Start-up Founder Retention Prediction (Checkpoint 2)

Team Members

1. Kavya Gupta - IMT2023016
2. Nainika Agrawal - IMT2023034
3. Pragya Rai - IMT2023529

Github Link: https://github.com/kavyagupta3011/Classification

Task

he objective is to predict whether a startup founder will remain with their venture or exit, based on a wide range of personal, professional, and startup-related attributes. This is a binary classification problem with the target variable retention_status.
Understanding founder retention is important for multiple real-world applications:

- **HR and Organizational Analytics:** Identifying early signals of founder burnout or disengagement.
- **Startup Performance Forecasting:** Retention patterns often reflect team stability and long-term viability.
- **Investor and Venture Capital Insights:** Predicting leadership continuity helps inform investment risk.
- **Entrepreneurship Research:** Enables data-driven study of founder behaviour, satisfaction, and resilience.

The goal is to develop a predictive model that uses the provided features to estimate the likelihood of a founder staying or leaving, and optimize performance.

Dataset and Features Description

This dataset contains detailed information about startup founders, including their demographics, responsibilities, work habits, satisfaction levels, and the overall performance and structure of the startup they lead.

- **train.csv**: 59611 x 24
- **test.csv:** 14900 x 23
- **sample_submission.csv:** 5 x 2

Below describes each feature in the dataset:

- **founder_id** – Unique identifier for each founder.
- **founder_age** – Age of the founder.
- **founder_gender** – Gender identity (Male/Female/Other).
- **years_with_startup** – Total years the founder has worked in the current startup.
- **founder_role** – Position held (CEO/CTO/Co-founder etc.).
- **monthly_revenue_generated** – Startup's monthly revenue (may contain missing values).

- **work_life_balance_rating** – Self-reported work-life balance level.
- **venture_satisfaction** – Founder's satisfaction with the venture.
- **startup_performance_rating** – Overall performance rating of the startup.
- **funding_rounds_led** – Number of funding rounds led by the founder.
- **working_overtime** – Whether the founder frequently works overtime (Yes/No).
- **distance_from_investor_hub** – Distance to nearest investor/innovation hub.
- **education_background** – Founder's educational domain.
- **personal_status** – Relationship/marital status.
- **num_dependents** – Number of financially dependent individuals.
- **startup_stage** – Growth stage of the startup (Seed/Series A/Growth/etc.).
- **team_size_category** – Size category of the startup team.
- **years_since_founding** – Age of the startup in years.
- **remote_operations** – Whether the startup operates primarily remotely (Yes/No).
- **leadership_scope** – Nature of leadership duties (Operational/Strategic/Both).
- **innovation_support** – Whether innovation initiatives are encouraged (Yes/No).
- **startup_reputation** – Public/industry reputation level.
- **founder_visibility** – Public visibility of the founder.
- **retention_status** – **Target variable** indicating whether the founder *Stayed* or *Left*.

The following section outlines the necessary steps for conducting essential **Exploratory Data Analysis (EDA)** to gain insights, detect potential issues, and prepare data for model training.

1. Import Libraries and Load Dataset

The initial step in the EDA process involves importing essential libraries such as pandas, numpy, matplotlib, tensorflow, seaborn, and relevant modules from scikit-learn for data preprocessing. The dataset is loaded into a pandas DataFrame for further analysis and manipulation. After loading the train, test, and sample submission files, the dataset shapes were inspected to confirm successful import.

2. Data Overview

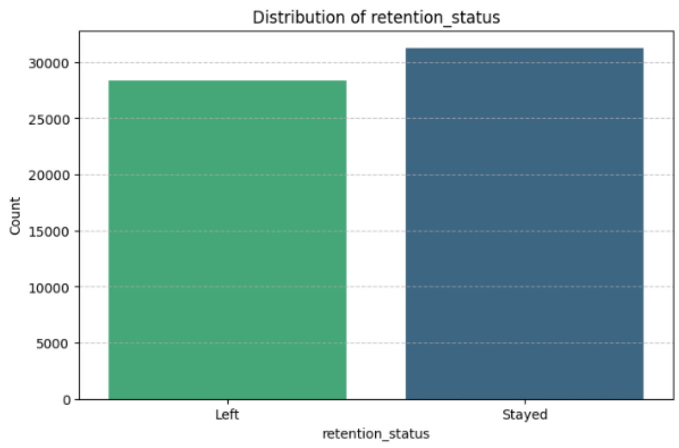
The dataset is first explored by viewing the initial rows, inspecting data types, checking for duplicates and invalid data, and detecting missing values to understand its structure and quality. A statistical summary of the numerical columns is then generated to examine their central tendencies, variability, and possible outliers.

3. Handling duplicates

We found 13 rows that have duplicates in this dataset. These were handled by dropping the duplicate row. New Train Shape: (59598, 24)

4. Handling target variable

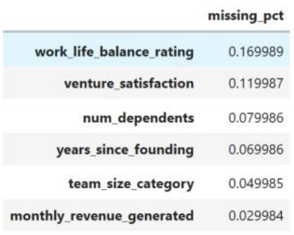
On analyzing the target variable retention_status, we saw it had 2 unique values: “Stayed” and” Left”. It had no missing values, and all feature columns were present in both train and test datasets. “Stayed” was the most frequent, appearing 31260 times.



