ML Project Report - Checkpoint 2

Team Members:
1. Subikshaa Sakthivel (IMT2023020)
2. Harshita Bansal (IMT2023035)
3. Vriddhi Agrawal (IMT2023611)

Github for Task 1: https://github.com/Harshita30-bansal/Financial-Risk-Profiling
Github for Task 2: https://github.com/Subikshaa22/Travel-Behavior-Insights

Task 1 (Financial Risk Profiling)

This task involves predicting borrower default probability from personal and financial indicators. It is a binary classification problem. The goal is to train different classification models to determine the one that can most accurately predict the result based on the given data.

EDA and Preprocessing

The purpose of this analysis was to explore the characteristics of the training dataset (train.csv) and prepare it, along with the corresponding test dataset (test.csv), for a machine learning classification task aimed at predicting the binary variable RiskFlag.

Data Dimensions and Structure

The original dataset contained 204,277 rows and 18 columns, consisting of both numeric (integer/float) and categorical (object/string) data types.

- **Target Variable:** RiskFlag (integer classification target)
- **Identifier:** ProfileID (unique string identifier)
- **Feature Distribution:** After removing the target and ID, the dataset consisted of 9 numeric features and 7 categorical features.

Missing Values Analysis

A complete analysis of missing values was carried out across all columns.

- **Finding:** No missing values were detected (0.0% missing in all columns).
- **Implication:** No immediate imputation was required.

- **Action:** A preprocessing pipeline was still implemented to handle possible inconsistencies and ensure robustness. Numeric values were imputed with the median, and categorical values with the mode.

Target Variable Distribution

The distribution of the binary target RiskFlag was analyzed to evaluate class balance.

- **Class 0 (No Risk):** 180,524 records (88.372%)
- **Class 1 (Risk):** 23,753 records (11.628%)

Exploratory Data Analysis (EDA) Findings

Numeric Feature Summary and Skewness

Summary statistics were computed for all 9 numeric variables.

- Skewness values for most numeric features were extremely low:
    - TrustMetric: 0.006
    - OfferRate: 0.005
    - AnnualEarnings: 0.001
    - RequestedSum: −0.002

Interpretation: Numeric features were approximately symmetric, and transformations such as logarithmic scaling were not required.

Class-wise Numeric Statistics (Risk vs No Risk)

Differences in key numeric features between the two target classes were examined:

- **ApplicantYears:**
    - Risk = 1: Mean age = 36.57
    - Risk = 0: Mean age = 44.40
- **OfferRate:**
    - Risk = 1: Mean = 15.86
    - Risk = 0: Mean = 13.17
- **AnnualEarnings:**
    - Risk = 1: Mean = 71,712.60
    - Risk = 0: Mean = 83,926.43

These differences align with expected patterns of financial risk.

Categorical Feature Cardinality

All 7 categorical features had low cardinality (maximum: 5 unique values).
Implication: One-Hot Encoding (OHE) is an appropriate and efficient encoding technique.

Preprocessing Steps

Feature Identification

The non-predictive columns (RiskFlag and ProfileID) were removed.
The final feature set included:

- **9 numeric features**
- **7 categorical features**

Missing Value Imputation

Although the dataset contained no missing values, a consistent imputation strategy was implemented:

- Numeric Columns: Median imputation
- Categorical Columns: Mode imputation

Categorical Feature Encoding (One-Hot Encoding)

Given the low cardinality of categorical features, One-Hot Encoding was applied.

- get_dummies() was used to expand the 7 categorical columns into binary indicator columns.
- The test dataset was aligned to match the training dataset's encoded column structure.
- Final dataset dimensionality**:** 33 columns after encoding.

Numeric Feature Scaling (Standardization)

To prevent numeric features from dominating model training:

- StandardScaler was used to scale all numeric features.
- The scaler was fitted exclusively on training data and applied to both training and testing feature matrices.
- The transformation produced standardized features with mean 0 and standard deviation 1.

