



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Machine Learning System Service for Earth Orientation Parameters Forecast

A Machine Learning System that predicts the Earth Orientation parameters

Master's thesis in Space, earth and environment

Sudeshna Lama

MASTER'S THESIS 2026

Machine Learning System Service for Earth Orientation Parameters Forecast

A Machine Learning System that predicts the Earth Orientation
parameters

SUDESHNA LAMA



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Gothenburg, Sweden 2026

Machine Learning System Service for Earth Orientation Parameters Forecast
A Machine Learning System that predicts the Earth Orientation parameters
SUDESHNA LAMA

© SUDESHNA LAMA, 2026.

Supervisor: Rüdiger Haas
Full Professor, Onsala Space Observatory
Department of Physics and Astronomy

Examiner: Rüdiger Haas
Full Professor, Onsala Space Observatory
Department of Physics and Astronomy

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in L^AT_EX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Machine Learning System Service for Earth Orientation Parameters Forecast
A Machine Learning System that predicts the Earth Orientation parameters
SUDESHNA LAMA
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Earth Orientation Parameters (EOP) polar motion (x_p, y_p), the difference between astronomically observed and atomic time ($UT1 - UTC$), length-of-day excess (LOD), and celestial pole offsets (dX, dY) describe irregularities in the Earth’s rotation that are essential inputs to satellite navigation, precise geodesy, and reference-frame transformations, yet are difficult to predict because they arise from the coupled, only partially observable exchange of angular momentum between the solid Earth, the atmosphere, and the oceans. This thesis presents the design, implementation, and evaluation of an end-to-end software platform that forecasts EOP using machine learning, framed primarily as a contribution to Computer Systems and Networks: the scientific forecasting problem motivates the system, but the central contribution is the architecture and engineering of a reproducible, modular, testable platform that carries a forecast from raw data acquisition through to a served, visualized prediction.

The platform ingests real International Earth Rotation and Reference Systems Service (IERS) EOP bulletins and real atmospheric/oceanic angular momentum data (AAM/OAM) from the GFZ Potsdam Earth System Modelling group, alongside a reproducible synthetic data generator used for offline engineering validation. Data are cleaned via robust, Median-Absolute-Deviation-based outlier detection and time-aware interpolation onto a strict daily calendar, merged with AAM drivers, and transformed into a supervised-learning representation using lag features, rolling statistics, and calendar encodings, with an explicit `shift(1)` guard against target leakage. Five model families – Ridge regression, Random Forest, XGBoost, an LSTM network with an additive attention mechanism, and a Split Conformal Prediction wrapper for uncertainty quantification – are trained independently for each of six EOP targets at four forecast horizons (7, 14, 30, and 365 days) using a strict chronological train/calibration/test split, and served through a FastAPI inference layer and a browser-based visualization interface.

The complete Targets $\times$ Horizons $\times$ Models experimental grid (120 trained artifacts)

was evaluated on the reproducible synthetic dataset, and is reported and analyzed as such; a smaller, targeted subset was additionally trained and evaluated directly on real IERS/AAM data to characterize the gap between synthetic-data and real-data performance. Results indicate that model accuracy degrades with forecast horizon for most targets, that no single model family is uniformly best across all targets, and that the implemented conformal prediction intervals achieve empirical coverage close to their 90% nominal target for most target/ horizon combinations but substantially undershoot it for others – evidence directly attributable to the non-stationarity of EOP series violating conformal prediction’s exchangeability assumption. The real-data subset shows that this gap is target-dependent: real-data forecast error for length-of-day was several times higher than on synthetic data, while polar-motion forecasts were comparable or better on real data. The thesis concludes that the platform successfully demonstrates an operational, reproducible, and honestly evaluated machine-learning forecasting service for EOP, while identifying concrete limitations – most importantly the synthetic-data scope of the full experimental grid – that bound the strength of its scientific claims.

Note on evidentiary status. All quantitative results reported in this thesis are drawn directly from experiment artifacts (generated figures, leaderboard tables, and training logs) produced by the implemented system; no numerical result is invented. Where a specific value could not be recovered from the available project records, it is marked **[TO BE VERIFIED FROM EXPERIMENTAL RECORDS]** rather than estimated.

Keywords: Earth Orientation Parameters, polar motion, UT1-UTC, length of day, celestial pole offsets, atmospheric angular momentum, machine learning, time-series forecasting, Ridge regression, Random Forest, XGBoost, long short-term memory, attention mechanism, conformal prediction, FastAPI, software architecture.

Acknowledgements

I would like to thank my thesis supervisor, Mr. Rüdiger Haas, for his guidance, feedback, and patience throughout the design, implementation, and evaluation of this project.

I also acknowledge the International Earth Rotation and Reference Systems Service (IERS) and the GFZ Potsdam Earth System Modelling group, whose openly published Earth Orientation Parameter and atmospheric/oceanic angular momentum data this thesis's real-data evaluation (Section 7.7) depends on.

Finally, I thank my family and friends for their encouragement and support throughout this degree.

Sudeshna Lama, Gothenburg, 2026-10-05

Contents

List of Figures	xv
List of Figures	xv
List of Tables	xvii
List of Tables	xvii
List of Abbreviations	xix
1 Introduction and Theoretical Framework	1
1.1 Background	1
1.2 Earth Orientation Parameters	1
1.3 Relevance and the Difficulty of Forecasting	2
1.4 Motivation	3
1.5 Theoretical Foundations	3
1.6 From Scientific Problem to Computational Problem	4
2 Statement of the Problem	5
2.1 The Problem	5
2.2 Scope	5
3 Purpose of the Study	7
4 Review of the Literature	9
4.1 Earth Orientation Theory and Observation	9
4.2 Physical Excitation Mechanisms	9
4.3 Classical EOP Prediction	10
4.4 Machine Learning for EOP and Time-Series Forecasting	10
4.5 Uncertainty Quantification	10

4.6	Time-Series Evaluation Practice	11
4.7	Summary and Research Gap	11
5	Research Questions	13
6	System Design, Methods and Procedures	15
6.1	Requirements	15
6.2	Architecture	15
6.3	Data Sources and Pipeline	16
6.4	Data Cleaning	16
6.5	Feature Engineering	17
6.6	Machine-Learning Models	18
6.6.1	Ridge Regression	18
6.6.2	Random Forest	18
6.6.3	XGBoost	18
6.6.4	LSTM with Attention	19
6.6.5	Split Conformal Prediction	19
6.7	Training Strategy and Evaluation	19
6.8	Model Persistence, Inference, and the API	20
6.9	Frontend	20
6.10	Testing	21
7	Experimental Results	23
7.1	Scope of the Reported Results	23
7.2	Model Comparison Across Targets and Horizons	23
7.3	Error Growth with Forecast Horizon	27
7.4	Conformal Prediction Coverage	29
7.5	Computational Cost	29
7.6	Actual versus Predicted	30
7.7	Real-Data Validation Subset	31
8	Discussion	33
8.1	Answering the Research Questions	33
8.2	Why These Results Occur	33
8.3	Relation to Prior Work	34
8.4	Trade-offs	34
9	Limitations and Delimitations	35
9.1	Limitations	35

9.2 Delimitations	35
10 Significance of the Study	37
11 Conclusion and Future Work	39
11.1 Conclusion	39
11.2 Future Work	39
Code Availability	41
Bibliography	43
Appendix A API Reference	I
Appendix B Configuration Reference	III
Appendix C Test Inventory	V
Appendix D Traceability Matrix	VII

List of Figures

7.1	MAE, RMSE, and R^2 for all five models at the 7-day horizon, by target. MAE/RMSE units are target-dependent (mas for dX/dY , ms for LOD, s for $UT1-UTC$, arcsec for x_p/y_p); R^2 is dimensionless.	24
7.2	MAE across the full target $\times$ horizon grid, per model. Values are in each target's native unit (mas for dX/dY , ms for LOD, s for $UT1-UTC$, arcsec for x_p/y_p); the color scale is shared across all five panels and is therefore only directly comparable within, not across, these unit groups.	26
7.3	MAE versus forecast horizon, per target, with per-target units.	28
7.4	Empirical conformal interval coverage vs. 90% nominal target, all evaluated combinations. The x-axis is grouped by target (bold labels below each six-bar cluster) with the forecast horizon labeled above, in place of 24 individually rotated labels.	29
7.5	Mean training time per model, averaged across all trained target/horizon combinations (seconds).	30
7.6	Actual vs. predicted values on the held-out test split, selected target/horizon combinations.	30
11.1	Local webpage of the server in the browser	42

List of Tables

6.1	Thesis chapter/topic to implementation module mapping.	16
7.1	Best-performing model per target/horizon by test MAE (excerpt from the full leaderboard).	27
7.2	Test MAE (native per-target unit): same model, same target/horizon, trained on real data vs. synthetic data.	31
A.1	Implemented API endpoints.	I
B.1	Key configuration constants.	III
C.1	Automated test suite, by file.	V
D.1	Thesis section to source-code module mapping.	VII