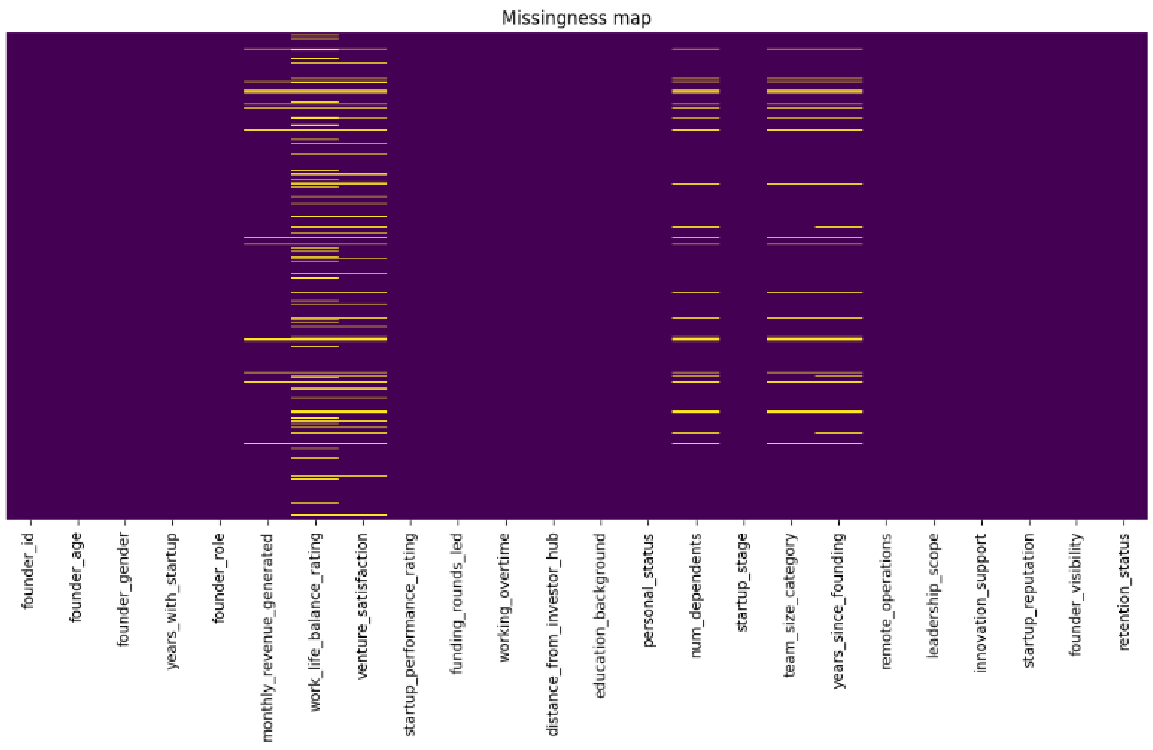
5. Handling Missing Values

A complete missing-value check was performed using column-wise missing percentage calculations. We saw that 6 features contained missing values.

```
work_life_balance_rating      10131
venture_satisfaction           7151
num_dependents                 4767
years_since_founding           4171
team_size_category             2979
monthly_revenue_generated      1787
```



	missing_pct
work_life_balance_rating	0.169989
venture_satisfaction	0.119987
num_dependents	0.079986
years_since_founding	0.069986
team_size_category	0.049985
monthly_revenue_generated	0.029984

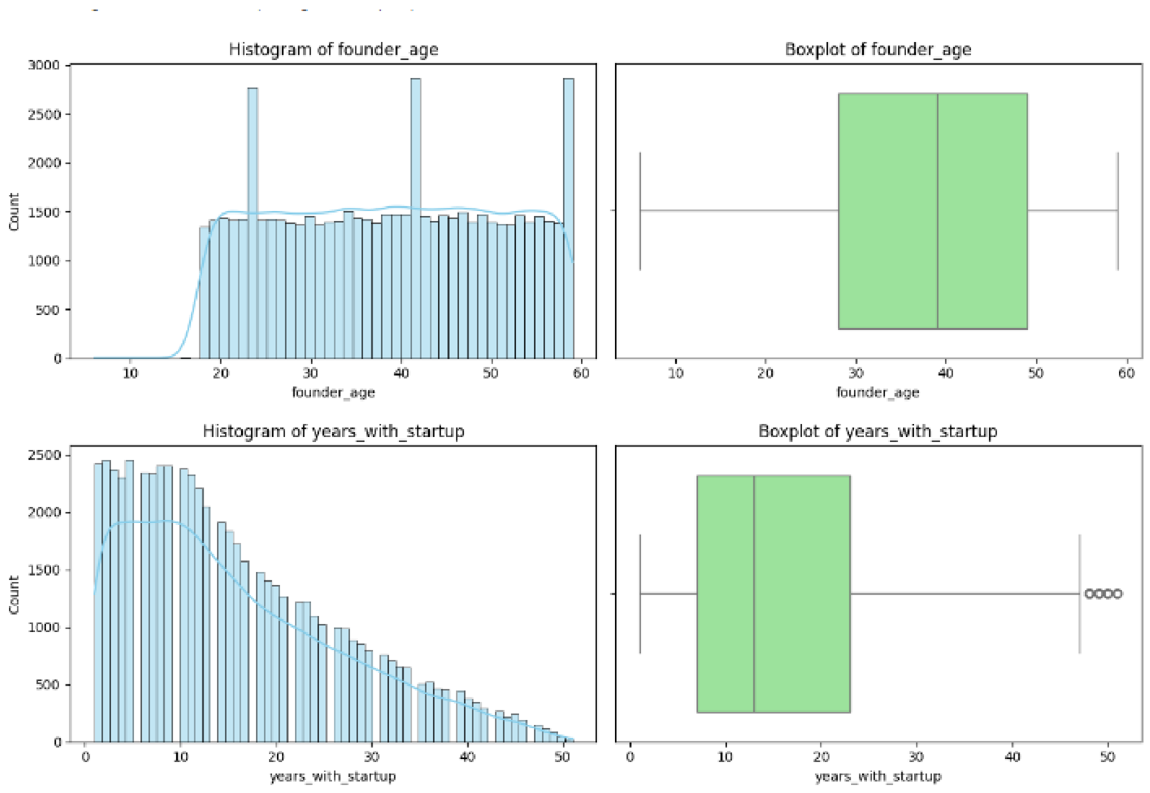


We created missing flags and imputed all the numeric missing features with the median, and categorical with most frequent.

