AI-DRIVEN FRAMEWORK FOR UNDERPRICED FITNESS PRODUCTS

Presented By: Miss. Tungana Bhavya

This Challenge is Sponsored by Databricks, organized by Code Basics and Indian Data Club

14 - Day AI Challenge with two phases

- **Phase 1 - Learning & Sharing: Jan 9 → Jan 22**
- **Phase 2 - Project Phase: Jan 24 → Jan 30**

Problem Statement: Market Pricing Challenge

Problem Context

- Fitness product pricing is **highly inconsistent** across online marketplaces.
- Buyers struggle to identify **fair** and **genuine deals** due to:
  - Wide price variations for similar products
  - Manual price comparison being **time-consuming** and **error-prone**
  - High-value deals hidden among **overpriced listings**

Project Objective

- **Designed** and **implemented** an AI-driven framework to:
  - Predict the **fair market price** of fitness products.
  - Identify and rank **underpriced products** using data-driven valuation.

no_of_ratings	discount_price	actual_price	_ingested_at
3,836	null	₹59	2026-01-30T11:09:38.279+00:...
3,018	₹279	₹699	2026-01-30T11:09:38.279+00:...
4,403	₹489	₹1,049	2026-01-30T11:09:38.279+00:...
8,901	₹376	₹999	2026-01-30T11:09:38.279+00:...
33,078	₹1,049	₹4,090	2026-01-30T11:09:38.279+00:...
165	₹320	₹1,699	2026-01-30T11:09:38.279+00:...
48	₹99	₹999	2026-01-30T11:09:38.279+00:...
5,158	₹449	₹499	2026-01-30T11:09:38.279+00:...
3,755	₹528	₹799	2026-01-30T11:09:38.279+00:...
990	₹99	₹590	2026-01-30T11:09:38.279+00:...

Market Problem & Impact

Example:

- **CGC-60 Bike:** Market Price ₹40,000 vs Predicted Fair Price ₹55,543
- **Price Gap:** ₹15,543 (28% undervalued)

Consequences of Mispricing:

- **For Buyers:** Risk of overpaying for products
- **For Retailers:** Mispriced inventory leading to revenue/margin loss

Goal:

- **AI-Generated Value Score (%)** to identify **underpriced products**, helping both buyers and retailers make data-driven decisions

Medallion Architecture Overview

Key Highlights:

1. **Bronze Layer (Raw Data):**
- Ingest raw data (e.g., from Amazon) with minimal processing.
- Example: bronze_db.amazon_fitness_raw (non-transformed).

2. **Silver Layer (Cleaned Data):**
- Clean, standardize, and transform data for further analysis.
- Example: silver_db.amazon_fitness_clean
  (missing values handled, outliers removed).

3. **Gold Layer (Final Data for Analysis/ML):**
- Ready for analysis and ML predictions.
- Example: gold_db.amazon_fitness_gold
  (final features for modeling).

Environment & Database Setup

Key Highlights:

- **Databases Created in Databricks:**

1. bronze_db – Stores raw ingested tables
2. silver_db – Stores cleaned and standardized tables
3. gold_db – Stores curated, ML-ready tables

- **All layers contain persistent Delta tables** (non-temporary).
- Verified database creation using:
  SHOW DATABASES LIKE '%_db%'

catalog_name	schema_name	schema_owner
workspace	gold_db	tbhavya054@gmail.com
workspace	bronze_db	tbhavya054@gmail.com
workspace	silver_db	tbhavya054@gmail.com

database	tableName	isTemporary
bronze_db	amazon_fitness_raw	false

database	tableName	isTemporary
silver_db	amazon_fitness_clean	false

database	tableName	isTemporary
gold_db	amazon_fitness_gold	false

Raw Amazon Dataset – Overview (Before Filtering)

1. **Total Records:**

```
# Get total count
total_count = raw_df.count()

# Print the total count
print(f"Total Records in Raw Amazon Dataset: {total_count}")

> See performance (1)
```

```
Total Records in Raw Amazon Dataset: 551585
```

