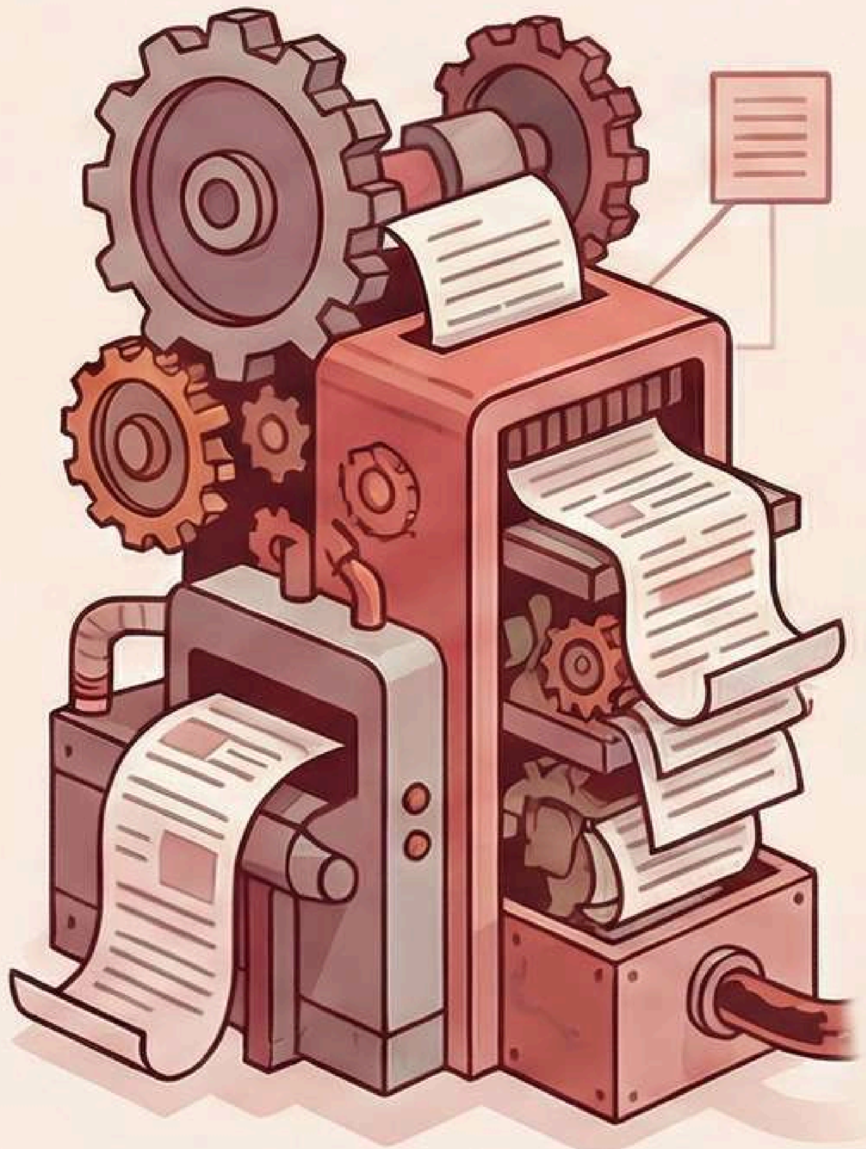
The AI Advantage in Credit Risk Prediction

The Problem with Traditional Credit Risk Rules



Traditional Systems Struggle to Scale

They cannot effectively handle the complexity and volume of modern loan applications.

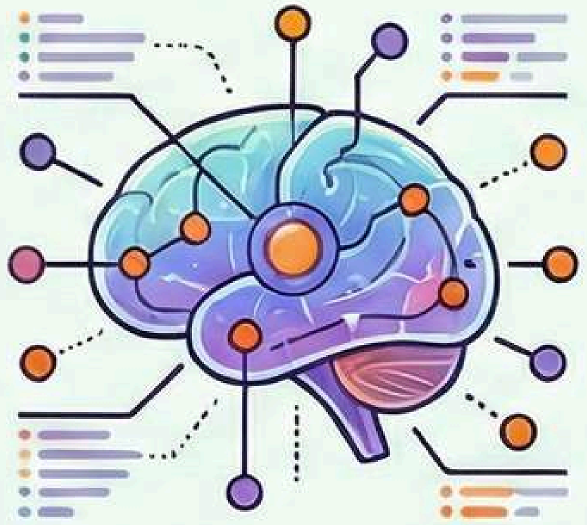
Fixed Rules Cannot Adapt to Reality

They miss complex relationships in data and fail to adjust to new borrower behaviors.

Decisions Become Inconsistent & Overly Conservative

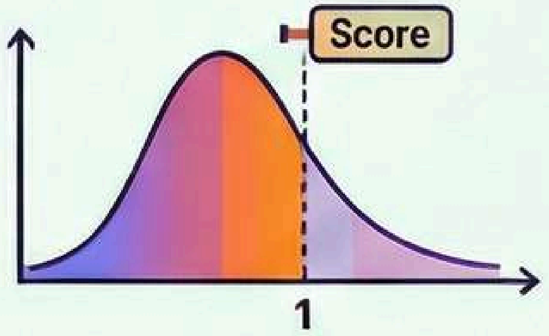
This slow, unreliable process hinders effective risk-aware lending.

The Machine Learning Solution & Business Impact



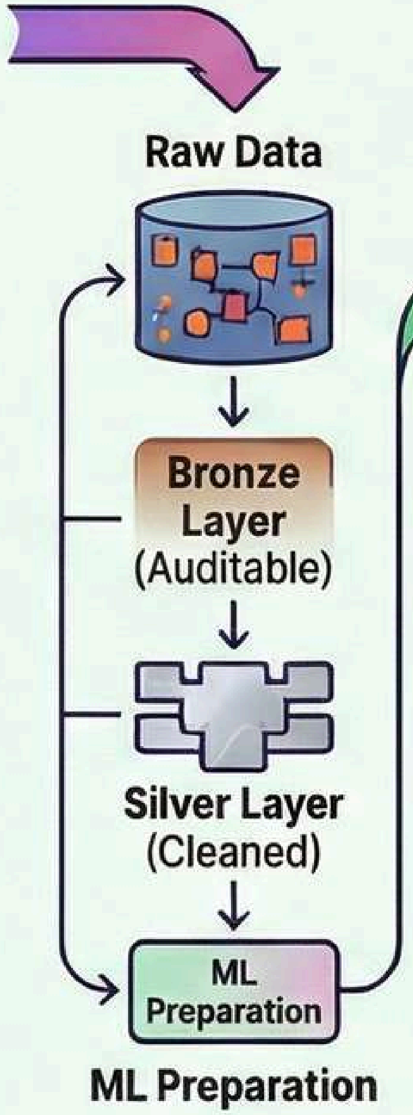
AI Learns Hidden Patterns from Past Data

It combines borrower and loan attributes to predict future loan outcomes.