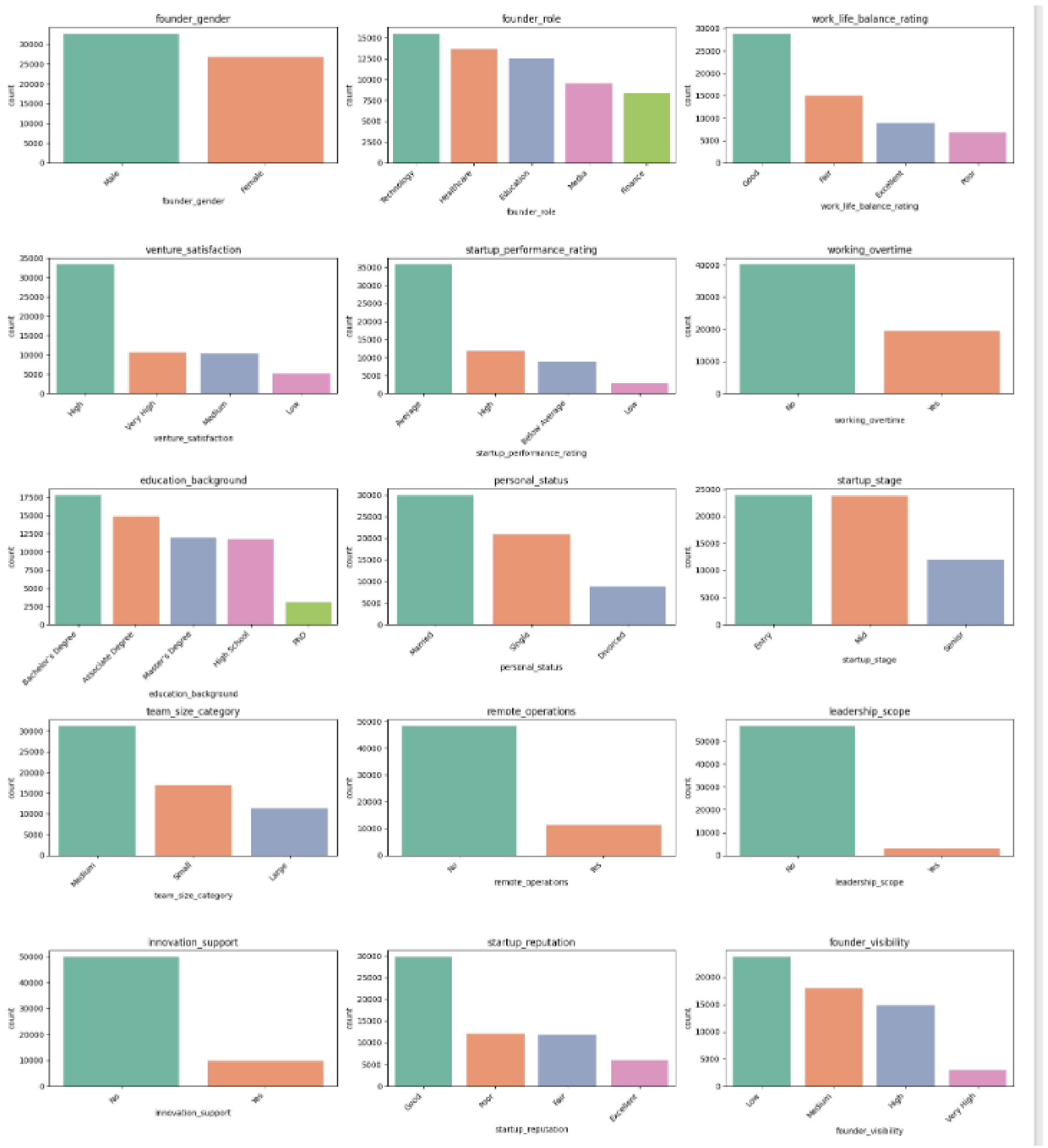
6. Exploratory Data Analysis (EDA)

Several plots were created to examine data distributions, identify outliers, and explore relationships between features. These visualizations offer valuable insights that guide the next steps in preprocessing and model development. The key visualizations and their interpretations are outlined below:

- **Numeric Feature Plots:**
  Plotted for all features, some shown below.

