International Institute of Information Technology Bangalore

ML Project Report: Start-up Founder Retention Prediction

(Checkpoint 2)

Team Name: Predictify

Team Members:

1. Kartikeya Dimri - IMT2023126
2. Ayush Mishra - IMT2023129
3. Harsh Sinha - IMT2023571

Github Repo Link:
https://github.com/Ayush-Mishra-0018/Start-up-Founder-Retention-Prediction/

November 27, 2025

Contents

1 Task

The objective of this project is to build a predictive model that determines whether a startup founder is likely to **remain with their venture** or **exit**. This is formulated as a binary classification problem based on demographic signals, professional engagement patterns, and startup performance metrics.

Because real-world startup datasets often contain more retained founders than exited ones, class imbalance is expected. To account for this, we evaluate all models using the **Macro F1 Score**, which balances performance across both classes, ensuring minority behavior (exits) is not overshadowed by majority behavior (retention).

2 Background

Founder retention is one of the strongest indicators of startup stability, long-term survival, and organizational resilience. A founder's decision to stay or leave is influenced by a mix of personal circumstances, leadership responsibilities, team dynamics, venture performance, and environmental factors.

In this project, we utilize a dataset of founders with attributes spanning demographics, role-specific responsibilities, satisfaction levels, work-life conditions, and startup stage indicators. By examining how these features interact, we aim to predict whether a founder will continue with their startup. This serves as a practical application of machine learning to high-dimensional, mixed-type data, offering insights relevant to incubators, investors, HR analytics teams, and entrepreneurship researchers.

3 Dataset Description

The dataset includes anonymized records of startup founders and the ventures they lead. Each entry corresponds to a unique founder and contains personal, professional, and organizational indicators. The target variable is `retention_status`, which specifies whether the founder stayed with the startup or exited.

The dataset contains a mix of numerical, categorical, and ordinal features, making it suitable for modeling tasks such as churn prediction, risk scoring, and early-warning analytics.

3.1 Feature Descriptions

Table 1: Startup Founder Dataset Column Descriptions

S.No	Column Name	Description
1	founder_id	Unique identifier assigned to each founder.
2	founder_age	Age of the founder in years.
3	founder_gender	Gender identity of the founder.
4	years_with_startup	Number of years the founder has worked with the current startup.
5	founder_role	Founder's designation (e.g., CEO, CTO, Co-founder).
6	monthly_revenue_generated	Monthly revenue generated by the startup; may contain missing values.
7	work_life_balance_rating	Self-rated work–life balance (Low / Medium / High).
8	venture_satisfaction	Overall satisfaction level of the founder with the venture.
9	startup_performance_rating	Performance rating of the startup (Excellent / Average / Poor).
10	funding_rounds_led	Number of funding rounds led or co-led by the founder.
11	working_overtime	Indicates whether the founder frequently works overtime (Yes / No).
12	distance_from_investor_hub	Distance between the founder and the nearest innovation or investor hub.
13	education_background	Founder's educational background or domain (e.g., Engineering, Business).
14	personal_status	Marital or relationship status.
15	num_dependents	Number of dependents financially supported by the founder.
16	startup_stage	Current funding or maturity stage of the startup (Seed, Series A, Growth, etc.).
17	team_size_category	Team size classification (Small / Medium / Large).
18	years_since_founding	Number of years since the startup's founding date.
19	remote_operations	Whether the startup primarily operates remotely (Yes / No).
20	leadership_scope	Nature of leadership responsibilities (Strategic / Operational / Both).
21	innovation_support	Indicates whether the startup invests in or supports innovation initiatives.
22	startup_reputation	Public or industry reputation of the startup (High / Moderate / Low).
23	founder_visibility	Level of public presence or visibility of the founder (High / Medium / Low).
24	**retention_status**	**Target variable:** Whether the founder stayed (Retained) or exited (Exited).

4 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) serves as a critical foundation for understanding the structure, behaviour, and underlying patterns within our dataset. In this project, we conducted a comprehensive EDA to investigate data distributions, identify missing values, detect outliers, examine feature relationships, and generate visual insights that guided our modelling choices. The purpose of this analysis is not only to gain familiarity with the data but also to uncover trends, anomalies, and correlations that influence downstream machine learning workflows.

All the code used to generate the figures, tables, and insights in this section is available at the following link:

https://github.com/Ayush-Mishra-0018/Start-up-Founder-Retention-Prediction/blob/main/EDA/KnowingData.ipynb

For this project, we have created **four separate EDA files**, each focusing on different aspects of the dataset. The subsections below correspond to each file.