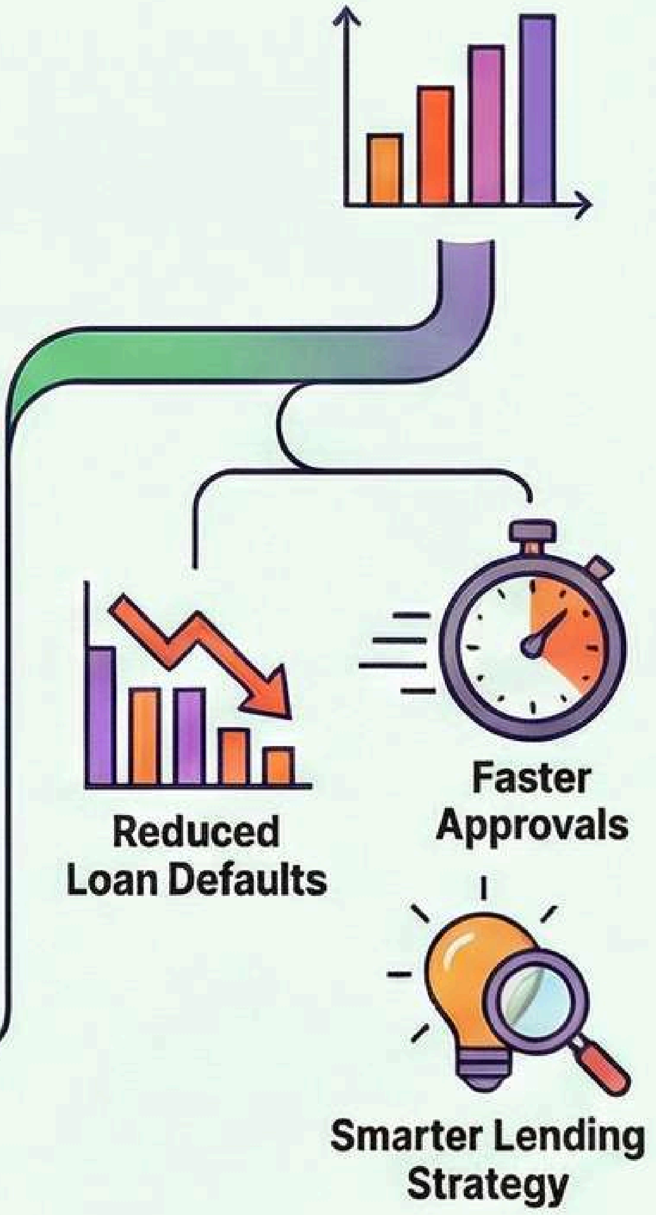


Delivers Probability-Based Risk Scores

ML replaces simple yes/no decisions with a nuanced score indicating default likelihood.



Data is Refined in a 4-Step Process

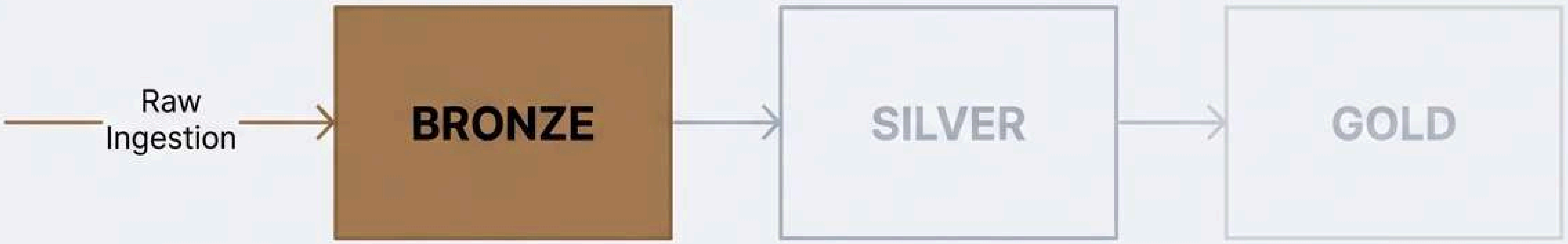


Drives Major Business Improvements

Results in reduced loan defaults, faster approvals, and a smarter lending strategy.

Constructing the Bronze Layer: Raw Data Ingestion

A foundational blueprint for preserving data fidelity in the Lakehouse Architecture




```
OBJECTIVE: Ingest credit_risk_raw_data
MANDATE: Maintain original state for traceability
STACK: Databricks / Spark / Delta Lake
```

The Bronze Mandate: Immutable & Traceable.

1. **No Modifications:** Load raw dataset without modifying content.
2. **Preservation:** Keep original data for traceability and debugging.
3. **Zero Cleaning:** No transformations at this stage.

The purpose of the Bronze layer is to load the raw dataset into the system without modifying its content. This layer preserves the original data for traceability, debugging, and reproducibility.

Architectural Note: If we clean data here, we lose the ability to debug issues in the source system later. Bronze is our source of truth.

The Dataset: Credit Risk Profile.

ENTITY: THE BORROWER

ENTITY: THE LOAN

person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_g
22	59000	RENT	123	PERSONAL	D

SOURCE PATH: /Volumes/workspace/finance/credit_risk_raw_data
DOMAIN: Financial Risk Analysis

Step 1: Ingesting the Raw CSV

```
df = spark.read.csv(
    '/Volumes/workspace/finance/credit_risk_raw_data',
    header=True,
    inferSchema=True
)
```

Preserves Semantics: Instructs Spark to use the first row of the CSV as column names.

Automated Typing: A critical **Bronze decision**. Spark scans the file to detect Integers vs. Strings automatically, adapting to the raw structure.

INDIAN DATA CLUB

Step 2: Validating the DataFrame Structure

loan_grade	loan_amnt	loan_int_rate	loan_percent_income	cb_person_default_on_file	person_age	person_income	loan_intent	loan_status	cb_person_cred_hist_lengtł
D	35000	16.02	0.59	Y	57	$3000000	loan-online service	Completed	22
B	1000	11.14	0.1	N	58	$250000	inanving	Completed	16
C	5500	12.87	0.57	N	56	$100000	loan intent	Completed	25
C	35000	15.23	0.53	N	63	$250000	inanving	Completed	39
A	2500	14.27	0.55	Y	50	$700000	loan intent	Completed	84
B	35000	7.14	0.25	N	58	$750000	lean-soonning service	Completed	34
A	35000	12.42	0.45	N	57	$750000	loan intent	Completed	36
D	1600	11.11	0.44	N	53	$700000	loan-soonning service	Completed	37
B	35000	8.9	0.42	N	60	$250000	loan-soonning service	Completed	17
A	35000	14.74	0.16	N	63	$700000	loan-soonning service	Completed	32
A	4500	10.37	0.41	N	58	$1000000	loan-soonning service	Completed	29
A	35000	8.63	0.45	N	53	$700000	loan intent	Completed	26
A	35000	7.9	0.37	N	57	$450000	loan intent	Completed	26
E	35000	18.39	0.32	N	57	$700000	loan intent	Completed	24

10,000+ rows | Truncated data | 20.54s runtime

Observation Log

- **Volume:** 10,000+ rows loaded.
- **Data Texture:** 'cb_person_default_on_file' contains binary flags (Y/N).
- **Risk Metrics:** "loan_percent_income" is a calculated ratio crucial for risk models.
- **Categorization:** 'loan_grade' provides categorical risk ratings.