2. **Columns in Dataset:**

```
|Column Names  |
+--------------+
|_c0           |
|name          |
|main_category |
|sub_category  |
|image         |
|link          |
|ratings       |
|no_of_ratings |
|discount_price|
|actual_price  |
+--------------+
```

Highlights:

- Shows dataset scale and structure before any filtering or cleaning
- Contains all product categories
- Prepares audience for Silver layer filtering

Category Diversity (Before Filtering)

Unique Categories	
main_category	**sub_category**
appliances	Air Conditioners
appliances	Refrigerators
electronics	Smartphones
electronics	Televisions
fitness	Treadmills
fitness	Exercise Bikes

Highlights:

- Shows diversity of dataset before filtering.
- Fitness category is target for AI pricing analysis.
- Smooth transition to Silver layer – cleaned and filtered fitness data.

Bronze Layer – Raw Fitness Data

1. **Total Records in Bronze Layer:**

```
> See performance (1)
Total Records in Raw Amazon Dataset: 551585
```

2. **Key Columns:**

[‘_c0’, 'name', 'main_category', 'sub_category', 'ratings',
'no_of_ratings', 'actual_price', 'discount_price', '_ingested_at']

```
1  bronze_df.printSchema()
```

```
root
 |-- _c0: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- main_category: string (nullable = true)
 |-- sub_category: string (nullable = true)
 |-- image: string (nullable = true)
 |-- link: string (nullable = true)
 |-- ratings: string (nullable = true)
 |-- no_of_ratings: string (nullable = true)
 |-- discount_price: string (nullable = true)
 |-- actual_price: string (nullable = true)
 |-- _ingested_at: timestamp (nullable = false)
```

Highlights:

- Bronze layer preserves all raw data (source of truth).
- Minimal transformations, only ingestion metadata (_ingested_at).
- Serves as input for Silver layer cleaning and filtering.

Silver Layer – Cleaned Fitness Data

1. **Record Counts & Columns after Cleaning:**

```
silver_df: pyspark.sql.connect.dataframe.DataFrame = [name: string, main_category: string ... 7 more fields]
Records after Deduplication: 12496
Duplicates Removed: 133
Remaining Columns: ['name', 'main_category', 'sub_category', 'ratings', 'no_of_ratings', 'discount_price', 'actual_price', '_ing
ted_at', 'sub_category_standardized']
```

Highlights:

- Deduplicated and filtered to fitness products only.
- Columns standardized and numeric types corrected for ML.
- Dataset ready for feature engineering and AI modeling.

2. **Distinct Standardized Sub-Categories:**

sub_category_standardized
Unknown
Badminton
Cycling
Cardio Equipment
Cricket
Camping & Hiking
Football
Fitness Accessories
Running
Strength Training
Yoga

Silver Layer – Null Handling, Data Cleaning & Correlation Check

Correlation Check: discount_price vs actual_price validated

```
> silver_df_cleaned:  pyspark.sql.connect.dataframe.DataFrame = [name: string, main_category: string ... 7 more fields]
> valid_assets_df:  pyspark.sql.connect.dataframe.DataFrame = [name: string, main_category: string ... 7 more fields]

Validation: Correlation between Discount and Actual Price is 0.0164
Silver Layer SUCCESS: 12496 rows persisted.
```

Highlights:

- Removed non-numeric characters from numeric columns using regexp_replace.
- Converted columns safely to double using try_cast.
- Replaced NULLs with 0 using coalesce.
- Filtered rows where actual_price = 0 for realistic price correlation.
- Cleaned Silver Layer persisted in silver_db.amazon_fitness_clean
- Total rows persisted: **12,496**

Silver Layer – Feature Selection & Dimensionality Reduction

1. **Columns Dropped:**

- **image** → not useful for ML models
- **link** → irrelevant for pricing prediction
- **_c0** → auto-generated index, redundant

2. **Main Category Check:**

- Performed variance check on main_category.
- silver_df_cleaned.select("main_category").distinct().count() →. returned 1.
- Dropped main_category (zero variance**).**

Highlights:

- Removes irrelevant / redundant columns.
- Reduces feature space dimensionality.
- Improves model training speed and interpretability.