Model Training

1. Gaussian Mixture Model

- Loaded the preprocessed training and test datasets and aligned their feature columns.
- Split the data into an 80–20 train–validation set and fitted separate GMMs for each target class.
- Classified each sample based on the class model that produced the higher likelihood score.
- Validation accuracy of 0.697
- Kaggle score: **0.700**

2. Polynomial Logistic Regression

- Loaded encoded train and test datasets and created polynomial interaction features.
- Performed an 80–20 train–validation split.
- Trained an L2-regularized Logistic Regression model on the feature set.
- Best Validation Accuracy of 0.884 at threshold of 0.53
- Kaggle score: **0.887**

3. Neural Network

- Loaded the training and test datasets and applied StandardScaler to numeric features and One-Hot Encoding to categorical features.
- Performed 3-fold cross-validation, training up to 35 epochs per fold with early stopping.
- Achieved a mean AUC of 0.750
- Kaggle score: **0.887**

4. Stacked Model (Logistic Regression + Linear SVM + Neural Network)

- Loaded and preprocessed the training and test datasets using scaling for numeric features and one-hot encoding for categorical features.
- Constructed a stacking ensemble combining Logistic Regression, a calibrated Linear SVM, and a medium-sized Neural Network, with Logistic Regression as the meta-learner.
- Used 3-fold cross-validation and evaluated model performance using AUC
- Achieved average AUC of 0.748
- Kaggle score: **0.887**

5. Pure SVM Model

- Loaded the preprocessed training and test datasets and aligned their feature spaces.

- Used LinearSVC as the base classifier and applied CalibratedClassifierCV to obtain probability estimates.
- Trained the calibrated SVM model on the complete training dataset.
- Generated probability predictions for the test dataset and converted them to binary predictions using a 0.5 threshold.
- Kaggle score: **0.885**

Comparative Model Analysis

Below is a comparative analysis of the performance of the above models in ascending order of their Kaggle scores.

1. **Gaussian Mixture Model (GMM)**
   Performed the worst (0.700) because its Gaussian assumptions do not fit high-dimensional, one-hot encoded financial data. As a generative density model, it struggles to form strong class boundaries compared to supervised discriminative methods.

2. **Pure SVM (Calibrated LinearSVC)**
   Scored 0.885, performing strongly in high-dimensional encoded spaces but limited by its linear decision boundary. Its lack of nonlinear modeling capability explains why it slightly trails the top models.

3. **Polynomial Logistic Regression**
   Achieved top performance (0.887) by capturing nonlinear interactions through polynomial features while retaining logistic regression's stability. Its strong validation–Kaggle consistency indicates excellent generalization.

4. **Neural Network**
   Matched the best score (0.887), benefiting from its ability to learn complex patterns in the data. Slightly lower AUC indicates some instability, but proper preprocessing and training allowed it to generalize well.

5. **Stacked Model (LR + SVM + NN)**
   Also reached 0.887 but did not outperform its components because the base models lacked sufficient diversity. The linear models and NN provided overlapping signals, limiting the benefit of stacking.

Task 2 (Travel Behavior Insights)

This task involves understanding how travelers plan, book, and experience their journeys across destinations. It is a multi-class classification problem. The goal is to train different classification models to determine the one that can most accurately predict the spend-category based on the given trip and traveller data.

Preprocessing

The data was cleaned and transformed following a structured pipeline to ensure quality and consistency between the train and test sets.

Missing Value Imputation

- **Numerical Columns:** Missing values in numerical features were imputed using the median value calculated from the training data.
- **Categorical Columns:** Missing values in all categorical features were imputed by filling them with the string "Unknown".

Outlier Handling

To mitigate the influence of extreme outliers without discarding data, the Interquartile Range (IQR) capping method was applied to the four original numerical columns: num_females, num_males, mainland_stay_nights, and island_stay_nights.

Log Transform

The log1p() transformation was applied to the highly skewed duration features, mainland_stay_nights and island_stay_nights, to reduce their skewness and stabilize variance.

Feature Engineering

Four new features were created to capture meaningful travel insights:

- **total_group_size**: The sum of num_females and num_males.
- **is_family_trip**: A binary indicator (1/0) for trips with 'Spouse' or 'Children' companions.
- **is_group_trip**: A binary indicator (1/0) for trips with 'Other Friends/Relatives' companions.
- **mainland_to_island_ratio**: The ratio of mainland stay nights to island stay nights.

Categorical Encoding and Alignment

- All categorical columns were converted into numerical features using One-Hot Encoding, with the drop_first=True argument to avoid multicollinearity.

- The encoded training and test sets were aligned using a left join to ensure they share the identical set of feature columns. Any dummy columns present in the training set but missing in the test set were added to the test set and filled with 0.

Standardization (Z-Score Scaling)

Finally, selected numerical features were scaled using StandardScaler. This centers the data around a mean of 0 with a standard deviation of 1.

Conclusion and Final Output

The preprocessing steps successfully handled missing data, mitigated the effect of outliers, reduced skewness in duration features, and converted all categorical variables into a numerical format.

