Start-up Founder Retention Prediction

AIT 511 / Machine Learning

Assignment 2: Binomial

Project Overview

Challenge

This dataset captures information about startup founders, their personal and professional attributes, and various factors influencing their decision to stay or leave their ventures. It can be used for predictive modeling, especially in understanding founder retention — i.e., identifying patterns that contribute to whether a founder continues with their startup or exits. References [1]

Dataset

Each participant in the dataset is represented by features describing daily routines, activity levels, social engagement, and expressive tendencies. The target variable is a personality cluster label, which represents a group of individuals who share similar behavioral patterns.

Data set folder taken from kaggle includes → train.csv: **59611 x 24** and test.csv: **14900 x 23**

Column/Feature Name	Type	Meaning
founder_id	int64 (Index column)	Unique identifier assigned to each founder
founder_age	int64	Age of the founder in years
founder_gender	object	Gender of the founder
years_with_startup	int64	Number of years the founder has been associated with the startup
founder_role	object	Primary role of the founder within the startup
monthly_revenue_generated	float64	Average monthly revenue generated by the startup
work_life_balance_rating	object	Subjective rating of work–life balance
venture_satisfaction	object	Founder's satisfaction level with the venture
startup_performance_rating	object	Perceived overall performance of the startup
funding_rounds_led	int64	Number of funding rounds led by the founder
working_overtime	object	Indicates whether the founder frequently works overtime
distance_from_investor_hub	int64	Distance from the nearest major investor hub
education_background	object	Highest educational qualification of the founder
personal_status	object	Personal status of the founder (e.g., single, married)
num_dependents	float64	Number of dependents supported by the founder
startup_stage	object	Current developmental stage of the startup
team_size_category	object	Categorical representation of team size
years_since_founding	float64	Years elapsed since the startup was founded
remote_operations	object	Indicates whether the startup operates remotely
leadership_scope	object	Level of leadership responsibility handled by the founder
innovation_support	object	Availability of innovation and R&D support
startup_reputation	object	Market reputation of the startup
founder_visibility	object	Public and media visibility of the founder
retention_status	object (Target column)	Founder retention outcome used as the prediction target

Table 1: Startup Founder Dataset Feature Description

Data Processing Steps

Step 1 : Import Libraries and Load Dataset

The initial step in the EDA process involves importing essential libraries such as pandas, numpy, matplotlib, seaborn, and relevant modules from scikit-learn for data preprocessing. The dataset is loaded into a pandas DataFrame for further analysis and manipulation.

'Libraries Used'

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.naive_bayes import GaussianNB

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.layers import BatchNormalization, Dropout, Dense
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

from sklearn.linear_model import LinearRegression, ElasticNetCV, Lasso, Ridge
from sklearn.kernel_ridge import KernelRidge
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from xgboost import XGBRegressor

