📋 Novartis Datathon 2025 - AI Agent Workflow Todo List

> **Project:** Pharmaceutical Generic Erosion Forecasting
> **Objective:** Predict 24-month sales volume after generic entry
> **Priority Metric:** Minimize Prediction Error (PE) with focus on Bucket 1 (high erosion)

🎯 Project Overview

Scenario	Input	Predict	Key Weight
Scenario 1	No post-generic data	Months 0-23	50% on months 0-5
Scenario 2	Months 0-5 provided	Months 6-23	50% on months 6-11

Bucket Weighting: Bucket 1 (high erosion) = 2×, Bucket 2 = 1×

Phase 0: Project Setup

0.1 Create Folder Structure

- [ ] Create `data/raw/` directory
- [ ] Create `data/processed/` directory
- [ ] Create `src/` directory for Python modules
- [ ] Create `notebooks/` directory for Jupyter notebooks
- [ ] Create `models/` directory for saved models
- [ ] Create `submissions/` directory for output CSVs
- [ ] Create `reports/` directory for markdown reports

0.2 Data Organization

- [ ] Copy `SUBMISSION/Data files/TRAIN/df_volume_train.csv` → `data/raw/`
- [ ] Copy `SUBMISSION/Data files/TRAIN/df_generics_train.csv` → `data/raw/`
- [ ] Copy `SUBMISSION/Data files/TRAIN/df_medicine_info_train.csv` → `data/raw/`
- [ ] Copy `SUBMISSION/Data files/TEST/df_volume_test.csv` → `data/raw/`
- [ ] Copy `SUBMISSION/Data files/TEST/df_generics_test.csv` → `data/raw/`
- [ ] Copy `SUBMISSION/Data files/TEST/df_medicine_info_test.csv` → `data/raw/`
- [ ] Copy `SUBMISSION/Metric files/metric_calculation.py` → `src/`
- [ ] Copy `SUBMISSION/Submission example/submission_template.csv` → `submissions/`

0.3 Update Dependencies

- [ ] Verify `requirements.txt` includes:
  - pandas, numpy, scipy
  - scikit-learn, xgboost, lightgbm
  - statsmodels, prophet

    - matplotlib, seaborn, plotly
    - jupyter, notebook, ipykernel
    - tqdm
- [ ] Install dependencies: `pip install -r requirements.txt`

Phase 1: Understand Requirements

1.1 Read and Summarize Objective

- [ ] Read `Docs/datathon_explanation.md` thoroughly
- [ ] Read `Docs/novartis_datathon_2025_instructions.md` (504 lines)
- [ ] Document: "Forecast 24-month post-generic sales erosion"
- [ ] Understand: Drug lifecycle → Patent expiry → LoE → Generic entry → Erosion

1.2 Document Prediction Scenarios

- [ ] **Scenario 1 Documentation:**
    - Input: Pre-generic historical data only
    - Output: Months 0-23 predictions (24 values per country-brand)
    - Test set: 228 observations
    - Key: Months 0-5 carry 50% weight
- [ ] **Scenario 2 Documentation:**
    - Input: Pre-generic data + months 0-5 actuals
    - Output: Months 6-23 predictions (18 values per country-brand)
    - Test set: 112 observations
    - Key: Months 6-11 carry 50% weight

1.3 Extract Vocabulary Definitions

- [ ] Document key terms:
    - **LoE (Loss of Exclusivity):** Patent expiration date
    - **Generic Entry / Month 0:** First month generics appear
    - **Generic Erosion:** Sales drop after generics enter
    - **Mean Generic Erosion:** Average normalized volume months 0-23
    - **Bucket 1:** High erosion (mean 0-0.25) - DOUBLE WEIGHTED
    - **Bucket 2:** Lower erosion (mean > 0.25)
    - **Avg_j:** 12-month pre-entry average volume (normalization factor)
    - **vol_norm_i:** Normalized volume = Vol_i / Avg_j
    - **n_gxs:** Number of generic competitors at each month
    - **Therapeutic Area:** Disease category
    - **Hospital Rate:** Fraction of sales through hospitals
    - **Biological:** Drug made from living cells
    - **Small Molecule:** Classic chemical drug

Phase 2: Data Loading & Validation

2.1 Load Datasets

- [ ] Load `df_volume_train.csv` into DataFrame
- [ ] Load `df_generics_train.csv` into DataFrame
- [ ] Load `df_medicine_info_train.csv` into DataFrame
- [ ] Load test datasets (volume, generics, medicine_info)
- [ ] Print shape and info for each DataFrame