List of Abbreviations

AAM	Atmospheric Angular Momentum
API	Application Programming Interface
BIPM	International Bureau of Weights and Measures (<i>Bureau International des Poids et Mesures</i>)
CORS	Cross-Origin Resource Sharing
CSN	Computer Systems and Networks
dX, dY	Celestial Pole Offsets (X and Y components)
EOP	Earth Orientation Parameter(s)
FCN	Free Core Nutation
GNSS	Global Navigation Satellite System
GFZ	German Research Centre for Geosciences (<i>Deutsches GeoForschungsZentrum</i>)
HTTP	Hypertext Transfer Protocol
IAU	International Astronomical Union
IERS	International Earth Rotation and Reference Systems Service
JSON	JavaScript Object Notation
LOD	Length of Day (excess)
LSTM	Long Short-Term Memory
MAD	Median Absolute Deviation
MAE	Mean Absolute Error

MAPE	Mean Absolute Percentage Error
MJD	Modified Julian Date
OAM	Oceanic Angular Momentum
R²	Coefficient of Determination
REST	Representational State Transfer
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
UT1	Universal Time 1 (observed Earth rotation angle time scale)
UTC	Coordinated Universal Time
VLBI	Very Long Baseline Interferometry
x_p, y_p	Polar Motion Coordinates (X and Y components)

1

Introduction and Theoretical Framework

1.1 Background

The Earth’s rotation is not perfectly uniform: both the orientation of its rotation axis and its rotation rate vary continuously and irregularly. These variations are described by a small set of quantities, Earth Orientation Parameters (EOP), that relate an Earth-fixed terrestrial reference frame to a space-fixed celestial frame at a given epoch [1]. EOP arise from the coupled, only partially observable exchange of angular momentum between the solid Earth, the atmosphere, the oceans, and (on longer timescales) the fluid core [2], which makes them only partially predictable from physical models alone. This partial predictability, combined with the practical importance of EOP described in Section 1.3, motivates this thesis: whether machine-learning methods, embedded in a properly engineered software platform, can provide a useful complementary forecasting approach.

1.2 Earth Orientation Parameters

The IERS Conventions define polar motion (x_p, y_p), $UT1 - UTC$, and the celestial pole offsets (dX, dY) [1]; length-of-day excess (LOD), the time derivative of $UT1 - UTC$, is reported alongside them. This project forecasts all six – implemented as `x_pole`, `y_pole`, `ut1_utc`, `lod`, `dX`, `dY` in `src/utils/config.py`.

Polar motion (x_p, y_p) is dominated by two quasi-periodic components: the Chandler wobble, a free oscillation with a period of approximately 433 days that persists only because it is continually re-excited (predominantly by atmospheric and oceanic angular momentum fluctuations), and an annual wobble forced directly by

seasonal mass and wind redistribution. Their superposition produces a slow, well-documented beat pattern in the combined amplitude [2].

$UT1 - UTC$ combines a long-term secular trend (dominated by tidal deceleration, partly offset by core-mantle and post-glacial-rebound effects) with shorter-term irregular variation driven by the same angular-momentum exchange that excites polar motion. Because leap seconds keep $|UT1 - UTC|$ bounded to 0.9s, the raw series exhibits a characteristic sawtooth pattern with direct consequences for this project's data pipeline (Chapter 7).

Length of day (LOD) excess reflects the same total-angular-momentum conservation: an increase in atmospheric or oceanic angular momentum is compensated by a change in the solid Earth's rotation rate [2]. This coupling is strong enough that atmospheric angular momentum (AAM) explains most sub-seasonal LOD variance, making LOD the target for which the AAM features used in this project (Section 4.2) are most directly informative.

Celestial pole offsets (dX, dY) are residuals between the observed celestial pole position and the adopted IAU precession-nutation model, dominated by Free Core Nutation, a free oscillation of the fluid core with a period near 430 days that the adopted model does not fully capture [1]. Consistent with this, the real IERS data used in this project shows systematically larger dX/dY residuals before the IAU 2000/2006 model was adopted – a reference-model artifact rather than a physical change, verified directly in Chapter 7.

1.3 Relevance and the Difficulty of Forecasting

EOP are a required input wherever a coordinate must be transformed between terrestrial and celestial frames, including GNSS processing, satellite orbit determination, and VLBI geodesy. Because final EOP values are only available weeks to months after the fact, real-time applications depend on *predicted* values; the IERS explicitly distinguishes final (Bulletin B) from predicted (Bulletin A) products for this reason [1].

Forecasting EOP is difficult for three reasons: they are **non-stationary** (LOD and $UT1 - UTC$ carry secular trends, and Chandler-wobble amplitude drifts over years); they are driven by **multiple, only partially observed** physical subsystems, so no single predictor fully determines future values; and their dominant periodic signals are only **quasi**-periodic, so harmonic extrapolation degrades over long hori-

zons. Classical operational approaches (harmonic analysis, autoregressive models, Kalman-filter combination) address this well but generally assume a fixed, low-order functional form and are not typically built as modular, testable, API-accessible software platforms. This thesis does not claim the implemented machine-learning models are inherently superior – Chapter 7 shows a simple linear model is often competitive with more complex ones – but that a machine-learning platform offers a natural architecture for systematically comparing modeling paradigms within one reproducible system.

1.4 Motivation

This thesis has a twofold motivation. The *scientific* motivation is to compare, empirically, several machine-learning paradigms (regularized linear regression, bagged trees, boosted trees, an attention-augmented recurrent network) on the same EOP forecasting problem under a consistent, leakage-free protocol, and to evaluate whether split conformal prediction can supply useful calibrated intervals. The *engineering* motivation – carrying particular weight for a Computer Systems and Networks thesis – is to design a platform in which data acquisition, cleaning, feature engineering, training, evaluation, persistence, and network-accessible inference are first-class, modular, testable components. Chapter 6 treats these as inseparable: the model comparison is meaningful only because it runs inside a reproducible system, and the system is interesting only because it serves a genuinely difficult problem.

1.5 Theoretical Foundations

This section briefly introduces the concepts used throughout; full mathematical treatment, tied to the exact implemented hyperparameters, follows in Chapter 6.

The problem is framed as **supervised regression**: given a feature vector built from information available up to day t , predict the target’s value h days later. Because time-series samples are not independent (temporal autocorrelation, seasonality), evaluation must use a **chronological**, not random, split, and lag/rolling features must strictly exclude information at or after time t – a random split or a leaking feature would let future information contaminate training. **Feature engineering** for time series typically adds lagged values, rolling summary statistics, and calendar encodings so a model has explicit access to known structure rather than needing to rediscover it.

Ensemble learning combines many simple base learners (trees). Bagging, underlying Random Forest [3], averages independently trained trees to reduce variance; boosting, underlying XGBoost [4], trains trees sequentially to correct the ensemble’s residual error, reducing bias.

Recurrent neural networks process a sequence while maintaining a hidden state, but plain RNNs suffer vanishing/exploding gradients over long sequences. **LSTM** networks [5] address this with a memory cell and gates that regulate what is written, kept, or read at each step. **Attention** [6] lets a model form a weighted combination of *all* hidden states rather than only the final one, with weights learned to emphasize whichever past steps matter most for the current prediction – the architecture this project implements over raw driver sequences (Chapter 6).

Conformal prediction constructs prediction intervals with a finite-sample marginal coverage guarantee from only an *exchangeability* assumption between calibration and test data, without assuming an error distribution [7]. Split conformal prediction, implemented here, holds out a calibration set, computes nonconformity scores, and adds a corrected empirical quantile of those scores as a fixed margin around a new point prediction [8]. This guarantee is conditional on exchangeability; for a non-stationary process such as EOP it can be violated, and observed coverage can undershoot the nominal target as a direct, expected consequence rather than an implementation error – revisited critically against real results in Chapter 7.

1.6 From Scientific Problem to Computational Problem

The physical picture above defines a forecasting problem, not yet a software system. Turning it into a reproducible, testable, network-accessible platform requires further engineering decisions: how heterogeneous raw data is acquired and validated; how a leakage-free supervised dataset is built from a physical time series; how multiple model families are trained, evaluated, and persisted consistently; how a trained model is served without retraining on every request; and how forecasts are exposed both programmatically and visually. Chapter 2 formalizes this transition, and Chapter 6 describes, module by module, how the implemented system addresses it.

2

Statement of the Problem

2.1 The Problem

EOP are time-dependent geophysical variables that are non-stationary, seasonal, oscillatory, noisy, driven by multiple physical processes, and subject to missing or delayed observations (Chapter 1). Classical harmonic and autoregressive prediction methods handle several of these properties well in operational settings, but generally assume a fixed, low-order functional form and are not usually built as general-purpose software platforms that support multiple targets, multiple horizons, multiple model families, programmatic access, and automated testing at once. This creates two related problems that this thesis addresses together.

The computational problem: given historical EOP observations and atmospheric/oceanic angular momentum (AAM) data, construct a machine-learning system capable of producing forecasts at several future horizons, together with a rigorous evaluation methodology and, where implemented, calibrated uncertainty estimates.

The system problem: how can this forecasting capability be implemented as a reproducible, modular, testable, and network-accessible software platform, rather than as a one-off modeling script?

2.2 Scope

This thesis addresses both problems through a single implemented system (described in full in Chapter 6): a data pipeline that ingests real IERS EOP bulletins and real GFZ AAM/OAM data (with a synthetic generator for offline engineering validation), a feature engineering stage, five model families trained across six EOP targets and four forecast horizons, a FastAPI inference service, and a browser-based visualization

2. Statement of the Problem

interface, all covered by an automated test suite. The scope is deliberately bounded to what is actually implemented; Section 9.2 states explicitly what the system does not attempt.

