📊 Complete Model Configurations & Sweep Reference

> **Novartis Datathon 2025 - All Models, Configs, and Hyperparameter Combinations**

This document provides a comprehensive reference for all model configurations, named sweep configurations, grid sweep combinations, and CLI commands available in this project.

📑 Table of Contents

Complete Model Inventory

All 20 Models Available

#	Model	Type	CLI Flag	Config File	Status
1	**XGBoost**	Gradient Boosting	`--model xgboost`	`model_xgb.yaml`	☑ Production
2	**LightGBM**	Gradient Boosting	`--model lightgbm`	`model_lgbm.yaml`	☑ Production
3	**CatBoost**	Gradient Boosting	`--model catboost`	`model_cat.yaml`	☑ Production
4	**Physics + LightGBM**	Hybrid	`--model hybrid`	`model_hybrid.yaml`	☑ Production
5	**Physics + XGBoost**	Hybrid	`--model hybrid`	`model_hybrid.yaml`	☑ Production
6	**Ridge Regression**	Linear	`--model linear --config-id default`	`model_linear.yaml`	☑ Baseline

#	Model	Type	CLI Flag	Config File	Status
7	**Lasso Regression**	Linear	`--model linear --config-id lasso`	`model_linear.yaml`	☑ Baseline
8	**ElasticNet**	Linear	`--model linear --config-id elasticnet`	`model_linear.yaml`	☑ Baseline
9	**Huber Regression**	Linear	`--model linear --config-id huber`	`model_linear.yaml`	☑ Baseline
10	**ARIHOW (SARIMAX+HW)**	Time Series	`--model arihow`	`model_arihow.yaml`	⚠ Specialized
11	**Neural Network (MLP)**	Deep Learning	`--model nn`	`model_nn.yaml`	⚠ Experimental
12	**LSTM**	Deep Learning	`--model lstm`	`model_lstm.yaml`	⚠ Experimental
13	**CNN-LSTM**	Deep Learning	`--model cnn_lstm`	`model_cnn_lstm.yaml`	⚠ Experimental
14	**KG-GCN-LSTM**	Graph + Deep Learning	`--model kg_gcn_lstm`	`model_kg_gcn_lstm.yaml`	⚠ Experimental
15	**Exponential Decay**	Baseline	`baselines.py`	-	☑ Baseline
16	**Linear Decay**	Baseline	`baselines.py`	-	☑ Baseline
17	**Naive Persistence (Flat)**	Baseline	`--model flat`	-	☑ Baseline
18	**Global Mean**	Baseline	`--model global_mean`	-	☑ Baseline
19	**Trend**	Baseline	`--model trend`	-	☑ Baseline
20	**Historical Curve**	Baseline	`--model historical_curve`	-	☑ Baseline

Model Categories

Category	Models	Use Case
Gradient Boosting	XGBoost, LightGBM, CatBoost	Best accuracy, production use
Hybrid (Physics+ML)	Physics+LightGBM, Physics+XGBoost	Interpretable, physics-informed
Linear Models	Ridge, Lasso, ElasticNet, Huber	Baseline, interpretability, feature selection
Time Series	ARIHOW (SARIMAX + Holt-Winters)	Brands with 12+ months history
Deep Learning	NN, LSTM, CNN-LSTM, KG-GCN-LSTM	Complex patterns, GPU recommended
Baselines	Flat, Trend, Global Mean, Decay	Sanity checks, lower bounds

---

Model Rankings

Accuracy Ranking (Official Metric - Lower is Better)

Based on actual test runs:

Rank	Model	Scenario 1	Scenario 2	Avg Score	Notes
🥇 1	**XGBoost**	0.7671	0.2976	0.5324	Best overall
🥈 2	**CatBoost**	0.7692	**0.2742** ⭐	0.5217	Best S2
🥉 3	**LightGBM**	0.7698	0.3019	0.5359	Fastest
4	Hybrid	~0.78	~0.31	~0.55	Physics-informed
5	ARIHOW	~0.80	~0.32	~0.56	Time series
6	CNN-LSTM	~0.82	~0.35	~0.59	Needs GPU
7	LSTM	~0.83	~0.36	~0.60	Needs GPU
8	Neural Network	~0.84	~0.38	~0.61	MLP
9	KG-GCN-LSTM	~0.85	~0.40	~0.63	Graph NN
10	Linear	~0.88	~0.42	~0.65	Baseline
11	Trend	~0.90	~0.50	~0.70	Simple
12	Historical Curve	~0.92	~0.55	~0.74	Simple
13	Global Mean	~0.95	~0.60	~0.78	Simplest
14	Flat	~1.00	~0.70	~0.85	No change

Speed Ranking (Training Time)