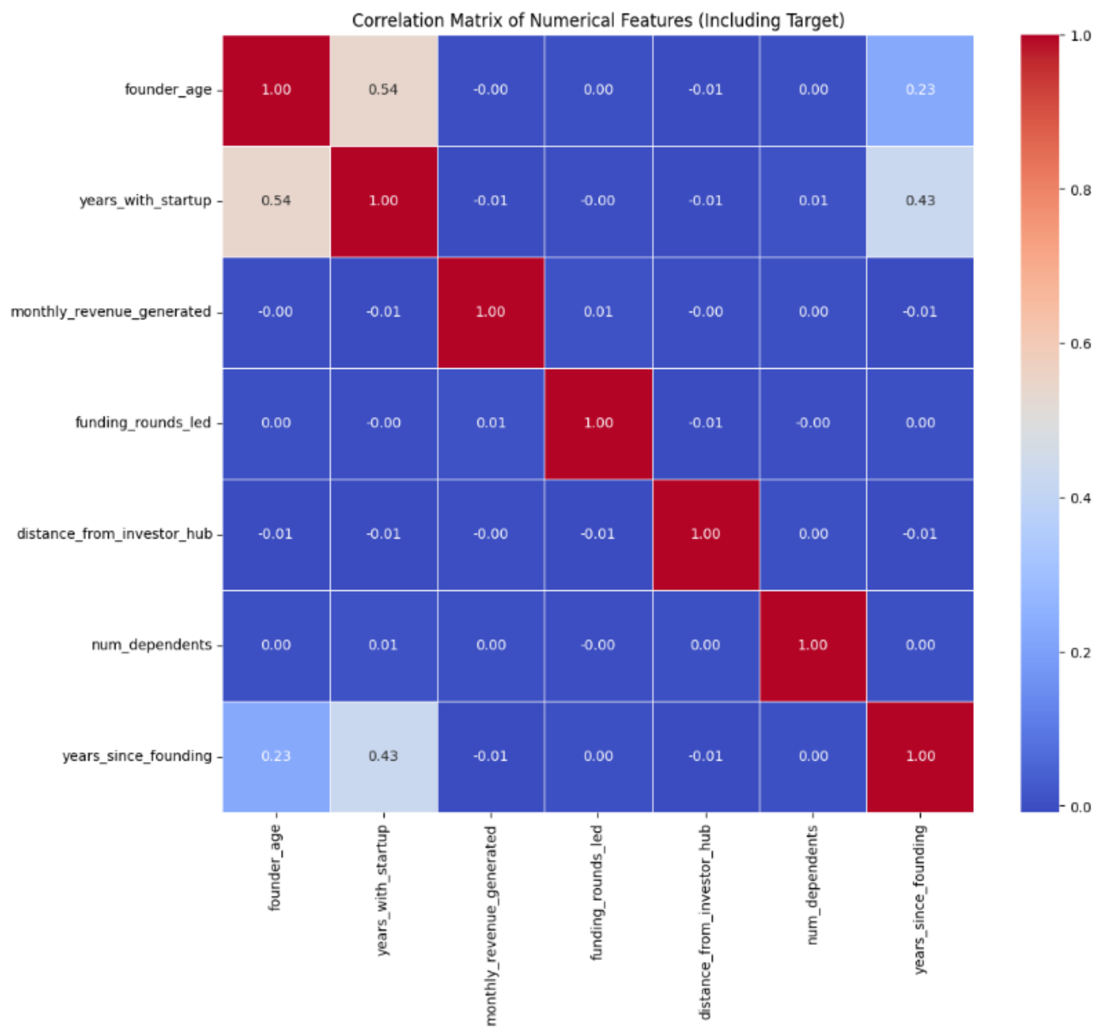


- **Categorical Feature Plots:**

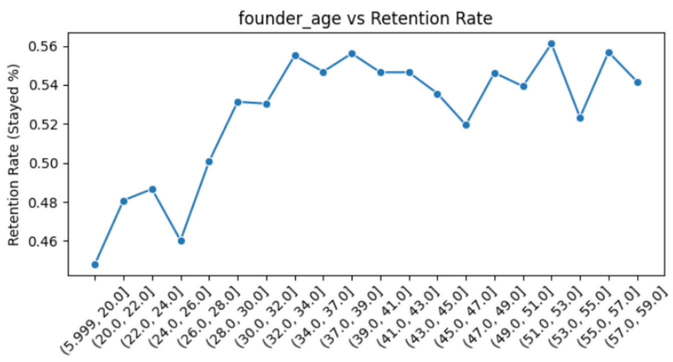


- **Correlation Matrix (Heatmap):**
  To understand how the numerical features relate to one another, we computed and visualized the correlation matrix. The heatmap displays pairwise Pearson correlation coefficients between all numerical variables in the dataset.
  Overall, the correlations among features are **very low**, indicating a dataset with minimal linear relationships between behavioral indicators. Most numerical features show correlation values close to 0.00, meaning: They are largely independent of each other and capture different aspects of the target variable. There is no multicollinearity problem. This is beneficial for tree-based models and neural networks because each feature provides unique information without redundancy.There is no risk of redundant predictors. The highest observed correlation is: founder_age vs years_with_startup: r ≈ 0.54
  Since none of the inputs have meaningful linear relationships:
    - The dataset is likely complex and non-linear in nature.

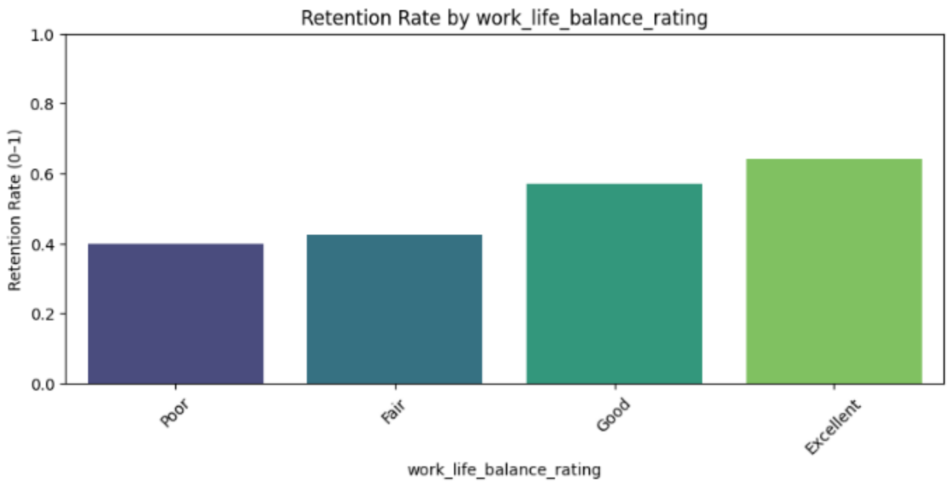
- The independence among features increases the importance of non-linear transformations, interactions, and deeper models to extract meaningful structure.



