🏥 Novartis Datathon 2025 - Generic Erosion Forecasting

> **Predict pharmaceutical brand sales erosion after generic drug entry**

Python 3.10+ License MIT

📋 Table of Contents

🎯 Overview

When generic drugs enter the market, branded drug sales typically decline—this is called **generic erosion**. This project predicts the **24-month post-generic sales trajectory** for pharmaceutical brands.

Two Scenarios:

Scenario	Task	Available Data
Scenario 1	Predict months 0-23	Only pre-entry data (months -24 to -1)
Scenario 2	Predict months 6-23	Pre-entry + actual months 0-5

🔬 Problem Statement

Goal: Predict monthly `volume` (sales units) for 24 months after generic entry.

Key Challenge: Bucket 1 brands (high erosion, mean normalized volume ≤ 0.25) are weighted **2×** in the evaluation metric.

```
Normalized Volume = Actual Volume / Pre-Entry Average (Avg_j)

Bucket 1: mean(normalized volume) ≤ 0.25  →  High erosion, 2× weight
Bucket 2: mean(normalized volume) > 0.25   →  Lower erosion, 1× weight
```

📁 Project Structure

```
Main_project/
│
├── src/                         # CORE LOGIC (Python modules)
│   ├── config.py                # Paths, constants, model parameters
│   ├── data_loader.py           # Load and validate data
│   ├── bucket_calculator.py     # Compute Avg_j, buckets, normalization
│   ├── feature_engineering.py   # Create 40+ features
│   ├── models.py                # Baseline + ML models
│   ├── evaluation.py            # PE metric computation
│   ├── submission.py            # Generate submission files
│   ├── pipeline.py              # End-to-end CLI pipeline
│   └── eda_analysis.py          # EDA computations
│
├── scripts/                     # EXECUTION SCRIPTS
│   ├── run_demo.py              # Quick demo (test everything)
│   ├── train_models.py          # Train and compare all models
│   ├── generate_final_submissions.py  # Create competition submissions
│   └── validate_submissions.py  # Validate before upload
│
├── notebooks/                   # VISUALIZATION (Jupyter)
│   ├── 01_eda_visualization.ipynb
│   ├── 02_feature_exploration.ipynb
│   └── 03_model_results.ipynb
│
├── data/
│   ├── raw/                     # Input CSV files
│   └── processed/               # Generated intermediate files
│
├── models/                      # Saved trained models (.joblib)
├── submissions/                 # Generated submission files
├── reports/                     # Model comparison results
│   └── figures/                 # Saved plots
│
├── requirements.txt             # Python dependencies
└── README.md                    # This file
```

🚀 Quick Start

1. Setup Environment

```
# Navigate to project
cd D:\Datathon\novartis_datathon_2025\Main_project
```

```
# Create virtual environment (if not exists)
python -m venv saeed_venv

# Activate environment
.\saeed_venv\Scripts\Activate.ps1

# Install dependencies
pip install -r requirements.txt
```

2. Run Quick Demo

```
python scripts/run_demo.py
```

This will:

- Load data
- Create features
- Train baseline model
- Evaluate predictions
- Generate sample submission

3. Train All Models

```
# Train for Scenario 1
python scripts/train_models.py --scenario 1

# Train for Scenario 2
python scripts/train_models.py --scenario 2
```

4. Generate Final Submissions

```
python scripts/generate_final_submissions.py --model baseline
```

5. Validate Submissions

```
python scripts/validate_submissions.py
```

📊 Data Description

Input Files (in `data/raw/`)

File	Description	Key Columns
`df_volume_train.csv`	Monthly sales volume	`country`, `brand_name`, `months_postgx`, `volume`
`df_generics_train.csv`	Generic competitor info	`country`, `brand_name`, `months_postgx`, `num_generics`
`df_medicine_info_train.csv`	Brand metadata	`country`, `brand_name`, `ther_area`, `hospital_rate`

Time Reference

```
months_postgx:
  -24 to -1  →  Pre-generic entry (historical)
    0        →  Generic entry month
    1 to 23  →  Post-generic entry (forecast period)
```

🔄 Pipeline Workflow

```
┌─────────────────┐
│    Load Data    │  data_loader.py
└────────┬────────┘
         │
         ▼
┌─────────────────┐
│ Compute Avg_j   │  bucket_calculator.py
│ Assign Buckets  │  (Pre-entry average volume)
└────────┬────────┘
         │
         ▼
┌─────────────────┐
│   Engineer      │  feature_engineering.py
│   Features      │  (40+ features)
└────────┬────────┘
         │
         ▼
┌─────────────────┐
│  Train Models   │  models.py
│  - Baseline     │
│  - LightGBM     │
│  - XGBoost      │
```

```
└─────────────────┘
         │
         ▼
┌─────────────────┐
│    Evaluate     │  evaluation.py
│   (PE Metric)   │
└─────────────────┘
         │
         ▼
┌─────────────────┐
│    Generate     │  submission.py
│    Submission   │
└─────────────────┘
```