3

Purpose of the Study

The **theoretical purpose** of this study is to investigate the suitability of several distinct machine-learning paradigms – regularized linear regression, bagged trees, boosted trees, and an attention-augmented recurrent network – for nonlinear, temporally dependent EOP forecasting, and to evaluate whether split conformal prediction can supply empirically useful uncertainty estimates for this problem.

The **practical purpose** is to produce an operational prototype that integrates data processing, machine learning, uncertainty estimation, inference, visualization, and testing into one platform. The goal is not to identify the single model with the smallest error in isolation, but to investigate how machine-learning forecasting can be integrated into a complete, usable computational system – consistent with the Computer Systems and Networks framing of this thesis (Chapter 1).

3. Purpose of the Study

4

Review of the Literature

4.1 Earth Orientation Theory and Observation

The physical definitions of polar motion, $UT1 - UTC$, LOD, and celestial pole offsets, and the conventions for transforming between terrestrial and celestial reference frames, are codified authoritatively in the IERS Conventions [1]. Operational EOP products are derived from a combination of techniques, principally VLBI (for $UT1 - UTC$ and celestial pole offsets) and satellite geodetic techniques, and are distributed by the IERS as both final, fully reconciled series (e.g. the EOP 14 C04 combined solution, [9]) and as rapid/predicted values (Bulletin A) for near-real-time use [1]. This project's real-data pipeline consumes exactly this finals/rapid combined product (Chapter 6).

4.2 Physical Excitation Mechanisms

Barnes et al. [2] established the now-standard framework in which atmospheric (and, by extension, oceanic) angular momentum is decomposed into mass and motion terms whose equatorial components excite polar motion and whose axial component excites LOD. This framework directly motivates including AAM/OAM data as model features in this project (Section 1.2), rather than treating EOP as a univariate series to be extrapolated from its own history alone. Core-mantle coupling and tidal braking additionally contribute to the long-term secular trends in $UT1 - UTC$ /LOD noted in Chapter 1, though these are slower processes than this project's forecast horizons primarily target.

4.3 Classical EOP Prediction

Operational and academic EOP prediction has long combined harmonic modeling of the Chandler and annual wobbles with autoregressive or autocovariance-based extrapolation of the residual [10]. The IERS Earth Orientation Parameters Prediction Comparison Campaign systematically compared such methods across contributing groups and forecast horizons, establishing empirical baselines for achievable short-, medium-, and long-term prediction accuracy [11]. This campaign is a natural point of reference for interpreting the horizon-dependent accuracy degradation observed in this project’s own results (Chapter 7), although a direct numerical comparison is out of scope here, since the campaign’s submissions used real operational data whereas this project’s full experimental grid uses synthetic data (Chapter 7 explains this scope decision in full).

4.4 Machine Learning for EOP and Time-Series Forecasting

More recently, deep-learning approaches have been applied directly to EOP forecasting: Guessoum et al. [12] used a one-dimensional convolutional network with AAM input to predict LOD at short horizons, reporting improvements over classical detrend-and-extrapolate baselines. This and similar work establish that nonlinear, data-driven models can be competitive for EOP forecasting, but – to the knowledge gathered for this review – such work typically evaluates a single model family against a single target, rather than systematically comparing multiple model families (linear, bagged-tree, boosted-tree, and attention-based recurrent) across every EOP target and multiple horizons within one common, reproducible software platform, which is the specific gap this thesis addresses. The general machine-learning methods used – regularized linear regression, bagging [3], boosting [4], and LSTM networks with attention [5, 6] – are each well established outside the EOP domain specifically (Chapter 1).

4.5 Uncertainty Quantification

Point forecasts alone do not communicate reliability. Conformal prediction [7] and its computationally cheap split variant [8] provide distribution-free prediction intervals under an exchangeability assumption, but their application to genuinely non-

stationary geophysical time series – as opposed to i.i.d. or weakly dependent data – remains comparatively under-explored in the EOP literature specifically; this thesis’s evaluation of conformal coverage against a known-non-stationary target (Chapter 7) is offered as a direct, empirical contribution on this point.

4.6 Time-Series Evaluation Practice

Sound evaluation of time-series models requires a chronological, not random, train/test split to avoid leaking future information into training, and the reporting of standard error metrics (MAE, RMSE, MAPE, R^2) alongside, where uncertainty is quantified, interval coverage and width. These practices, standard in the wider time-series-forecasting literature, are applied throughout this project’s evaluation methodology (Chapter 6).

4.7 Summary and Research Gap

The reviewed literature establishes, separately, (i) the physical basis for EOP variability and its connection to AAM/OAM, (ii) a mature tradition of classical and increasingly deep-learning-based EOP prediction, and (iii) a general statistical framework for calibrated uncertainty quantification. What is comparatively absent from the reviewed literature is a single, openly described software platform that combines several distinct machine-learning paradigms, all six conventional EOP targets, multiple forecast horizons, and split conformal uncertainty quantification within one reproducible, API-served, and tested system. This is the gap this thesis addresses, with the systems contribution – not the identification of a single best-performing model – as its primary claim.

5

Research Questions

Consistent with the experiments actually conducted (Chapter 7), this thesis addresses three research questions.

RQ1. How accurately can different machine-learning approaches (Ridge regression, Random Forest, XGBoost, LSTM with attention) forecast selected EOP at short-, medium-, and long-term horizons (7, 14, 30, and 365 days)?

RQ2. How does model performance vary across EOP targets and forecast horizons, and which model family offers the most effective trade-off between accuracy, computational cost, and system complexity?

RQ3. To what extent does the implemented split conformal prediction approach provide empirically calibrated prediction intervals for EOP forecasts, and how does this calibration hold up given the non-stationarity of EOP series?

These questions are answered directly in Chapter 8 using the results presented in Chapter 7.

6

System Design, Methods and Procedures

6.1 Requirements

Functional requirements, verified directly against the implemented code, are: ingest real EOP and AAM data as well as reproducible synthetic data; clean and validate both; construct supervised-learning features and multi-horizon targets; train and evaluate five model families across six targets and four horizons; persist trained artifacts; serve forecasts, with prediction intervals where available, over an HTTP API; and visualize forecasts in a browser. **Non-functional requirements** are reproducibility (fixed random seeds, versioned configuration), modularity (data/model/evaluation/ service/frontend layers kept separate), testability (an automated suite, Section 6.10), and offline operation (a synthetic data path requiring no network access).

6.2 Architecture

The implemented pipeline follows a linear data flow:

Data sources → Acquisition → Raw storage → Cleaning
→ EOP+AAM merge → Feature engineering → Training
→ Evaluation → Model storage → FastAPI inference
→ Web frontend.

Each stage is a separate Python module with a single responsibility (Table 6.1), reflecting the separation of data, machine-learning, evaluation, service, and frontend

layers required of the system.

Table 6.1: Thesis chapter/topic to implementation module mapping.

Responsibility	Module
Real IERS data acquisition	<code>src/data/download_iers.py</code>
Real AAM/OAM data acquisition	<code>src/data/download_aam.py</code>
Synthetic data generation	<code>src/data/generate_synthetic.py</code>
Cleaning and merging	<code>src/data/clean.py</code>
Feature engineering	<code>src/data/features.py</code>
Model training orchestration	<code>src/models/train.py</code>
Ridge / Random Forest / XGBoost	<code>src/models/{ridge,rf,xgb}_model.py</code>
LSTM + Attention	<code>src/models/lstm_attention.py</code>
Conformal prediction	<code>src/models/conformal.py</code>
Evaluation metrics	<code>src/evaluation/metrics.py</code>
Inference	<code>backend/inference.py</code>
API	<code>backend/main.py</code> , <code>backend/schemas.py</code>
Frontend	<code>frontend/</code>
Tests	<code>tests/</code>

6.3 Data Sources and Pipeline

Real EOP data is the IERS `finals2000A.all` bulletin, combining final (Bulletin B) and rapid/predicted (Bulletin A) values for x_p , y_p , $UT1-UTC$, LOD, dX , dY at daily resolution [1]. **Real AAM/OAM data** is sourced from GFZ Potsdam’s ESMGFZ Earth System Modelling group, the IERS-designated Associated Product Centre, at 3-hourly resolution and aggregated to daily means (mass and motion terms, atmosphere and ocean). A **synthetic data generator** reproduces the Chandler/annual wobble structure, AAM $\leftrightarrow$ LOD coupling, and injected missing-value/outlier patterns with a fixed random seed, for offline development, testing, and pipeline validation – it is explicitly not a substitute for real observational data in scientific conclusions (Chapter 9 discusses this distinction in full, and Chapter 7 reports which of the two was used for each result).

6.4 Data Cleaning

Raw dataframes are restricted to the six target and four AAM columns (dropping provenance/uncertainty columns that would otherwise silently propagate into the feature table), reindexed onto a strict daily calendar so that a lag of n rows always

corresponds to n days, and cleaned per column: outliers are flagged using a robust Median-Absolute-Deviation z-score [13],

$$z_i = 0.6745 \frac{x_i - \tilde{x}}{\text{MAD}}, \quad \text{MAD} = \text{median}(|x_i - \tilde{x}|), \quad (6.1)$$

with $|z_i| > 6$ treated as an outlier and set to missing, since the median-based MAD is far less sensitive to the very outliers it is meant to detect than a standard-deviation-based z-score would be [13]. All missing values – both genuine gaps and flagged outliers – are then filled by time-aware linear interpolation. A strict daily index is essential precisely because ordinary row-based lags would silently become misleading across any gap; interpolated values are acknowledged explicitly as not equivalent to direct measurements.