Rank	Model	S1 Time	S2 Time	Total	GPU Required
🥇 1	Flat/Mean/Trend	<1s	<1s	<1s	No
🥈 2	Linear	~0.5s	~0.5s	~1s	No
🥉 3	**LightGBM**	2.17s	2.38s	**4.55s**	No
4	**XGBoost**	4.30s	4.34s	**8.64s**	No
5	Hybrid	~10s	~10s	~20s	No
6	ARIHOW	~30s	~30s	~60s	No
7	Neural Network	~60s	~60s	~120s	Optional
8	**CatBoost**	59.85s	42.81s	**102.66s**	Optional
9	LSTM	~120s	~120s	~240s	Recommended
10	CNN-LSTM	~180s	~180s	~360s	Recommended
11	KG-GCN-LSTM	~300s	~300s	~600s	Required

Quick Decision Guide

```
Need SPEED?      → LightGBM (25x faster than CatBoost)
Need ACCURACY S1? → XGBoost (0.7671)
Need ACCURACY S2? → CatBoost (0.2742) - 8% better than XGBoost
```

```
Need BOTH?        → XGBoost (best trade-off)
Need BASELINE?    → Linear (Ridge)
Need ENSEMBLE?    → XGBoost + LightGBM + CatBoost
Need INTERPRETABILITY? → Hybrid (Physics + ML)
Have LIMITED DATA? → ARIHOW (time series approach)
```

Overview

The project uses a **YAML-based configuration system** with support for:

Feature	Description
Named Configs	Pre-defined hyperparameter sets (`sweep_configs`)
Grid Sweeps	Cartesian product of parameter lists (`sweep.grid`)
Active Config ID	Select a specific named config at runtime
Base Parameters	Default params used when no config is selected
Scenario-Specific	Best params stored for Scenario 1 and Scenario 2

Configuration Files Location

```
configs/
├── model_xgb.yaml      # XGBoost (PRIMARY)
├── model_lgbm.yaml     # LightGBM (SECONDARY)
├── model_cat.yaml      # CatBoost (TERTIARY)
├── model_nn.yaml       # Neural Network (EXPERIMENTAL)
├── model_linear.yaml   # Linear Models (BASELINE)
├── model_hybrid.yaml   # Hybrid Physics+ML
├── data.yaml           # Data loading settings
├── features.yaml       # Feature engineering settings
└── run_defaults.yaml   # Training defaults
```

How to Use Configurations

Method 1: Single Run with Default Parameters

```
python -m src.train --scenario 1 --model xgboost --model-config configs/model_xgb.yaml
```

Method 2: Run a Specific Named Config

```
python -m src.train --scenario 1 --model xgboost --config-id low_lr --model-config
configs/model_xgb.yaml
```

Method 3: Sweep All Named Configs

```
python -m src.train --scenario 1 --model xgboost --sweep --model-config
configs/model_xgb.yaml
```

Method 4: Quick Sweep (First 3 Configs Only)

```
python -m src.train --scenario 1 --model xgboost --sweep --quick-sweep --model-config
configs/model_xgb.yaml
```

Method 5: Full Pipeline (Both Scenarios)

```
python -m src.train --full-pipeline --model xgboost --model-config configs/model_xgb.yaml
```

Method 6: All Models Sweep

```
python -m src.train --scenario 1 --all-models
```

Model Priority & Recommendations

Priority	Model	Use Case	Training Speed	Accuracy
1 (PRIMARY)	XGBoost	Best overall performance	Medium	★★★★★
2 (SECONDARY)	LightGBM	Fast training, ensemble	Fast	★★★★
3 (TERTIARY)	CatBoost	Categorical features, ensemble diversity	Slow	★★★★
4 (BASELINE)	Linear	Baseline, interpretability	Very Fast	★★
5 (EXPERIMENTAL)	Neural Net	Complex patterns	Slow	★★★

---

1 XGBoost Configuration

File: `configs/model_xgb.yaml`
Priority: 1 (PRIMARY MODEL)
Best For: Overall best performance on official metric

Base Parameters

Parameter	Value	Description
`booster`	`gbtree`	Tree-based booster

Parameter	Value	Description
`objective`	`reg:squarederror`	Regression objective
`eval_metric`	`rmse`	Early stopping metric
`max_depth`	6	Maximum tree depth
`min_child_weight`	1	Minimum sum of instance weight
`learning_rate`	0.03	Learning rate (eta)
`n_estimators`	3000	Max iterations (uses early stopping)
`gamma`	0	Minimum loss reduction
`reg_alpha`	0	L1 regularization
`reg_lambda`	1	L2 regularization
`subsample`	0.8	Row subsampling
`colsample_bytree`	0.8	Column subsampling
`early_stopping_rounds`	50	Early stopping patience
`seed`	42	Random seed