The final preprocessed datasets were saved as train_preprocessed.csv and test_preprocessed.csv.

EDA

Data Structure

The initial training dataset contained 12,654 rows and 25 columns, while the test dataset contained 5,852 rows and 24 columns. The difference in column count is due to the target variable, spend_category, being present only in the training set.

Missing Values

Several columns had significant missing values, requiring imputation:

- has_special_requirements had the largest number of missing values.
- arrival_weather and days_booked_before_trip were also major contributors to missing data.
- A small number of missing values were also present in other columns like travel_companions, total_trip_days, and several binary features.

Target Variable Analysis

Spend Category Distribution

The target variable, spend_category, is a multiclass variable with three labels: 0 (Low spenders), 1 (Medium spenders), and 2 (High spenders). The distribution of classes was:

- Category 0: 6245 samples
- Category 1: 4911 samples
- Category 2: 1464 samples

The dataset exhibits class imbalance, with "Low spenders" being the most frequent class and "High spenders" being the least frequent.

Numerical Feature Analysis

Summary Statistics and Outliers

An inspection of the descriptive statistics revealed a few key characteristics:

- num_females and num_males had mean values close to 1, but maximum values of 49.0 and 58.0, respectively, indicating the presence of extreme outliers.
- mainland_stay_nights and island_stay_nights also had high maximum values (365 and 240, respectively) relative to their median (6 and 0), confirming the presence of outliers and suggesting high positive skewness.

Skewness Check

A formal skewness check confirmed that all four numerical features were highly positively skewed, with skewness values ranging from 9.65 to 16.71. This necessitates a transformation to approximate a normal distribution for better model performance.

Models Used for Training

1. **SVM**
    - We first loaded the preprocessed training and testing data and ensured alignment between their features.
    - We then split the data into training and validation sets (80-20 split).
    - The SVM model was trained on the training set, to get an F1 score of 0.6751 on the validation set.
    - Finally, we re-trained the model on the entire train.csv data (training + validation set), before getting the predictions for test.csv.
    - Kaggle Score: **0.707**

2. **Logistic Regression**
    - We first loaded the preprocessed training and testing data and ensured alignment between their features.
    - We then split the data into training and validation sets (80-20 split).
    - The multinomial logistic regression model was trained on the training set, to get a validation accuracy of 0.7516.
    - Kaggle score for predictions on test.csv : **0.673**

3. **GMM (Gaussian Mixture Modelling)**
    - We first loaded the preprocessed training and testing data and ensured alignment between their features.
    - We then split the data into training and validation sets (80-20 split).
    - The GMM model was trained on the training set, to get a validation accuracy of 0.5852.
    - Kaggle score for predictions on test.csv : **0.534**

4. **Neural Networks**
    - We first loaded the preprocessed training and testing data and ensured alignment between their features.
    - Additionally, we scaled the features and encoded the target as Neural Networks perform better with scaled values.
    - We then split the data into training and validation sets (80-20 split).
    - The Neural Networks model was trained on the training set, to get a validation score of 0.6705.
    - Kaggle score for predictions on test.csv : **0.649**

5. **SVM + NN + Logistic Regression Stacking**
    - We first loaded the training and testing data and ensured alignment between their features.
    - Additionally, we scaled the features and encoded the target.
    - We then split the data into training and validation sets (80-20 split).
    - We stacked the SVM and NN (MLP) models and used Logistic Regression as a meta-learner.
    - We got a validation accuracy of 0.7675.
    - Kaggle score for predictions on test.csv : **0.667**
    -

Comparative Model Analysis

1. **Kernel SVM (Kaggle: 0.707)**

Kernel SVM performed the best, likely because the nonlinear kernel captured complex relationships in travel behaviour patterns. Its strong validation and higher Kaggle score show good generalization and robustness to mixed feature types.

2. **Logistic Regression (Kaggle: 0.673)**

Logistic Regression achieved good validation accuracy but underperformed on Kaggle F1, suggesting it captures linear trends well but struggles with the nonlinear structure of behavioural data. The model's simplicity limits its ability to adapt to complex decision boundaries.

3. **Stacked Model (SVM + NN + LR) (Kaggle: 0.667)**

The stacked model showed the highest validation accuracy but dropped on Kaggle, indicating mild overfitting. The models provided overlapping signals, so the meta-learner could not leverage enough complementary information to surpass Kernel SVM.

4. **Neural Network (Kaggle: 0.649)**

The neural network moderately captured nonlinear patterns but did not outperform SVM due to limited architecture depth or optimization. Its validation and Kaggle scores suggest decent learning but insufficient expressiveness for the task.

