OVERVIEW

Predicting Loan Default Risk using Medallion Data Pipeline

CS6111 ; Assignment 1
Lim Kai Zhuo ; kz.lim.2023@mitb.smu.edu.sg

1 **Objective:** Build a **data pipeline** to train, tune, evaluate, and deploy a model to **estimate customer default probability** based on financial, attributes, and behaviors (FAB).
Given customer FAB before loan given: to **predict and identify high-risk borrowers** to enable loan rejection, proactive intervention, and credit optimization.

2 **Approach**: Implement a full Medallion architecture (Bronze → Silver → Gold) in PySpark in a Docker environment.

FEATURES

Dataset Overview:
Financial – income, loans, utilization ratios (Financial health metrics)
Attributes – age, occupation, demographics (Customer characteristics)
Behaviors – Clickstream events (fe_1–fe_{20}) (Customer behaviors)

Bronze Overview: Extract raw data from backend data source

Silver Overview : Data cleaning of raw Bronze data

Numerical: Outliers handling, null/ placeholder coercion, string to float/int cleaning
Categorical: Regular expression cleaning, binning, binary/indicator variables (e.g. Type of loan split into 1 column for 0/1 for each loan type)
Imputation: Missing values fill with mean of group (e.g. NA salary filled with avg. of job)

Gold Overview : Feature Engineering to create new features that are either simpler, new insights, or introduces non-linearities (esp. useful for linear models)

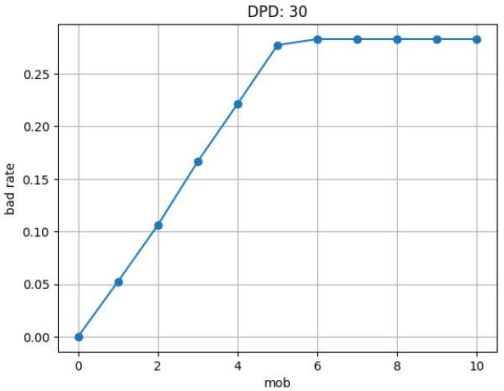
Ratio Metrics: Taking ratios of variables of absolute monetary values into new scale-independent financial ratios. E.g. Debt-to-income, Dept-per-loan, EMI-to-income
Non-linear Transformations: Log, squared, cubed
Threshold / Binning: Simplify continuous variables into bins or ordinal categories

Environment & Operations: Running in a docker container. Entire data pipeline is in a ***single main.py file*** that can be scheduled to run automatically using airflow.

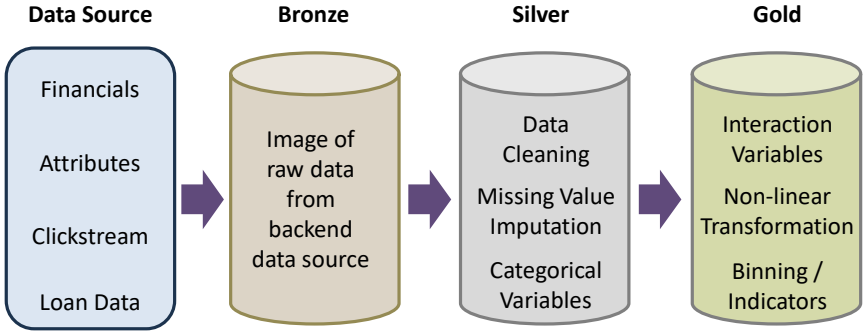
3 **Conclusion**: Medallion architecture setup to provide a clean dataset with feature engineering to train, tune, evaluate, and deploy predictive model.
Next Action: Train various models (linear regression / XGB), perform feature selection, hyperparameter tuning, to select best model to deploy

LABELS

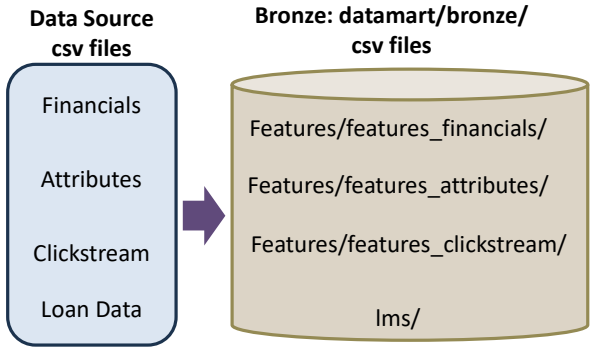
Data Source for Label: Installment date, overdue amount
Labelling of Customer Default: Rather than waiting for customer to default, use days past due to estimate
Days Past Due(DPD) Calculation: Each snapshot ➔ If customer has overdue ➔ Subtract snapshot date from first overdue date
Default Labelling: If DPD > 30 ➔ Label as default
Threshold DPD of 30: Selected as threshold as the bad rate plateaus afterwards