Named Sweep Configurations

Config ID	Description	max_depth	learning_rate	n_estimators	reg_lambda	Other
`default`	Default balanced	6	0.03	3000	1	-
`low_lr`	Lower LR, more trees	6	0.02	5000	1	-
`shallow`	Shallow trees	4	0.05	3000	3	-
`deep`	Deeper trees	8	0.02	3000	5	min_child_weight=3
`regularized`	Strong regularization	5	0.03	3000	10	reg_alpha=1, subsample=0.7, colsample_bytree=0.7
`s1_best`	Best for Scenario 1	6	0.03	3000	1	-
`s2_best`	Best for Scenario 2	4	0.05	3000	1	-

Grid Sweep Combinations (48 total)

```yaml
sweep.grid:
  max_depth: [3, 4, 5, 6]          # 4 values
  learning_rate: [0.02, 0.03, 0.05, 0.07]  # 4 values
  reg_lambda: [1, 3, 5]            # 3 values
```

Total Combinations: 4 × 4 × 3 = **48 experiments**

#	max_depth	learning_rate	reg_lambda
1	3	0.02	1
2	3	0.02	3
3	3	0.02	5
4	3	0.03	1
5	3	0.03	3
6	3	0.03	5
7	3	0.05	1
8	3	0.05	3
9	3	0.05	5
10	3	0.07	1
11	3	0.07	3
12	3	0.07	5
13	4	0.02	1
14	4	0.02	3
15	4	0.02	5
16	4	0.03	1
17	4	0.03	3
18	4	0.03	5
19	4	0.05	1
20	4	0.05	3
21	4	0.05	5
22	4	0.07	1
23	4	0.07	3
24	4	0.07	5
25	5	0.02	1
26	5	0.02	3
27	5	0.02	5
28	5	0.03	1
29	5	0.03	3
30	5	0.03	5
31	5	0.05	1
32	5	0.05	3

#	max_depth	learning_rate	reg_lambda
33	5	0.05	5
34	5	0.07	1
35	5	0.07	3
36	5	0.07	5
37	6	0.02	1
38	6	0.02	3
39	6	0.02	5
40	6	0.03	1
41	6	0.03	3
42	6	0.03	5
43	6	0.05	1
44	6	0.05	3
45	6	0.05	5
46	6	0.07	1
47	6	0.07	3
48	6	0.07	5

Sweep Presets

Preset	Parameters	Total Combinations
`fast`	max_depth: [4,5,6,7], learning_rate: [0.02,0.03,0.05,0.07]	16
`full`	max_depth: [3,4,5,6], learning_rate: [0.02,0.03,0.05,0.07], reg_lambda: [1,3,5]	48
`focused`	max_depth: [5,6,7], learning_rate: [0.02,0.03,0.04]	9

Best Known Results

Scenario	max_depth	learning_rate	reg_lambda	Official Metric
Scenario 1	6	0.03	1	0.7499
Scenario 2	4	0.05	1	0.2659

---

2 LightGBM Configuration

File: `configs/model_lgbm.yaml`
Priority: 2 (SECONDARY MODEL)
Best For: Fast training, ensemble with XGBoost

Base Parameters

Parameter	Value	Description

Parameter	Value	Description
`boosting_type`	`gbdt`	Gradient boosting
`objective`	`regression`	Regression task
`metric`	`rmse`	Early stopping metric
`num_leaves`	63	Number of leaves per tree
`max_depth`	-1	No limit
`min_data_in_leaf`	20	Minimum samples in leaf
`learning_rate`	0.05	Learning rate
`n_estimators`	3000	Max iterations
`lambda_l1`	0.0	L1 regularization
`lambda_l2`	0.0	L2 regularization
`feature_fraction`	0.8	Column subsampling
`bagging_fraction`	0.8	Row subsampling
`bagging_freq`	5	Bagging frequency
`early_stopping_rounds`	50	Early stopping patience
`seed`	42	Random seed

Named Sweep Configurations

Config ID	Description	num_leaves	learning_rate	min_data_in_leaf	Other
`default`	Default balanced	63	0.05	20	-
`low_lr`	Lower LR, more trees	63	0.02	20	n_estimators=5000
`high_leaves`	More complex patterns	127	0.03	10	-
`conservative`	Reduce overfitting	31	0.03	40	feature_fraction=0.7, bagging_fraction=0.7
`regularized`	L1/L2 regularization	63	0.05	30	lambda_l1=0.1, lambda_l2=0.1
`s1_best`	Best for Scenario 1	63	0.05	20	-
`s2_best`	Best for Scenario 2	31	0.05	20	-