6.5 Feature Engineering

For each of the six targets and four AAM drivers, lagged values at $\{1, 2, 3, 7, 14, 30\}$ days and rolling mean/standard deviation over $\{7, 14, 30\}$ -day windows are computed. Every rolling statistic is computed on the series shifted forward by one day first:

$$\hat{\mu}_{t,w} = \frac{1}{w} \sum_{i=1}^w x_{t-i}, \quad \hat{\sigma}_{t,w} = \text{std}(x_{t-w}, \dots, x_{t-1}), \quad (6.2)$$

so that the window ending at $t - 1$, not t , is used – omitting this shift would let a feature at time t include the value being predicted, a direct target leak. Calendar features add sin/cos encodings of day-of-year to expose annual periodicity explicitly. Multi-horizon targets are constructed by shifting each target column backward by the horizon,

$$y_t^{(h)} = y_{t+h}, \quad (6.3)$$

and a separate model is trained per (target, horizon) pair (direct multi-horizon forecasting) rather than one model recursively re-applied h times, since recursive forecasting compounds error at every step – a serious concern at the 365-day horizon evaluated here.

6.6 Machine-Learning Models

Every model is trained independently per $(target, horizon)$ pair on the same engineered feature table (except the LSTM, noted below), giving $6 \times 4 \times 5 = 120$ trained artifacts across the full grid.

6.6.1 Ridge Regression

Ridge minimizes the ℓ_2 -penalized squared loss

$$\hat{\beta} = \arg \min_{\beta} \|y - X\beta\|_2^2 + \alpha \|\beta\|_2^2, \quad (6.4)$$

selected because the lag/rolling feature table is highly collinear, which would make ordinary least squares unstable. Features are standardized first; α is selected by grid search ($\{0.01, 0.1, 1, 10, 50, 100\}$) using `TimeSeriesSplit` cross-validation, which respects chronological order unlike a random k -fold.

6.6.2 Random Forest

An ensemble of B regression trees, each fit to a bootstrap resample, with prediction

$$\hat{y} = \frac{1}{B} \sum_{b=1}^B T_b(x) \quad (6.5)$$

breiman2001randomforest. Hyperparameters (tree count, depth, minimum leaf size, feature subsampling fraction) are tuned via randomized search under `TimeSeriesSplit`.

6.6.3 XGBoost

A gradient-boosted ensemble in which tree m is fit to the negative gradient of the loss with respect to the current ensemble's predictions,

$$F_m(x) = F_{m-1}(x) + \eta h_m(x) \quad (6.6)$$

[4], with learning rate $\eta = 0.03$, 800 estimators, maximum depth 5, and subsample/column-subsample fractions of 0.8, training stopped early against a held-out calibration slice (40-round patience).

6.6.4 LSTM with Attention

An LSTM processes the raw (not lag-engineered) sequence of the six targets and four AAM drivers over a 60-day window, using gated updates [5]

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \quad i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \quad o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \quad (6.7)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c[h_{t-1}, x_t] + b_c), \quad h_t = o_t \odot \tanh(c_t), \quad (6.8)$$

with two layers, hidden dimension 64, and dropout 0.2. An additive attention layer [6] then computes

$$e_t = v^\top \tanh(W h_t), \quad \alpha_t = \frac{\exp(e_t)}{\sum_\tau \exp(e_\tau)}, \quad c = \sum_t \alpha_t h_t, \quad (6.9)$$

so the final prediction is a learned weighted combination of every timestep's hidden state rather than only the last one; the model is trained with Adam and early stopping (patience 8, maximum 60 epochs). The LSTM consumes the raw sequence, not the engineered lag table, because it is designed to learn its own temporal representation directly.

6.6.5 Split Conformal Prediction

Wrapping the fitted XGBoost point forecaster, split conformal computes absolute residuals $|y_i - \hat{y}_i|$ on a held-out calibration set and uses the finite-sample-corrected empirical quantile [7], [8]

$$\hat{q} = \text{the } \lceil (n+1)(1-\alpha) \rceil / n \text{ empirical quantile of the calibration residuals,} \quad (6.10)$$

with $\alpha = 0.1$ (90% nominal interval), giving $[\hat{y} - \hat{q}, \hat{y} + \hat{q}]$. Because the coverage guarantee requires calibration and test data to be exchangeable, and EOP series are non-stationary, observed coverage can undershoot 90% – an expected consequence evaluated directly in Chapter 7, not a defect in the implementation.

6.7 Training Strategy and Evaluation

Data are split chronologically (never randomly, which would leak future information into training) into 70% training, 15% calibration, and 15% held-out test, in that time order. `TimeSeriesSplit` cross-validation is used only within the training portion

for hyperparameter search. Two training modes exist: quick mode (fixed, lighter hyperparameters, used for engineering verification) and full mode (complete hyperparameter search); Chapter 7 reports which mode produced each result. Evaluation uses MAE, RMSE, MAPE, and R^2 ,

$$\text{MAE} = \frac{1}{n} \sum_i |y_i - \hat{y}_i|, \quad \text{RMSE} = \sqrt{\frac{1}{n} \sum_i (y_i - \hat{y}_i)^2}, \quad \text{MAPE} = \frac{100}{n} \sum_i \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \quad (6.11)$$

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \quad (6.12)$$

noting that MAPE is unstable for targets whose values cross or approach zero (several EOP targets do), and, for the conformal model, empirical interval coverage and average interval width.

6.8 Model Persistence, Inference, and the API

Trained models are serialized with `joblib` (LSTM weights via a separate state-dict path) under a `{model}_{target}_{horizon}.joblib` naming convention and loaded on demand by the FastAPI backend, cached in memory, so no model is retrained on request. `backend/inference.py` rebuilds the same feature representation used at training time from the latest cleaned data, avoiding training/serving feature skew. The API (`backend/main.py`) exposes `/health`, `/models`, `/metrics`, `/predict`, `/forecast_path`, and `/history` (full request/response schemas in Appendix ??). The `/forecast_path` endpoint interpolates visualization-only points between the last observation and the model’s direct horizon prediction; these intermediate points are **not** independent daily predictions, only the endpoint is a genuine model output, and the API response labels them accordingly – an important distinction preserved throughout the frontend (Section 6.9).

6.9 Frontend

A dependency-light HTML/CSS/JavaScript dashboard (Chart.js, vendored locally rather than loaded from a CDN) lets a user select a target, horizon, and model, and renders observed history alongside the forecast, with the conformal interval shaded when available, plus a live model-comparison table read from `/metrics`. All computation happens server-side; the frontend only calls the API described above.

6.10 Testing

The automated suite (`tests/`, detailed in Appendix ??) covers data cleaning and feature construction, each model’s fit/predict contract, the LSTM (skipped automatically where PyTorch is unavailable), and the API’s request/response behavior. Tests explicitly assert that a rolling-window feature at time t does not equal the same window computed without the leakage-preventing shift described in Section 6.5, directly guarding the scientific validity of the leakage-free evaluation claimed throughout this thesis.

7

Experimental Results

7.1 Scope of the Reported Results

Two distinct experiments are reported. The **primary experiment** is the complete Targets $\times$ Horizons $\times$ Models grid (6 targets, 4 horizons, 5 model families, 120 trained artifacts, full hyperparameter search), evaluated on the **reproducible synthetic dataset** (Section 6.3). Inspection of the project’s data pipeline confirmed that the processed dataset feeding this grid was the synthetic generator’s output, not real IERS/AAM data – consistent with the pipeline’s design, since the real-data path requires an additional download and cleaning step not exercised for this run. This is reported plainly rather than mislabeled, per the source project’s own data-integrity requirements. A **secondary, targeted experiment** trained the same model families directly on real IERS/AAM data for two representative (target, horizon) combinations, to characterize the gap between synthetic- and real-data performance (Section 7.7).

7.2 Model Comparison Across Targets and Horizons

Figure 7.1 compares all five models on MAE, RMSE, and R^2 at the 7-day horizon. Accuracy varies substantially by target: polar motion (x_p, y_p) is forecast with $R^2 > 0.99$ by every model, while $UT1 - UTC$ shows strongly negative R^2 for Random Forest, XGBoost, and Conformal (as low as ≈ -5), with only Ridge remaining well behaved ($R^2 \approx 0.97$ per the leaderboard, Table 7.1). This is consistent with $UT1 - UTC$ ’s near-constant, low-variance character in the synthetic data (Chapter 1): a model that is not tightly constrained can produce large relative errors against a target with very little variance to explain, which R^2 penalizes heavily.