Preventing Data Leakage: To prevent data leakage, during training, use X(t-6),y(t) as features and labels. By joining features from a snapshot date 6 months prior to the labels snapshot date, this ensure there is no possible temporal data leakage. Imagine as using all possible information at snapshot t-6 months to predict customer default at snapshot t months



Bronze Layer | Predicting Loan Default Risk using Medallion Data Pipeline



1 **Objective: Bronze Layer – Data Ingestion and Standardization:**

- Raw data ingestion to consolidate source files for downstream cleaning (silver) and feature engineering (gold).
- Captures data exactly as received, unmodified, and traceable
- Ensures full auditability and reproducibility

2 **Method: Data Source (csv files) ➔ Bronze Datamart (csv files)**

Raw data ingestion to consolidate source files from data source to bronze datamart
Append vs Overwrite table

Append Table (clickstream): New data appended as new rows to the table

Load source file *spark.read.csv(…)* ➡ **Filter by snapshot date** *.filter(col('snapshot_date') == snapshot_date)* ➡ **Save to Bronze Datamart** *spark.toPandas.to_csv(…)*

Overwrite Table (financials & attributes): If customer exists, overwrite existing row

Load source file *spark.read.csv(…)* ➡ **Filter by snapshot date** *.filter(col('snapshot_date') <= snapshot_date)* ➡ **Save to Bronze Datamart** *spark.toPandas.to_csv(…)*

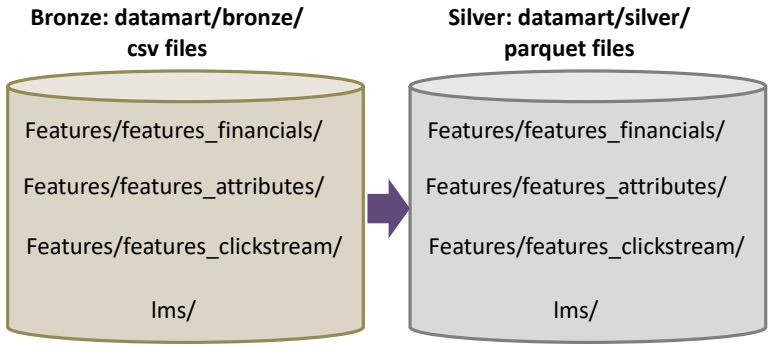
3 **Key Considerations:**

- Partitioning by snapshot date for efficient time-based queries
- Standardization of datamart into separate tables (folders) for the 3 types of features and 1 label table
- Append vs Overwrite table.
  - Since overwrite table will have 1 row per customer_ID only, when filtering by snapshot date, we must keep the previous entries with older snapshot_date to ensure all existing customer data are captured
  - For append table, each snapshot adds new roles for existing customers in the same snapshot_date, hence we can filter by snapshot_date exactly

Silver Layer — Predicting Loan Default Risk using Medallion Data Pipeline

1 Objective: Silver Layer – Data Cleaning:

- Transforms raw Bronze data into clean, consistent, and analysis-ready datasets
- Remove noise and inconsistencies from raw inputs.
- Ensure column-level data validity and completeness.
- Create one cleaned dataset per source type (financial, clickstream, attributes).



2 Method: Bronze Datamart (csv files) ➔ Silver Datamart (parquet files). 1 Folder per Table

Schema Enforcement

Enforce correct datatypes for each columns
(e.g. float, int, str, date)

Null / Placeholder / Invalid Characters

- Strip leading and trailing unwanted characters
- Replace default values and missing with standardized placeholder values
- String to numeric

Numerical

- Outlier floor and ceil to bounded values (e.g. age)
- String to value (e.g. credit history age Years and Months ➔ months as int)

Categorical

- Binary/Nominal variables (e.g. spend level low/med/high ➔ 0/1/2)
- Explode groups into individual indicator variable (loan type can have 1 or more groups in the same string ➔ convert to 1 column per loan type with 0 or 1 value)

Imputations

- Missing values impute with default 0 values
- OR impute with average by group (e.g. missing salary impute with occupation mean)

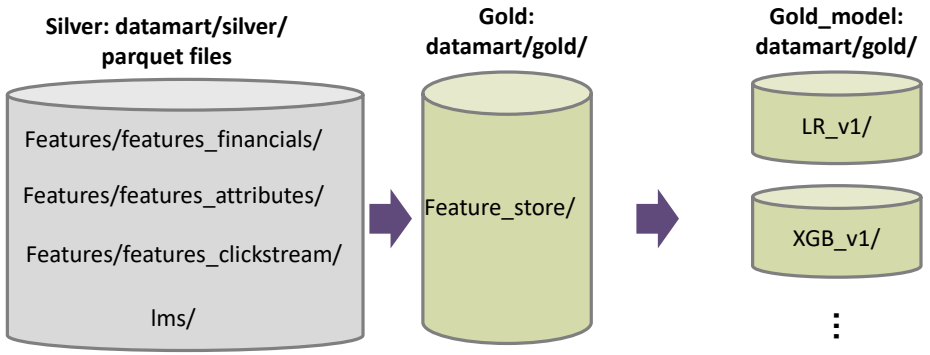
3 Key Considerations:

- Validity checks: regular expression and trimming to handle incorrect input values
- Missing values and imputations: proper default values handling and imputation to ensure no missing data
- Outlier treatment: ensure no extreme values
- Categorical variables: proper handling of indicator/binary variable to ensure downstream models can use such data (e.g. linear regression must have 1 hot encoding for nominal categorical variables rather than integer values)

Gold Layer — Predicting Loan Default Risk using Medallion Data Pipeline

1 Objective: Gold Layer – Feature Engineering and Model Preparation:

- Gold layer combines all Silver tables into a unified, model-ready feature store
- Merge financial, behavioral, and demographic features for each customer
- Engineer features to enable better model training and performance
- Store final dataset in a consistent schema for model training and inference.



2 Method: Silver Datamart (parquet files) ➔ Gold Datamart (parquet files)
Firstly, join all features in silver layer by customer and snapshot date into 1 table. Then perform feature engineering

Ratio Metrics

- Taking ratios of variables
- Financial meaningful ratios to help prediction
- Introduces non-linearity, useful for linear models and XGB
- E.g. Debt-to-income, Dept-per-loan

Threshold / Binning

- Simplify continuous variables into small number of bins (e.g. age bin to 0,1,2)
- Add threshold to flag out (indicator) based on financial knowledge. E.g. Debt to income ratio exceed certain threshold value

Non-linear Transformations

- Log scale: helps with values with long tail distribution (e.g. income, debt)
- Square/cube: helps emphasis risk is non-linearly worse for higher debt values
- Introduces non-linearity (similar to ratio metrics)

Model-Specific Gold Tables

- Prepare features by selecting subset for specific model training
- E.g. XGB can use larger number of features and even categorical features (without one-hot encoding)
- But linear regression need to select and prepare features that are compatible with linear models
- E.g. remove perfect multi-collinearity columns (or highly correlated columns)
- Create 1 gold_model table for each model
- Additionally, after training an intital model with a lot of features, based on features importance (e.g. SHAP or high L1 regularization term), we can identify a subset of features to use for the next V2 iteration of the model and so on

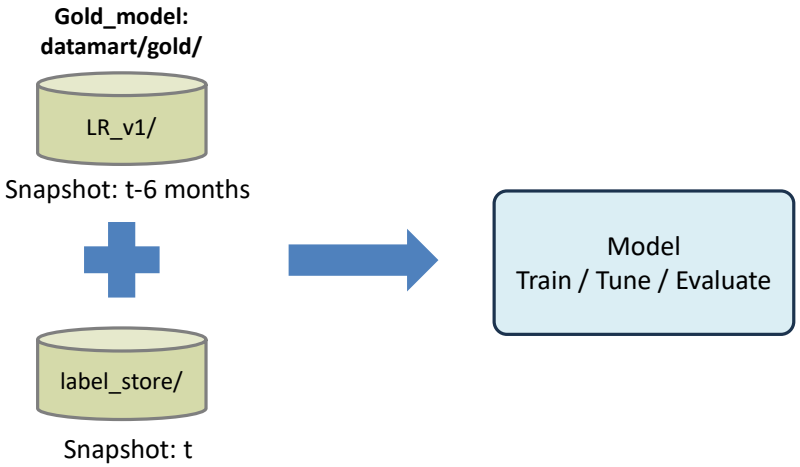
3 Key Considerations:

- Joining of features from various silver tables into single gold table
- Engineer features to simplify, enhance, and introduce non-linearity ➔ At this stage, more is better, can use feature importance to subset later on
- Prepare model-specific gold tables as each model has different needs.

Model Training | Predicting Loan Default Risk using Medallion Data Pipeline

1 Objective: Combine Features and Labels for Model Training

Load gold model feature table and gold label table for model training



2 Method: Gold Datamart (parquet files) – Features (t-6), Labels (t)

- Select the desired gold model feature table based on the model being trained
- Perform join on gold label table
- To prevent temporal data leakage, join on feature tables with a snapshot date 6 months prior.
- Since the feature table contains all information up to time t-6 and this is used to predict the label table with information at time t ➔ this ensures there is no possible temporal data leakage from features to labels

3 Key Considerations:

- Selecting the correct gold model feature table for the desired model
- Prevent data leakage by joining features table from 6 months before labels table