Deciphering the Schema.

Translation Matrix

`person_emp_length`	**Employment Stability (Years)**
`cb_person_cred_hist_length`	**Credit History Depth**
`loan_intent`	**Borrowing Reason**
`loan_status`	**Target Variable (Default/Repaid)**

Understanding these definitions is required for the Silver Layer transformations that will follow.

Step 3: Schema Inspection & Type Verification.

Schema Verification

col_name	data_type
person_age	int ✅
person_income	int ✅
person_home_ownership	string
person_emp_length	double
loan_int_rate	double ✅
loan_grade	string

Technical Check:

Spark successfully recognized the mathematical nature of the financial columns (Integers and Doubles) versus the categorical columns (Strings), validating the 'inferSchema' parameter.

The Urge to 'Fix'.

```
2918, 'John Doe', null, 'CA', 'mEdical', 150000, '2023-10-27T14:30:00Z'
```

The Scenario:
We spot nulls, outliers, or inconsistent casing.

The Discipline:
In the Bronze layer, we DO NOT touch it.

The Reason:
If we fix a value here, we mask a data quality issue from the upstream provider. The Bronze layer must remain an exact mirror of the source.

Step 4: Committing to the Lakehouse.

Format Upgrade: Converts raw CSV to Delta Lake format, enabling ACID transactions and versioning.

```
df.write.format('delta') \
  .mode('overwrite') \
  .saveAsTable('finance.bronze_credit_risk')
```

Idempotency: Replaces the table entirely on run, preventing duplicates for this batch.

Metastore Registration: makes data queryable via SQL as 'finance.bronze_credit_risk'.

Final Validation: The Bronze Table

```
SELECT * FROM bronze_credit_risk LIMIT 10
```

STATUS: PERSISTED.

	person_age	person_income	person_home_ownership
1	22	50000	RENT
2	21	6000	OWN
3	25	8600	MORTGAGE
4	23	65500	RENT
5	24	54400	RENT
6	21	9800	OWN
7	26	77100	RENT
8	24	78958	RENT
9	24	83000	RENT
10	21	10000	OWN

- **Result:** Structured, queryable table ‘`finance.bronze_credit_risk`’.
- **Schema:** Enforced.
- **Performance:** ~3 seconds execution.

Helvetica Now Display

Blueprint Recap: The Bronze Pipeline

JetBrains Mono

SOURCE:
Raw CSV

INGEST:
Spark Read +
Schema Inference

VALIDATE:
Visual
Inspection

PERSIST:
Write to Delta
(Overwrite)

JetBrains Mono

RESULT:
finance.bronze_credit_risk

Ready for Refinement.

We have successfully established the Bronze Layer.

Current State: Raw, immutable, historic.

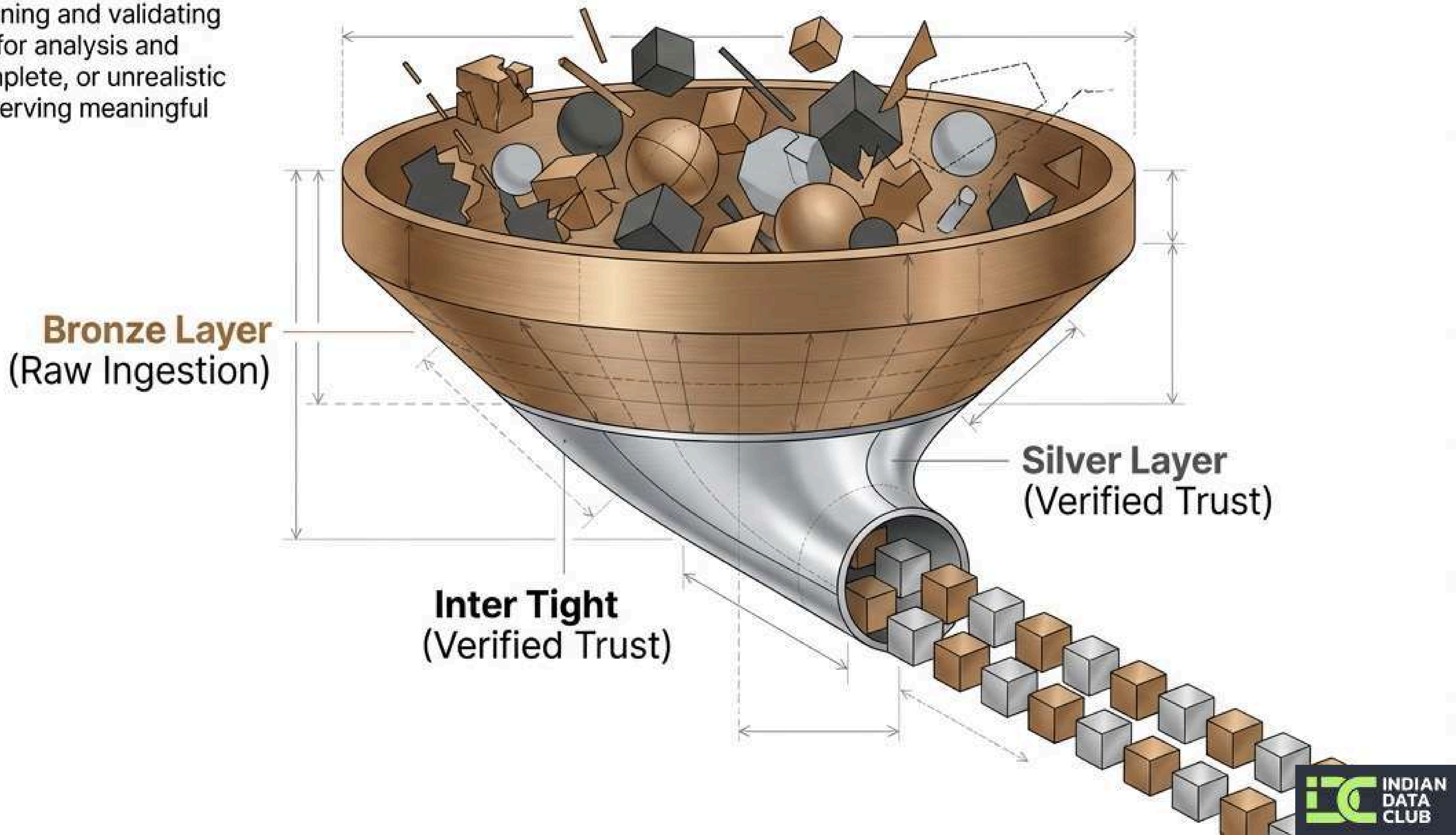
NEXT OBJECTIVE: THE SILVER LAYER

- Cleanse null values
- Standardize categories
- Enforce business logic

The foundation is poured. Now we build the structure.

Silver Layer Construction: Engineering High-Fidelity Credit Risk Data

The Silver layer focuses on cleaning and validating raw data to ensure it is reliable for analysis and machine learning. Invalid, incomplete, or unrealistic records are removed while preserving meaningful business information.



