📚 Project Explanation - Novartis Datathon 2025

> **This document explains EXACTLY what each file does, step-by-step, in simple terms.**
> Based on the competition requirements from `datathon_explanation.md` .

🎯 The Big Picture: What Are We Doing?

The Business Problem (In Plain English)

Imagine you're Novartis, a big pharmaceutical company. You sell a drug called "BrandX" for $100.

Then the patent expires.

Now other companies can make **generic copies** that do the same thing but cost only $20.

What happens? Your sales **DROP** (this is called **generic erosion**).

The Question: How much will sales drop over the next 24 months?

Why Does This Matter?

Novartis needs to:

- Plan budgets
- Allocate resources
- Make business decisions

Your job: Build a model that predicts this sales decline accurately.

📊 The Two Scenarios

Scenario	What You Have	What You Predict
Scenario 1	Only BEFORE generic entry	Months 0-23 (all 24 months)
Scenario 2	Before + First 6 months actual	Months 6-23 (remaining 18 months)

Think of it like weather forecasting:

- **Scenario 1** = Predicting next week's weather with only historical data
- **Scenario 2** = Predicting next week's weather knowing today's weather

---

🪣 The Bucket System (VERY IMPORTANT!)

Not all drugs decline the same way:

Bucket	Mean Erosion	What It Means	Scoring Weight

Bucket	Mean Erosion	What It Means	Scoring Weight
Bucket 1	≤ 0.25	SEVERE erosion (sales crash to <25% of original)	**2× (double!)**
Bucket 2	> 0.25	Moderate erosion	1× (normal)

Why This Matters:

If you predict Bucket 1 drugs wrong, your score gets **penalized TWICE as much!**

Strategy: Focus extra effort on predicting high-erosion drugs accurately.

📁 File-by-File Explanation

1️⃣ `src/config.py` - The Settings File

What it does: Stores ALL settings and constants in ONE place.

What's inside:

```
# WHERE to find files
PROJECT_ROOT = Path(__file__).parent.parent
DATA_RAW = PROJECT_ROOT / "data" / "raw"
MODELS_DIR = PROJECT_ROOT / "models"

# COMPETITION RULES
PRE_ENTRY_MONTHS = 12      # Use 12 months before generic entry
POST_ENTRY_MONTHS = 24     # Predict 24 months after
BUCKET_1_THRESHOLD = 0.25  # Below this = Bucket 1 (high erosion)

# SCORING WEIGHTS for Scenario 1
S1_SUM_0_5_WEIGHT = 0.5    # First 6 months = 50% of score!
S1_SUM_6_11_WEIGHT = 0.2   # Months 6-11 = 20%
S1_SUM_12_23_WEIGHT = 0.1  # Months 12-23 = 10%

# MODEL SETTINGS
LGBM_PARAMS = {...}  # LightGBM hyperparameters
XGB_PARAMS = {...}   # XGBoost hyperparameters
```

In simple terms: This is like a "control panel" where you adjust all the knobs in one place instead of hunting through every file.

2️⃣ `src/data_loader.py` - The Data Reader

What it does: Loads the 3 CSV files and combines them.

The 3 datasets:

File	Contains	Key Columns
`df_volume`	Sales history	`volume` (units sold each month)
`df_generics`	Competition info	`n_gxs` (number of generic competitors)
`df_medicine_info`	Drug characteristics	`ther_area`, `hospital_rate`, etc.

Key functions:

```
# Load one dataset
volume_df = load_volume_data(train=True)

# Load all 3 datasets
volume, generics, medicine = load_all_data(train=True)

# MERGE them together into ONE table
merged = merge_datasets(volume, generics, medicine)
```

In simple terms: Like gathering ingredients from 3 different shelves and putting them on ONE cutting board.

Output: A single DataFrame with ALL information per (country, brand, month).

---

3 `src/bucket_calculator.py` - The Normalization Engine

What it does: Calculates key metrics for evaluation.

Step-by-step:

Step A: Calculate `Avg_j` (Pre-entry Average)

```
For each drug:
    Avg_j = average volume from month -12 to -1
```

Example:

- BrandX sold 1000, 1100, 900, ... units in the 12 months BEFORE generics arrived
- Avg_j = average of these = 1000 units

Why? This is the "baseline" to compare against. If you predict 500 units and Avg_j was 1000, that's 50% of baseline.