from sklearn.compose import TransformedTargetRegressor

```

Step 2 : Feature Grouping and Null Values

Table 2: Feature Classification by Category for Startup Founder Dataset

Category	Feature Name	null values
Categorical	founder_gender	0
	founder_role	0
	work_life_balance_rating	10144
	venture_satisfaction	7164
	startup_performance_rating	0

Continued on next page

Category	Feature Name	null values
	working_overtime	0
	education_background	0
	personal_status	0
	startup_stage	0
	team_size_category	2992
	remote_operations	0
	leadership_scope	0
	innovation_support	0
	startup_reputation	0
	founder_visibility	0
	retention_status	0
Numerical	founder_age	0
	years_with_startup	0
	monthly_revenue_generated	1800
	funding_rounds_led	0
	distance_from_investor_hub	0
	num_dependents	4780
	years_since_founding	4184
Identifier	founder_id	0

We grouped features so that dealing with the same kind of features can be done in a simple pipeline process (instead of using pipeline, it can also be done using arrays and for loops)

Step 3 : Exploratory Data Analysis (EDA)

Various plots were generated to analyse the data distribution, detect outliers, and explore relationships among features. These visualisations provide critical insights that inform subsequent preprocessing and modelling decisions. Key visualisations and their interpretations are detailed below:

Boxplots

Boxplots were used to examine each numerical feature for potential outliers, which appear as points beyond the plot's whiskers.

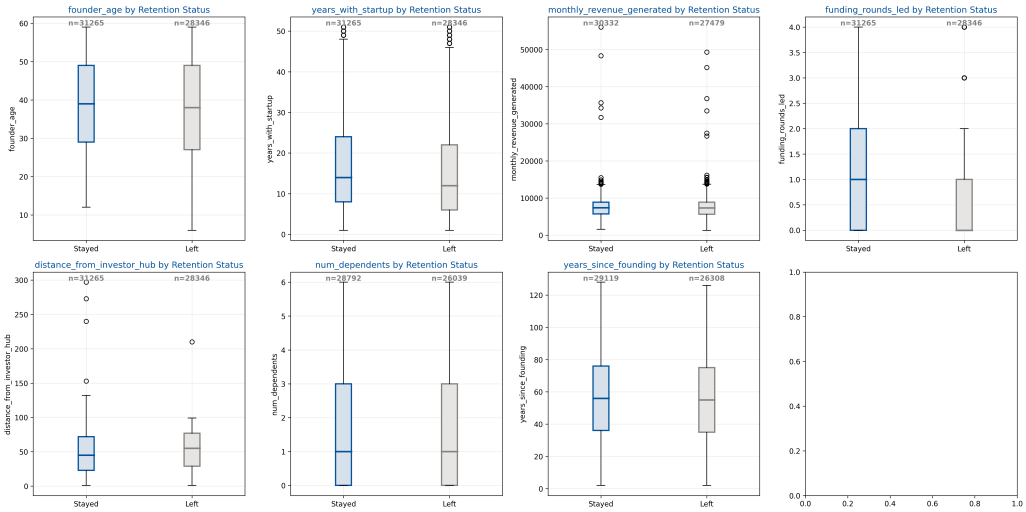

Figure 1: Box plot

Key Takeaways

- Medians for most numerical features differ only slightly between Stayed vs Left.
- Funding rounds led is low in both groups — median around 1.