4.1 EDA File 1: Comprehensive Data Health Check

The first EDA script performs an extensive data health assessment to ensure that the dataset is clean, well-understood, and suitable for downstream modelling. This step is essential because any inconsistencies, missing information, or imbalanced distributions in the data can significantly affect the performance and reliability of machine learning models. The script systematically evaluates the dataset across several diagnostic dimensions:

- **Dataset Structure and Types:** We begin by loading the training dataset and inspecting its overall shape, i.e., the number of rows and columns, along with the data types of each feature. This provides a foundational understanding of the dataset's composition and helps identify whether features are numerical, categorical, or mixed.
- **Missing Values Analysis:** A detailed breakdown of missing values is computed, including both the absolute count and the percentage of missing entries for each column. This is crucial for determining which features require imputation, removal, or further investigation due to insufficient data quality.
- **Duplicate Records:** Duplicate rows are detected to ensure data integrity. Removing duplicates prevents model bias and ensures that no sample is unintentionally overrepresented.
- **Cardinality of Features:** For every column, the script reports the number of unique values. Understanding cardinality is important for decisions related to encoding categorical variables and assessing feature variability.
- **Target Variable Distribution:** The distribution of the target variable (`retention_status`) is computed in percentage form. This helps in identifying class imbalance, which could necessitate techniques such as reweighting, resampling, or use of specialized metrics.

- **Numerical Feature Statistics:** Descriptive statistics (mean, standard deviation, quartiles, etc.) are generated for all numerical features. These statistics help in identifying outliers, skewed variables, and potential normalization requirements.
- **Categorical Feature Breakdown:** For all categorical features (excluding the target), the script lists the top five most frequent categories. This provides insight into dominant categories, rare values, and potential encoding strategies.

Overall, this script serves as a foundational data audit, ensuring that the dataset is well-understood before more advanced analyses and modelling efforts are undertaken.

4.2 EDA File 2: Visual Exploration of Numerical and Categorical Features

The second EDA script focuses on generating visual insights to better understand how numerical and categorical features behave individually and in relation to the target variable. Visual analysis is a crucial complement to statistical summaries, as it helps identify patterns, relationships, anomalies, and separability that are not always evident from numerical diagnostics alone. This script includes two major components: analysis of numerical variables and analysis of categorical variables.

A. Numerical Features Analysis

- **Correlation Matrix:** A heatmap of pairwise correlations among all numerical features is generated. This helps identify strongly correlated variables, potential multicollinearity concerns, and redundant features that may impact model performance. It also highlights which features are most associated with the target variable.
- **Distribution Plots:** For every numerical feature, the script plots its histogram along with a kernel density estimate. These plots allow us to visually assess skewness, modality, and the overall spread of the data. Understanding these characteristics is essential for decisions related to normalization, transformation, or outlier handling.
- **Boxplots Against the Target:** Each numerical feature is also plotted against the target variable (`retention_status`) using boxplots. This reveals whether different target classes show distinguishable value ranges, variations, or outlier patterns. Such visual separations help assess the predictive potential of features.

B. Categorical Features Analysis

- **Countplots by Target Class:** For each categorical feature (excluding the target itself), the script generates countplots split by the target variable. These plots illustrate how category frequencies differ across retention outcomes, making it easier to spot dominant categories, rare occurrences, and meaningful class-level separation.
- **Handling High-Cardinality Categories:** When a feature contains many unique categories, the x-axis labels are rotated for better readability. This ensures that even wide categorical distributions can be interpreted clearly without losing structural information.

Overall, this script provides a comprehensive set of visual diagnostics that enhance our understanding of feature behaviour, class separability, and potential data transformation needs. These insights guide the modelling pipeline by highlighting which features are informative and how they should be preprocessed.

4.3 EDA File 3: Statistical Diagnostics, Normality Checks, and Multivariate Interactions

The third EDA script extends the earlier analyses by incorporating statistical distribution diagnostics, categorical association strength, and multivariate interaction patterns. While some steps (such as identifying numerical columns or plotting distributions) overlap with the previous two EDA files, this script applies deeper statistical tools that were not covered earlier, namely skewness analysis, normality assessment through Q–Q plots, Cramer's V for categorical association, and interaction-based visualizations. These techniques provide more rigorous insight into data behaviour and feature relationships.

Part 1: Numerical Distribution Shape and Normality (Skewness & Q–Q Analysis)

This section revisits numerical columns (previously identified in Files 1 and 2), but now introduces statistical shape descriptors to evaluate how closely each feature follows a normal distribution.