The Raw Material: Assessing the Bronze Layer.

```
df_bronze = spark.table("finance.bronze_credit_risk")
display(df_bronze)
```

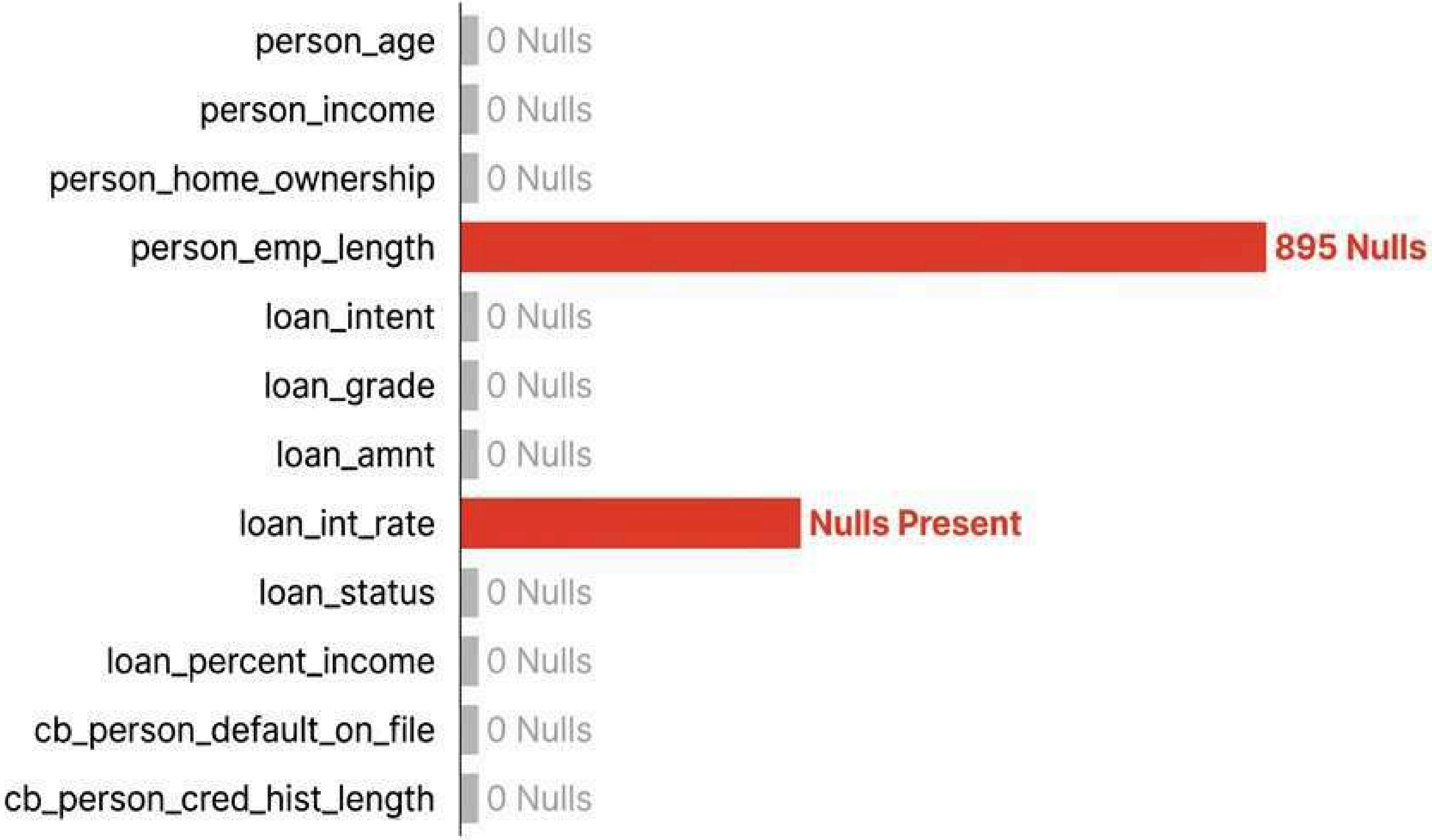
Total Rows In: 32,581

```
root
 |-- person_age: integer (nullable = true)
 |-- person_income: integer (nullable = true)
 |-- person_home_ownership: string (nullable = true)
 |-- person_emp_length: double (nullable = true)
 |-- loan_intent: string (nullable = true)
 |-- loan_grade: string (nullable = true)
 |-- loan_amnt: integer (nullable = true)
 |-- loan_int_rate: double (nullable = true)
 |-- loan_status: integer (nullable = true)
 |-- loan_percent_income: double (nullable = true)
 |-- cb_person_default_on_file: string (nullable = true)
 |-- cb_person_cred_hist_length: integer (nullable = true)
```

RISK: Critical fields are flagged as nullable. Directly feeding this schema into a Machine Learning model will cause training failures due to missing vectors.

Diagnostic: Identifying the Null Hypothesis.

Null Value Counts by Field.



Machine learning models are sensitive to missing or **incorrect data**. We must

remove records with missing values in critical fields such as income, loan amount, and interest rate to avoid misleading predictions.

Quality Gate 1: Structural Integrity

Technical Action

```
rows_after_drop = df_bronze.dropna(subset=[
    "person_income",
    "loan_amnt",
    "loan_int_rate",
    "loan_status"
])
```

Target Variable (Must exist for Supervised Learning)

Impact

The Drop

32,581

(Starting Count)

	32,581	Charcoal
-	**3,116**	Dropped Rows
=	**29,465**	Remaining

Quality Gate 2: Business-Driven Filtering

Income Validation

```
filter(col("person_income") > 0)
```

Exclude zero or negative income.

Pass

Loan Amount Validation

```
filter(col("loan_amnt") > 0)
```

Exclude invalid loan requests.

Pass

"Certain values are not realistic in a financial context. A loan applicant with zero or negative income cannot reasonably repay a loan."

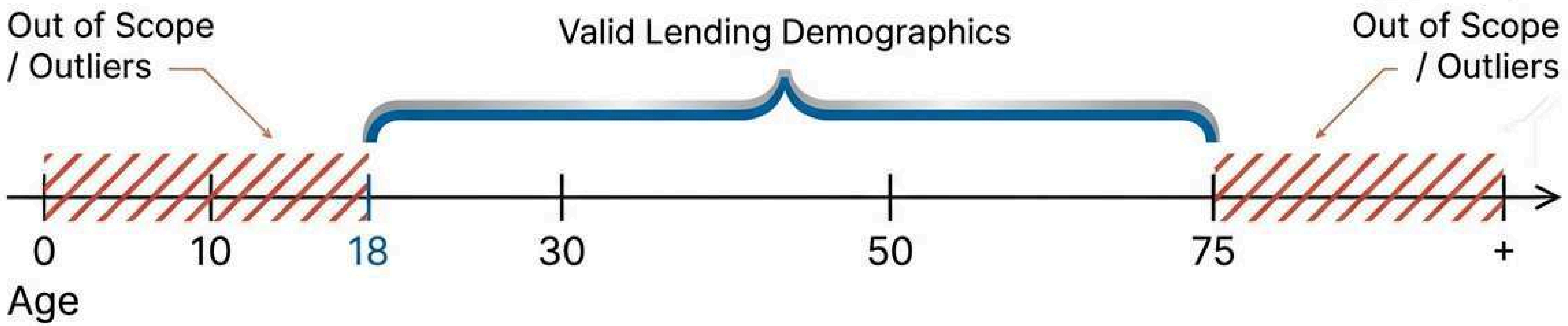
Refining the Feature Set

Selected Columns
(The Survivors)

- person_age
- person_income
- person_home_ownership
- person_emp_length
- loan_intent
- loan_grade
- loan_amnt
- loan_int_rate
- loan_percent_income
- loan_status

person_age	person_income	loan_intent	loan_grade	loan_amnt
22	59000	PERSONAL	D	35000
21	9600	EDUCATION	B	1000

Quality Gate 3: Demographic Consistency




```
df_silver = df_silver.filter(
        (col("person_age") >= 18) & (col("person_age") <= 75) )
```

Secondary Cleanup: A final dropna() ensures feature vector completeness.