- **Numerical features vs target**
  To understand how each numerical feature influences the probability that a founder stays with the startup, we plotted Retention Rate (percentage of founders who stayed) against binned ranges of each numeric variable. This produces a smooth trend line for visualizing the relationship between a feature and the target variable. It answers: Does this numeric feature have a meaningful trend that affects the probability of retention?
  An upward trend indicates that higher feature values increase the probability of a founder staying.
  A downward trend suggests higher values decrease retention probability.
  A flat line means the feature does not meaningfully impact retention.
  Also, these plots highlight non-linear behaviors that correlation matrices cannot detect.

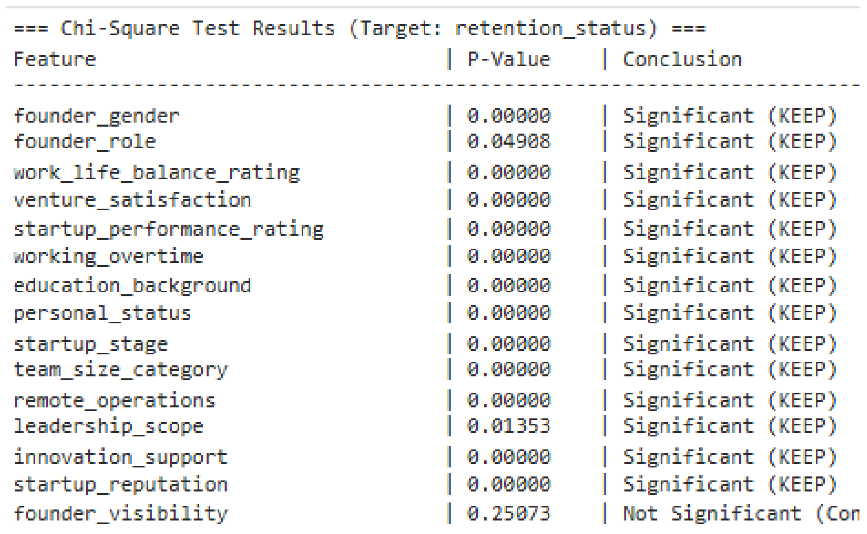


- **Categorical features vs target**
  - For each categorical feature, we calculated the average retention rate (mean of the target variable) within each category and visualized it using barplots. This helps us understand how founder retention varies across different groups such as work-life balance, founder role, education background, startup stage, etc. The height of the bar shows the proportion of founders who stayed. Categories with higher bars indicate groups that are more likely to retain founders. Categories with lower bars indicate groups associated with a higher exit likelihood. If bars for different categories show clear separation, the feature is highly informative.



- **Chi- Square Test for feature importance**

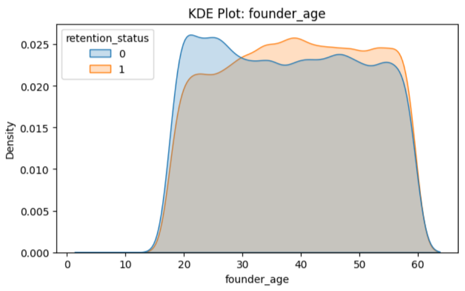
To evaluate the statistical association between each categorical feature and the target variable, we performed a Chi-Square independence test. This test helps determine whether the distribution of target differs significantly across different categories of each feature. Features with low p-values (< 0.05) are considered significantly associated with the target and are therefore informative for classification. The results clearly separate the features into two groups — significant predictors and non-significant predictors.



```
=== Chi-Square Test Results (Target: retention_status) ===
Feature                                   | P-Value    | Conclusion
---------------------------------------------------------------------
founder_gender                            | 0.00000    | Significant (KEEP)
founder_role                              | 0.04908    | Significant (KEEP)
work_life_balance_rating                  | 0.00000    | Significant (KEEP)
venture_satisfaction                      | 0.00000    | Significant (KEEP)
startup_performance_rating                | 0.00000    | Significant (KEEP)
working_overtime                          | 0.00000    | Significant (KEEP)
education_background                      | 0.00000    | Significant (KEEP)
personal_status                           | 0.00000    | Significant (KEEP)
startup_stage                             | 0.00000    | Significant (KEEP)
team_size_category                        | 0.00000    | Significant (KEEP)
remote_operations                         | 0.00000    | Significant (KEEP)
leadership_scope                          | 0.01353    | Significant (KEEP)
innovation_support                        | 0.00000    | Significant (KEEP)
startup_reputation                        | 0.00000    | Significant (KEEP)
founder_visibility                        | 0.25073    | Not Significant (Con
```

- **KDE Plots**

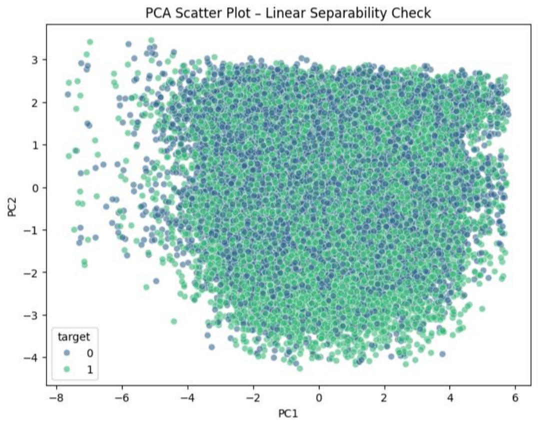
To further understand how numeric behavioral features vary across personality clusters, Kernel Density Estimate (KDE) plots were generated for the key numeric features. KDE plots provide a smooth estimate of the probability distribution of each feature while overlaying multiple target values in different colors, allowing us to visually inspect separation, overlap, and distribution shape.
These visualizations are particularly useful for assessing the discriminative power of each feature and for identifying whether it contributes meaningful signal for classification. Across most KDE plots, the curves show significant overlap, indicating that the numeric features tend to follow similar distribution patterns regardless of



retention.