- **Skewness and Kurtosis Summary:** For each numerical feature, the script computes skewness and kurtosis to quantify distribution asymmetry and tail behaviour. This provides a statistical basis for determining whether features require transformations such as log scaling or power transforms—especially important for algorithms sensitive to non-normality.
- **Transformation Recommendations:** Based on skewness thresholds, the script automatically suggests whether a feature should undergo transformation, scaling, or can be left as-is. This adds actionable preprocessing guidance beyond the visual distributions shown in EDA File 2.
- **Histogram & Q–Q Plots for Skewed Variables:** For features with high skew, detailed histogram and Q–Q plots are generated. Q–Q plots allow a more rigorous assessment of normality by comparing quantiles of the feature distribution to those of a theoretical normal distribution—something not included in the first two EDA scripts.

Part 2: Categorical Predictive Strength (Cramer's V)

While EDA File 2 visually explored categorical variables, this script introduces a statistical measure—Cramer's V—to quantify the strength of association between each categorical feature and the target variable.

- **Cramer's V Computation:** Using chi-square contingency tables, the script calculates Cramer's V (ranging from 0 to 1), offering a numeric measure of how informative each categorical variable is with respect to retention status.

- **Ranked Feature Importance Chart:** The resulting association scores are visualised as a bar chart, enabling clear comparison of which categorical variables carry the most predictive value.

This section provides statistical significance where earlier scripts provided only frequency-based visuals.

Part 3: Multivariate Interaction Exploration

This final part examines how multiple features interact simultaneously, capturing relationships that cannot be detected through univariate or bivariate plots used earlier.

- **Scatterplot (Age vs Revenue vs Retention):** A multi-axis scatterplot visualises whether the relationship between founder age and revenue differs across retention classes, helping identify interaction effects.
- **Violin Plot (Satisfaction vs Revenue):** This plot explores whether revenue patterns vary meaningfully across levels of venture satisfaction and retention status, offering insight into combined behavioural trends.
- **Pairplot of Key Numerical Factors:** A pairplot (corner view) is generated for selected numerical variables to reveal clustering behaviour, linear relationships, or separability patterns across the target classes.

Overall, this script enhances the EDA pipeline by supplementing earlier visual and structural analyses with statistical distribution diagnostics, categorical association metrics, and multivariate interaction insights.

4.4 EDA File 4: Target-Oriented Comparative Analysis and Normalized Categorical Insights

The fourth EDA script focuses on visual comparisons between input features and the target variable, with an emphasis on understanding how different numerical and categorical attributes vary across retention outcomes. While parts of this analysis overlap with earlier EDA files—such as correlation heatmaps and boxplots—this script introduces more structured and target-oriented comparison techniques, including normalized stacked bar charts that reveal proportional differences across categories. The goal is to gain a clearer, more interpretable understanding of how feature behaviour shifts between retained and non-retained founders.

1. Correlation Heatmap (Numerical Features)

A correlation heatmap similar to the one generated in EDA File 2 is produced again for completeness. This version uses a triangular mask to remove redundant correlations and improve readability. Since numerical correlations were already discussed earlier, the emphasis here is simply on reaffirming which features exhibit meaningful linear relationships, particularly in relation to potential multicollinearity.

2. Numerical Features vs Target (Boxplots)

Boxplots comparing each numerical feature with the target variable are generated once more. This method was previously used in EDA File 2; however, the present script extends the analysis by placing all boxplots into a compact grid layout. This makes it easier to visually scan for median shifts or distributional differences across all numerical inputs simultaneously, supporting quicker, high-level comparisons.

3. Categorical Features vs Target (Normalized Stacked Bar Charts)

This section introduces a new visual technique not used in any of the earlier scripts: normalized stacked bar charts. Unlike the countplots in EDA File 2 or the association scores from EDA File 3, this method shows the *proportional* distribution of retention outcomes within each category of a feature.

- **Contingency Table Normalization:** For each categorical feature with a manageable number of unique values, a crosstab is created and normalized row-wise so that each bar sums to 100%. This reveals how retention probabilities vary within categories, independent of category size.
- **Stacked Bar Visualization:** The normalized proportions are displayed as stacked bars, enabling easy interpretation of whether certain categories skew heavily toward retention or non-retention.
- **Percentage Labeling:** Labels are added to segments large enough to be interpretable, which improves the clarity of category-level retention patterns.

This visualization technique is particularly useful for categorical variables with moderate cardinality, offering a more nuanced perspective on class distribution than raw counts or frequencies.

Overall, the fourth script consolidates earlier insights while adding fresh comparative visualisations that help contextualize how each feature behaves with respect to the target variable. This deepens the understanding of which features exhibit meaningful separation and can contribute to improved model interpretability and performance.