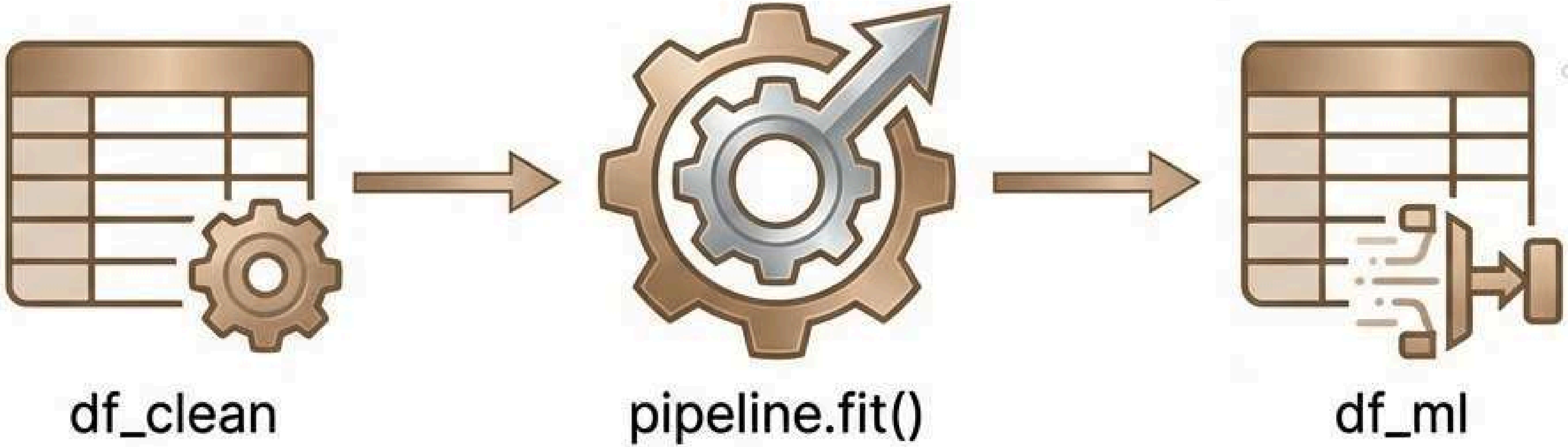
The Silver Standard: Validation & Persistence.

Final Row Count: 29,457

```
df_silver.write.format("delta") \
    .mode("overwrite") \
    .saveAsTable("finance.silver_credit_risk")
```

After cleaning, the dataset contains only valid and consistent records.
This cleaned dataset is now ready for feature engineering.

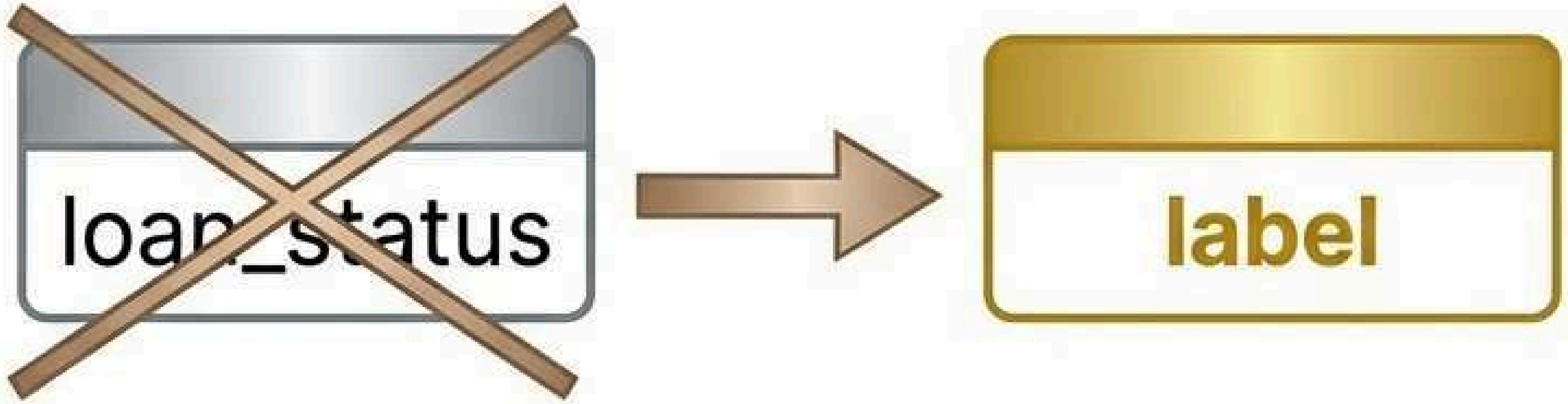
Initializing the ML Pipeline.




```
from pyspark.ml import Pipeline
stages = []
pipeline = Pipeline(stages=stages)
df_ml = pipeline.fit(df_clean).transform(df_clean)
```

We wrap the data flow in a Spark ML Pipeline object, ensuring that all future transformations are reproducible for inference.

Final Transformation: The 'Gold' Handoff.




```
df_ml = df_ml.withColumnRenamed("loan_status", "label")
```