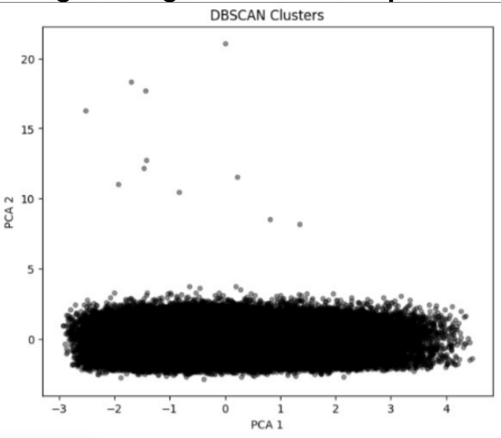
- **PCA and t-SNE**

Dimensionality reduction techniques (PCA and t-SNE) reveal that the ‘Stayed’ and ‘Left’ classes are heavily overlapped and do not form distinctly separable clusters. This indicates the absence of linear separability and only weak nonlinear patterns. Pairplot analyses further show that no single feature effectively discriminates between the two classes, suggesting that founder retention is governed by complex interactions



between multiple variables

DBSCAN for data clustering among two PCA Components:



7. Data Preprocessing
Now the data is preprocessed for modeling through the following steps:

- **Encoding categorical variables** using One Hot Encoding (or other encodings like label, target, etc, based on models) to convert categories into numerical values, enabling algorithms to interpret them effectively.
- **Standardizing numerical features** with a StandardScaler (or other Scalers like Robust, etc, based on models) to bring all features onto a similar scale and reduce model sensitivity to differences in magnitude.
- **Feature engineering**, when necessary, to create new features that capture nonlinear relationships or additional patterns in the data.

8. Splitting the Dataset
The train.csv dataset is divided into training and validation sets to allow model performance assessment on unseen data, typically using an 80/20 split. This step ensures the data is ready for both model training and evaluation.

Summary
The EDA and preprocessing steps we did include:

- Conducted a thorough inspection of data types, missing values, duplicates, and basic statistical properties.
- Performed exploratory data analysis to visualize distributions, identify outliers, and examine relationships between features and the target variable.
- Encoded categorical variables, standardized numeric features, and can implement feature engineering to capture additional patterns
- Split the dataset into training and validation sets (80/20) to prepare it for model training and performance evaluation.

Find all the preprocessing done here:
https://github.com/kavyagupta3011/Classification/blob/main/BinaryClassification/EDA_Binary.ipynb

Models Used For Training

1. Decision Tree
   - Randomized Search CV identified Gini impurity and min_ samples_ leaf = 1 as the best split configuration. This combination allowed the tree to capture essential retention patterns while keeping splits simple and statistically meaningful, resulting in a stable and competitive F1 score of 0.716.

2. K nearest neighbours
   - KNN performs poorly (F1 = 0.684) because the dataset violates almost all assumptions that KNN relies on.
   - The high-dimensional, one-hot-encoded features destroy meaningful distance relationships (curse of dimensionality). The classes overlap heavily, making local neighborhoods mixed and noisy.
   - KNN is highly sensitive to outliers and treats all features equally, causing irrelevant or noisy dimensions to dominate leading to significantly lower F1 scores compared to tree-based and kernel-based models.

3. Random Forest
   - Adding RandomizedSearchCV with 3-fold stratified CV allowed model to explore meaningful hyperparameters efficiently, improving generalization. This tuning of cross validation parameter raised performance best F1 of 0.734.

4. XGBoost
   - Initially XG Boost gave F1 score of 0.740 and ensembling with other models did not perform that well.

5. Gaussian Mixture Models
   - Using Bayesian posterior probabilities and optimizing the decision threshold (best threshold = **0.37**) reduced false positives and false negatives more effectively than a default 0.5 cutoff. This calibration step significantly boosted the model's recall-precision balance, resulting in a competitive test F1 score of **0.719**.

6. Bayes Classifier
   - The optimized version tested multiple PCA dimensions (6 to 50 components) and automatically selected the number that maximized validation F1. This removed guesswork and ensured the classifier received the most informative, noise-reduced representation, leading to a more efficient decision boundary for Naive Bayes.
   - Instead of fixing PCA at 30 components, the optimized pipeline evaluated several PCA sizes on a stratified validation split. This ensured the model used exactly the dimensionality that generalized best, reducing both underfitting (too few components) and overfitting (too many components).
   - The improved pipeline tuned PCA before fitting the final model, then refit PCA on the full dataset using the optimal number of components. This created a more stable, well-regularized feature space for Gaussian Naive Bayes, increasing the test F1 score from 0.698 to 0.721.

7. Catboost
   - The optimized model introduced ratio features (e.g., revenue_per_year, life_investment_ratio) and interaction features (e.g., satisfaction × balance), which captured deeper founder-startup relationships. This enriched feature

space allowed CatBoost to learn stronger patterns, improving F1 performance.
- By merging infrequent categories into an “OTHER” class, the model avoided overfitting to extremely rare values. This stabilized tree splits, improved categorical representation, and enhanced generalization on unseen data.
- Replacing a single train-validation split with 5-fold cross-validation produced five diverse CatBoost models whose predictions were averaged. This ensemble effect reduced variance, prevented overfitting, and significantly boosted the F1 score from 0.677 to 0.744.

