Machine Learning

# Objectives:

* Build different models
* 3 supervised models:
  1. Logistic regression
  2. Random Forest
  3. Gradient Boosting
* 2 Unsupervised Models:
  1. k-Means Clustering
  2. Isolation Forest
* Compare the performances of the different models

# RoadMap

## Week 1: Data Prep & Proxy Target

**Day 1: Data Ingestion & Initial Profiling**

* **Load the CSV** into a DataFrame and confirm its shape and column names.
* **Preview** the first 10–20 rows to understand typical entries.
* **Run** df.info() to check for nulls, data types, and memory usage.
* **Note anomalies** (e.g., odd strings in numeric columns, unexpected zeros).
* **Jot down** any questions/concerns to address later (e.g., “Why is debt\_ratio sometimes 0?”).

### Day 2: Defining & Generating Your Proxy Target

* **Review** candidate features (payment\_delinquency\_count, bnpl\_debt\_ratio, over\_indebtedness\_flag, etc.).
* **Select thresholds** (e.g., ≥3 delinquencies, debt\_ratio >1.5) and any additional flags.
* **Write a small function** that applies your rule and returns 0/1.
* **Apply** it row-wise (or via vectorized logic) to create default\_flag.
* **Inspect** the resulting class balance (value counts, percentage of 1s vs. 0s).
* **Adjust thresholds** if you end up with an extremely imbalanced split (e.g., fewer than 5% defaults).

### Day 3: Feature Encoding & Scaling

**Goals:** Prepare raw features so they play nicely with ML algorithms.

* **Drop** CustomerID (no predictive power).
* **Identify** which features need scaling (all continuous/numerical) vs. which are already 0/1.
* **Set up a StandardScaler** (or MinMaxScaler) for numeric columns.
* **Build an sklearn ColumnTransformer**:
  + Passthrough for binary flags.
  + Scaler for numeric.
* **Test-transform** a handful of rows to confirm ranges look sensible (mean≈0, std≈1 for scaled).
* **Save** your transformer pipeline (e.g., via joblib) for reuse.

### Day 4: Exploratory Data Analysis (EDA)

**Goals:** Understand distributions, relationships, and potential pitfalls.

* **Plot histograms** (or KDEs) for each numeric feature—watch for skew.
* **Boxplots or violin plots** of key risk metrics, split by your new default\_flag.
* **Correlation matrix** (heatmap) to spot collinearity and strong predictors.
* **Pairwise scatterplots** for top 3–4 correlated pairs (e.g., debt\_ratio vs. stress\_score).
* **Document** any surprises (e.g., features highly skewed, redundant columns).

### Day 5: Feature Engineering

**Goals:** Create new variables that capture interactions or domain insights.

* **Brainstorm** 2–3 composites or interactions (e.g., stress\_score × debt\_ratio, or delinquencies per loan).
* **Implement** these transforms in code (add new columns).
* **Re-run histograms** or boxplots on engineered features to check for informativeness.
* **Check correlation** of new features with default\_flag (e.g., via point biserial correlation).
* **Decide** which engineered features to keep in your pipeline.

### Day 6: Train/Validation/Test Split

**Goals:** Lay the foundation for unbiased model evaluation.

* **Decide split ratios** (e.g., 60% train, 20% validation, 20% test).
* **Use train\_test\_split** twice (train vs. temp, then val vs. test) with a fixed random\_state.
* **Verify** each set has a similar default rate (stratify on default\_flag).
* **Persist** the splits (e.g., save DataFrames or index lists) so they can be reused exactly.

### Day 7: Pipeline Finalization & Documentation

**Goals:** Turn all preprocessing into a reusable, version-controlled artifact.

* **Wrap** your ColumnTransformer and any custom feature-engineering steps into a single Pipeline.
* **Test** the pipeline end-to-end: fit on train, transform on val/test, ensure no errors.
* **Save** the final pipeline object (scaler + feature engineering) with joblib or pickle.
* **Write up** a short methodology note:
  + Data cleaning steps taken
  + How your proxy target was defined
  + Feature transformations applied
* **Commit** code and notes into your Git repo under a week1/ folder.

## Week 2 Supervised Baselines & Tuning