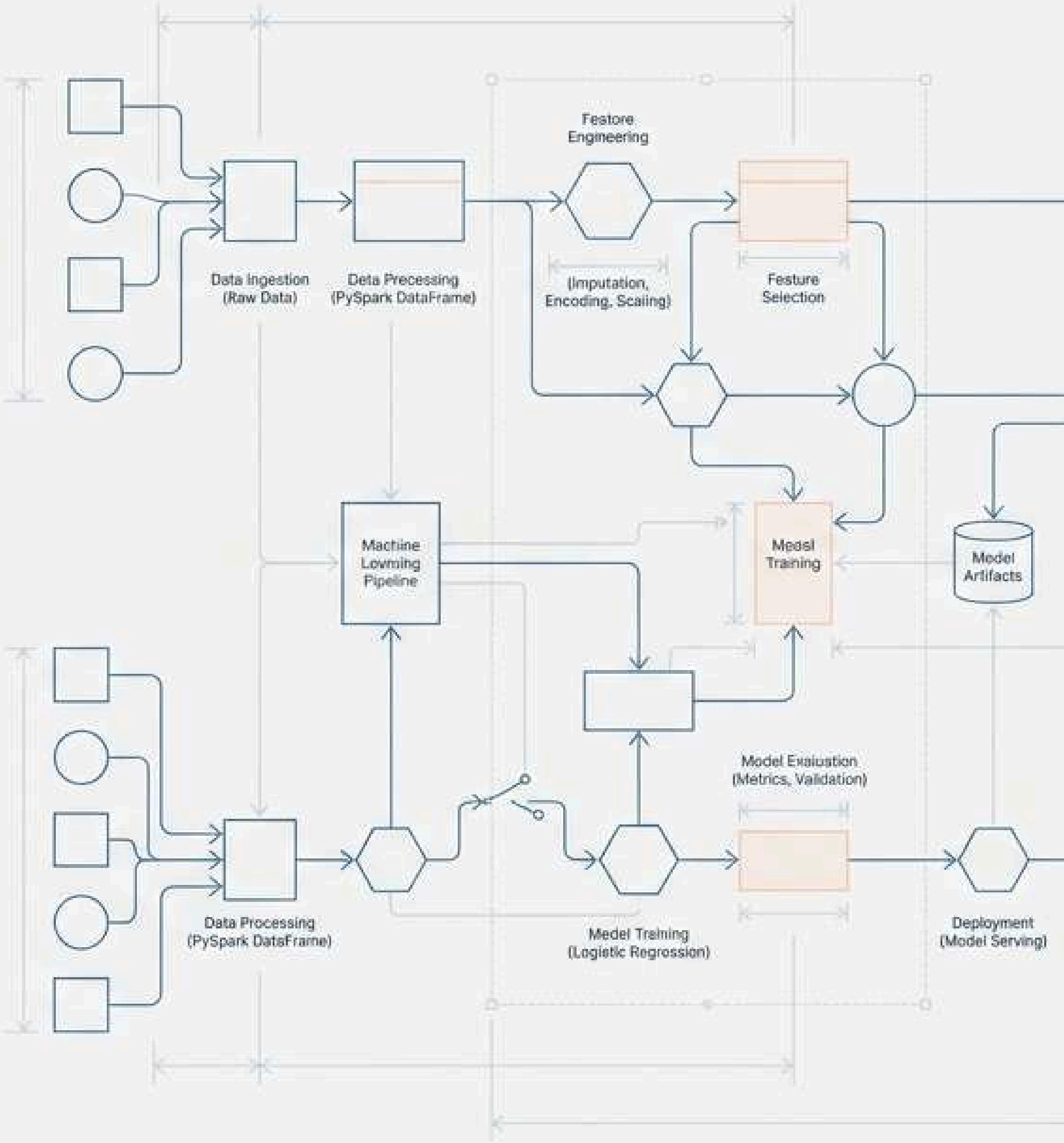
Persistence: ML Ready.

```
.saveAsTable("finance.ml_ready_credit_risk")
```

The journey is complete. We started with a messy CSV dump and ended with a structured, **validated**, Delta-backed **feature store** ready for model training.

Building a Credit Risk Machine Learning Pipeline

A Step-by-Step Guide to Feature Engineering and Logistic Regression in PySpark



DATABRICKS IMPLEMENTATION GUIDE

INDIAN DATA CLUB

From Silver Layer Data to Predictive Insight

Source: finance.silver_credit_risk

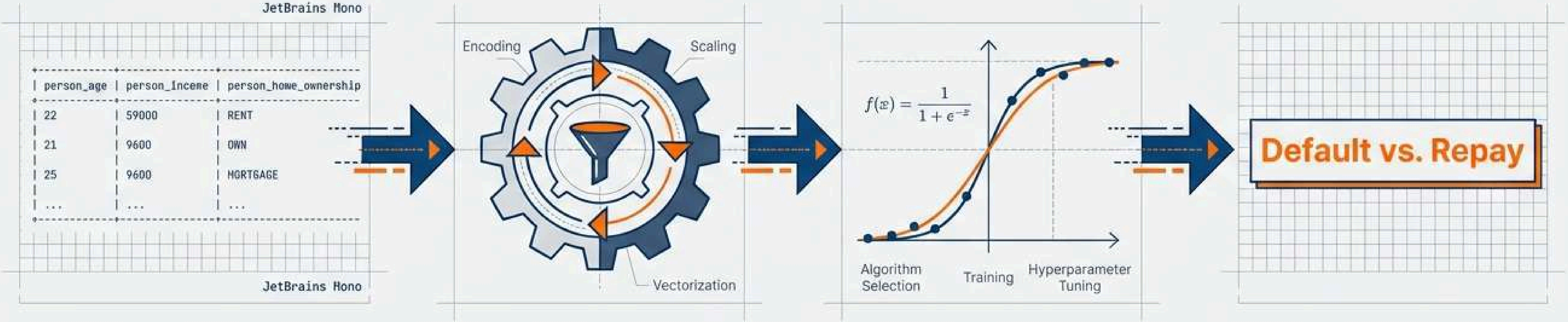
Target: loan_status

Input: Silver Table

Refinery: Feature Engineering

Engine: Model Training

Output: Prediction



The Blueprint: Defining Input Signals

Categorical Features (Strings)

- person_home_ownership
- loan_intent
- loan_grade

Numeric Features (Integers/Floats)

