	max
WorkDuration	204277.0	59.508511	34.645589	0.0	30.0	59.0	90.0	119.0
ActiveAccounts	204277.0	2.502078	1.116898	1.0	2.0	3.0	4.0	4.0

Figure 5: Summary statistics for important features

	count	mean	std	min	25%	50%	75%	max
OfferRate	204277.0	13.488147	6.636060	2.0	7.76	13.45	19.24	25.0
RepayPeriod	204277.0	36.010926	16.944827	12.0	24.00	36.00	48.00	60.0

Figure 6: Summary statistics for important features

	count	mean	std	min	25%	50%	75%	max
DebtFactor	204277.0	0.500579	0.230914	0.1	0.3	0.5	0.7	0.9

Figure 7: Summary statistics for important features

2.2 Data Preprocessing

Prior to training the models, the dataset underwent a structured preprocessing pipeline designed to handle heterogeneous feature types and improve the robustness of model performance.

Binary Feature Encoding. Three binary attributes *OwnsProperty*, *FamilyObligation*, and *JointApplicant* were converted from their original `Yes/No` format to numerical values using a simple binary mapping (`Yes` $\rightarrow$ 1, `No` $\rightarrow$ 0). This transformation ensures the order is maintained (yes ¿ no in the context here).

Ordinal Encoding. Two features with inherent ordering were encoded using an ordinal scheme. The levels for *QualificationLevel* were defined as

$$\text{High School} < \text{Bachelor's} < \text{Master's} < \text{PhD},$$

while the *WorkCategory* levels were ordered as

$$\text{Unemployed} < \text{Part-time} < \text{Full-time} < \text{Self-employed}.$$

An `OrdinalEncoder` was used to map these categories to integer values, with unseen categories during inference assigned a default value of -1.

Nominal Feature Handling. The nominal variables *RelationshipStatus* and *FundUseCase*, which do not possess any meaningful ordering, were converted into indicator variables using `OneHotEncoder`. This prevents the models from inferring false ordinality between categories.

Outlier Removal. To reduce the influence of extreme values, outliers were identified within all numerical features using the interquartile range (IQR) rule. Specifically, observations lying outside the interval

$$[Q_1 - 1.5 \times IQR,\ Q_3 + 1.5 \times IQR]$$

for any numerical attribute. But in the dataset, we found no outliers.

Feature Scaling and Final Transformation. A `ColumnTransformer` was used to unify preprocessing across feature groups. Numerical variables were standardized using `StandardScaler`, nominal variables were one-hot encoded, ordinal features were passed through as encoded integers, and binary variables were retained as-is.

2.3 Models, Hyperparameters, and Test Scores

Model	Preprocessing	Best Parameters	F1 Score	Notes
Logistic Regression	StandardScaler	`solver=lbfgs`, `class_weight=balanced`, `penalty=l2`	0.676	Linear model struggled to capture non-linear structure.
SVM (RBF)	RobustScaler	`kernel=rbf`, `C=3`, `gamma=scale`, `class_weight=balanced`	0.885	Strong performance due to non-linear separation capability.
Naive Bayes (Gaussian)	RobustScaler	Default	0.885	Surprisingly strong; feature distributions align well with Gaussian assumptions.
Neural Network (MLP)	RobustScaler	`hidden_layers=(64,32)`, `activation=relu`, `optimizer=adam`, `batch_size=32`	**0.887**	Best model; captured interactions between features effectively.

Table 1: Model Comparison and F1 Scores for Financial Risk Profiling

2.4 Observations

- The **Neural Network (MLP) achieved the highest F1 score** of 0.887.
- SVM and Naive Bayes tied closely behind with F1 scores of 0.885.
- Logistic Regression performed significantly worse, showing the dataset is not linearly separable.
- Non-linear models clearly outperform linear methods for this task.

3 Multi Class Classification = Travel Behavior Insights

3.1 Objective

The goal of this project is to predict the `spend_category` of travelers using trip-related information such as travel companions, stay duration, activities, and accommodations. This is a **multi-class classification** problem, and the aim is to identify models that generalize well on unseen data.

3.2 Exploratory Data Analysis (EDA)

- Dataset contains approximately 12,000 rows with a mix of numeric and categorical features.
- Target variable: `spend_category`.
- Key observations:
  - `num_males` and `num_females` are small integers with occasional outliers.
  - Stay durations (`mainland_stay_nights`, `island_stay_nights`) are skewed.
  - Many categorical features have missing values ($< 2\%$) or redundant information.
- Visualizations:
  - **Boxplots** to detect outliers and determine clipping thresholds.
  - **Histograms** to examine distribution and skewness.
  - **Summary statistics** to verify ranges and detect anomalies.