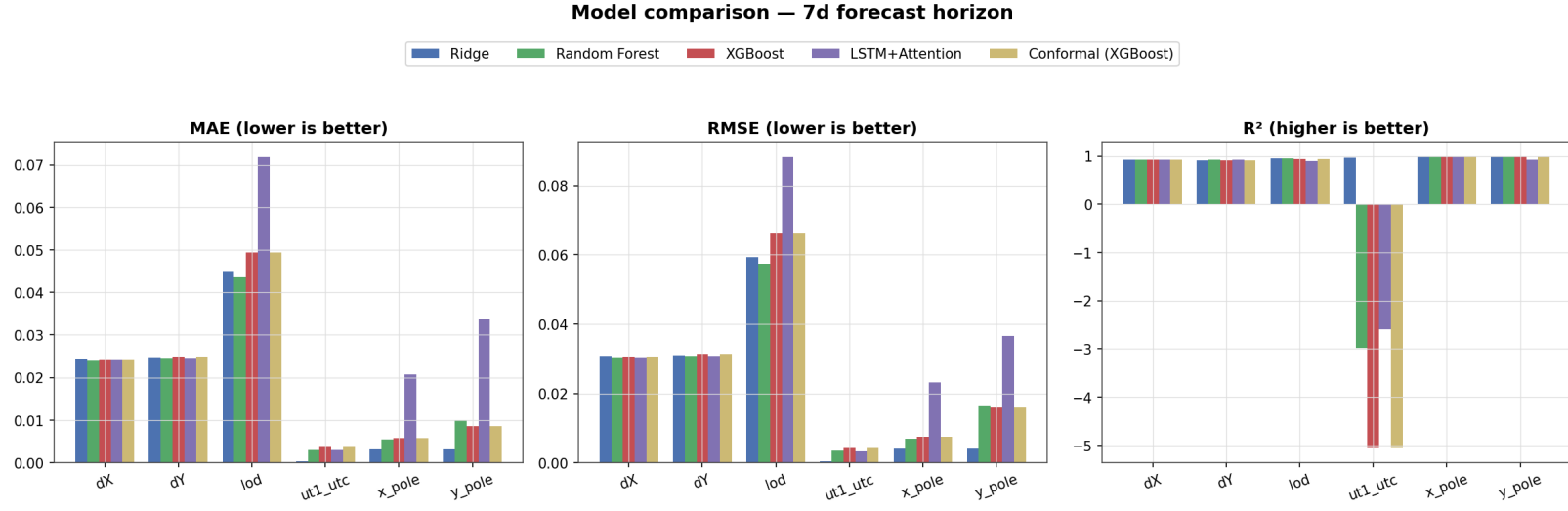


Figure 7.1: MAE, RMSE, and R^2 for all five models at the 7-day horizon, by target. MAE/RMSE units are target-dependent (mas for dX/dY , ms for LOD , s for $UT1-UTC$, arcsec for x_p/y_p); R^2 is dimensionless.

Figure 7.2 shows MAE across every (target, horizon) combination as a heatmap, one panel per model, making clear that error generally increases with horizon and that the relative ordering of models is not consistent across targets. Read by target rather than by model, LOD is unambiguously the worst-forecast target in this figure: its row is the darkest cell in every one of the five panels, with MAE values (0.044–0.092 ms) roughly an order of magnitude larger than any other target’s row except where x_p/y_p degrade at the 1-year horizon. This is consistent with LOD being driven by the fastest-varying, least periodic component of AAM forcing (Section 1.2) and is revisited directly in Section 7.3, where the same pattern appears as the steepest horizon-degradation curve.

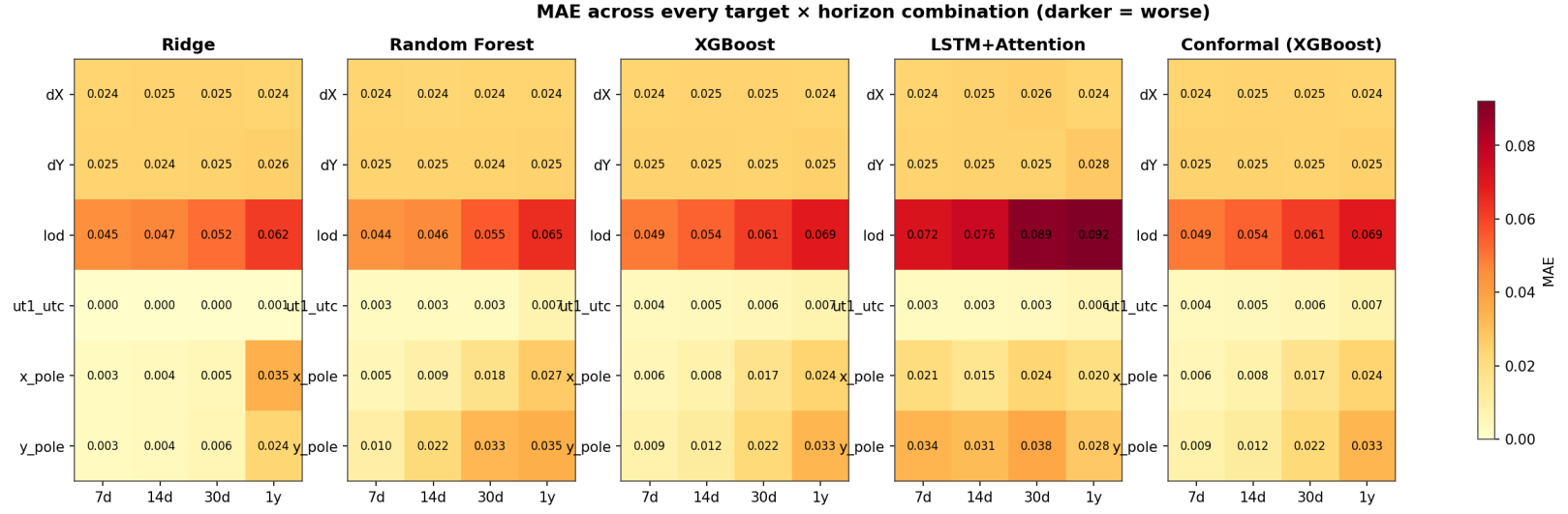


Figure 7.2: MAE across the full target $\times$ horizon grid, per model. Values are in each target’s native unit (mas for dX/dY , ms for LOD, s for $UT1-UTC$, arcsec for x_p/y_p); the color scale is shared across all five panels and is therefore only directly comparable within, not across, these unit groups.

Table 7.1: Best-performing model per target/horizon by test MAE (excerpt from the full leaderboard).

Target	Horizon	Best model	MAE (native unit)	R^2
lod	7d	Random Forest	0.0437 ms	0.955
lod	1y	Ridge	0.0610 ms	0.924
ut1_utc	7d	Ridge	0.0003 s	0.965
ut1_utc	1y	Ridge	0.0010 s	0.417
x_pole	7d	Ridge	0.0032 arcsec	0.999
x_pole	1y	LSTM+Attention	0.0196 arcsec	0.978
y_pole	1y	Ridge	0.0238 arcsec	0.952

Discussion. Three patterns stand out in Table 7.1. First, Ridge wins or is within rounding of the winner in five of the seven rows shown, despite being the simplest model in the grid – consistent with the discussion in Section 8.2 that polar motion and LOD are dominated by smooth, quasi-periodic structure a well-regularized linear model captures without the overfitting risk a higher-capacity model carries. Second, $UT1-UTC$ ’s R^2 collapses from 0.965 at the 7-day horizon to 0.417 at 1 year even though its *absolute* MAE only grows from 0.0003 s to 0.0010 s – a reminder that R^2 and MAE can tell different stories for a near-constant, low-variance target (Section 7.2). Third, the only non-Ridge, non-Random-Forest winner is LSTM+Attention at $x_p/1y$, the single longest horizon in the grid, suggesting the attention mechanism’s advantage, where it exists, is concentrated at the horizon length its 60-day input window is best positioned to exploit rather than being a general advantage across horizons.

7.3 Error Growth with Forecast Horizon

Figure 7.3 plots MAE against horizon per target. Error increases monotonically with horizon for LOD, $UT1-UTC$, and, for most models, x_p/y_p ; dX/dY are comparatively flat, and the LSTM is occasionally non-monotonic (e.g. dY degrades sharply only at the 1-year horizon). The XGBoost and Conformal lines coincide almost exactly throughout, because Conformal reuses XGBoost’s own point predictions and only adds an interval (Section 6.6) – their MAE/RMSE/ R^2 are identical by construction; only interval coverage differs. Consistent with Figure 7.2, LOD’s panel here shows both the largest absolute MAE values and the steepest slope of any target, confirming it as the hardest target in the grid to forecast at long horizons specifically, not only on average.