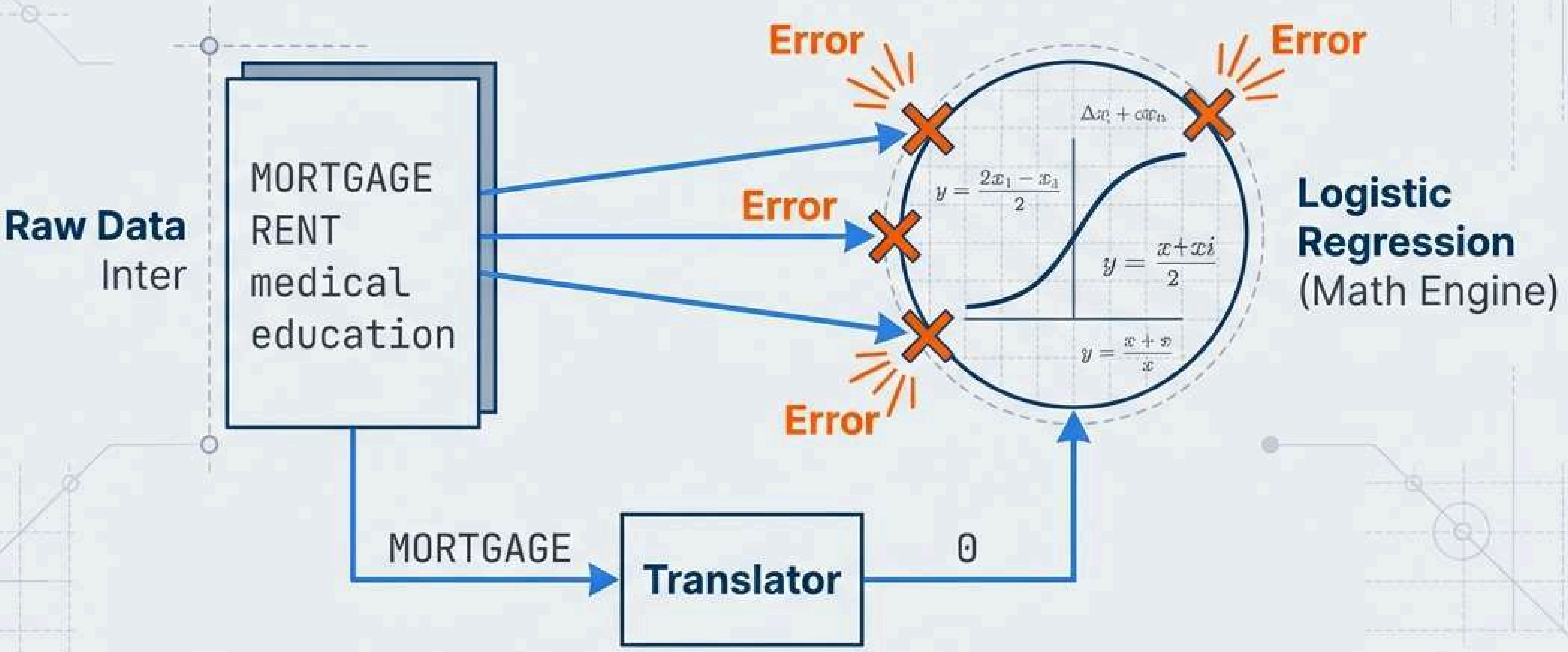
- person_age
- person_income
- person_emp_length
- loan_amnt
- loan_int_rate
- loan_percent_income

```
categorical_cols = [
    "person_home_ownership",
    "loan_intent",
    "loan_grade"
]
```

```
numeric_cols = [
    "person_age",           "loan_amnt",
    "person_income",        "loan_int_rate",
    "person_emp_length",    "loan_percent_income"
]
```

The Translation Challenge

Why we cannot feed text directly into the model

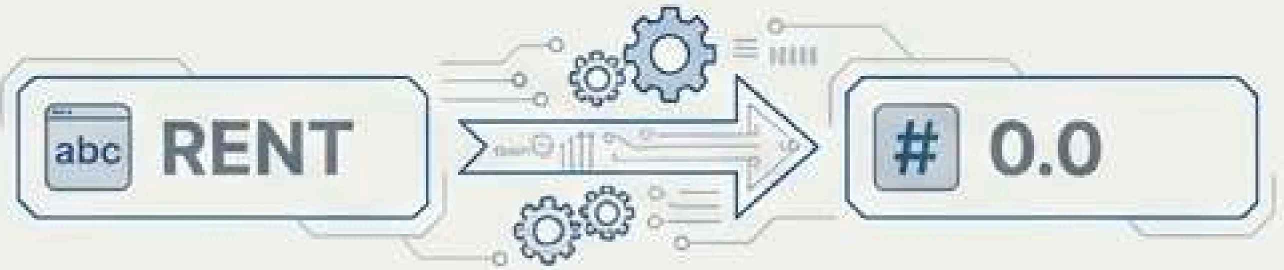


Algebra requires numerical inputs. Strings must be encoded.

Indexing Categorical Variables

Technique: StringIndexer

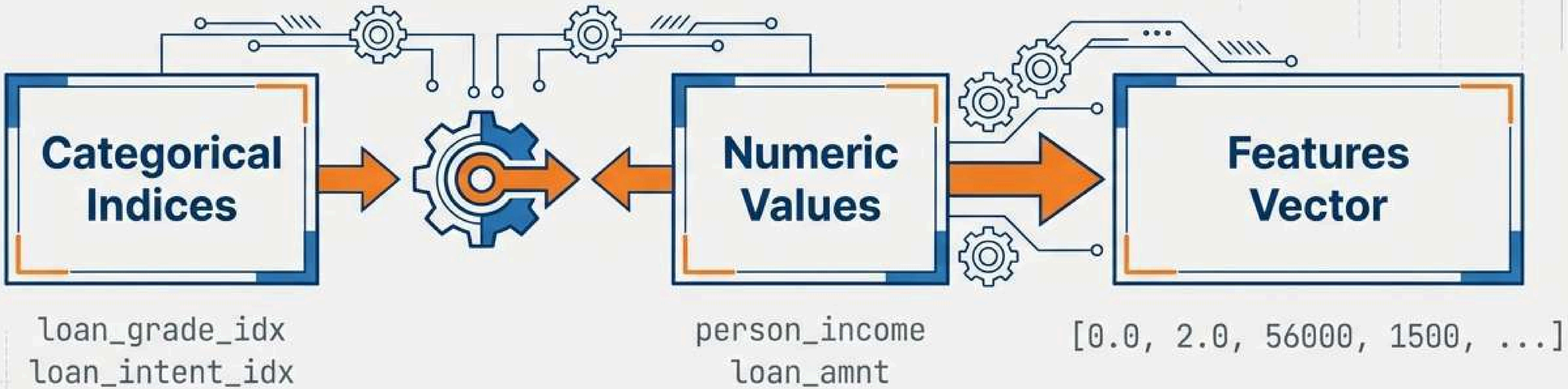
- Mechanism: Assigns a unique numerical index to each category based on frequency.
  - **Most Frequent = 0**
  - **Next Frequent = 1**
- Handling Invalid Data: ‘keep’ (prevents crashes on unseen labels).




```
indexers = [
    StringIndexer(
        inputCol=c,
        outputCol=f"{c}_idx",
        handleInvalid="keep"
    )
    for c in categorical_cols
]
```

Creating the Feature Vector

Technique: VectorAssembler




```
assembler = VectorAssembler(inputCols=[f"{c}_idx" for c in categorical_cols] + numeric_cols,
                            outputCol="features")
```

Finalizing the ML-Ready Dataset

1. **Standardize Target**: Rename '**loan_status**' to '**label**'.
2. **Select Artifacts**: Keep only 'features' (vector) and 'label' (int).
3. **Persist:** Save as Delta Table for reproducibility.

```
df_ml = df_ml.withColumnRenamed("loan_status",
"label")
```

```
df_ml = assembler.transform(df_ml)
  .select("features", "label")
  .write.format("delta").mode("overwrite")
  .saveAsTable("finance.ml_ready_credit_risk")
```

finance.ml_ready_credit_risk

Training Strategy: The Split

Helvetica Now Display
(RGB 0, 51, 102)

Slate Grey
(RGB 64, 78, 97)

Random Seed = 42
(Reproducible)
Inter

Code JetBrains Mono
(RGB ℞GB, 51, 102)

```
train_df, test_df = df_ml.randomSplit([0.7, 0.3], seed=42)
```

Configuring the Logistic Regression Model

Model Configuration

Algorithm:	Logistic Regression	Binary classification model based on sigmoid function.
Input Col:	"features"	Column containing the vector of features for training.
Target Col:	"label"	Column containing the true labels (0 or 1).
Max Iterations:	20	Maximum number of iterations for optimization.
Reg Param:	0.01	Regularization to prevent overfitting.