Grid Sweep Combinations (27 total)

```
sweep.grid:
  num_leaves: [31, 63, 127]          # 3 values
```

```
    learning_rate: [0.02, 0.03, 0.05]    # 3 values
    min_data_in_leaf: [20, 40, 80]       # 3 values
```

Total Combinations: 3 × 3 × 3 = **27 experiments**

#	num_leaves	learning_rate	min_data_in_leaf
1	31	0.02	20
2	31	0.02	40
3	31	0.02	80
4	31	0.03	20
5	31	0.03	40
6	31	0.03	80
7	31	0.05	20
8	31	0.05	40
9	31	0.05	80
10	63	0.02	20
11	63	0.02	40
12	63	0.02	80
13	63	0.03	20
14	63	0.03	40
15	63	0.03	80
16	63	0.05	20
17	63	0.05	40
18	63	0.05	80
19	127	0.02	20
20	127	0.02	40
21	127	0.02	80
22	127	0.03	20
23	127	0.03	40
24	127	0.03	80
25	127	0.05	20
26	127	0.05	40
27	127	0.05	80

Sweep Presets

Preset	Parameters	Total Combinations

Preset	Parameters	Total Combinations
`fast`	num_leaves: [31,63,127], learning_rate: [0.02,0.03,0.05]	9
`full`	num_leaves: [31,63,127], learning_rate: [0.02,0.03,0.05], min_data_in_leaf: [20,40,80]	27
`focused`	num_leaves: [47,63,95], learning_rate: [0.03,0.05,0.07]	9

Best Known Results

Scenario	num_leaves	learning_rate	min_data_in_leaf	Notes
Scenario 1	63	0.05	20	Close to XGBoost
Scenario 2	31	0.05	20	Close to XGBoost

---

3 CatBoost Configuration

File: `configs/model_cat.yaml`
Priority: 3 (TERTIARY MODEL)
Best For: Native categorical handling, ensemble diversity

> ⚠ **Note:** CatBoost consistently underperforms XGBoost on official_metric. Use primarily for ensemble diversity.

Base Parameters

Parameter	Value	Description
`loss_function`	`RMSE`	Loss function
`eval_metric`	`RMSE`	Early stopping metric
`depth`	6	Tree depth
`min_data_in_leaf`	20	Min samples in leaf
`learning_rate`	0.03	Learning rate
`iterations`	3000	Max iterations
`l2_leaf_reg`	3.0	L2 regularization
`random_strength`	1.0	Random strength
`bagging_temperature`	1.0	Bayesian bootstrap
`early_stopping_rounds`	100	Early stopping patience
`random_seed`	42	Random seed

Categorical Features (Native Handling)

```yaml
categorical_features:
  - "ther_area"
  - "main_package"
  - "time_bucket"
  - "hospital_rate_bin"
  - "n_gxs_bin"
```

```
```

Named Sweep Configurations

Config ID	Description	depth	learning_rate	l2_leaf_reg	Other
`default`	Default balanced	6	0.03	3.0	-
`shallow`	Shallow trees	4	0.05	1.0	-
`conservative`	Strong regularization	5	0.02	10.0	random_strength=2.0
`s1_best`	Best for Scenario 1	6	0.03	3.0	-
`s2_best`	Best for Scenario 2	6	0.03	3.0	-

Grid Sweep Combinations (4 total)

```yaml
sweep.grid:
  depth: [4, 6]              # 2 values
  learning_rate: [0.03, 0.05] # 2 values
```

Total Combinations: 2 × 2 = **4 experiments**

#	depth	learning_rate
1	4	0.03
2	4	0.05
3	6	0.03
4	6	0.05

Sweep Presets

Preset	Parameters	Total Combinations
`minimal`	depth: [4,6], learning_rate: [0.03,0.05]	4

---

4 Neural Network Configuration

File: `configs/model_nn.yaml`
Priority: 5 (EXPERIMENTAL)
Framework: PyTorch
Best For: Complex non-linear patterns

Architecture Options

Type	Description
`mlp`	Multi-Layer Perceptron
`tabnet`	TabNet architecture
`transformer`	Transformer-based

Base Parameters

Parameter	Value	Description
`hidden_layers`	[256, 128, 64]	Layer sizes
`activation`	`relu`	Activation function
`dropout`	0.2	Dropout rate
`batch_norm`	true	Batch normalization
`epochs`	100	Training epochs
`batch_size`	256	Batch size
`learning_rate`	0.001	Learning rate
`weight_decay`	0.00001	L2 regularization
`optimizer`	`adam`	Optimizer
`early_stopping.patience`	20	ES patience