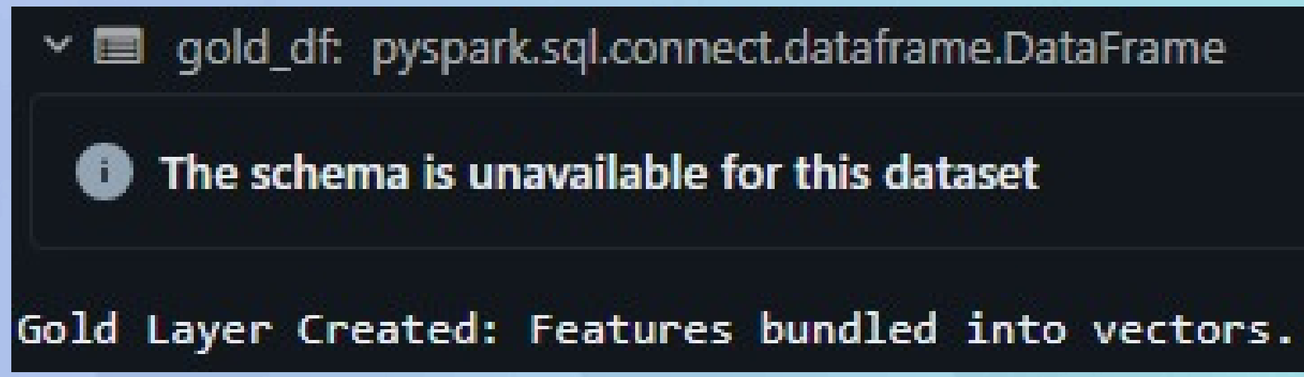
```
1  silver_df.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- main_category: string (nullable = true)
 |-- sub_category: string (nullable = true)
 |-- ratings: string (nullable = true)
 |-- no_of_ratings: string (nullable = true)
 |-- discount_price: string (nullable = true)
 |-- actual_price: string (nullable = true)
 |-- _ingested_at: timestamp (nullable = false)
 |-- sub_category_standardized: string (nullable = true)
 |-- actual_price_clean: float (nullable = true)
 |-- discount_price_clean: float (nullable = true)
```

Gold Layer: Categorical Encoding & Feature Vector Assembly

Converting Categories to Numeric Features for ML

- Categorical text (sub_category_standardized) → numeric (sub_cat_index).
- Used StringIndexer from PySpark.
- Handles unknown/missing categories automatically (Unknown → 0).
- Ready for ML models and feature vector assembly.



Bundling Features into ML-Ready Vectors

1. **Why**:
- ML models require all features in a single numeric vector
- Combines categorical (sub_cat_index) + numeric features
2. **Features Used:**
- ["sub_cat_index", "ratings", "no_of_ratings", "discount_price"]
3. **Technique:**
- Used Vector Assembler from PySpark
- Target column actual_price renamed as label

Gold Layer: Outlier Detection & Filtering

Filtered Unrealistic Price Values

- **Why**: Identified outliers (e.g., **₹6.1 Crore)** and **removed** zero-priced items, as they **skewed** the model's predictions.
- **Step 1: Described** price distribution to detect anomalies.
- **Step 2:** Filtered unrealistic product prices:
  - label > 100
  - label < 100000
  - discount_price > 0
- **Step 3:** Reduced dataset from **12496** records to **11221** after filtering.

summary	label
count	12496
mean	7893.890125640205
stddev	546496.5794390141
min	0.0
max	6.108299E7

Data Splitting: Train & Test Sets

Splitting Data into Training and Testing Sets

```python
# Partition data using an 80/20 split
# Seed=42 ensures consistent results across multiple executions
train_data, test_data = gold_dfc.randomSplit([0.8, 0.2], seed=42)

print(f"Training set count: {train_data.count()}")
print(f"Testing set count: {test_data.count()}")
```

- **Why:** Dividing data into training and testing sets ensures that the model learns from one set and tests on unseen data for accurate performance evaluation.

```
> test_data: pyspark.sql.connect.dataframe.DataFrame
> train_data: pyspark.sql.connect.dataframe.DataFrame

Training set count: 9033
Testing set count: 2190
```

Step 1: Split the filtered Gold Layer dataset into two parts:
Training set: 9033 record
Testing set: 2186 records
Step 2: Used 80% for training and 20% for testing to maintain a balanced split.