5. **Gaussian Mixture Model (GMM) (Kaggle: 0.534)**

GMM performed the weakest due to its Gaussian assumptions not matching the dataset's categorical–numerical mix. As a generative density model, it failed to separate behaviour classes effectively, leading to low validation and Kaggle scores.

Model Performance Analysis: Full Dataset vs. 20% Dataset

(for Task 1 - Binary Classification)

The objective of this experiment was to evaluate how training data size influences model performance when predicting loan risk using a neural network classifier. Two datasets were used:

- Full Dataset (100%)
- Reduced Dataset (20%)

Performance was compared using two key metrics commonly used in credit risk modeling:

- Accuracy
- AUC (Area Under ROC Curve)

Neural Networks

Dataset	Accuracy	AUC
Full Dataset	0.8853	0.7478
20% Dataset	0.8830	0.7296

Interpretation of Metrics

Accuracy

Accuracy measures the proportion of correctly predicted labels.
Both datasets achieved similar accuracy **(~88%)**, suggesting:

- The dataset has strong predictive features.
- Even 20% of the data captures the major patterns needed to classify most customers.
- The model performs consistently well in identifying the majority class.

AUC (Area Under ROC Curve)

AUC evaluates the model's ability to separate high-risk vs. low-risk applicants across all probability thresholds.

The AUC values show a clearer distinction between models:

- The Full Dataset achieves **AUC = 0.7478**
- The 20% Dataset achieves **AUC = 0.7296**

This decrease in AUC indicates that:

- With less training data, the model struggles to distinguish borderline risky cases.
- Predictions from the smaller dataset are less reliable for probability-based ranking.
- The model is more likely to incorrectly assign similar risk scores to distinctly different applicants.

Impact of Data Size on Model Quality

Why Accuracy Stays Similar

Accuracy depends on a single threshold (0.5).
The model needs only to be "good enough" at classifying the dominant pattern which the 20% dataset captures.

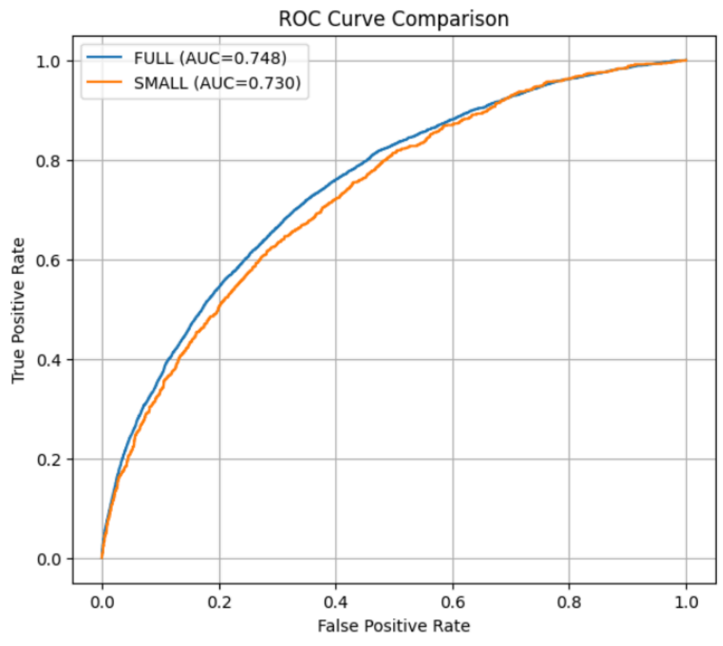
Hence, accuracy remains high even with reduced data.