Named Sweep Configurations

Config ID	Description	hidden_layers	learning_rate	dropout
`default`	Default configuration	[256, 128, 64]	0.001	0.2
`small`	Smaller, faster	[128, 64]	0.001	0.1
`large`	Larger capacity	[512, 256, 128, 64]	0.0005	0.3
`deep`	Deeper network	[256, 256, 128, 128, 64]	0.0001	0.2

Grid Sweep Combinations (27 total)

```yaml
sweep.grid:
  learning_rate: [0.001, 0.0005, 0.0001]  # 3 values
  hidden_layers_idx: [0, 1, 2]             # 3 values (maps to layer configs)
  dropout: [0.1, 0.2, 0.3]                 # 3 values
```

Total Combinations: 3 × 3 × 3 = **27 experiments**

---

5 Linear Models Configuration

File: `configs/model_linear.yaml`
Priority: 4 (BASELINE)
Best For: Baseline comparison, interpretability

Available Model Types

Type	Description	Key Parameters
`ridge`	Ridge Regression (L2)	alpha

Type	Description	Key Parameters
`lasso`	Lasso Regression (L1)	alpha
`elasticnet`	ElasticNet (L1+L2)	alpha, l1_ratio
`huber`	Huber Regression	epsilon, alpha

Named Sweep Configurations

Config ID	Description	model_type	alpha	l1_ratio	epsilon
`default`	Default Ridge	ridge	1.0	-	-
`ridge_strong`	Strong Ridge	ridge	10.0	-	-
`ridge_weak`	Weak Ridge	ridge	0.1	-	-
`lasso`	Lasso feature selection	lasso	1.0	-	-
`lasso_strong`	Sparse Lasso	lasso	10.0	-	-
`elasticnet`	Balanced ElasticNet	elasticnet	1.0	0.5	-
`elasticnet_l1`	L1-heavy ElasticNet	elasticnet	1.0	0.8	-
`elasticnet_l2`	L2-heavy ElasticNet	elasticnet	1.0	0.2	-
`huber`	Robust to outliers	huber	0.0001	-	1.35

Grid Sweep Combinations (15 total)

```yaml
sweep.grid:
  alpha: [0.01, 0.1, 1.0, 10.0, 100.0]  # 5 values
  model_type: ["ridge", "lasso", "elasticnet"]  # 3 values
```

Total Combinations: 5 × 3 = **15 experiments**

#	model_type	alpha
1	ridge	0.01
2	ridge	0.1
3	ridge	1.0
4	ridge	10.0
5	ridge	100.0
6	lasso	0.01
7	lasso	0.1
8	lasso	1.0
9	lasso	10.0
10	lasso	100.0

#	model_type	alpha
11	elasticnet	0.01
12	elasticnet	0.1
13	elasticnet	1.0
14	elasticnet	10.0
15	elasticnet	100.0

---

6 Hybrid Physics+ML Configuration

File: `configs/model_hybrid.yaml`
Priority: 2 (SECONDARY)
Architecture: Physics baseline + ML residual learning

How It Works

```
final_prediction = physics_baseline + ML_residual

physics_baseline = exp(-decay_rate × months_post_gx)
ML_residual = LightGBM/XGBoost predictions
```

Base Parameters

Parameter	Value	Description
`decay_rate`	0.05	Exponential decay rate
`ml_model.type`	`lightgbm`	ML residual model type
`clip_min`	0.0	Min prediction clip
`clip_max`	2.0	Max prediction clip

Named Sweep Configurations

Config ID	Description	decay_rate	learning_rate	num_leaves	n_estimators
`default`	Default balanced	0.05	0.05	31	500
`slow_decay`	More ML learning	0.03	0.05	63	500
`fast_decay`	Less ML correction	0.10	0.03	31	500
`ml_heavy`	More ML capacity	0.05	0.07	63	1000

Grid Sweep Combinations (24 total)

```yaml
sweep.grid:
  decay_rate: [0.03, 0.05, 0.07, 0.10]   # 4 values
  learning_rate: [0.03, 0.05, 0.07]      # 3 values
  num_leaves: [31, 63]                    # 2 values
```

Total Combinations: 4 × 3 × 2 = **24 experiments**

Grid Sweep Combinations Summary

Model	Named Configs	Grid Combinations	Total Experiments
XGBoost	7	48	55
LightGBM	7	27	34
CatBoost	5	4	9
Neural Network	4	27	31
Linear Models	9	15	24
Hybrid	4	24	28
TOTAL	**36**	**145**	**181**

Per Scenario (×2)

Since each configuration can be run on Scenario 1 and Scenario 2:

Total Named Configs	Total Grid Combos	Grand Total Experiments
72	290	**362**

CLI Command Reference

Basic Commands

```
# Train with default parameters
python -m src.train --scenario 1 --model xgboost --model-config configs/model_xgb.yaml

# Train with specific named config
python -m src.train --scenario 1 --model xgboost --config-id low_lr --model-config
configs/model_xgb.yaml

# Sweep all named configs
python -m src.train --scenario 1 --model xgboost --sweep --model-config
configs/model_xgb.yaml

# Quick sweep (first 3 only)
python -m src.train --scenario 1 --model xgboost --sweep --quick-sweep --model-config
configs/model_xgb.yaml
```

Full Pipeline Commands

```
# Train both scenarios
python -m src.train --full-pipeline --model xgboost --model-config configs/model_xgb.yaml
```

```
# Train both scenarios in parallel
python -m src.train --full-pipeline --model xgboost --parallel --model-config
configs/model_xgb.yaml
```

Multi-Model Commands

```
# Sweep all models (XGBoost + LightGBM)
python -m src.train --scenario 1 --all-models

# Quick sweep all models
python -m src.train --scenario 1 --all-models --quick-sweep
```

Cross-Validation Commands

```
# 5-fold CV
python -m src.train --scenario 1 --model xgboost --cv --n-folds 5 --model-config
configs/model_xgb.yaml

# Sweep with CV
python -m src.train --scenario 1 --model xgboost --sweep-cv --n-folds 3 --model-config
configs/model_xgb.yaml
```

Hyperparameter Optimization (Optuna)

```
# Run HPO with 50 trials
python -m src.train --scenario 1 --model xgboost --hpo --hpo-trials 50 --model-config
configs/model_xgb.yaml

# HPO with timeout
python -m src.train --scenario 1 --model xgboost --hpo --hpo-trials 100 --hpo-timeout
3600 --model-config configs/model_xgb.yaml
```

All Models Quick Reference

Model	CLI Flag	Config File
XGBoost	`--model xgboost`	`configs/model_xgb.yaml`
LightGBM	`--model lightgbm`	`configs/model_lgbm.yaml`
CatBoost	`--model catboost`	`configs/model_cat.yaml`
Neural Network	`--model nn`	`configs/model_nn.yaml`
Linear	`--model linear`	`configs/model_linear.yaml`
Hybrid	`--model hybrid`	`configs/model_hybrid.yaml`

Model	CLI Flag	Config File
ARIHOW	`--model arihow`	`configs/model_arihow.yaml`
LSTM	`--model lstm`	`configs/model_lstm.yaml`
CNN-LSTM	`--model cnn_lstm`	`configs/model_cnn_lstm.yaml`
KG-GCN-LSTM	`--model kg_gcn_lstm`	`configs/model_kg_gcn_lstm.yaml`
Flat	`--model flat`	-
Trend	`--model trend`	-
Global Mean	`--model global_mean`	-
Historical Curve	`--model historical_curve`	-

---

7 ARIHOW Configuration (SARIMAX + Holt-Winters)

File: `configs/model_arihow.yaml`
Priority: 4 (Specialized)
Best For: Brands with 12+ months of historical data

Architecture

```
ARIHOW = β × ARIMA + (1-β) × Holt-Winters
- ARIMA captures trend and autocorrelation
- Holt-Winters captures level and trend with exponential smoothing
- β is learned via grid search or optimization
- Falls back to exponential decay for brands with insufficient history
```

Base Parameters

Component	Parameter	Value	Description
ARIMA	`order`	[1, 1, 1]	(p, d, q) - AR, differencing, MA
ARIMA	`seasonal_order`	[1, 0, 1, 12]	(P, D, Q, s) - SARIMAX seasonal
ARIMA	`trend`	'c'	Constant trend
Holt-Winters	`trend`	'add'	Additive trend
Holt-Winters	`seasonal`	null	No seasonal (in ARIMA)
Holt-Winters	`damped_trend`	true	Damped trend
Weights	`initial_beta`	0.5	Starting ARIMA weight
Weights	`method`	'grid'	Grid search for β
Fallback	`min_history_months`	12	Minimum history required
Fallback	`decay_rate`	0.02	Exponential decay fallback

Named Sweep Configurations

Config ID	Description	ARIMA Weight	Notes
`default`	Balanced ARIMA/HW	0.5	Grid search optimization
`arima_heavy`	More ARIMA weight	0.7	Better for trending
`hw_heavy`	More Holt-Winters	0.3	Better for level shifts

CLI Examples

```
python -m src.train --scenario 1 --model arihow --model-config configs/model_arihow.yaml
python -m src.train --scenario 1 --model arihow --config-id arima_heavy --model-config
configs/model_arihow.yaml
```

---

8 LSTM Configuration

File: `configs/model_lstm.yaml`
Priority: 5 (Experimental)
Framework: PyTorch
Best For: Sequential patterns, ablation study vs CNN-LSTM