5 Preprocessing Strategies

Based on the insights gathered during the Exploratory Data Analysis (EDA), multiple preprocessing pipelines were designed to prepare the founder–startup dataset for machine learning models. Since the data contains a mix of numerical, categorical, ordinal, and binary attributes, each preprocessing script explored different combinations of encoding, imputation, scaling, and feature engineering choices.

The primary objective behind these preprocessing variations was to understand how different representations of founder demographics, startup characteristics, and performance indicators influence the predictive power of the models. Given the mild class imbalance in the `retention_status` variable, all pipelines were also built to support fair evaluation using the Macro F1 Score.

Overall, the preprocessing scripts covers mainly:

- how missing values were handled for revenue- and distance-related fields,
- which encoding schemes were applied to categorical and ordinal variables,
- whether numerical features were scaled,
- whether engineered features such as founder–startup alignment metrics were included,
- and how binary attributes were standardized across models.

These variations allowed us to compare multiple data representations and select the preprocessing strategy that produced the most stable and generalizable performance across different classifiers.

5.1 P1_preprocess.py (Baseline Feature Engineering + Full Preprocessing)

This script served as the primary preprocessing pipeline for the founder retention prediction task. It implemented a combination of engineered features, imputation strategies, and mixed-type encodings to transform the raw dataset into a fully numeric matrix suitable for machine learning models.

- **Feature Engineering:** Three derived variables were introduced to capture founder–startup alignment patterns:
  - `age_at_founding`: estimated founder age at the time of the startup's creation,
  - `tenure_ratio`: proportion of years the founder has stayed relative to the startup's age,
  - `unhappy_overtime`: interaction between overtime frequency and satisfaction level.
- **Target Encoding:** The `retention_status` variable was mapped to a binary numeric label (Left = 1, Stayed = 0).
- **Numerical Preprocessing:** Continuous fields such as revenue, distance, founder age, and engineered metrics were median-imputed and standardized using `StandardScaler`.

- **Ordinal Encoding:** Ratings such as work–life balance, satisfaction, performance, reputation, and visibility were encoded using custom-defined ordinal mappings based on their natural rating order.
- **Nominal Encoding:** Categorical attributes including gender, founder role, education, startup stage, team size, and binary Yes/No fields were one-hot encoded using `OneHotEncoder`.
- **Pipeline Integration:** A `ColumnTransformer` combined numerical scaling, ordinal encoding, and nominal one-hot encoding into a unified preprocessing pipeline applied consistently to both training and test sets.
- **Output:** The script generated `X_processed`, `X_test_processed`, the binary target vector `y`, and the original test founder IDs for later submission.

5.2 Data Cleaning, Feature Engineering, and Encoding

The first stage of preprocessing focuses on converting the raw dataset into a clean, consistent, and fully numerical form suitable for machine learning algorithms. This pipeline performs several critical transformations, including deduplication, anomaly filtering, missing-value handling, feature engineering, categorical encoding, and final scaling. The complete workflow is outlined below.

1. Data Cleaning and Integrity Checks

- **Duplicate Removal:** All duplicate rows are removed to ensure that no sample is unintentionally overrepresented in the learning process.
- **Anomaly Filtering:** Two logical constraints are enforced:
  - Founders must be at least 18 years old.
  - The inferred start age (`founder_age - years_with_startup`) must be at least 16, preventing unrealistic or contradictory timelines.

  These constraints eliminate biologically or temporally impossible records.

2. Missing Value Imputation

- **Numerical Features:** Skew-sensitive numerical columns are imputed using the median, a robust measure that prevents extreme values from distorting the distribution.
- **Categorical Features (Psychological Ratings):** Missing survey responses for variables such as work–life balance and venture satisfaction are imputed as `Unknown`, preserving the semantic signal that a founder chose not to respond.
- **Other Categorical Features:** Structural variables (e.g., team size category) are imputed using the mode to retain the most common category.

3. Feature Engineering

A new feature, **start age**, is created by subtracting years with the startup from current founder age. This captures the age at which the founder began their entrepreneurial journey, which may hold predictive significance. Additionally, the target variable is encoded numerically (`Stayed` $\rightarrow$ 0, `Left` $\rightarrow$ 1) to align with standard churn modelling practice.

4. Ordinal and Binary Encoding

Several categorical features possess meaningful order or binary structure, and are encoded manually to preserve semantic relationships:

- **Ordinal Ratings:** Work–life balance, satisfaction, startup reputation, performance, and visibility are mapped to ordered numerical scales, ensuring the model recognises that `Excellent`,`Good` or `Low`, etc.
- **Binary Variables:** Yes/No attributes such as overtime, remote operations, innovation support, and leadership scope are encoded as 1 and 0.
- **Startup Stage:** Mapped to an approximate progression scale to reflect increasing maturity.