- Founder age, years with startup, and years since founding show mild spread difference but still overlapping.
- High revenue outliers exist but do not distinctly separate retention groups.
- This confirms: numeric features alone don't strongly distinguish retention behavior.

Histogram

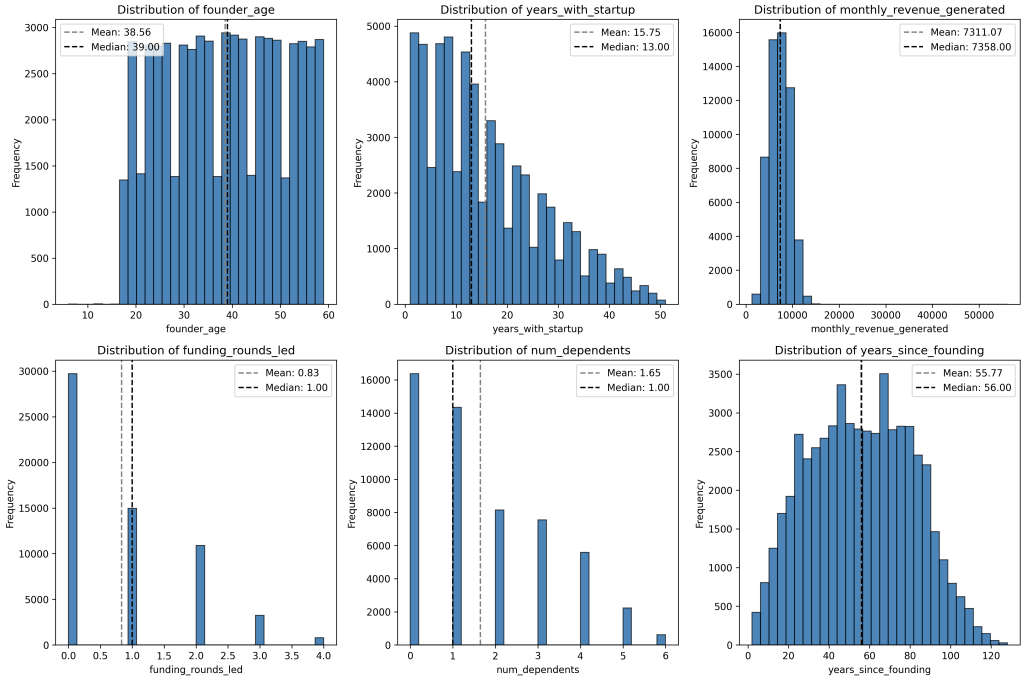

Figure 2: Numerical Distribution

Key Takeaways

- Founder age centers near 38–40 years with near-uniform spread across middle range.
- Years with startup is right-skewed — most founders are early-stage (≈ 5–15 years).
- Monthly revenue highly right-skewed — majority under ¤10K– ¤15K range.
- Funding rounds are sparse — mostly 0 or 1 per founder.
- Years since founding centered around 55 years showing long-running startups.
- Overall distributions indicate wide variance but heavy clustering toward lower values for most numeric features.

Violin Plots

Key Takeaways

- Distribution shapes are mostly similar between Stayed & Left groups, indicating no drastic separation.
- FSlightly higher years_with_startup and years_since_founding noticed among those who stayed.
- Revenue & funding distributions are skewed, with long right tails (few high-value founders).
- Number of dependents is low for most — rarely exceeds 3–4.
- Overall suggests retention is weakly dependent on numeric attributes at distribution level.

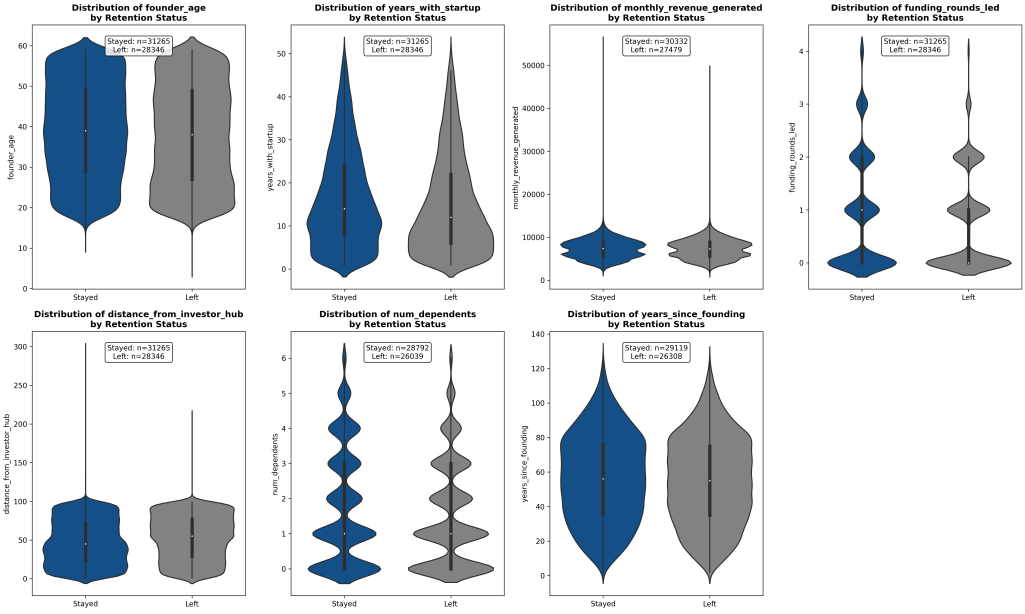

Figure 3: Violin plots

Correlation Heat Map

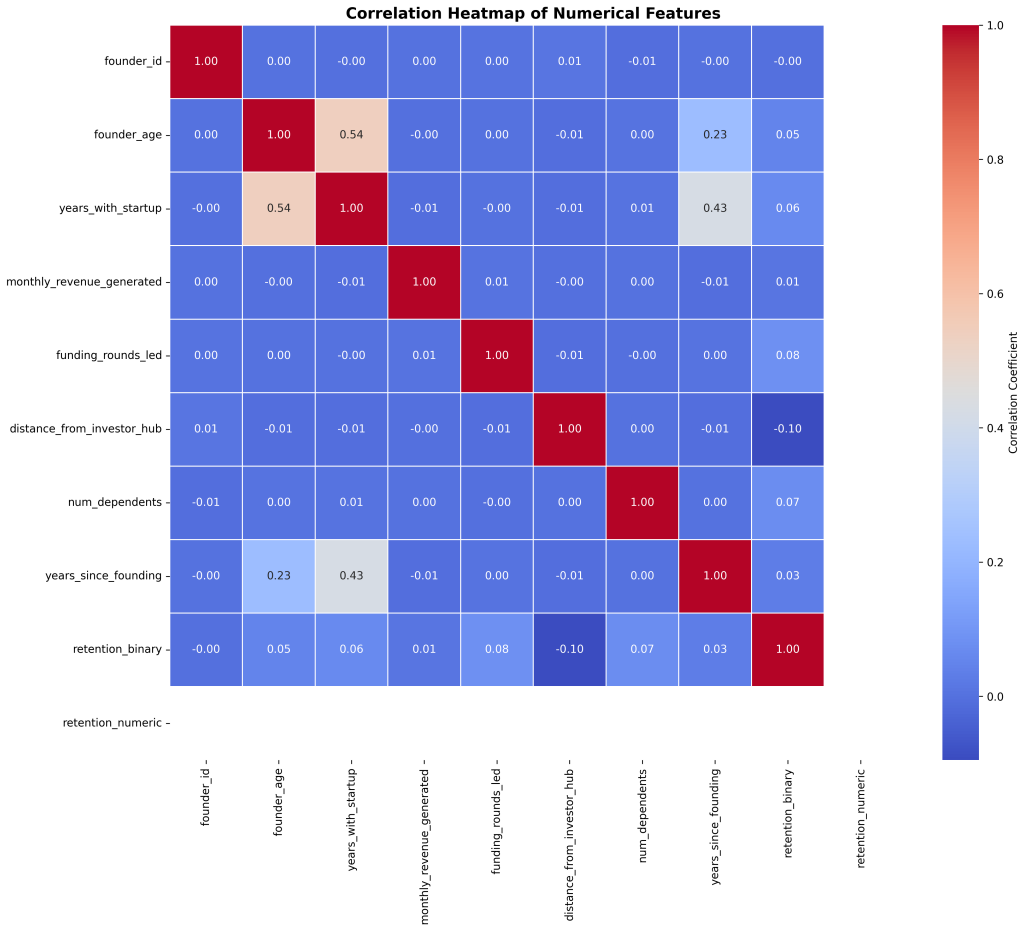

Figure 4: Correlation heat map using only Numeric Columns

Key Takeaways

- Most numerical features show very low mutual correlation.
- Founder age moderately correlates with years with startup (≈ 0.54).
- Years with startup and years since founding have a positive relation, indicating experience alignment.