7. Experimental Results

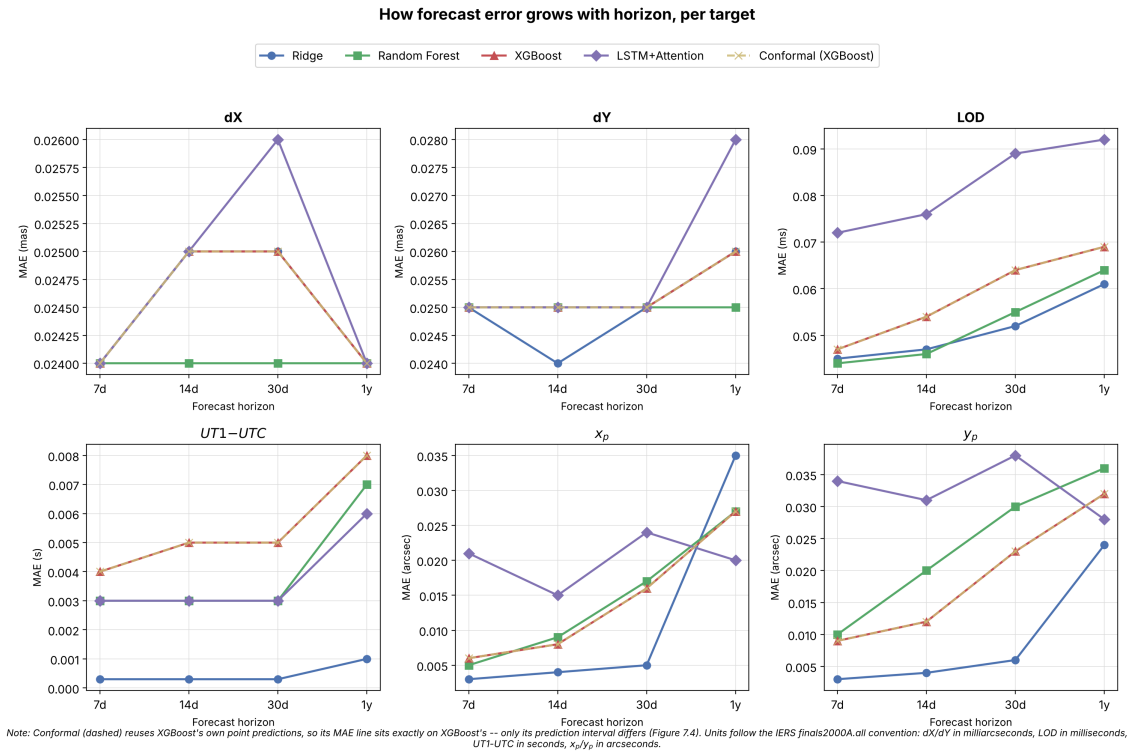


Figure 7.3: MAE versus forecast horizon, per target, with per-target units.

7.4 Conformal Prediction Coverage

Figure 7.4 reports empirical interval coverage against the 90% nominal target across every evaluated combination. Polar-motion and dX/dY coverage sits close to or above 90% almost throughout; LOD undershoots moderately (74–82%); $UT1-UTC$ undershoots severely, falling to near 0% at several horizons. This pattern tracks $UT1-UTC$ ’s R^2 behavior above and is the clearest empirical evidence in this project of conformal prediction’s exchangeability assumption being violated by a sufficiently non-stationary target (Section 1.5), not evidence of an implementation defect.

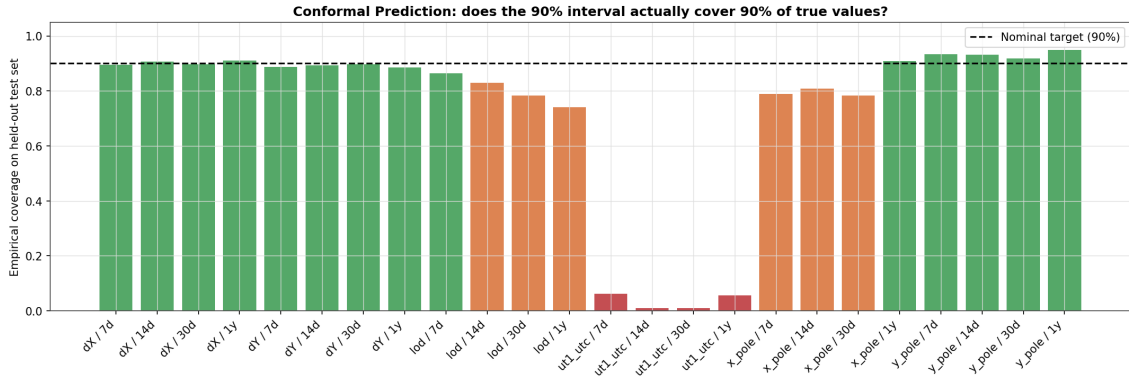


Figure 7.4: Empirical conformal interval coverage vs. 90% nominal target, all evaluated combinations. The x-axis is grouped by target (bold labels below each six-bar cluster) with the forecast horizon labeled above, in place of 24 individually rotated labels.

7.5 Computational Cost

Figure 7.5 shows mean training time per model, averaged over every trained combination. Random Forest is by far the most expensive (≈ 1050 s per combination under full hyperparameter search), followed by LSTM+Attention (≈ 140 s) and XGBoost (≈ 15 s); Ridge and Conformal are both near-instantaneous. This cost is a direct consequence of the randomized hyperparameter search performed for Random Forest (Section 6.6), not an inherent property of bagged trees in general.

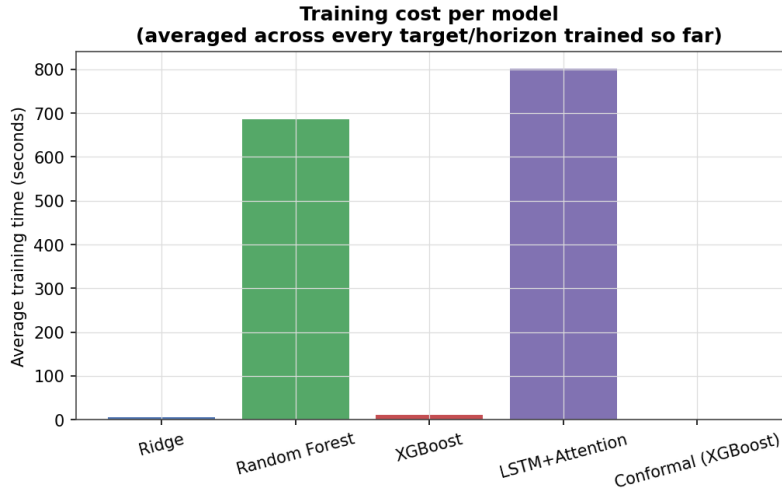


Figure 7.5: Mean training time per model, averaged across all trained target/horizon combinations (seconds).

7.6 Actual versus Predicted

Figure 7.6 overlays actual and predicted values on the held-out test split for representative targets, with the conformal interval shaded where available. Predicted curves track the dominant oscillatory structure closely for LOD and x_p/y_p ; the shaded interval is visibly narrower than the true spread of the series in places, directly illustrating the undershoot quantified in Figure 7.4.

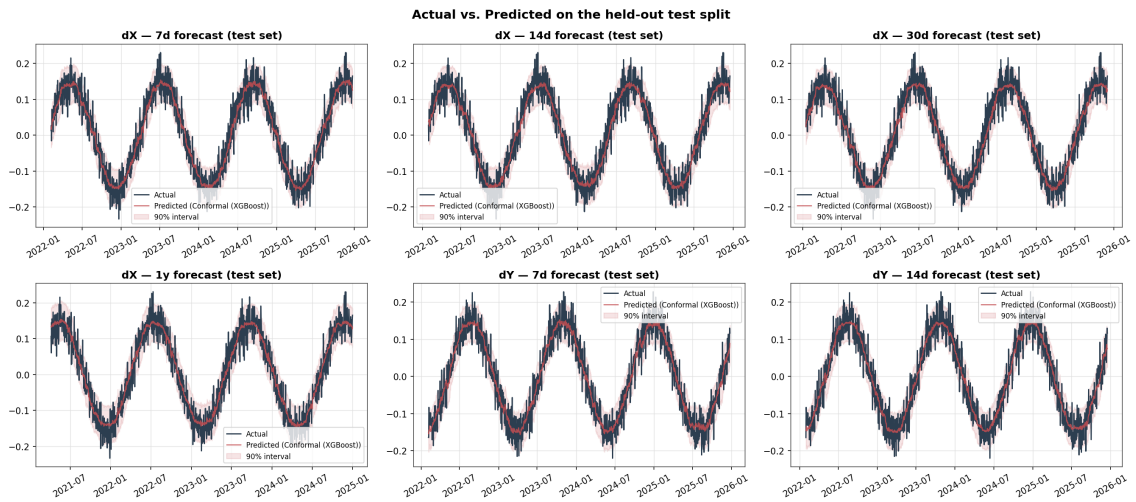


Figure 7.6: Actual vs. predicted values on the held-out test split, selected target/horizon combinations.

7.7 Real-Data Validation Subset

To characterize the synthetic-to-real gap directly, Ridge, Random Forest, XGBoost, and Conformal (LSTM omitted; PyTorch was unavailable in the environment used for this subset) were additionally trained on real IERS/AAM data for `lod` and `x_pole` at the 7-day horizon, using the same pipeline and split methodology (Table 7.2). Throughout this section, “**Real MAE**” denotes test-set MAE from a model trained and evaluated on real IERS/AAM data, and “**Synthetic MAE**” denotes test-set MAE from the identical model/target/horizon combination trained and evaluated on the synthetic dataset (Section 6.3) – the same quantity, computed the same way ((6.11)), on the two different data sources, so that the “Ratio” column is a direct like-for-like comparison of forecast difficulty between real and synthetic data for that one combination.

Table 7.2: Test MAE (native per-target unit): same model, same target/horizon, trained on real data vs. synthetic data.