5. One-Hot Encoding

Nominal categorical variables without inherent order (e.g., gender, role, education background, personal status, team size category) are one-hot encoded. The use of `drop_first=True` prevents multicollinearity by removing redundant dummy columns.

6. Train–Test Split

The processed dataset is split into an 80–20 training–testing ratio using stratified sampling to preserve the target distribution across splits. This ensures a fair and representative evaluation.

7. Feature Scaling

All numerical features are standardized using **z-score scaling** (via `StandardScaler`). This step ensures that variables with large numerical ranges (e.g., revenue) do not dominate those with smaller scales (e.g., age), thereby stabilising gradient-based models and improving convergence.

Overall, this preprocessing pipeline produces a fully cleaned, validated, encoded, and scaled dataset, ready for downstream model training and evaluation.

5.3 Unified Forensic Cleaning and Cross-Dataset Preparation

This preprocessing stage applies a unified transformation pipeline to both the training and test datasets to ensure consistent feature space alignment and identical preprocessing logic. The steps below describe the operations performed before model training, focusing exclusively on the data preparation components.

1. Dataset Merging for Consistency

The training and test datasets are concatenated into a single combined frame after tagging each instance with an `is_train` indicator. This ensures that all preprocessing steps (such as imputations, mappings, and encodings) are applied uniformly, preventing train–test discrepancies and feature mismatches during inference.

2. Noise Reduction Through Column Elimination

A set of columns identified as uninformative, redundant, or excessively noisy is removed from the dataset. This "kill list" includes identifiers and subjective evaluator variables that introduce little predictive signal:

`founder_id`, `founder_role`, `leadership_scope`, `founder_visibility`, `innovation_support`, `team_size_category`

Dropping such features simplifies the feature space and reduces the risk of overfitting.

3. Skew Correction

The variable `monthly_revenue_generated` exhibits strong positive skew, and is therefore transformed using `log1p`. This stabilizes variance and makes the distribution more Gaussian-like, benefiting algorithms sensitive to scale or distributional irregularities.

4. Missing Value Imputation

Several targeted imputations are applied:

- **Numerical Columns:** `years_since_founding` is imputed with the median, and `num_dependents` with the mode to reflect the most frequent structural characteristic.
- **Psychological and Satisfaction Ratings:** Missing values in work–life balance and satisfaction ratings are imputed with `Unknown`, preserving behavioural signal and preventing loss of context.

5. Feature Engineering

Two engineered transformations are applied:

- Founder age is clipped to a minimum of 18 years to remove biologically implausible records.
- A new variable, **start age**, is constructed by subtracting years with the startup from the founder's age. This captures entry timing into entrepreneurship, a potentially predictive behavioural marker.

6. Ordinal and Binary Encoding

Ordered categorical variables representing performance, reputation, work–life balance, and satisfaction are manually mapped to numeric ordinal scales to preserve semantic relationships among categories.

Similarly, binary variables such as overtime and remote operation indicators are encoded using {0, 1}, ensuring compatibility with numerical algorithms.

The startup stage variable is mapped onto an approximate progression scale capturing relative maturity.

7. One-Hot Encoding

Nominal categorical attributes (gender, education background, and personal status) are one-hot encoded with `drop_first=True` to prevent multicollinearity. This expands categorical variables into separate binary indicators, allowing them to be used by linear and non-linear models.

8. Train–Test Reconstruction and Scaling

After preprocessing, the dataset is split back into training and test subsets based on the `is_train` flag. Numerical features are standardized using z-score scaling via `StandardScaler`, a requirement for algorithms such as SVMs and neural networks, ensuring stable convergence and balanced feature influence.

—

Overall, this pipeline produces a harmonized, noise-filtered, fully encoded, and scaled dataset, ensuring that both training and testing inputs share an identical preprocessing structure.

5.4 Advanced Feature Engineering and Interaction-Based Transformations

The third preprocessing pipeline introduces a more sophisticated transformation strategy designed to enrich the feature space and capture deeper behavioural and structural patterns within the dataset. While several core operations (such as imputations, ordinal encoding, and one-hot encoding) remain consistent with earlier preprocessing stages, this pipeline distinguishes itself by generating multiple interaction features that target founder behaviour, efficiency, and burnout risk. The key components are described below.

1. Cleaning and Structural Refinement

A reduced feature space is constructed by removing several noisy or low-signal columns, including founder identifiers and subjective perception variables that were found to contribute minimal predictive value. The following columns were excluded:

`founder_id`, `founder_visibility`, `innovation_support`

This step helps minimize overfitting and maintains a more stable modelling pipeline.