Code Initialization & Fitting

```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(
    featuresCol="features",
    labelCol="label",
    maxIter=20,
    regParam=0.01
)

lr_model = lr.fit(train_df)
```

Generating Predictions

	label	probability	prediction
1	0	[0.806…, 0.193…]	0.0
2	0	[0.979…, 0.020…]	0.0
3	0	[0.966…, 0.033…]	0.0
4	0	[0.761…, 0.238…]	0.0
5	0	[0.990…, 0.009…]	0.0

Probability vector shows confidence for [Class 0, Class 1]

Model Evaluation

0.8518

Area Under ROC (AUC)

Interpretation: Strong baseline predictive power. The model effectively discriminates between borrowers likely to repay and those likely to default.

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

evaluator = BinaryClassificationEvaluator(
    labelCol='label',
    metricName='areaUnderROC'
)

auc = evaluator.evaluate(predictions)
auc
```

Under the Hood: Model Coefficients

Inspect the weights driving the risk prediction.

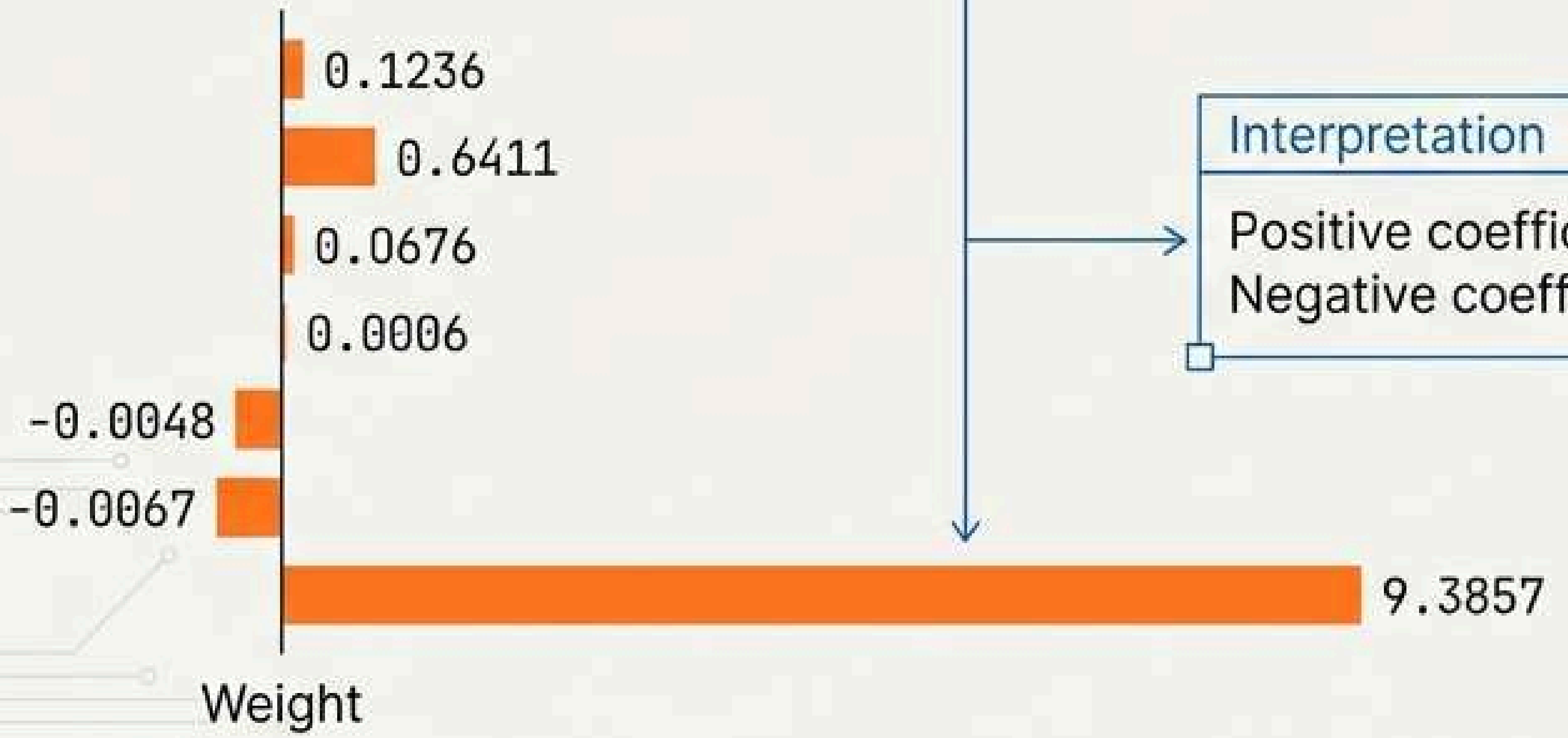
`lr_model.coefficients` — Coefficients (Weights)

```
DenseVector([-0.0048, 0.1236, 0.6411, 0.0006, -0.0,
             -0.0067, -0.0, 0.0676, 9.3857])
```

`lr_model.intercept`

-4.036

Intercept (Bias)



Interpretation

Positive coefficients increase default risk.
Negative coefficients decrease default risk.

Distribution Analysis: Predicted vs. Actual

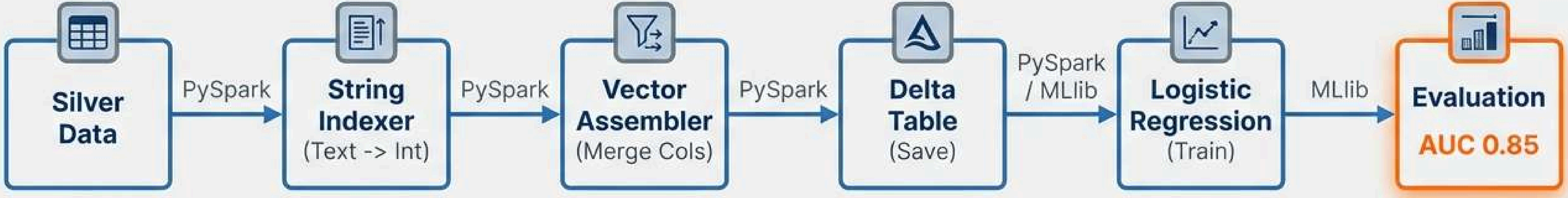
Model Predictions	
0.0 (Repay):	7392
1.0 (Default):	1135

Actual Labels	
0 (Repay):	6706
1 (Default):	1821

Model is conservative.
Under-predicts total defaults.

```
predictions.groupBy("prediction").count().show()
predictions.groupBy("label").count().show()
```

Pipeline Architecture Recap



Conclusion & Next Steps

Summary:

We engineered a robust feature pipeline, transforming raw banking data into a performant predictive model with a **0.85 AUC score**.

Next Steps:

1. Hyperparameter Tuning: Adjust `maxIter` and `regParam` to optimize accuracy.
2. Model Selection: Experiment with Random Forest or Gradient Boosted Trees.
3. Operations: Integrate with MLflow for experiment tracking and deployment.