- No strong linear predictors for retention emerge from correlations alone.
- Retention-related variables show weak independent influence, suggesting retention depends on multiple interacting features rather than a single numeric factor.

Step 5 : Data Preprocessing

Data Preprocessing includes:

1. Feature Engineering
2. Handling missing values
3. Scaling of dataset
4. Encoding

Feature Engineering

We tried but achieved poor results so we dropped it.

Handling missing values

1. Numerical Columns
   Handling numerical columns you need to see the data distribution of the feature and based on that you need to decide whether to use mean or median to fill missing values. When we saw the variance comparison for all the numerical data features all showed replacing them with median is good. see Figure 2.

   We got **MEDIAN** better.

2. Categorical Columns
   For categorical columns, we took a look at the count of each category the feature has. If a certain category count far exceeds others then we replaced the missing values with it, but if they are nearly equal then may be replace them with **'Unknown'**.

   We got **MODE** better.

Scailing of Data set

Scaling of all numerical columns is done using ***Standard Scaling method***. Why ? Because our dataset does not many outliers as observed in Figure 1.

Encoding

We used ***One hot*** encoding to encode the categorical columns.

Step 6 : Training-Validation Split

The dataset is split into training and testing sets, ensuring that model performance can be evaluated on unseen data. A common split ratio is 80% for training and 20% for testing. This step prepares the dataset for model training and evaluation.

'Split Data into Training and Validation'