Base Parameters

Parameter	Value	Description
`lstm_hidden_dim`	128	Hidden state dimension
`lstm_num_layers`	2	Number of LSTM layers
`bidirectional`	false	Bidirectional LSTM
`dropout`	0.2	Dropout rate
`learning_rate`	0.001	Learning rate
`epochs`	100	Training epochs
`batch_size`	64	Batch size

Named Sweep Configurations

Config ID	Description	hidden_dim	num_layers	bidirectional
`default`	Default LSTM	128	2	false
`small`	Smaller, faster	64	1	false
`large`	Larger capacity	256	3	false
`bidirectional`	Bidirectional	128	2	true

Grid Sweep Combinations (81 total)

```
sweep.grid:
  lstm_hidden_dim: [64, 128, 256]      # 3 values
  lstm_num_layers: [1, 2, 3]           # 3 values
  learning_rate: [0.001, 0.0005, 0.0001]  # 3 values
  dropout: [0.1, 0.2, 0.3]             # 3 values
```

Total Combinations: 3 × 3 × 3 × 3 = **81 experiments**

CLI Examples

```
python -m src.train --scenario 1 --model lstm --model-config configs/model_lstm.yaml
python -m src.train --scenario 1 --model lstm --config-id bidirectional --model-config
configs/model_lstm.yaml
```

---

9 CNN-LSTM Configuration

File: `configs/model_cnn_lstm.yaml`
Priority: 4 (Experimental)
Framework: PyTorch
Best For: Local feature extraction + temporal dependencies

Architecture

```
Input → CNN (local features) → LSTM (temporal) → Dense → Output

1. CNN: Extracts local patterns from feature windows
2. LSTM: Captures temporal dependencies
3. Fusion: Combines representations
```

Base Parameters

Component	Parameter	Value	Description
CNN	`filters`	[32, 64]	Conv filter counts
CNN	`kernel_size`	3	Convolution kernel
CNN	`pool_size`	2	Max pooling size
LSTM	`hidden_dim`	64	LSTM hidden size
LSTM	`num_layers`	1	LSTM layers
Training	`learning_rate`	0.001	Learning rate
Training	`dropout`	0.2	Dropout rate

Named Sweep Configurations

Config ID	Description	CNN Filters	LSTM Hidden	Dropout
`default`	Default balanced	[32, 64]	64	0.2
`small`	Smaller, faster	[16, 32]	32	0.1
`large`	Larger capacity	[64, 128]	128	0.3
`deep_cnn`	Deeper CNN	[32, 64, 128]	64	0.2

Grid Sweep Combinations (27 total)

```
sweep.grid:
  lstm_hidden_dim: [32, 64, 128]           # 3 values
  learning_rate: [0.001, 0.0005, 0.0001]   # 3 values
  dropout: [0.1, 0.2, 0.3]                 # 3 values
```

CLI Examples

```
python -m src.train --scenario 1 --model cnn_lstm --model-config
configs/model_cnn_lstm.yaml
python -m src.train --scenario 1 --model cnn_lstm --config-id large --model-config
configs/model_cnn_lstm.yaml
```

🔟 Baseline Models

File: `src/models/baselines.py`
Purpose: Sanity checks, lower bounds for model performance

Available Baselines

Model	CLI Flag	Formula	Use Case
Naive Persistence (Flat)	`--model flat`	`vol = avg_vol`	No change baseline
Global Mean	`--model global_mean`	`vol = mean(all)`	Global average
Trend	`--model trend`	Linear trend	Simple trend
Historical Curve	`--model historical_curve`	Historical pattern	Use past patterns
Linear Decay	`baselines.py`	`vol = avg × (1 - rate × t)`	Linear erosion
Exponential Decay	`baselines.py`	`vol = avg × exp(-rate × t)`	Exponential erosion

Exponential Decay

```python
from src.models.baselines import BaselineModels

# Generate exponential decay predictions
predictions = BaselineModels.exponential_decay(
    avg_j_df=avg_volumes,         # DataFrame with avg volumes
    months_to_predict=[0, 1, ..., 23],
    decay_rate=0.05,              # 5% monthly decay
    volume_col='avg_vol'
)

# Tune decay rate on validation data
best_rate, results = BaselineModels.tune_decay_rate(
    actual_df=val_data,
    avg_j_df=avg_volumes,
    decay_type='exponential',
    decay_rates=[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.10]
)
```

Linear Decay

```python
# Generate linear decay predictions
predictions = BaselineModels.linear_decay(
    avg_j_df=avg_volumes,
    months_to_predict=[0, 1, ..., 23],
    decay_rate=0.03,              # 3% monthly decay
    volume_col='avg_vol'
)
# Formula: volume = avg_vol × max(0, 1 - decay_rate × month)
```