2.2 Validate Schema

- [ ] **Volume Dataset Columns:**
  - `country` (string)
  - `brand_name` (string)
  - `month` (datetime)
  - `months_postgx` (int, negative=pre-entry, 0=entry, positive=post-entry)
  - `volume` (float, TARGET VARIABLE)
- [ ] **Generics Dataset Columns:**
  - `country`, `brand_name`, `months_postgx`
  - `n_gxs` (int, number of generic competitors)
- [ ] **Medicine Info Columns:**
  - `country`, `brand_name`
  - `therapeutic_area` (categorical)
  - `hospital_rate` (float 0-1)
  - `main_package` (categorical)
  - `biological` (boolean)
  - `small_molecule` (boolean)

2.3 Check Data Quality

- [ ] Check missing values per column per dataset
- [ ] Check data types (convert if needed)
- [ ] Check unique country-brand combinations (Train: 1,953, Test: 340)
- [ ] Verify months_postgx range: should be -24 to +23 for train
- [ ] Check for duplicates on (country, brand_name, months_postgx)

2.4 Merge Datasets

- [ ] Merge volume + generics on (country, brand_name, months_postgx)
- [ ] Merge result + medicine_info on (country, brand_name)
- [ ] Verify merge did not drop rows unexpectedly
- [ ] Create unified modeling table

2.5 Save Processed Data

- [ ] Save merged training data to `data/processed/merged_train.csv`

- [ ] Save merged test data to `data/processed/merged_test.csv`

Phase 3: Compute Erosion Buckets

3.1 Compute Pre-Entry Average (Avg_j)

- [ ] For each (country, brand_name):
  - Filter rows where months_postgx in [-12, -1]
  - Compute: `Avg_j = mean(volume)` over these 12 months
- [ ] Handle edge cases: brands with fewer than 12 pre-entry months
- [ ] Store Avg_j as a feature

3.2 Compute Normalized Volume

- [ ] For each post-entry row (months_postgx >= 0):
  - Compute: `vol_norm_i = volume / Avg_j`
- [ ] Handle division by zero (Avg_j = 0 or NaN)

3.3 Compute Mean Generic Erosion

- [ ] For each (country, brand_name):
  - Filter rows where months_postgx in [0, 23]
  - Compute: `mean_erosion = mean(vol_norm_i)`

3.4 Assign Erosion Buckets

- [ ] Apply bucket logic:

```
if 0 <= mean_erosion <= 0.25:
    bucket = 1  # High erosion (PRIORITY)
else:
    bucket = 2  # Lower erosion
```

- [ ] Count bucket distribution (expect Bucket 1 to be minority)
- [ ] Save bucket labels for all training country-brand pairs

3.5 Store Auxiliary Data

- [ ] Create `data/processed/aux_bucket_avgvol.csv` with:
  - country, brand_name, Avg_j, mean_erosion, bucket

Phase 4: Exploratory Data Analysis (EDA)

4.1 Time Series Visualization

- [ ] Plot 5-10 sample country-brand pairs showing full lifecycle

- [ ] Mark Month 0 (generic entry) with vertical line
- [ ] Show pre-entry stability vs post-entry erosion
- [ ] Save plots to `reports/figures/`

4.2 Bucket Comparison Analysis

- [ ] Compare Bucket 1 vs Bucket 2 erosion curves
- [ ] Analyze:
  - Drop speed (how fast does erosion happen?)
  - Final steady-state level
  - Volatility/noise level
  - Recovery patterns (if any)
- [ ] Create bucket-wise aggregate plots

4.3 Competitive Analysis

- [ ] Analyze relationship between `n_gxs` and erosion rate
- [ ] Plot: Number of generics vs mean erosion
- [ ] Identify if more generics = faster/deeper erosion

4.4 Product Feature Analysis

- [ ] Analyze by `therapeutic_area`:
  - Which therapeutic areas have highest erosion?
- [ ] Analyze by `hospital_rate`:
  - Do hospital-distributed drugs erode differently?
- [ ] Analyze by `biological` vs `small_molecule`:
  - Do biologicals have different erosion patterns?
- [ ] Analyze by `main_package`

4.5 Generate EDA Report

- [ ] Create `reports/eda_report.md` with:
  - Data summary statistics
  - Missing value analysis
  - Bucket distribution
  - Key visualizations (embedded or linked)
  - Initial hypotheses for modeling

Phase 5: Feature Engineering

5.1 Time-Based Features

- [ ] **Lag Features:**
  - `lag_1`, `lag_2`, `lag_3` (previous month volumes)
  - `lag_12` (same month previous year if available)
- [ ] **Rolling Statistics:**
  - `rolling_mean_3`, `rolling_mean_6` (pre-entry)

    - `rolling_std_3`, `rolling_std_6`
- [ ] **Pre-Entry Slope:**
    - Linear regression slope over last 12 months pre-entry
- [ ] **Growth Rate:**
    - `growth_rate = (vol_t - vol_t-1) / vol_t-1`