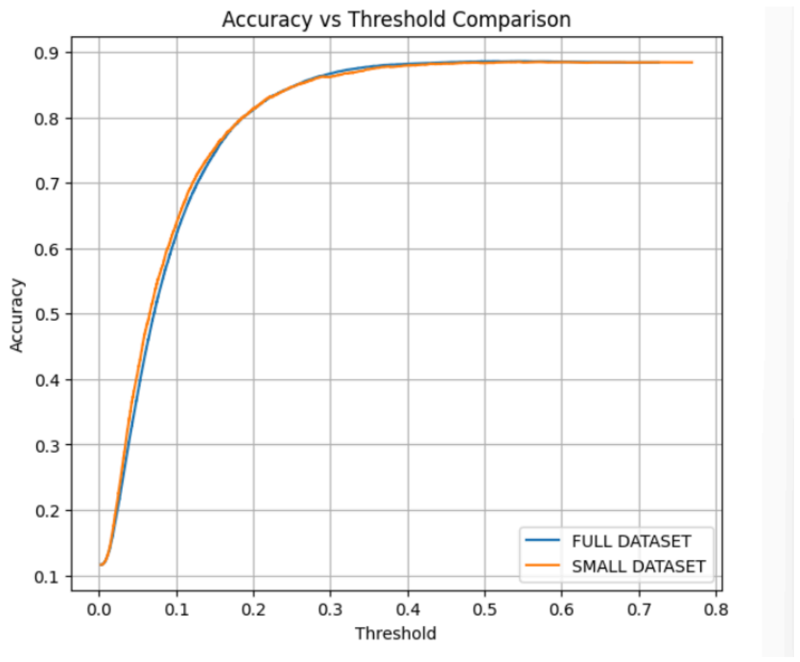
Why AUC Drops With Less Data

AUC reflects the model's ability to rank/order customers by risk level a fundamental requirement in credit scoring.

The slight yet meaningful drop in **AUC (0.7478 → 0.7296)** suggests:

- The full dataset provides richer examples of subtle risk patterns.
- The 20% dataset lacks diversity in edge cases.
- Decision boundaries become less refined with limited data.





SVMs

Metric	Full Dataset	Small Dataset
Accuracy	0.8848	0.8842
AUC	0.7463	0.7421

Interpretation of Metrics

Accuracy

Accuracy reflects the proportion of applicants correctly classified as low-risk or high-risk.
Both datasets achieve nearly **identical accuracy (~88.4%)**, indicating:

- The core signal in the dataset is strong and easy to learn.

- Even 20% of the data captures the main trends required for risk classification.
- The model performs consistently well at identifying the predominant class (typically low-risk customers).

AUC (Area Under ROC Curve)

AUC measures the model's ability to **separate** risky customers from safe customers across all probability thresholds—an essential requirement in credit scoring.

The AUC results show a more meaningful difference:

- **Full Dataset AUC = 0.7463**, indicating stronger discrimination ability
- **20% Dataset AUC = 0.7421**, slightly weaker separation

Although the drop appears small, it carries important implications:

- Smaller datasets reduce exposure to rare but critical “edge-case” risk patterns.
- The model trained on 20% data is less reliable at ranking borderline applicants.
- With fewer samples, the SVM learns a less refined separating hyperplane.

Impact of Dataset Size on Model Behavior

Why Accuracy Appears Unchanged

Accuracy remains similar for both datasets because:

- The dataset likely contains strong linear relationships between predictors and risk.
- Linear SVMs require relatively few samples to learn stable decision boundaries.
- Class imbalance inflates accuracy — the model correctly predicts many low-risk cases regardless of dataset size.

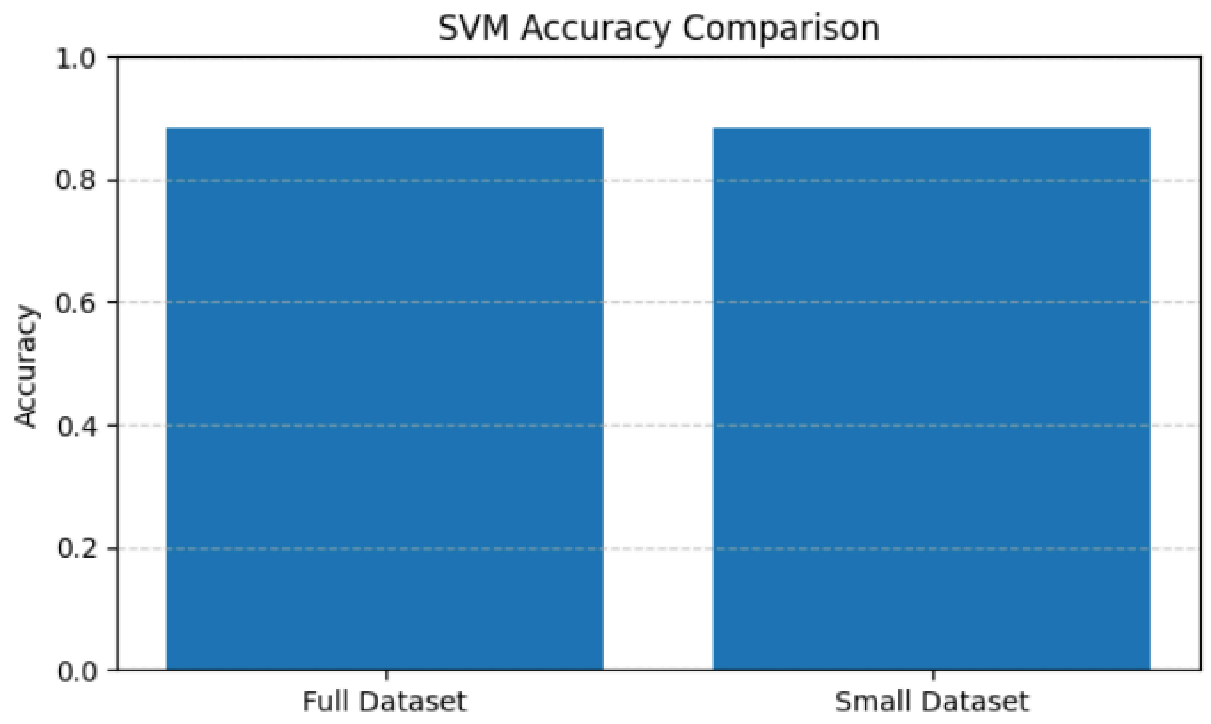
As a result, even 20% of the data is sufficient for achieving high accuracy.

