Training Strategy Analysis

Current Training Approach

How Training Works Now

The current pipeline (`train_models.py`) trains models **twice** - once for each scenario:

```
┌─────────────────────────────────────────────────────────────────┐
│                    CURRENT TRAINING FLOW                        │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  ┌─────────────────┐      ┌─────────────────┐                  │
│  │ SCENARIO 1      │      │ SCENARIO 2      │                  │
│  │ Training        │      │ Training        │                  │
│  ├─────────────────┤      ├─────────────────┤                  │
│  │Train data:      │      │Train data:      │                  │
│  │months 0-23      │      │months 6-23      │                  │
│  │1,953 brands     │      │1,953 brands     │                  │
│  └────────┬────────┘      └────────┬────────┘                  │
│           │                        │                            │
│           ▼                        ▼                            │
│  ┌─────────────────┐      ┌─────────────────┐                  │
│  │Models saved:    │      │Models saved:    │                  │
│  │scenario1_*.jbl  │      │scenario2_*.jbl  │                  │
│  └─────────────────┘      └─────────────────┘                  │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Current Training Details

Aspect	Scenario 1	Scenario 2
Training brands	1,953	1,953
Training months	0-23 (post-entry)	6-23 (post-entry)
Training rows	~46,872	~35,154
Models trained	7 models	7 models
Time	~3-5 minutes	~3-5 minutes

Models Trained (Per Scenario)

1. `baseline_no_erosion` - Predicts avg_vol (no decline)
2. `baseline_exp_decay` - Exponential decay (tunes decay rate)
3. `lightgbm` - Gradient boosting

4. `xgboost` - Gradient boosting
5. `hybrid_lightgbm` - Physics + LightGBM residuals
6. `hybrid_xgboost` - Physics + XGBoost residuals
7. `arihow` - SARIMAX + Holt-Winters

Total: 14 models trained (7 × 2 scenarios)

The Problem

Why Separate Training is Redundant

```
┌──────────────────────────────────────────────────────────────────┐
│                         DATA ANALYSIS                            │
├──────────────────────────────────────────────────────────────────┤
│                                                                  │
│  TRAINING DATA (Same for both scenarios!)                        │
│  ├── 1,953 brands                                                │
│  ├── Complete data: months -24 to 23                             │
│  └── Used to learn erosion patterns                              │
│                                                                  │
│  TEST DATA (Different brands, split by scenario)                 │
│  ├── Scenario 1: 228 brands (predict months 0-23)                │
│  ├── Scenario 2: 112 brands (predict months 6-23)                │
│  └── Total: 340 brands (NO overlap with training)                │
│                                                                  │
└──────────────────────────────────────────────────────────────────┘
```

Key Insight: The test brands are **completely different** between scenarios. It's not the same brand with different forecast horizons - they are **different brands altogether**.

What This Means

Question	Answer
Are S1 and S2 the same brands?	**NO** - completely different 340 brands
Do we need different models?	**NO** - one model can predict any month
Why train twice?	**Unnecessary** - same training data
What determines S1 vs S2?	**Test data availability** (pre-entry only vs pre-entry + 6 months)

Recommended Approach

Option 1: Single Model (Simplest)

Train **ONE model** using all training data (months 0-23), then use it for both scenarios:

```
┌─────────────────────────────────────────────────────────────────┐
│                   RECOMMENDED: SINGLE TRAINING                  │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  ┌─────────────────────────────────────────┐                    │
│  │         UNIFIED TRAINING                │                    │
│  ├─────────────────────────────────────────┤                    │
│  │ Train data: months 0-23                 │                    │
│  │ Brands: 1,953                           │                    │
│  │ Features include: months_postgx         │                    │
│  └─────────────────────────────────────────┘                    │
│                    │                                            │
│                    ▼                                            │
│  ┌─────────────────────────────────────────┐                    │
│  │         ONE MODEL                       │                    │
│  │         (e.g., lightgbm.joblib)         │                    │
│  └─────────────────────────────────────────┘                    │
│                    │                                            │
│          ┌─────────┴─────────┐                                  │
│          ▼                   ▼                                  │
│  ┌───────────────┐  ┌───────────────┐                           │
│  │ S1 Prediction │  │ S2 Prediction │                           │
│  │ months 0-23   │  │ months 6-23   │                           │
│  │ 228 brands    │  │ 112 brands    │                           │
│  └───────────────┘  └───────────────┘                           │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Benefits:

- ✅ Cuts training time in half
- ✅ Conceptually cleaner
- ✅ One model learns all erosion patterns
- ✅ `months_postgx` feature tells model the time horizon

Option 2: Scenario-Specific Validation (Current + Improvement)

Keep training twice but with **proper validation splits**:

- **S1 validation**: Hold out brands, evaluate on months 0-23
- **S2 validation**: Hold out brands, evaluate on months 6-23

This mimics the actual test scenario better for hyperparameter tuning.

Option 3: Scenario-Specific Features (Advanced)

For Scenario 2, we have **6 months of actual post-entry data**. We could:

1. Use months 0-5 actuals as **features** (not just training data)
2. Create features like `actual_erosion_month_0_5`, `actual_vol_month_5`

3. This gives S2 an advantage - we know how the brand actually started eroding

```
# For S2 brands, we can use actual early erosion as a feature
df['early_erosion_actual'] = df.groupby('brand')['volume'].transform(
    lambda x: x[x['months_postgx'] <= 5].mean() / avg_vol
)
```

Comparison Table

Approach	Training Time	Model Files	Complexity	Accuracy
Current (2× training)	~10 min	14 models	Medium	Baseline
Single model	~5 min	7 models	Low	Same/Better
S2 with actual features	~7 min	10 models	High	Potentially Better

---

When Separate Training Makes Sense

Separate training would be justified if:

1. ❌ S1 and S2 were the **same brands** at different time points - **NOT the case**
2. ❌ Different features available for training - **NOT the case** (same train data)
3. ☑️ Different **evaluation metrics** per scenario - **TRUE** (different month weights)
4. ☑️ Hyperparameter tuning per scenario - **Possible benefit**

Recommendation

For Competition Speed: Use Single Training

```
# Train once on all months 0-23
model.fit(X_train, y_train)  # Uses months_postgx as feature

# Predict for S1 brands (months 0-23)
s1_pred = model.predict(X_s1)

# Predict for S2 brands (months 6-23)
s2_pred = model.predict(X_s2)

# Combine into unified submission
submission = pd.concat([s1_pred, s2_pred])
```

For Best Accuracy: Add S2-Specific Features

For Scenario 2 test brands, we have actual volumes for months 0-5. Use them:

```
# S2 brands have actuals for months 0-5
# Use this as additional feature for S2 predictions
s2_features['actual_early_erosion'] = actual_vol_month_5 / avg_vol
s2_features['actual_erosion_slope'] = (vol_month_5 - vol_month_0) / 5
```

Summary

Current State	Issue	Recommendation
Train S1 and S2 separately	Redundant - same train data	Train once, predict for both
14 models total	Doubled work	7 models sufficient
~10 min training	Wasted time	~5 min with single training
S2 ignores actual early data	Missed opportunity	Use months 0-5 actuals as S2 features

Bottom Line: The current approach works but is inefficient. A single trained model can predict for both scenarios since `months_postgx` is a feature that tells the model which time horizon to predict.