Step B: Calculate Normalized Volume

```
Normalized Volume = Actual Volume / Avg_j
```

Example:

- Month 3 actual: 400 units
- Avg_j: 1000 units
- Normalized: 400/1000 = 0.4 (40% of original sales)

Step C: Calculate Mean Erosion

```
Mean Erosion = average(Normalized Volume) over months 0-23
```

Example:

- BrandX normalized volumes: [0.8, 0.6, 0.4, 0.3, 0.2, ...]
- Mean erosion = average = 0.35

Step D: Assign Buckets

```
if Mean Erosion ≤ 0.25:
    Bucket = 1  (HIGH erosion - severe sales crash)
else:
    Bucket = 2  (LOWER erosion - moderate decline)
```

Output file: `aux_bucket_avgvol.csv` containing:

- `country`, `brand_name`
- `avg_vol` (the pre-entry average)
- `mean_erosion`
- `bucket` (1 or 2)

---

4 `src/feature_engineering.py` - The Feature Factory

What it does: Creates ~40 features for the ML model to learn from.

Feature Categories:

A. Lag Features (Past Values)

```
volume_lag_1  = volume from 1 month ago
volume_lag_3  = volume from 3 months ago
volume_lag_6  = volume from 6 months ago
volume_lag_12 = volume from 12 months ago
```

Why? Recent sales predict future sales.

B. Rolling Features (Trends)

```
rolling_mean_3  = average of last 3 months
rolling_std_3   = volatility of last 3 months
rolling_mean_6  = average of last 6 months
rolling_mean_12 = average of last 12 months
```

Why? Captures momentum and stability.

C. Competition Features (Generics)

```
n_gxs               = number of generic competitors NOW
n_gxs_cummax        = maximum competitors seen so far
months_with_generics = how long generics have been in market
```

Why? More competitors = more erosion.

D. Pre-Entry Features (CRITICAL for Scenario 1)

```
avg_vol             = pre-entry average (the baseline)
pre_entry_slope     = was sales growing or declining BEFORE generics?
pre_entry_volatility = how stable were sales before?
```

Why? In Scenario 1, this is ALL you have to predict with!

E. Time Features

```
months_postgx = months since generic entry (0, 1, 2, ...)
month_sin/cos = capture seasonality
is_early_postgx = is this in first 6 months? (binary)
```

Output: DataFrame with original data + ~40 new feature columns.

---

5 `src/models.py` - The Prediction Models

What it does: Implements different prediction strategies.

A. Baseline Models (Simple)

1. No Erosion Baseline:

```
prediction = avg_vol  # (assume sales stay the same forever)
```

This is the WORST case - it ignores erosion completely.

2. Linear Decay:

```
prediction = avg_vol × (1 - 0.03 × month)
```

Sales drop by 3% each month in a straight line.

3. Exponential Decay: ☑ BEST PERFORMER

```
prediction = avg_vol × exp(-0.05 × month)
```

Sales drop quickly at first, then slow down (like real erosion!).

Why exponential works best: Generic erosion follows this pattern naturally:

- Month 0-3: Big drop (generics are new, doctors switch)
- Month 12+: Slower decline (loyal patients stay)

B. ML Models (Complex)

LightGBM / XGBoost:

- Use ALL 40 features
- Learn patterns from historical data
- Can capture non-linear relationships

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

Current Result: Baseline exponential (PE=1.18) beats ML models (PE=2.84+)

Why? The decay pattern is so consistent that a simple formula works better than complex ML on this data.

---

6 `src/evaluation.py` - The Scoring System

What it does: Calculates the official competition metric (PE = Prediction Error).

Scenario 1 PE Formula:

```
PE = 0.2 × (monthly errors normalized)
   + 0.5 × (error in months 0-5 sum)      ← 50% WEIGHT!
   + 0.2 × (error in months 6-11 sum)
   + 0.1 × (error in months 12-23 sum)
```

In plain English:

- Getting months 0-5 right is HALF your score
- Monthly individual errors = 20%
- Later months matter less

Scenario 2 PE Formula:

```
PE = 0.2 × (monthly errors normalized)
   + 0.5 × (error in months 6-11 sum)     ← 50% WEIGHT!
   + 0.3 × (error in months 12-23 sum)
```

Final Score Calculation:

```
Final Score = (2 × avg_PE_bucket1 + 1 × avg_PE_bucket2) /
total_weighted_brands
```

The key insight: Bucket 1 errors count DOUBLE.

Example:

- Bucket 1 average PE: 0.5 (10 brands)
- Bucket 2 average PE: 0.3 (100 brands)
- Final = (2×0.5×10 + 1×0.3×100) / (2×10 + 1×100) = (10 + 30) / 120 = 0.33

---

7 `src/submission.py` - The Output Generator

What it does: Creates the CSV file you upload to the competition.

Required Format:

country	brand_name	months_postgx	volume
COUNTRY_001	BRAND_ABC	0	1234.56
COUNTRY_001	BRAND_ABC	1	1100.23
...	...	...	...
COUNTRY_001	BRAND_ABC	23	456.78

Validation Checks:

```
☑ All required columns present
☑ No missing values
☑ No negative volumes
☑ Correct months for scenario (0-23 or 6-23)
☑ Every brand has all required months
☑ Total rows = brands × months
```

Example output:

- Scenario 1: 340 brands × 24 months = 8,160 rows
- Scenario 2: 340 brands × 18 months = 6,120 rows

---

8 `src/pipeline.py` - The Orchestrator

What it does: Runs EVERYTHING in the correct order.