2. Missing Value Treatment

Numerical fields such as revenue, years since founding, and dependent counts are imputed using median or mode values, while psychological survey items are assigned the label `Unknown`. This ensures that structurally meaningful information is retained while preventing disruption of downstream encodings.

3. New Interaction and Behavioural Features

A central innovation of this pipeline is the introduction of several engineered variables that combine multiple raw features into higher-level behavioural indicators:

- **Revenue Efficiency:** Defined as

$$\frac{\log(1 + \text{monthly\_revenue})}{\text{years\_since\_founding} + 1},$$

  this feature captures how effectively revenue has been generated relative to the startup's maturity.

- **Founder Experience Gap:** Calculated as the difference between founder age and years spent with the startup, this variable approximates prior professional experience and business maturity.

- **Binary Conversion for Operational Variables:** Overtime and remote operation indicators are mapped to binary form $(0, 1)$ to enable mathematical interaction with satisfaction scores.

- **Burnout Index:** Using the satisfaction score mapping and binary overtime status, a burnout proxy is defined as:

$$\text{burnout\_index} = \frac{\text{working\_overtime}}{\text{satisfaction\_score} + 1}.$$

  This feature reflects the tension between workload and perceived satisfaction, which may strongly influence retention behaviour.

These engineered variables introduce meaningful non-linear relationships and provide the model with deeper conceptual signals beyond raw feature values.

4. Ordinal Encoding

All rating-based attributes (work–life balance, venture satisfaction, performance, and reputation) are mapped to an ordinal scale using a unified satisfaction mapping function. Similarly, the startup stage variable is encoded into a rough maturity progression scale. These encodings preserve the inherent order of perceived quality and progression levels.

5. One-Hot Encoding

Nominal variables without natural ordering (such as gender, education background, personal status, founder role, team size category, and leadership scope) are expanded using one-hot encoding with `drop_first=True` to avoid redundant representation. This step ensures compatibility with linear and non-linear models.

6. Train–Test Reconstruction and Scaling

After preprocessing, the dataset is split back into training and test subsets based on the `is_train` indicator. All numerical features are standardized using z-score normalization through `StandardScaler`, producing the matrices:

$$X_{\text{scaled}}, \quad X_{\text{submit\_scaled}}, \quad y.$$

Standardization is essential for stability in gradient-based models and neural networks, ensuring that variables operate on comparable scales.

Overall, this advanced preprocessing stage enriches the feature space using behavioural, temporal, and interaction-driven transformations while maintaining numerical consistency across train and test partitions. This results in a significantly more expressive set of features for downstream ensemble modelling.

6 Model Training

A diverse set of supervised learning models was trained to predict founder retention using the various preprocessing pipelines developed. The focus was on understanding how different algorithms respond to operational, behavioural, and business-related founder attributes rather than traditional psychometric modelling.

Multiple preprocessing variants were tested to provide clean, standardized, and fully engineered feature spaces. All models were trained and hyperparameter tuning was performed using randomized or grid search depending on the model family.

The following subsections outline the key model families explored, the motivation for selecting them, and the files corresponding to their implementations.

6.1 Primary Models: Neural Networks, SVM, and Logistic Regression

These three model families formed the core of our experiments, covering linear baselines (Logistic Regression), geometric decision boundaries (SVM), and deep nonlinear representation learning (Neural Networks). Together, they offer a clear comparison of how varying model complexities respond to the transformed founder-level feature space.

6.1.1 Neural Networks (Deep Learning)

Explanation: Multi-Layer Perceptrons (MLPs) were used to model non-linear interactions within the engineered founder–level feature space. Compared to linear models, MLPs can learn deeper hierarchical patterns and capture subtle behavioural or operational signals through their hidden layers. Regularization, hyperparameter search, and careful optimization strategies were used to stabilize training and prevent overfitting.

Files:

- `MLP_P1_721.py`

  Best Hyperparameters (after tuning):

  - Solver: `adam`
  - Learning Rate: `0.01`
  - Hidden Layer Sizes: `(128,)`
  - Regularization (α): `0.01`
  - Activation: `tanh`

- `Bayes_NLP_Log_Lgbm_Xgb_Cat_P4.py` (Runs multiple models, including an MLP variant.)

- `AllModels_P2.ipynb`

- `AllModels_P3.ipynb`

  Best Hyperparameters (after tuning):

  - Learning Rate: `0.001`
  - Hidden Layer Sizes: `(50,)`
  - Regularization (α): `0.001`

* Activation: `relu`

Note: Selected MLP outputs were later incorporated into ensemble experiments, which are documented in the Ensemble section.