```
# Split processed data into 80% train and 20% validation
X_train, X_val, y_train, y_val = train_test_split(
    X_processed_df,  y, test_size = 0.2, random_state = 42 )
```

Models Used and Hyperparameter tuning

Models Used:

1. MLP, SVM
2. XGBoost, KNN
3. Logsitic regression
4. Naive Bayes

Support Vector Machine (SVM)

A Support Vector Machine (SVM) classifier with a radial basis function (RBF) kernel was employed to model nonlinear decision boundaries. The model configuration is as follows:

- Kernel: RBF
- Probability estimation: Enabled
- Random state: 42

Input features were standardized before training. Model evaluation was carried out using accuracy, F1-score, AUC, and 5-fold stratified cross-validation.

Multi-Layer Perceptron (MLP) - **Best Model**

A Multilayer Perceptron (MLP) classifier was implemented using the TensorFlow/Keras framework for binary classification. The network consists of four hidden layers with the following configuration:

- Layer sizes: $\{256, 128, 64, 32\}$
- Activation functions: GELU, ReLU and tanh

The output layer uses a sigmoid activation function for probability estimation. The model was trained using the Adam optimizer and the binary cross-entropy loss function with the following training parameters:

- Batch size: 32
- Maximum epochs: 20
- Validation split: 80:20
- Early stopping patience: 10

Early stopping was employed to prevent overfitting by monitoring the validation loss and restoring the best-performing weights. Model performance was evaluated using classification accuracy and macro F1-score on the validation set.

Logistic Regression

A Logistic Regression classifier was implemented as a baseline linear model for classification. The model was trained with the following configuration:

- Solver: Default (LBFGS)
- Maximum iterations: 1000
- Random state: 42

Feature scaling was applied prior to training. Model performance was evaluated using accuracy, F1-score, AUC, and 5-fold stratified cross-validation.

K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) classifier was used as a distance-based, instance learning method. The default sklearn configuration was adopted:

- Number of neighbors: Default
- Distance metric: Euclidean
- Weighting scheme: Uniform

All input features were standardized prior to training. Model performance was assessed using accuracy, F1-score, AUC, and cross-validation.

Naive Bayes

A Gaussian Naive Bayes (GNB) classifier was implemented under the assumption that all features follow class-conditional Gaussian distributions and are conditionally independent. The model was trained using the default sklearn parameters without explicit hyperparameter tuning. Since Naive Bayes is sensitive to feature scales, standardized input features were used. Performance was evaluated using accuracy, F1-score, AUC, and stratified cross-validation.

XGBoost

The Extreme Gradient Boosting (XGBoost) classifier was used as an ensemble tree-based boosting model. The following configuration was applied:

- Objective function: Binary logistic loss
- Evaluation metric: Logarithmic loss
- Random state: 42

XGBoost was trained on unscaled feature values. Model performance was evaluated using accuracy, F1-score, AUC, and 5-fold stratified cross-validation.

> **Note**
>
> We tried Grid Search CV for Neural Networks models but it is taking lot of time so we done tuning manually for some values.

Summary

S.No	Submission File	Score	Remarks
1	**mlp_submission_v8**	**0.749**	**Feature-engineered MLP with scaling.**
2	mlp_submission_v2	0.741	Scaled features with MLP.
3	mlp_submission_v1	0.747	Base features with normalization.
4	submission_ensemble	0.744	Soft-voting ensemble of models.
5	submission_stacking	0.726	Stacked meta-learning architecture.
6	XG_Boost	0.741	Gradient boosting on raw features.
7	logistic_regression	0.641	Linear model with standardization.
8	svm_submission_v2	0.699	RBF kernel with scaled inputs.
9	bayesian_modelling	0.686	Gaussian Naive Bayes classifier.

Table 3: Summary of Submission Files and Model Variants

Team Members

Project Katakam - (Team Name)

S.No.	Name (Roll No.)	Role
1	Mohit Jagini (IMT2023528)	Member
2	Katakam Shashidhar Sai (IMT2023567)	Team Leader
3	Hardhik Dhavala (IMT2023579)	Member

> **Note**
>
> Click here to email the team leader with team members in CC

References

[1] Kaggle, "Start-up founder retention prediction challenge." https://www.kaggle.com/competitions/start-up-founder-retention-prediction, 2025. Kaggle Competition.

[2] P. Katakam, "Start-up founder retention prediction challenge." https://github.com/DHardhik/ML-Challenge, 2025. Accessed on November 27, 2025.