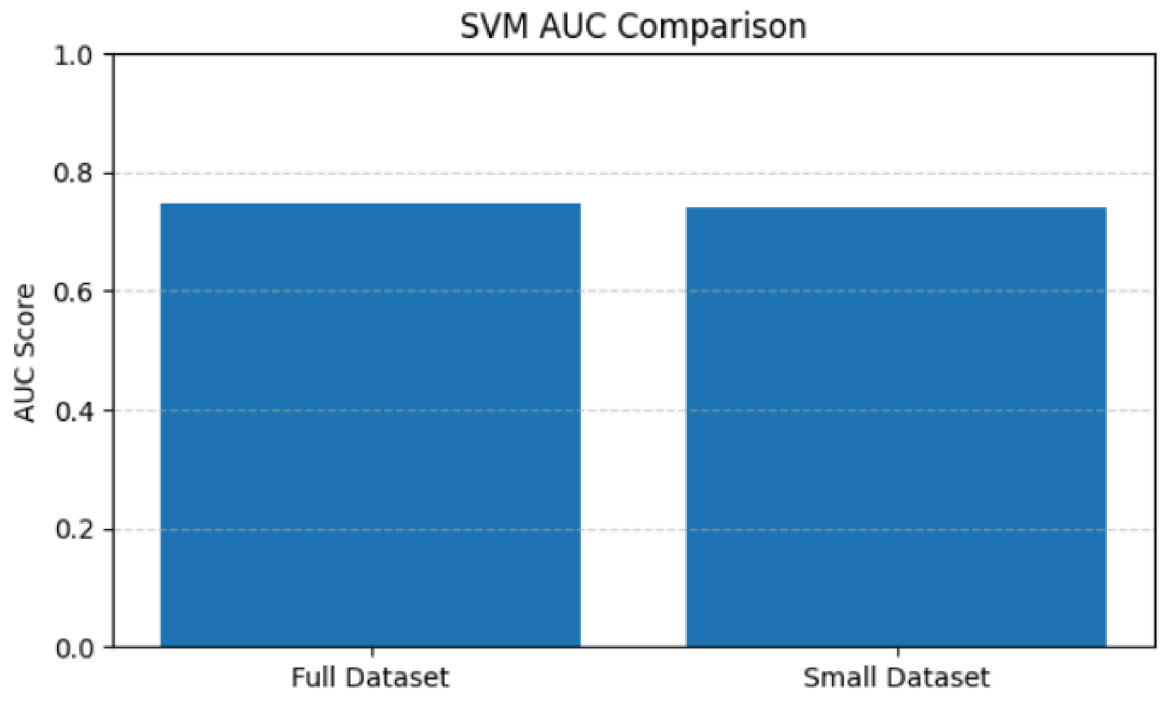
Why AUC Shows More Sensitivity

AUC captures the ranking quality of risk scores, making it more sensitive to data volume.

The observed decrease in AUC reflects the fact that:

- The full dataset contains diverse examples of nuanced risk patterns.
- The smaller dataset does not cover the full spectrum of applicant profiles.
- Subtle risk distinctions are harder to learn with reduced training size.


SVM Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
Full Dataset
Small Dataset


SVM AUC Comparison
AUC Score
1.0
0.8
0.6
0.4
0.2
0.0
Full Dataset
Small Dataset