Model Comparative Analysis

- **Why:** Compared multiple regression models to identify the best for fair price prediction.
- **Models Tested:**
  - Linear Regression (interpretable and transparent)
  - Random Forest Regressor (captures non-linear patterns)
  - Gradient Boosted Trees (balanced accuracy and complexity)

- **Process:**
  - Trained models on **80%** of the training data
  - Predicted and evaluated on **20%** testing data

- **Evaluation Metrics:**
  - RMSE (Root Mean Squared Error): Measures average valuation
  - R^2 (R-squared): Indicates the proportion of variance explained by the model.

predictions: pyspark.sql.connect.dataframe.DataFrame

```
--- Linear Regression Performance ---
RMSE: 2292.00 (Avg. Price Valuation Error in ₹)
R2 Score: 0.8776 (Model Accuracy / Variance Explained)
----------------------------------
--- Random Forest Performance ---
RMSE: 2533.29 (Avg. Price Valuation Error in ₹)
R2 Score: 0.7688 (Model Accuracy / Variance Explained)
----------------------------------
--- Gradient Boosted Trees (GBT) Performance ---
RMSE: 2500.68 (Avg. Price Valuation Error in ₹)
R2 Score: 0.8280 (Model Accuracy / Variance Explained)
----------------------------------
```

- **Key Insight:**
  Linear Regression achieved the lowest RMSE and the highest R^2, providing the best fit for predicting fair prices with high accuracy.

Market Pricing Insights: Identifying Underpriced Fitness Products

1. Final Model Selection:

- Trained the final Linear Regression model on the training data.
- Estimated expected market prices for fitness products.

2. Model Coefficients (Feature Impact):

```
> analysis_df: pyspark.sql.connect.dataframe.DataFrame
> final_predictions: pyspark.sql.connect.dataframe.DataFrame
> undervalued_market_assets: pyspark.sql.connect.dataframe.DataFrame = [Product Name: string, Product Category: string ... 4 more fields]
Final pricing model trained successfully.

Model Feature Weights (Impact on Price Prediction):
product_category: 18.7438
customer_rating: 44.5259
rating_volume: 0.0102
listed_price: 1.6387
```

3. Underpriced Products Identification:

- Calculated the price gap between predicted and actual market prices.
- Filtered products with a price gap above ₹500 and calculated the undervaluation percentage.

4. Strategic Insights:

- Top underpriced products identified, with high value gaps.
- Retailers and consumers can make data-driven purchasing and pricing decisions based on this analysis.
- Opportunities for retailers to adjust pricing for better profit margins or to avoid overpricing.

Product Name	Product Category	Current Market Price (₹)	Predicted Fair Market Price (₹)	Price Gap vs Market Expectation (₹)	Undervaluation Percentage (%)
CGC-60 Group Cycling Bike	Cardio Equipment	40,000	55,543.13	15,543.13	28
CEB-80 E	Cardio Equipment	30,000	45,233.33	15,233.33	33.7
Montra Downtown 7X3 Geared with Disc 700CX	Cycling	25,500	37,480.78	11,980.78	32
KAMACHI SB-08 Spin Bike with 13 Kg Fly Whe	Cardio Equipment	34,999	45,571.44	10,572.44	23.2
Avon TM-165 (5 HP Peak) Motorized Treadmil	Cardio Equipment	60,462	70,802.47	10,340.47	14.6
Jordan Fitness JF-38 (4 HP DC Peak) Motori...	Cardio Equipment	66,000	74,677.02	8,677.02	11.6
JOOLA Carbon Pro Professional Racket	Badminton	13,999	22,351.68	8,352.68	37.4
Jordan Fitness 3.5 HP DC Peak with Incline...	Cardio Equipment	64,500	72,265.19	7,765.19	10.7
Sg King Cobra Grade 1 English Willow Crick...	Cricket	14,999	21,881.47	6,882.47	31.5
CEB-60 U	Cardio Equipment	24,000	30,765.53	6,765.53	22

Thank You