🤖 Models

1. Baseline Models

Model	Formula	Best For
No Erosion	`volume = avg_j`	Upper bound baseline
Linear Decay	`volume = avg_j × (1 - λ × month)`	Simple decay
Exponential Decay	`volume = avg_j × exp(-λ × month)`	✅ **Best performer**

2. Machine Learning Models

Model	Type	Features Used
LightGBM	Gradient Boosting	40+ engineered features
XGBoost	Gradient Boosting	40+ engineered features

Feature Categories

- **Lag Features:** `volume_lag_1`, `volume_lag_3`, `volume_lag_6`, `volume_lag_12`
- **Rolling Features:** `rolling_mean_3`, `rolling_std_3`, `rolling_mean_6`, etc.
- **Competition:** `num_generics`, `months_with_generics`, `generics_growth_rate`
- **Time Features:** `months_postgx`, `month_sin`, `month_cos`
- **Pre-entry:** `pre_entry_slope`, `pre_entry_volatility`, `avg_vol`

📏 Evaluation Metric

The competition uses a custom **Prediction Error (PE)** metric:

```
PE_brand = w₁×|monthly_errors| + w₂×|sum_0_5| + w₃×|sum_6_11| + w₄×|sum_12_23|
```

Scenario 1 Weights:

Component	Weight
Monthly errors	20%
Sum months 0-5	**50%** ← Focus here!
Sum months 6-11	20%
Sum months 12-23	10%

Scenario 2 Weights:

Component	Weight
Monthly errors	20%
Sum months 6-11	**50%** ← Focus here!
Sum months 12-23	30%

Final Score:

```
Final PE = (2 × avg_PE_bucket1 + 1 × avg_PE_bucket2) / (2 × n_bucket1 + 1 × n_bucket2)
```

Lower is better!

📈 Results

Model Comparison

Model	Scenario 1 PE	Scenario 2 PE
No Erosion Baseline	1.84	2.18
Exponential Decay (λ=0.05)	**1.18** ✅	**1.10** ✅
XGBoost	2.84	3.39
LightGBM	14.93	14.96

Best Model: Exponential Decay baseline with λ=0.05

Final Submissions

File	Rows	Brands

File	Rows	Brands
`scenario1_baseline_final.csv`	8,160	340
`scenario2_baseline_final.csv`	6,120	340

💻 Usage Examples

Using the Pipeline Module

```python
from src.pipeline import run_pipeline

# Run full pipeline for Scenario 1
run_pipeline(scenario=1, model_type='lightgbm', generate_test_submission=True)
```

Loading and Merging Data

```python
from src.data_loader import load_all_data, merge_datasets

# Load training data
volume, generics, medicine = load_all_data(train=True)
merged = merge_datasets(volume, generics, medicine)
print(merged.shape)  # (93744, 11)
```

Computing Buckets

```python
from src.bucket_calculator import create_auxiliary_file

aux_df = create_auxiliary_file(merged, save=True)
print(aux_df['bucket'].value_counts())
# Bucket 1: 130 brands (high erosion)
# Bucket 2: 1823 brands (lower erosion)
```

Creating Features

```python
from src.feature_engineering import create_all_features, get_feature_columns

featured = create_all_features(merged, avg_j)
feature_cols = get_feature_columns(featured)
print(f"Total features: {len(feature_cols)}")  # ~40 features
```

Training a Model

```
from src.models import GradientBoostingModel

model = GradientBoostingModel(model_type='lightgbm')
model.fit(X_train, y_train, X_val, y_val)
predictions = model.predict(X_test)
model.save("scenario1_lightgbm")
```

Evaluating Predictions

```
from src.evaluation import evaluate_model

results = evaluate_model(actual_df, pred_df, aux_df, scenario=1)
print(f"Final Score: {results['final_score']:.4f}")
```

🔧 Configuration

Key settings in `src/config.py`:

```
# Bucket threshold
BUCKET_1_THRESHOLD = 0.25  # Mean erosion ≤ 0.25 = Bucket 1

# Metric weights - Scenario 1
S1_SUM_0_5_WEIGHT = 0.5    # 50% weight on early months

# Model parameters
LGBM_PARAMS = {
    'n_estimators': 500,
    'learning_rate': 0.05,
    'max_depth': 6,
    ...
}
```

📓 Notebooks

Open in Jupyter or VS Code:

Notebook	Purpose
`01_eda_visualization.ipynb`	Data quality, distributions, erosion curves
`02_feature_exploration.ipynb`	Feature correlations and importance
`03_model_results.ipynb`	Model comparison and submission analysis

```
jupyter notebook notebooks/
```

🎯 Key Insights

1. **Exponential decay baseline outperforms ML models** - The decay pattern is well-captured by a simple formula
2. **Bucket 1 is critical** - Only ~7% of brands but 2× weight in metric
3. **Early months matter most** - 50% weight on months 0-5 (S1) or 6-11 (S2)
4. **ML models need tuning** - Current implementation predicts raw volume; should predict normalized volume

---

📚 References

- Novartis Datathon 2025 Guidelines
- Question Set
- Metric Calculation

👥 Team

Branch: Saeed

📄 License

This project is for the Novartis Datathon 2025 competition.

🏆 Good luck with the competition! 🏆