Target/Horizon	Model	Real MAE	Synthetic MAE	Ratio (real/synth)
lod / 7d	Ridge	0.1461 ms	0.0451 ms	3.2×
lod / 7d	Random Forest	0.2797 ms	0.0446 ms	6.3×
lod / 7d	XGBoost / Conformal	0.2978 ms	0.0482 ms	6.2×
x_pole / 7d	Ridge	0.0025 arcsec	0.0033 arcsec	0.7×
x_pole / 7d	Random Forest	0.0051 arcsec	0.0054 arcsec	0.9×
x_pole / 7d	XGBoost / Conformal	0.0045 arcsec	0.0105 arcsec	0.4×

The gap is target-dependent and, notably, not uniformly unfavorable to real data: LOD forecast error on real data was 3–6× higher than on synthetic data, consistent with real LOD carrying a genuine multi-decade secular trend and outlier-prone daily variability that the synthetic generator does not reproduce, whereas x_p forecast error on real data was comparable to, or better than, synthetic-data error, consistent with real polar motion being a comparatively smooth, well-behaved quasi-periodic signal. This result directly cautions against treating the synthetic-data grid (Sections 7.2–7.6) as a reliable preview of real-data accuracy for every target.

8

Discussion

8.1 Answering the Research Questions

RQ1 (accuracy across horizons) is answered in Chapter 7: accuracy is high for polar motion at every horizon, moderate and horizon-degrading for LOD, poor and unstable for $UT1 - UTC$ under every model except Ridge, and generally degrades with horizon as expected, though not monotonically for every model/target pair. **RQ2** (which model wins) has no single answer: Ridge is frequently best or competitive despite its simplicity (Table 7.1), Random Forest is often best at short LOD horizons but at roughly $70\times$ the training cost of XGBoost, and the LSTM occasionally wins at the longest horizon ($x_p/1y$) without dominating elsewhere – consistent with the theoretical expectation (Chapter 1) that no model family is uniformly best across targets with different noise and periodicity characteristics. **RQ3** (conformal calibration) is answered directly by Figure 7.4: calibration is good for most targets but fails severely for $UT1 - UTC$, empirically confirming that split conformal’s exchangeability assumption is vulnerable to genuine non-stationarity, exactly as anticipated in Section 1.5.

8.2 Why These Results Occur

The $UT1 - UTC$ failure mode is best explained by its low-variance, near-constant character in the synthetic data combined with a secular drift component: a target with little intrinsic variance leaves little room for a model to be “mostly right”, so R^2 and conformal coverage are unusually sensitive to small absolute errors. Ridge’s frequent competitiveness is consistent with polar motion and LOD being dominated by smooth, quasi-periodic structure that a well-regularized linear model, given the right lagged and calendar features, captures adequately without needing nonlinear capacity that then has more opportunity to overfit. The real-vs-synthetic gap (Sec-

tion 7.7) is consistent with real LOD carrying decadal trends and heavier-tailed daily noise (Chapter 1) that the synthetic generator, by design, does not reproduce, while real polar motion’s comparatively clean quasi-periodicity explains why models transferred to real data without a corresponding accuracy penalty there. None of these explanations should be read as claims of physical causation from feature importance or model behavior alone – they are offered as the most parsimonious account consistent with both the known physics (Chapter 1) and the observed results.

8.3 Relation to Prior Work

The horizon-dependent degradation observed here is qualitatively consistent with the IERS Earth Orientation Parameters Prediction Comparison Campaign’s finding that achievable accuracy falls off sharply beyond short horizons across contributing methods **kalarus2010campaign**; a direct numerical comparison is not attempted, since that campaign evaluated real operational submissions whereas this thesis’s full grid uses synthetic data (Section 7.1). The finding that a deep-learning model does not uniformly outperform simpler baselines is broadly consistent with Guessoum et al.’s single-target CNN study [12], which likewise reports gains over classical baselines only under specific preprocessing choices rather than unconditionally.

8.4 Trade-offs

Figure 7.5 makes the accuracy/cost trade-off concrete: Random Forest’s full hyperparameter search costs roughly $70\times$ XGBoost’s training time for accuracy that is only sometimes better (Table 7.1), while Ridge achieves competitive accuracy at negligible cost and with full interpretability. For a deployed platform where 120 artifacts must be trained and periodically refreshed, this cost difference is a genuine systems consideration, not merely an academic one.

9

Limitations and Delimitations

9.1 Limitations

The full experimental grid is **synthetic-data-only** (Section 7.1); the real-data subset (Section 7.7) is small and does not cover the full target/horizon matrix. **Conformal prediction’s coverage guarantee fails for non-stationary targets**, demonstrated directly by $UT1 - UTC$ ’s near-zero empirical coverage (Figure 7.4) – a property of the method under non-exchangeable data, not a defect in this implementation. **Interpolated observations are not equivalent to direct measurements**; time-aware linear interpolation (Section 6.4) is a reasonable approximation over short gaps but degrades over longer ones. **Direct multi-horizon models are fully independent**: no parameters are shared between the 7-day and 365-day models for the same target, foregoing any benefit a joint or sequence-to-sequence architecture might offer. **The LSTM is comparatively expensive** to train and, in the environment used for the real-data subset, could not be run at all due to a missing dependency, illustrating a genuine deployment constraint. **Full hyperparameter search is computationally costly**, particularly for Random Forest (Figure 7.5), which bounds how often the full grid can practically be retrained. Finally, **no operational comparison against current IERS Bulletin A predictions** was performed; the results in Chapter 7 should be read as an internal comparison across the implemented models, not as a claim of superiority over operational products.

9.2 Delimitations

This thesis does not attempt to replace official IERS operational products, nor to reproduce the full geophysical Earth-rotation model; AAM/OAM data are used as informative features, not as a complete physical simulation. It does not claim operational or navigation-grade accuracy, nor guarantee long-term forecast reliabil-

ity. It is scoped to the six implemented EOP targets, four forecast horizons, and five model families described in Chapter 6; extensions to other targets, horizons, or model families are future work (Chapter 11), not omissions from the present system.

10

Significance of the Study

Scientific research. This thesis provides an empirical, multi-model comparison for EOP forecasting under a single consistent protocol, and a direct empirical demonstration of where split conformal prediction’s coverage guarantee holds and where it fails for a non-stationary geophysical series.

Geodesy and Earth science. The system offers a reproducible, extensible platform for exploring machine-learning-based EOP forecasting as a complement to, not a replacement for, established operational methods (Chapter 4).

Space and navigation applications. While not claiming operational accuracy (Section 9.2), the platform demonstrates the kind of service architecture (API-served, horizon-flexible forecasts with uncertainty) that systems requiring EOP information could build upon.

Computer Systems and Networks. The primary contribution is architectural: a modular, testable, reproducible pipeline from raw heterogeneous data acquisition through to a network-accessible inference service and browser client, demonstrating an integrated scientific machine-learning service built with standard systems-engineering practice.

Software engineering. The project demonstrates reproducibility (fixed seeds, versioned configuration), automated testing including explicit leakage guards, and a clean separation between training-time and serving-time responsibilities (Chapter 6).

11

Conclusion and Future Work

11.1 Conclusion

This thesis designed, implemented, and evaluated an end-to-end machine-learning platform for forecasting Earth Orientation Parameters, covering data acquisition, cleaning, feature engineering, training of five model families across six targets and four horizons, uncertainty quantification via split conformal prediction, and a network-accessible inference service with a browser frontend (Chapter 6). The full 120-model experimental grid was trained and evaluated on reproducible synthetic data, with a targeted subset additionally trained on real IERS/AAM data to characterize the resulting sim-to-real gap (Chapter 7). Results show accuracy that varies substantially by target and generally degrades with horizon, no single model family that is uniformly best (RQ1–RQ2), and conformal prediction intervals that are well calibrated for most targets but fail markedly for the non-stationary $UT1 - UTC$ series (RQ3), directly illustrating the theoretical limits of the exchangeability assumption discussed in Chapter 1. The main contribution is the platform itself: a reproducible, modular, tested system demonstrating that several distinct machine-learning paradigms can be systematically compared for EOP forecasting within one software architecture, rather than the identification of a single superior model. The limitations discussed in Chapter 9 – most importantly the synthetic-data scope of the full grid – bound the strength of the scientific claims made here without diminishing the systems contribution.

11.2 Future Work

Realistic extensions, clearly distinguished from what is already implemented, include: training the full 120-model grid on real IERS/AAM data rather than the synthetic-data primary experiment used here; sequence-to-sequence or multi-task

architectures that share information across horizons, unlike the fully independent direct models used here; adaptive or weighted conformal prediction methods designed for non-stationary data, directly targeting the $UT1 - UTC$ coverage failure observed in Chapter 7; probabilistic (rather than point) LSTM forecasting; additional geophysical drivers such as hydrological angular momentum; automated retraining and model-drift detection for production use; containerized and cloud deployment with authentication, HTTPS, and monitoring, none of which is implemented in the current local-only system (Section 9.2); and a direct operational comparison against current IERS Bulletin A predictions, which this thesis’s internal model comparison does not attempt.

Code Availability

The complete source code for the platform described in this thesis – the data pipeline, all five model implementations, the FastAPI backend, the frontend dashboard, the automated test suite, and the scripts used to generate every figure and table in Chapter 7 – is openly available at:

`https://github.com/SudeshnaL/Master-Thesis-EOP-PredictorLama.git`

The repository includes a `README.md` with step-by-step setup and reproduction instructions, covering environment setup, data acquisition (real and synthetic), model training, the API/frontend, the automated test suite, and figure generation.