8. SVM

- The optimized model added ratio features and interaction features (e.g., revenue_per_year, commitment × team, satisfaction × balance). These handcrafted nonlinear signals made the dataset more separable, allowing the SVM—especially after Nystroem mapping—to better identify differences between “Stayed” and “Left” founders.
- The initial model used limited Nystroem components (100–500), which restricted the expressiveness of the approximate RBF kernel.
  The optimized version increased this to up to 1200 components, giving the SVM a much more powerful nonlinear feature space, closely mimicking a full RBF kernel while remaining computationally efficient. This directly increased the model’s ability to capture complex retention patterns.
- The optimized RandomizedSearch used a narrower, better-chosen search space (gamma values, n_components, and C) that was specific to this dataset. This allowed the SVM to find a more optimal balance between kernel smoothness (gamma), nonlinear capacity (n_components), and margin complexity (C), improving the F1 score from 0.741 to 0.746.

9. Neural Network:

The improved MLP introduced:

- A smaller network (64 to 32 units)
- High L2 (alpha=0.05) regularization
- The optimised neural architecture focused on learning general retention patterns instead of memorizing noise. This controlled complexity increased test F1.
- The optimized version used Stratified 5-Fold cross-validation, producing five independently trained neural networks whose predictions were averaged. This ensemble-like effect lowered variance, captured more stable patterns, and improved the final F1 score.
- The F1 score improved from 0.736 to 0.742

10. Logistic Regression

- The optimized version performed a grid search over regularization strengths (C) combined with fine-grained threshold tuning (0.15 – 0.85).
- This jointly optimized the classifier’s margin and the probability cutoff for the highest weighted F1 score, resulting in a small improvement from 0.741 to 0.742.

11. Catboost+KNN

- CatBoost transforms categorical and numerical features into powerful tree-leaf embeddings, capturing non-linear interactions that raw KNN cannot detect. This makes distances between samples more meaningful, improving KNN’s ability to separate “Stayed” vs “Left”.
- The score for pure KNN improved from 0.684 to 0.708.

11. LightGBM

- LightGBM performed exceptionally well because its leaf-wise gradient boosting structure is highly effective at capturing complex, nonlinear interactions present in the founder-startup dataset. The model benefits from its ability to split deeper on informative features such as revenue-derived ratios, satisfaction scores, and team-related attributes, enabling it to learn subtle retention patterns that simpler models often miss.
- Additionally, the optimized hyperparameters (num_leaves, learning rate, and n_estimators) helped LightGBM balance bias and variance, producing strong generalization on unseen data. Its native handling of mixed-type features, robustness to missing values, and fast histogram-based training further contributed to achieving the strongest performance among tree-based models in this setup.
- It achieved an F1 score of 0.748.

12. SVM and Lightgbm

- The LightGBM + SVM ensemble model integrates both gradient-boosted decision trees and kernel-based nonlinear separation to capture complementary patterns in founder retention behavior. LightGBM leverages engineered features such as revenue_per_year, life_investment_ratio, and satisfaction × balance interactions to model complex tree-based hierarchies, while the Nystroem-enhanced SVM component projects the data into a high-dimensional RBF feature space, enabling sharper decision boundaries for subtle retention signals.
- Using cross-validated hyperparameter tuning for both models, the ensemble blends LightGBM probabilities with SVM's calibrated scores through weighted soft-voting, balancing tree-based interpretability with SVM's nonlinear discrimination. This hybrid architecture reduces model variance, strengthens generalization, and it achieved a Kaggle F1 score of 0.748.

13. Catboost + SVM

- The CatBoost + SVM hybrid model leverages the strengths of two fundamentally different learning paradigms to capture complex patterns in founder retention behavior. CatBoost first learns deep nonlinear decision structures from mixed numerical-categorical data and transforms each sample into high-level "leaf embeddings"; compact vectors representing the exact tree paths the model believes best describe a founder's profile and startup environment.
- These embeddings encode very rich nonlinear interactions (e.g., age × team size × satisfaction levels) that traditional preprocessing cannot capture. After generating these structured embeddings, a Linear SVM is trained on top to find the most discriminative hyperplane in this transformed feature space.
- This two-stage process effectively compresses nonlinear tree logic into a form that SVMs can separate linearly, reducing overfitting while enhancing generalization on unseen founders. As a result, the hybrid model achieved a strong F1 performance of 0.746 on Kaggle.

14. PCA + SVM

- The PCA + SVM model provides a fast, memory-efficient baseline that reduces the high-dimensional, mixed categorical-numerical feature space into a compact set of 50 principal components before classification. (Because 50 PCA components preserve most of the dataset's variance while removing noise, giving the best balance between accuracy and speed for SVM)
- By working in this compressed space, a linear SVM can identify a separating hyperplane without overfitting on sparse one-hot-encoded features or noisy categorical interactions.
- Although PCA is a linear transformation and cannot fully capture the complex

nonlinear dependencies present in founder retention behavior, it still extracts strong global structure, enabling the SVM to achieve a solid F1 performance of 0.740.

15. PCA + NN

- The PCA-Neural Network pipeline compresses the original high-dimensional, noisy feature space into 40 principal components that retain the most informative variance while removing redundant correlations introduced by one-hot encoding.
- This transformation produces a cleaner, denser representation on which the neural network can learn smoother decision boundaries. By using a compact architecture, ReLU activations, standardized inputs, and Adam optimization, the model efficiently captures nonlinear founder-startup relationships while avoiding overfitting.
- Although PCA discards some fine-grained categorical structure, the reduced dimensionality dramatically stabilizes MLP training and improves generalization, achieving a Kaggle score of 0.705 with very fast training time.