5.2 Categorical Encoding

- [ ] Encode `country` (label encoding or target encoding)
- [ ] Encode `therapeutic_area` (one-hot or target encoding)
- [ ] Encode `main_package` (one-hot or target encoding)

5.3 Competition Features

- [ ] `n_gxs` (number of generics at each month)
- [ ] `n_gxs_cummax` (cumulative max generics seen)
- [ ] `n_gxs_growth` (change in number of generics)

5.4 Normalization Features

- [ ] `Avg_j` (pre-entry average volume)
- [ ] `vol_norm` (normalized volume)
- [ ] `bucket` (as categorical feature)

5.5 Product Features

- [ ] `hospital_rate` (numeric)
- [ ] `biological` (binary)
- [ ] `small_molecule` (binary)

5.6 Store Feature Tables

- [ ] Save `data/processed/features_train.csv`
- [ ] Save `data/processed/features_test.csv`
- [ ] Document feature definitions in `reports/feature_dictionary.md`

Phase 6: Model Development

6.1 Baseline Models

- [ ] **Naive Persistence:** Predict last known pre-entry volume
- [ ] **Moving Average:** Predict 12-month pre-entry average (= Avg_j)
- [ ] **Linear Decay:** Fit linear trend post-entry
- [ ] Compute PE for baselines on validation set

6.2 Machine Learning Models

- [ ] **LightGBM/XGBoost:**
    - Frame as regression: features → volume prediction

    - Train on historical post-entry data
    - Use time-series cross-validation
    - Hyperparameter tuning (GridSearchCV or Optuna)

- [ ] **Reference:** Review `Other_projects/Pharma-Sales-Analysis-and-Forecasting-main/`
    - Check preprocessing approach
    - Check model configuration
    - Adapt relevant techniques

6.3 Time-Series Models

- [ ] **ARIMA/ARIMAX:**
    - Fit per country-brand
    - Include exogenous variables (n_gxs)
- [ ] **Prophet:**
    - Use yearly_seasonality
    - Add n_gxs as regressor
    - Consider holidays/events
- [ ] **(Optional) Deep Learning:**
    - LSTM/GRU for sequence prediction
    - Transformer if data permits

6.4 Scenario-Specific Models

- [ ] **Scenario 1 Model:**
    - Input: Pre-entry features only
    - Output: 24 monthly predictions (months 0-23)
    - Training: Use full training set
- [ ] **Scenario 2 Model:**
    - Input: Pre-entry features + months 0-5 actuals
    - Output: 18 monthly predictions (months 6-23)
    - Training: Can use Scenario 1 test data actuals

6.5 Model Evaluation

- [ ] Implement local PE calculation using `src/metric_calculation.py`
- [ ] Create validation split (time-based, not random)
- [ ] Compute PE for each model variant
- [ ] Track results in `reports/model_comparison.md`

6.6 Optimization Focus

- [ ] **Scenario 1 Priority:**

    - 50% weight on months 0-5 → optimize early predictions
    - Consider ensemble giving more weight to early months

- [ ] **Scenario 2 Priority:**
    - 50% weight on months 6-11 → optimize using 0-5 actuals
    - Leverage observed erosion pattern from months 0-5

- [ ] **Bucket 1 Priority:**
    - Double weighted in final metric
    - Consider separate model for high-erosion cases
    - Augment Bucket 1 training data if possible

Phase 7: Implement Official Metric

7.1 Scenario 1 PE Formula

- [ ] Implement 4-component formula:

```
PE_j = (0.2 * sum_abs_monthly_error_0_23 / (24 * Avg_j) +
        0.5 * abs_sum_error_0_5 / (6 * Avg_j) +
        0.2 * abs_sum_error_6_11 / (6 * Avg_j) +
        0.1 * abs_sum_error_12_23 / (12 * Avg_j))
```

- [ ] Verify against `metric_calculation.py`

7.2 Scenario 2 PE Formula

- [ ] Implement 3-component formula:

```
PE_j = (0.2 * sum_abs_monthly_error_6_23 / (18 * Avg_j) +
        0.5 * abs_sum_error_6_11 / (6 * Avg_j) +
        0.3 * abs_sum_error_12_23 / (12 * Avg_j))
```

- [ ] Verify against `metric_calculation.py`

7.3 Bucket-Weighted Aggregation

- [ ] Implement final aggregation:

```
PE_final = (2/n_B1) * sum(PE_j for j in Bucket1) +
           (1/n_B2) * sum(PE_j for j in Bucket2)
```

- [ ] Note: Perfect score = 0, Poor score > 3

7.4 Metric Unit Tests

- [ ] Create `src/test_metric.py`
- [ ] Test with known inputs/outputs from `auxiliar_metric_computation_example.csv`
- [ ] Test edge cases (Avg_j = 0, missing months)
- [ ] Verify matches official metric calculation