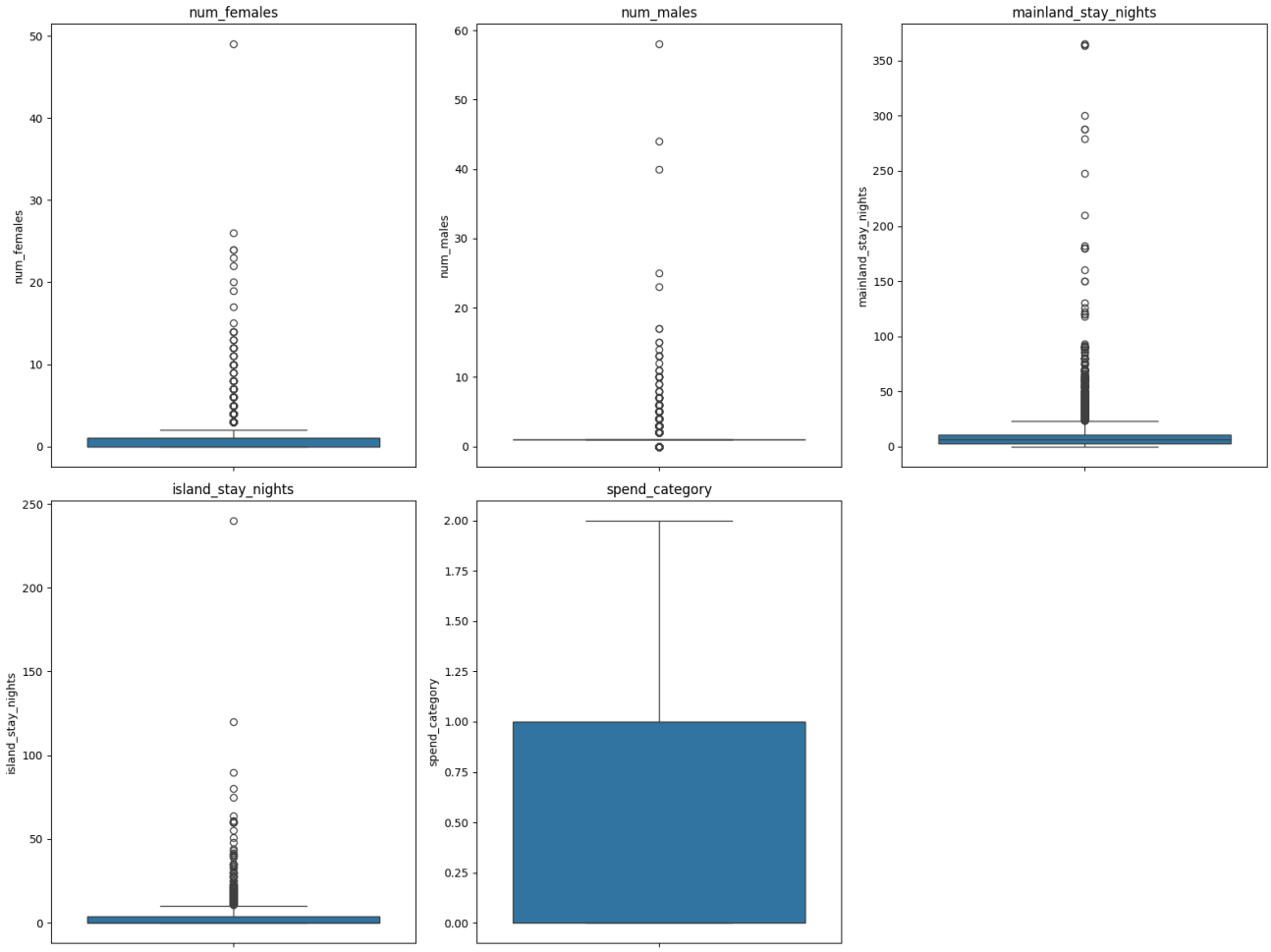

Figure 8: Boxplots for numeric features

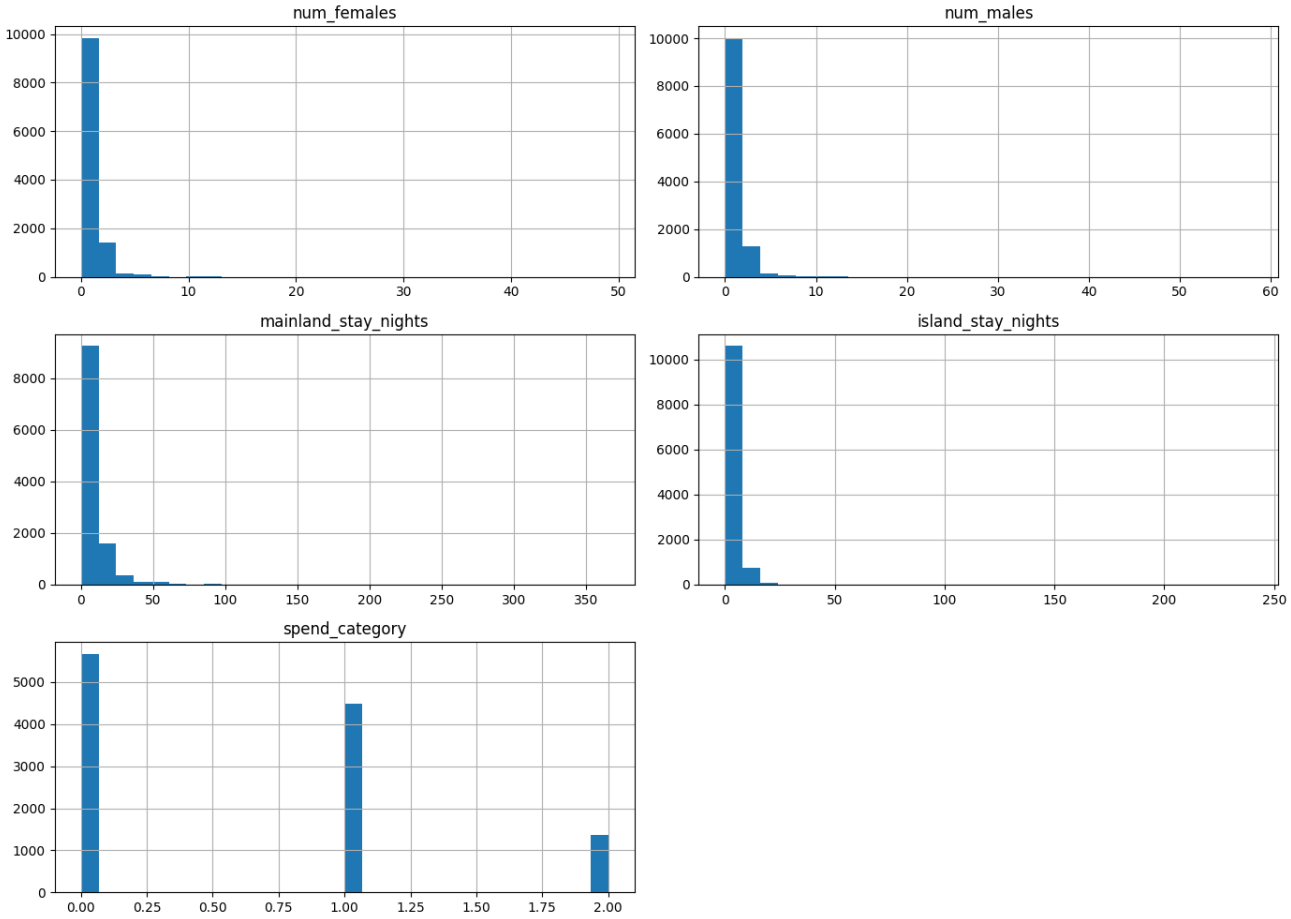

Figure 9: Histograms showing feature distributions

	num_females	num_males	mainland_stay_nights	island_stay_nights
count	11505.000000	11505.000000	11505.000000	11505.000000
mean	0.949066	1.012169	9.206780	2.522555
std	1.295324	1.273400	14.868802	5.170178
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	3.000000	0.000000
50%	1.000000	1.000000	6.000000	0.000000
75%	1.000000	1.000000	11.000000	4.000000
max	49.000000	58.000000	365.000000	240.000000

	spend_category
count	11505.000000
mean	0.625120
std	0.686026
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	2.000000

Figure 10: Summary statistics for important features

4 Preprocessing

A structured preprocessing pipeline was designed to clean the dataset, handle missing values, remove noisy features, engineer meaningful variables, and prepare the data for model training. All preprocessing rules were derived strictly from the training set and then applied consistently to both training and test data.

4.1 Handling Missing Values

4.1.1 Row-wise Decisions

- Rows with missing target values (`spend_category`) were dropped.
- For columns with very low missingness ($< 2\%$), corresponding rows were removed entirely to avoid introducing bias through imputation.

4.1.2 Column-wise Decisions

- Columns with more than 40% missing values were removed as they contribute little meaningful signal.
- The column `arrival_weather` was dropped manually because it had no conceptual relationship with spend behavior.