The Pipeline Steps:

```
STEP 1: Load Data
    ↓
STEP 2: Create Auxiliary File (avg_vol, buckets)
    ↓
```

```
STEP 3: Feature Engineering (create 40 features)
    ↓
STEP 4: Split Train/Validation
    ↓
STEP 5: Prepare X (features) and y (target)
    ↓
STEP 6: Train Model
    ↓
STEP 7: Evaluate on Validation Set
    ↓
STEP 8: Generate Submission File
```

Usage:

```
python src/pipeline.py --scenario 1 --model lightgbm
```

---

9 `src/eda_analysis.py` - The Data Explorer

What it does: Analyzes and understands the data BEFORE modeling.

Key analyses:

```
# Data quality
- Missing values per column
- Duplicate records
- Negative volumes

# Distribution analysis
- Volume distribution (heavily right-skewed)
- Brands per country
- Brands per therapeutic area

# Erosion analysis
- Average erosion curve over 24 months
- Erosion by bucket
- Impact of competition on erosion

# Bucket analysis
- How many Bucket 1 vs Bucket 2?
- Characteristics of high-erosion drugs
```

📜 Scripts Explained

`scripts/run_demo.py`

Purpose: Quick test to make sure everything works.

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

What it does:

1. Loads small sample of data
2. Creates features
3. Trains baseline model
4. Evaluates predictions
5. Generates sample submission

Use when: You want to quickly verify the code works.

`scripts/train_models.py`

Purpose: Train and compare ALL models.

```
python scripts/train_models.py --scenario 1
python scripts/train_models.py --scenario 2
```

What it does:

1. Trains No Erosion baseline
2. Trains Exponential Decay baseline (tunes λ)
3. Trains LightGBM
4. Trains XGBoost
5. Compares all models
6. Saves best models to `models/`
7. Saves comparison to `reports/model_comparison_scenarioX.csv`

`scripts/generate_final_submissions.py`

Purpose: Create the final competition submission files.

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

Output:

- `submissions/scenario1_baseline_final.csv`
- `submissions/scenario2_baseline_final.csv`

`scripts/validate_submissions.py`

Purpose: Check submissions BEFORE uploading.

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

Checks:

- ☑ Correct column names
- ☑ No missing values
- ☑ No negative volumes
- ☑ Correct months per scenario
- ☑ All brands present
- ☑ Correct total row count

📓 Notebooks Explained

Notebook	Purpose
`01_eda_visualization.ipynb`	See data distributions, erosion curves, bucket breakdown
`02_feature_exploration.ipynb`	Visualize features, correlations, importances
`03_model_results.ipynb`	Compare model performance, analyze submissions

These are for VISUALIZATION only - all logic is in `src/` files.

🔄 Complete Workflow

```
┌──────────────────────────────────────────────────────────────────┐
│                          YOUR WORKFLOW                           │
└──────────────────────────────────────────────────────────────────┘

1. SETUP
   pip install -r requirements.txt

2. QUICK TEST
   python scripts/run_demo.py

3. EXPLORE DATA (optional)
   Open notebooks/01_eda_visualization.ipynb
```

```
4. TRAIN MODELS
   python scripts/train_models.py --scenario 1
   python scripts/train_models.py --scenario 2

5. CHECK RESULTS
   Look at reports/model_comparison_scenario1.csv
   Open notebooks/03_model_results.ipynb

6. GENERATE SUBMISSIONS
   python scripts/generate_final_submissions.py --model baseline

7. VALIDATE
   python scripts/validate_submissions.py

8. SUBMIT
   Upload submissions/*.csv to competition
```

📈 Current Results

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

Winner: Simple exponential decay beats complex ML!

🎯 Key Takeaways

1. **Bucket 1 is CRITICAL** - Double weighted, focus on high-erosion drugs
2. **Early months matter most** - 50% of score from first 6 months
3. **Simple models can win** - Exponential decay captures the physics
4. **Normalize everything** - All errors divided by pre-entry average
5. **Validate before submit** - One wrong format = rejected submission

---

📚 Quick Reference

Task	Command
Test everything	`python scripts/run_demo.py`
Train models	`python scripts/train_models.py --scenario 1`
Generate submission	`python scripts/generate_final_submissions.py --model baseline`

Task	Command
Validate submission	`python scripts/validate_submissions.py`
Run full pipeline	`python src/pipeline.py --scenario 1 --model lightgbm`

Good luck with the competition! 🏆