Phase 8: Generate Submission Files

8.1 Scenario 1 Submission

- [ ] Generate predictions for 228 test observations
- [ ] Create CSV with columns: `country`, `brand_name`, `months_postgx`, `volume`
- [ ] Verify: months_postgx covers 0-23 for each country-brand
- [ ] Total rows: 228 × 24 = 5,472
- [ ] Validate against `submission_template.csv` format

8.2 Scenario 2 Submission

- [ ] Generate predictions for 112 test observations
- [ ] Create CSV with columns: `country`, `brand_name`, `months_postgx`, `volume`
- [ ] Verify: months_postgx covers 6-23 for each country-brand
- [ ] Total rows: 112 × 18 = 2,016
- [ ] Validate against template format

8.3 Submission Validation

- [ ] Check no missing values in predictions
- [ ] Check no negative volume predictions
- [ ] Check column names match exactly
- [ ] Check row counts match expected

8.4 Save Submissions

- [ ] Save to `submissions/scenario1_YYYYMMDD_HHMMSS.csv`
- [ ] Save to `submissions/scenario2_YYYYMMDD_HHMMSS.csv`
- [ ] Keep backup of best submissions

Phase 9: Optional Automation Tasks

9.1 Auto-Generate Reports

- [ ] Script to generate `reports/eda_report.md` from notebook outputs
- [ ] Script to generate `reports/model_decisions.md` documenting choices
- [ ] Auto-embed key visualizations in reports

9.2 Feature Importance Analysis

- [ ] Extract top 10 features from XGBoost/LightGBM
- [ ] Create SHAP summary plots
- [ ] Document in `reports/feature_importance.md`

9.3 Error Analysis

- [ ] Generate predicted vs actual scatter plots
- [ ] Plot error distribution by bucket
- [ ] Identify worst-performing country-brands
- [ ] Save analysis to `reports/error_analysis.md`

9.4 Bucket 1 Focus Report

- [ ] Deep dive on high-erosion cases
- [ ] Identify patterns in hardest-to-predict cases
- [ ] Document strategies for Bucket 1 improvement
- [ ] Save to `reports/bucket1_focus_report.md`

9.5 End-to-End Pipeline Script

- [ ] Create `src/pipeline.py` with stages:
    1. Load raw data
    2. Process and merge
    3. Compute buckets
    4. Engineer features
    5. Train models
    6. Generate predictions
    7. Create submission files
- [ ] Make runnable with single command

📊 Progress Tracking

Phase Completion Status

Phase	Status	Notes
0. Project Setup	☐ Not Started	
1. Understand Requirements	☐ Not Started	
2. Data Loading & Validation	☐ Not Started	
3. Compute Erosion Buckets	☐ Not Started	
4. EDA	☐ Not Started	
5. Feature Engineering	☐ Not Started	
6. Model Development	☐ Not Started	

Phase	Status	Notes
7. Implement Metric	☐ Not Started	
8. Generate Submissions	☐ Not Started	
9. Optional Automation	☐ Not Started	

Key Deadlines

- ☐ Phase 1 submission deadline: TBD
- ☐ Final submission selection: TBD

🔗 Reference Files

File	Location	Purpose
`datathon_explanation.md`	`Docs/`	Plain-language objective summary
`novartis_datathon_2025_instructions.md`	`Docs/`	Official challenge instructions
`metric_calculation.py`	`SUBMISSION/Metric files/`	Official PE metric implementation
`submission_template.csv`	`SUBMISSION/Submission example/`	Required output format
`df_volume_train.csv`	`SUBMISSION/Data files/TRAIN/`	Training volume data
`df_generics_train.csv`	`SUBMISSION/Data files/TRAIN/`	Training generics data
`df_medicine_info_train.csv`	`SUBMISSION/Data files/TRAIN/`	Training product info

🧠 Key Insights for AI Agent

1. **Bucket 1 is CRITICAL** - High erosion cases are double-weighted. Prioritize accuracy here.
2. **Early months matter most** - Months 0-5 (Scenario 1) and 6-11 (Scenario 2) carry 50% weight each.
3. **Normalization is essential** - All errors normalized by Avg_j (pre-entry average).
4. **Use provided metric code** - `metric_calculation.py` is the ground truth for scoring.
5. **Leverage test data wisely** - Scenario 2 test actuals (months 0-5) can inform Scenario 1 training.
6. **Consider ensemble approaches** - Different models may excel at different erosion patterns.

7. **Reference project available** - Check `Other_projects/Pharma-Sales-Analysis-and-Forecasting-main/` for similar time-series approaches.

☑ **Mark tasks as complete using `[x]` as you progress**

🤖 **This Todo list is optimized for AI Agent workflow automation**