CLI Examples for Baselines

```bash
python -m src.train --scenario 1 --model flat
python -m src.train --scenario 1 --model trend
python -m src.train --scenario 1 --model global_mean
python -m src.train --scenario 1 --model historical_curve
```

Best Practices

1. Start with Named Configs

```bash
# Test the best known configs first
python -m src.train --scenario 1 --model xgboost --config-id s1_best --model-config configs/model_xgb.yaml
python -m src.train --scenario 2 --model xgboost --config-id s2_best --model-config configs/model_xgb.yaml
```

2. Use Quick Sweep for Exploration

```
# Quick test of first 3 configs
python -m src.train --scenario 1 --model xgboost --sweep --quick-sweep --model-config
configs/model_xgb.yaml
```

3. Full Sweep for Final Tuning

```
# Complete sweep when you have time
python -m src.train --scenario 1 --model xgboost --sweep --model-config
configs/model_xgb.yaml
```

4. Model Priority Order

1. **XGBoost** - Best overall, start here
2. **LightGBM** - Fast, good for ensemble
3. **CatBoost** - Best Scenario 2 accuracy
4. **Hybrid** - Physics-informed, interpretable
5. **Linear** - Baseline comparison

5. Scenario-Specific Recommendations

Scenario	Best Model	Best Config	Official Metric
Scenario 1 (no actuals)	XGBoost	`s1_best`	0.7671
Scenario 2 (6-month actuals)	CatBoost	`s2_best`	0.2742

6. Ensemble Strategy

```
# Train all 3 boosting models
python -m src.train --full-pipeline --model xgboost --model-config configs/model_xgb.yaml
python -m src.train --full-pipeline --model lightgbm --model-config
configs/model_lgbm.yaml
python -m src.train --full-pipeline --model catboost --model-config
configs/model_cat.yaml

# Then blend predictions with learned weights
```

Appendix: Full Parameter Reference

XGBoost All Parameters

```yaml
params:
  booster: "gbtree"
  objective: "reg:squarederror"
  eval_metric: "rmse"
  max_depth: 6
  min_child_weight: 1
  max_leaves: 0
  learning_rate: 0.03
  n_estimators: 3000
  gamma: 0
  reg_alpha: 0
  reg_lambda: 1
  subsample: 0.8
  colsample_bytree: 0.8
  colsample_bylevel: 1.0
  colsample_bynode: 1.0
  early_stopping_rounds: 50
  verbosity: 0
  n_jobs: -1
  seed: 42
```

LightGBM All Parameters

```yaml
params:
  boosting_type: "gbdt"
  objective: "regression"
  metric: "rmse"
  num_leaves: 63
  max_depth: -1
  min_data_in_leaf: 20
  min_sum_hessian_in_leaf: 0.001
  learning_rate: 0.05
  n_estimators: 3000
  lambda_l1: 0.0
  lambda_l2: 0.0
  feature_fraction: 0.8
  bagging_fraction: 0.8
  bagging_freq: 5
  cat_smooth: 10
  early_stopping_rounds: 50
  verbose: -1
  n_jobs: -1
  seed: 42
```

CatBoost All Parameters

```yaml
params:
  loss_function: "RMSE"
  eval_metric: "RMSE"
  depth: 6
```

```
    min_data_in_leaf: 20
    learning_rate: 0.03
    iterations: 3000
    l2_leaf_reg: 3.0
    random_strength: 1.0
    bagging_temperature: 1.0
    early_stopping_rounds: 100
    random_seed: 42
    verbose: 100
    thread_count: -1
```

ARIHOW All Parameters

```
arima:
  order: [1, 1, 1]
  seasonal_order: [1, 0, 1, 12]
  trend: 'c'
  enforce_stationarity: false
  enforce_invertibility: false

holt_winters:
  trend: 'add'
  seasonal: null
  seasonal_periods: 12
  damped_trend: true
  initialization_method: 'estimated'

weights:
  initial_beta: 0.5
  method: 'grid'
  grid_values: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

fallback:
  min_history_months: 12
  decay_rate: 0.02
```

Hybrid All Parameters

```
physics:
  decay_rate: 0.05
  scenario_decay_rates:
    scenario1: 0.05
    scenario2: 0.05

ml_model:
  type: "lightgbm"  # or "xgboost"

  lightgbm:
    num_leaves: 31
    learning_rate: 0.05
    n_estimators: 500
```

```yaml
    xgboost:
      max_depth: 5
      learning_rate: 0.05
      n_estimators: 500
```

Document generated for Novartis Datathon 2025
Last updated: November 30, 2025