6.1.2 Support Vector Machines (SVM)

Explanation: Support Vector Machines were explored as a strong non-linear baseline. Using the RBF kernel, SVMs project the data into a higher-dimensional space where complex retention boundaries become more separable. Their margin-maximizing nature makes them robust on mixed-type, moderately imbalanced datasets.

Files:

- `AllModels_P2.ipynb`
- `AllModels_P3.ipynb`

  (Also includes ensemble experiments built on SVM outputs.)

  Best Hyperparameters (after tuning):

  * Kernel: `rbf`
  * Gamma: `scale`
  * C: `1`

6.1.3 Logistic Regression

Explanation: Logistic Regression was used as the primary linear baseline. It models retention probability through a sigmoid function, offering speed, interpretability, and a reference point to judge whether the underlying founder–startup relationships are linearly separable.

Files:

- `Bayes_MLP_Log_Lgbm_Xgb_Cat_P4.py` (Runs multiple models, including Logistic variant.)
- `Stack_P4.ipynb`
- `AllModels_P2.ipynb`

6.2 Other Models

To ensure a robust comparison, we trained a variety of other algorithms to benchmark against our primary models.

6.2.1 Naive Bayes Variant

Explanation: Naive Bayes models were included as lightweight probabilistic baselines.
Files:

- `Bayes_MLP_Log_Lgbm_Xgb_Cat_P4.py`
  (Contains implementations of Naive Bayes variant used.)

6.2.2 LightGBM (LGBM)

Explanation: LightGBM is a gradient-boosting framework optimized for speed and efficiency on large, high-dimensional tabular data. Its leaf-wise tree growth and regularization mechanisms make it highly effective at capturing non-linear interactions while maintaining fast training times.
Files:

- `Lgbm1_P1_751.py`
- `Lgbm2WithMoreHyperParam_P1_740.py`

  Best Hyperparameters (after tuning):
  - Subsample: `0.8`
  - Regularization (λ): `0`
  - L1 Penalty (α): `0`
  - Num Leaves: `20`
  - Min Child Samples: `10`
  - Max Depth: `6`
  - Learning Rate: `0.01`
  - Column Sample by Tree: `0.6`
- `Bayes_MLP_Log_Lgbm_Xgb_Cat_P4.py`
- `Xg_LgbM_Cat_P1_736.ipynb`

  (Used as part of an ensemble experiment.)

6.2.3 CatBoost

Explanation: CatBoost is a boosting algorithm designed for categorical-heavy tabular data. It handles categorical features natively and avoids target leakage through ordered boosting, making it highly stable even without heavy preprocessing.
Files:

- `Bayes_MLP_Log_Lgbm_Xgb_Cat_P4.py`
- `Cat_P1_738.py`

- `Xg_LgbM_Cat_P1_736.ipynb`

  (CatBoost used within ensemble experiments.)

6.2.4 XGBoost (XGB)

Explanation: XGBoost is a powerful boosting algorithm that builds additive trees with optimized regularization. It is known for excellent performance on structured/tabular data and was tested to benchmark against LGBM and CatBoost.

Files:

- `Bayes_MLP_Log_Lgbm_Xgb_Cat_P4.py`
- `Xg_LgbM_Cat_P1_736.ipynb`

  (Includes XGBoost outputs used inside ensemble models.)

6.2.5 Random Forest

Explanation: Random Forests aggregate multiple decision trees using bagging, reducing variance and improving stability. They were used as a classical ensemble baseline for comparison with boosting methods.

Files:

- `AllModels_P2.ipynb`

  Best Hyperparameters:

  * Max Depth: `10`
  * Min Samples Leaf: `1`
  * Min Samples Split: `2`
  * Number of Trees: `200`

- `AllModels_P3.ipynb`

  Best Hyperparameters:

  * Number of Trees: `200`
  * Min Samples Split: `5`
  * Min Samples Leaf: `4`
  * Max Depth: `10`

6.2.6 Gradient Boosting

Explanation: Gradient Boosting builds trees sequentially, each correcting the errors of the previous ones. It is more interpretable and stable than more aggressive boosting frameworks, making it an excellent mid-complexity model.

Files:

- `AllModels_P2.ipynb`
- `AllModels_P3.ipynb`

Best Hyperparameters:

- Subsample: `0.8`
- Number of Trees: `200`
- Max Depth: `3`
- Learning Rate: `0.05`

6.2.7 Ensemble Methods

Explanation: Ensemble learning combines the strengths of multiple models to produce more stable and accurate predictions than any single estimator alone.

Files:

- `Xg_LgbM_Cat_P1_736.ipynb`

  (Implements an ensemble of **XGBoost**, **LightGBM**, and **CatBoost**, including probability averaging and weighted voting schemes.)