16. Comparison of SVM and Neural Network on Full Dataset vs 20% Subset

Data Splitting:

- Full Dataset:
  - Training Set: 47,688 rows
  - Validation Set: 11,923 rows
- 20% Subset Dataset:
  - Training Set: 11,922 rows (20% of the full dataset)
  - Validation Set: 47,689 rows (80% of the full dataset)

1. Full Dataset:

- Model Training and Evaluation:
Support Vector Machine (SVM): Validation F1 Score: 0.7529
Neural Network (NN): Validation F1 Score: 0.7632
- Kaggle Submission Results:
SVM F1 Score on Kaggle: 0.734
NN F1 Score on Kaggle: 0.657

- Conclusion:
When trained on the full dataset, NN performed slightly better than SVM, achieving a higher F1 score (0.7632 vs 0.7529). Both models performed well on the full dataset.
On Kaggle, despite NN performing better on the training data, it achieved a lower F1 score compared to SVM, indicating that SVM may generalize better in this case.

2. 20% Subset Dataset:

- Model Training and Evaluation:
Support Vector Machine (SVM): Validation F1 Score: 0.7488
Neural Network (NN): Validation F1 Score: 0.6980
- Kaggle Submission Results (20% Subset):
SVM F1 Score on Kaggle: 0.734
NN F1 Score on Kaggle: 0.657

- Conclusion:
When trained on the 20% subset, SVM outperformed NN, achieving a higher F1 score (0.7488 vs 0.6980). Similar to the full dataset, SVM achieved a significantly higher F1 score on Kaggle than the Neural Network. This result suggests that SVM generalizes better when trained on a smaller dataset

Discussion on the Performance of Different Approaches:

- Tree-based models dominated because they naturally capture nonlinear interactions.Frameworks like CatBoost, XGBoost, and LightGBM performed strongly as they model hierarchical splits, automatically handle mixed numeric–categorical features, and adapt to complex relationships such as satisfaction × team size or revenue × age.
- Algorithms like Logistic Regression and KNN faced difficulty because the dataset contains many one-hot encoded features and nonlinearly separable classes, making Euclidean distances and linear boundaries ineffective.
- Kernel-based and feature-lifting methods provided major performance boosts.
- Models using Nystroem kernel approximation (SVM) or tree-leaf embeddings (CatBoost+SVM) significantly improved separation in the transformed feature space, demonstrating that nonlinear projection is essential for this task.
- Neural networks captured nonlinear structure but required careful regularization. While MLPs could learn complex relationships, they were sensitive to noise and missing data patterns. Smaller architectures with regularization performed better, but still lagged behind optimized SVM and boosting models.
- Dimension-reduction methods offered speed but not top accuracy. Linear and distance-based models struggled due to high dimensionality and class overlap.
- PCA-based models (PCA+SVM, PCA+NN) produced fast, stable baselines, yet struggled to retain the fine-grained categorical interactions required to fully separate “Stayed” vs “Left” founders.
- Nonlinear and ensemble approaches consistently delivered the highest F1 scores.
- Techniques that combined multiple learning perspectivesm like LightGBM + SVM (0.748 F1) or CatBoost + SVM, outperformed single linear models. This showed that leveraging both tree-based structure and kernel-based separation leads to stronger generalization.

Overall insight:
The founder retention dataset exhibits high overlap, mixed features, and nonlinear dependencies, meaning models that transform the feature space or model deep nonlinear interactions outperform simpler, linear, or distance-based classifier.

Why Tree-Based Embeddings Improved pure SVM

- Pure SVM struggled because the original dataset is extremely high-dimensional (due to one-hot encoding), noisy, and full of nonlinear interactions that a simple kernel approximation cannot fully capture.
- In this raw space, many features are sparse, weakly informative, or correlated, making the class boundary far from linearly separable. However, tree-based models like CatBoost and LightGBM automatically discover deep nonlinear splits, interactions, and hierarchical patterns.
- Their leaf embeddings compress these complex interactions into a dense, compact, and highly informative representation where classes become much closer to linearly separable.
- When a Linear SVM is applied on top of these tree-generated embeddings, it finally sees a clean, structured, high-dimensional feature space, making margin-based separation effective. This is why SVM alone reached around 0.746, but SVM combined with tree models reached higher scores up to 0.748.

Our interpretation of why SVM ensemble Gave the highest F1 Score:

- LightGBM captures global nonlinear feature interactions, while SVM focuses on local margin-based separation in high-dimensional space.
- Features like *revenue_per_year*, *life_investment_ratio*, and *satisfaction × balance* are modeled through deep, leaf-wise splits that reveal hierarchical patterns other models miss.
- Mapping the data into an approximate RBF space gives SVM a powerful nonlinear representation, enabling sharper class boundaries where tree models struggle.
- LightGBM contributes stable probabilities, while SVM corrects ambiguous boundary regions—together reducing misclassifications. LightGBM reduces underfitting, SVM reduces overfitting, and their combination captures both coarse and fine-grained retention signals.
- SVM alone achieved 0.746, but the ensemble improved this to 0.748, making it the best-performing approach overall.

Interesting Observations:

- **Nonlinear structure dominated the dataset**, which is why models like CatBoost, LightGBM, and SVM with RBF features consistently outperformed linear or distance-based methods.
- **Interaction and ratio features had a big impact**, revealing that relationships (e.g., satisfaction × balance, revenue_per_year) were more informative than raw variables.
- **High-dimensional one-hot encoding hurt simpler models**, while PCA compression made SVMs and neural networks more stable and less prone to overfitting.
- **Tree-based leaf embeddings worked surprisingly well**, allowing even a linear SVM to achieve strong performance by operating on CatBoost's non-linear tree representations.
- **Ensembles consistently improved results**, showing that LightGBM's hierarchical splits and SVM's margin-based separation capture complementary patterns in founder retention.
- **Distance-based and probabilistic models struggled**, indicating that the feature space is complex and not well-separated using simple similarity metrics or Gaussian assumptions.

References:

https://scikit-learn.org/stable/supervised_learning.html

https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/

https://www.geeksforgeeks.org/machine-learning/neural-networks-a-beginners-guide/

https://www.geeksforgeeks.org/machine-learning/lightgbm-light-gradient-boosting-machine/