4.2 Categorical Imputation Rules

For selected categorical features, domain-aware imputations were applied:

- `travel_companions` $\rightarrow$ "Alone"
- `days_booked_before_trip` $\rightarrow$ "61–90"

These rules were derived from the most common and reasonable default choices based on the distribution of each feature.

4.3 Inferring Trip Duration

The dataset included partially missing or inconsistent trip duration fields. A new feature, `total_trip_days`, was inferred using the function:

$$\texttt{total\_trip\_days} = \begin{cases} \texttt{total\_trip\_days}, & \text{if already provided} \\ \text{bucket}(\text{mainland\_stay\_nights} + \text{island\_stay\_nights}), & \text{otherwise} \end{cases}$$

Where the buckets were:

$$1\text{–}6, \quad 7\text{–}14, \quad 15\text{–}30, \quad 30+$$

This ensured consistency across records and avoided missing-trip-duration issues.

4.4 Outlier Handling

Domain-aware clipping (winsorization) was applied instead of the IQR rule since the features represent counts and stay durations with natural upper limits:

- `num_females`, `num_males` clipped to $[0, 5]$.
- `mainland_stay_nights` clipped to $[0, 30]$.
- `island_stay_nights` clipped to $[0, 21]$.

4.5 Feature Engineering

A new feature was created to better capture group-level dynamics:

$$\texttt{total\_people} = \texttt{num\_males} + \texttt{num\_females}.$$

After the combination, the original gender count columns were dropped.

4.6 Final Preprocessing Pipeline

After cleaning the dataset, approximately $\sim 11k$rows remained. The following transformations were applied before model training:

- Categorical columns were encoded using **One-Hot Encoding** with `handle_unknown=ignore`.
- Numerical columns were scaled using **RobustScaler** to reduce the effect of remaining outliers.
- The exact same transformation pipeline was fitted on the training set and then used for the test set.

This preprocessing pipeline ensured that the models received clean, well-structured inputs while maintaining consistency across training and evaluation.

4.7 Models, Hyperparameters, and Test Scores

Model	**Preprocessing**	**Best Parameters**	**Test Accuracy**	**Notes**
Logistic Regression	StandardScaler + PolynomialFeatures	`max_iter=1000`, `multi_class=multinomial`, `class_weight=balanced`, `solver=lbfgs`	0.704	Polynomial features helped capture non-linear patterns.
SVM (RBF)	RobustScaler	`kernel=rbf`, `C=3`, `gamma=scale`, `class_weight=balanced`	0.706	Best performance; handles high-dimensional OHE features effectively.
Neural Network (MLP)	RobustScaler	`hidden_layer_sizes=(64,32)`, `activation=relu`, `solver=adam`, `learning_rate=adaptive`, `batch_size=32`, `alpha=0.0005`, `max_iter=500`, `early_stopping=True`	0.687	Slight overfitting observed; early stopping improved stability.
Naive Bayes (Gaussian)	RobustScaler	Default	0.247	Strong independence assumptions do not fit mixed-type features.

Table 2: Model Comparison and Test Accuracy

4.8 Observations

- The **SVM (RBF) model achieved the highest accuracy** of 0.706.
- Neural network underperformed due to limited dataset size ($\sim 11k$ rows).
- Logistic regression benefited from polynomial features but did not surpass SVM.
- Naive Bayes performed poorly because its assumptions do not hold for this dataset.

Analysis of Train–Test Split Performance Differences in Binary Classification

This section explains why model performance changes when training with 20% of the data versus 100% of the data, for the binary classification task.

Training Dataset Size	**svm**	**nn**
20% dataset	0.8335	0.8192
100% dataset	0.8315	0.8372

Table 3: Model accuracies under different train–test splits for binary classification.

Neural Network

When trained on only 20% of the dataset, the neural network attains an accuracy of **0.8192**, which is lower than the corresponding SVM performance. This suggests that the neural network does not generalize as effectively when data are limited, likely because it relies on larger sample sizes to learn stable and expressive parameter representations.

With access to the full training dataset, its accuracy rises to **0.8372**, surpassing the SVM. This behaviour illustrates a key characteristic of neural networks: given sufficient data, they are capable of modelling richer and more flexible decision boundaries, ultimately yielding superior predictive performance.

Support Vector Machine

The SVM performs strongly even under restricted data availability, achieving an accuracy of **0.8335** with only 20% of the training set—higher than the neural network under the same conditions. This aligns with the theoretical expectations for SVMs, which, due to their margin-maximization principle, tend to be robust and less prone to overfitting on small datasets.

When trained on the full dataset, the SVM reaches an accuracy of **0.8315**, exhibiting only a marginal change compared to the smaller training split. This indicates that the SVM was already close to identifying its optimal separating hyperplane even with limited data, and that additional samples offered minimal further improvement.