- `AllModels_P3.ipynb`

  (Contains broader ensemble-style experimentation using outputs from **Gradient Boosting**, **Random Forest**, **SVM**, and **Neural Networks**.)

7 Observations

- **Boosting Methods Lead the Pack:** LightGBM emerged as the strongest model with a Macro F1 score of **0.751**. This indicates that tree-based learners, especially boosting frameworks, are highly effective at capturing the subtle, non-linear dependencies between founder behaviour, startup characteristics, and retention outcomes.
- **Neural Networks Were Competitive but Not Superior:** Despite their ability to model complex interactions, the best Neural Network configuration reached a Macro F1 of **0.732**. While competitive, it struggled to match the stability and structured feature handling of LGBM and Random Forest.
- **SVM Performs Respectably:** The best SVM model achieved **0.733**, confirming that margin-based separation helps uncover meaningful boundaries in the founder feature space. However, it still fell short of boosted tree methods, likely due to the mixed categorical–numerical nature of the dataset.
- **Feature Independence Helps Tree Models:** EDA showed weak pairwise correlations among most features. This favoured ensemble tree methods, which naturally exploit many low-correlation variables without requiring heavy feature interaction engineering.
- **Outliers Carry Predictive Signal:** Removing outliers consistently worsened performance. Many "outliers" corresponded to realistic but rare founder behaviours (e.g., unusually high revenue with low satisfaction). Keeping them improved minority-class detection.
- **Class Imbalance Requires Macro F1:** The dataset is skewed toward retained founders. Macro F1 penalised models that ignored the "Left" class, exposing that some linear models underperformed despite high accuracy. Boosted trees and SVMs handled the imbalance notably better.
- **Overall Winner: LightGBM (P1 pipeline)** offered the best combination of interpretability, computational efficiency, and predictive performance, making it the most reliable choice for startup founder retention prediction.

8 Results

The experimental results showed that **ensemble tree-based methods** were the strongest performers for the founder–retention prediction task. **LightGBM** achieved the highest Macro F1 Score of **0.751**. These models handled heterogeneous, mixed-type features and non-linear interactions exceptionally well, making them a natural fit for this dataset.

SVM (0.733) and the **Neural Network (MLP)** (0.732) also performed competitively, confirming that the dataset does contain meaningful non-linear structure. However, they did not surpass the stability and generalization ability of the boosting-based models.

CatBoost and **Gradient Boosting** delivered good and decent performance too.

Model Type	Best Script / File	Macro F1 Score
LightGBM	`Lgbm1_P1_751.py`	**0.751**
CatBoost	`Bayes_MLP_Log_Lgbm_Xgb_Cat_P4.py`	0.744
Gradient Boosting	`AllModels_P2.ipynb`	0.744
Random Forest	`AllModels_P3.ipynb`	0.735
SVM (RBF)	`AllModels_P3.ipynb`	0.733
Neural Network (MLP)	`Neural_Lgbm_P4.py`	0.732
Naive Bayes	`Bayes_MLP_Log_Lgbm_Xgb_Cat_P4.py`	0.725

Table 2: Best Performance Achieved by Each Model Architecture

9 Interpretation

The overall results showed that **LightGBM** emerged as the strongest performer with a Macro F1 Score of **0.751**, while **SVM** (**0.733**) and the **Neural Network (MLP)** (**0.732**) followed closely behind. This performance gap reflects the nature of our dataset and how different model families respond to it.

LightGBM excelled because the dataset contains many mixed-type features (ordinal, binary, one-hot encoded categories) and several interaction-based engineered variables. Boosting methods like LightGBM thrive in such environments—each tree corrects the mistakes of the previous one, allowing the model to pick up subtle, high-frequency patterns without requiring heavy scaling or strict feature distributions.

SVM and MLP also performed strongly, indicating the presence of meaningful non-linear relationships. However:

- **SVM**: Benefited from RBF kernels but struggled slightly with the large number of encoded categorical features, making it harder to optimize margins consistently across dimensions.
- **Neural Network**: Learned complex patterns well but was more sensitive to hyperparameters and required careful regularization; even with tuning, it slightly underperformed compared to LightGBM on tabular, engineered data.

In short, the dataset's **tabular structure**, **mixed feature types**, and **interaction-heavy engineered features** collectively favored **LightGBM**'s boosting mechanism over purely geometric (SVM) or fully non-linear (MLP) approaches.

References

[1] *Lend or Lose: Loan Default Prediction Project*. GitHub Repository. https://github.com/standing-on-giants/Lend-or-lose-Loan-Default-prediction-project

[2] *Getting Started with Classification*. https://www.geeksforgeeks.org/getting-started-with-classification/