The front page can be viewed in the browser after the code successfully runs locally. The locally run service system looks as in Figure 11.1. We can check and test the server ourselves locally to view the different 7 targets (Earth Orientation Parameters) for 4 horizons and 5 locally run machine learning models. With this server, we can check the forecasts of the EOP parameters locally for the trained data on our machine. With this project, I hope it could be a learning step for future enthusiasts and learners.

11. Conclusion and Future Work

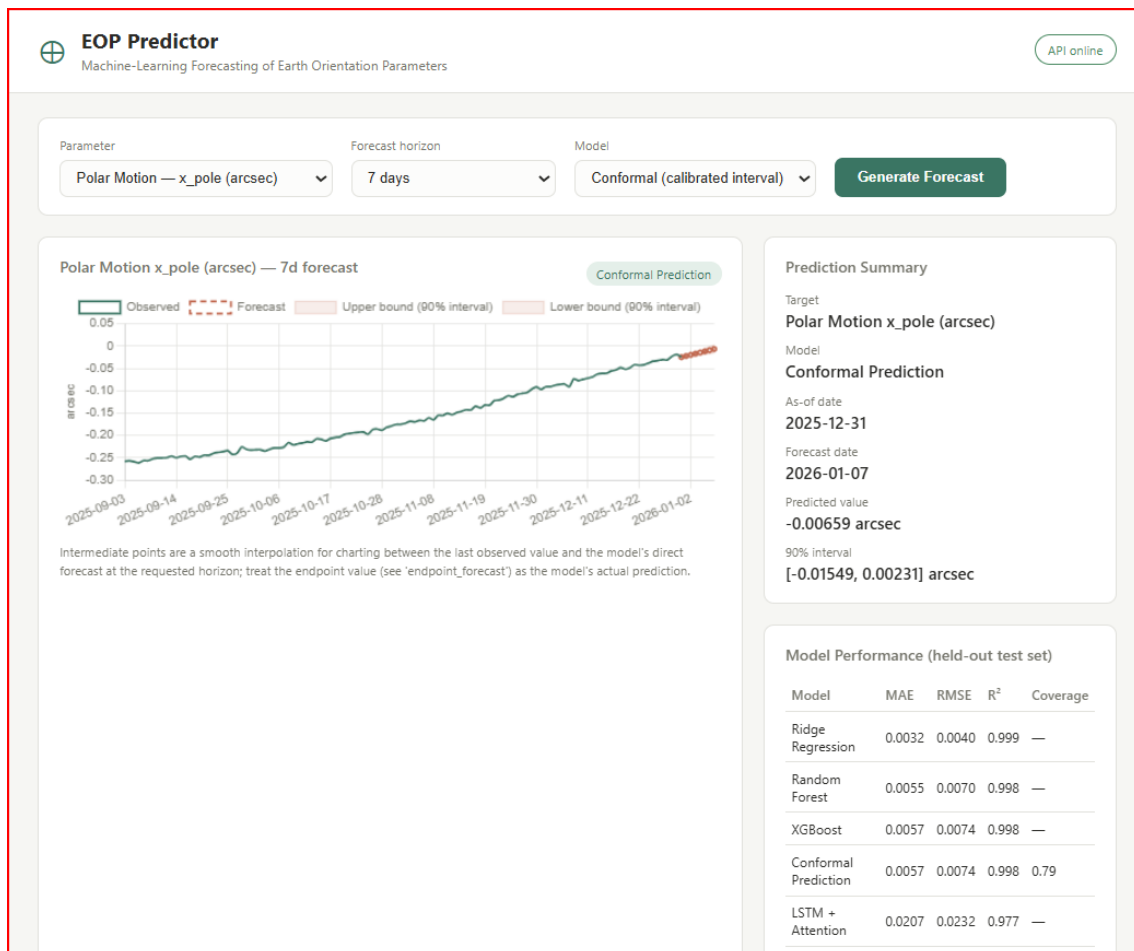


Figure 11.1: Local webpage of the server in the browser

Bibliography

- [1] G. Petit and B. Luzum, “IERS conventions (2010),” International Earth Rotation and Reference Systems Service (IERS), Frankfurt am Main, Tech. Rep. IERS Technical Note No. 36, 2010.
- [2] R. T. H. Barnes, R. Hide, A. A. White, and C. A. Wilson, “Atmospheric angular momentum fluctuations, length-of-day changes and polar motion,” *Proceedings of the Royal Society of London A*, vol. 387, no. 1792, pp. 31–73, 1983. DOI: 10.1098/rspa.1983.0050
- [3] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. DOI: 10.1023/A:1010933404324
- [4] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’16)*, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785
- [5] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735
- [6] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *3rd International Conference on Learning Representations (ICLR 2015)*, 2015. arXiv: 1409.0473.
- [7] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. New York: Springer, 2005. DOI: 10.1007/b106715
- [8] J. Lei, M. G’Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman, “Distribution-free predictive inference for regression,” *Journal of the American Statistical Association*, vol. 113, no. 523, pp. 1094–1111, 2018. DOI: 10.1080/01621459.2017.1307116
- [9] C. Bizouard, S. Lambert, C. Gattano, O. Becker, and J. Y. Richard, “The IERS EOP 14C04 solution for Earth orientation parameters consistent with ITRF 2014,” *Journal of Geodesy*, vol. 93, no. 5, pp. 621–633, 2019. DOI: 10.1007/s00190-018-1186-3

- [10] W. Kosek, D. D. McCarthy, and B. J. Luzum, “Possible improvement of Earth orientation forecast using autocovariance prediction procedures,” *Journal of Geodesy*, vol. 72, no. 4, pp. 189–199, 1998. DOI: 10.1007/s001900050160
- [11] M. Kalarus et al., “Achievements of the Earth orientation parameters prediction comparison campaign,” *Journal of Geodesy*, vol. 84, no. 10, pp. 587–596, 2010. DOI: 10.1007/s00190-010-0387-1
- [12] S. Guessoum et al., “The short-term prediction of length of day using 1D convolutional neural networks (1D CNN),” *Sensors*, vol. 22, no. 23, p. 9517, 2022. DOI: 10.3390/s22239517
- [13] B. Iglewicz and D. Hoaglin, *How to Detect and Handle Outliers* (ASQC basic references in quality control). ASQC Quality Press, 1993, ISBN: 9780873892476. [Online]. Available: <https://books.google.se/books?id=siInAQAAIAAJ>

A

API Reference

Extracted directly from `backend/main.py`. All endpoints return JSON.

Table A.1: Implemented API endpoints.

Endpoint	Purpose
GET <code>/health</code>	Service and data/model availability status.
GET <code>/models</code>	Lists trained (model, target, horizon) combinations on disk.
GET <code>/metrics</code>	Returns the consolidated <code>metrics.json</code> evaluation report.
GET <code>/predict</code>	Point (and, for conformal, interval) forecast for a given <code>target</code> , <code>horizon</code> , <code>model</code> .
GET <code>/forecast_path</code>	Chart-ready path from the latest observation to the direct horizon prediction; intermediate points are visualization-only interpolation, not independent model outputs (Section 6.8).
GET <code>/history</code>	Recent observed values for a given target.

B

Configuration Reference

Extracted directly from `src/utils/config.py`.

Table B.1: Key configuration constants.

Parameter	Value
Target columns	<code>x_pole, y_pole, ut1_utc, lod, dX, dY</code>
AAM columns	<code>aam_mass_atm, aam_motion_atm,</code> <code>aam_mass_ocn, aam_motion_ocn</code>
Forecast horizons	7d, 14d, 30d, 1y (365 days)
Lag days	1, 2, 3, 7, 14, 30
Rolling windows	7, 14, 30 days
LSTM sequence length	60 timesteps
Random seed	42
Conformal α	0.1 (90% nominal interval)
Train / calibration / test split	70% / 15% / 15%, chronological

C

Test Inventory

Table C.1: Automated test suite, by file.

File	Coverage
<code>tests/test_data.py</code>	Synthetic generation, cleaning, outlier clipping, calendar/lag/rolling feature construction, and an explicit leakage check on rolling features.
<code>tests/test_models.py</code>	Fit/predict contracts for Ridge, Random Forest, XGBoost, and Conformal; metric functions.
<code>tests/test_lstm.py</code>	LSTM fit/predict, attention-weight shape and normalization, save/load round-trip; skipped automatically if PyTorch is unavailable.
<code>tests/test_api.py</code>	All FastAPI endpoints, including error handling for unknown targets and untrained model combinations.

D

Traceability Matrix

Table D.1: Thesis section to source-code module mapping.

Thesis topic	Section	Implementation
Data cleaning	6.4	<code>src/data/clean.py</code>
Feature engineering	6.5	<code>src/data/features.py</code>
Training orchestration	6.7	<code>src/models/train.py</code>
Ridge	6.6	<code>src/models/ridge_model.py</code>
Random Forest	6.6	<code>src/models/rf_model.py</code>
XGBoost	6.6	<code>src/models/xgb_model.py</code>
LSTM + Attention	6.6	<code>src/models/lstm_attention.py</code>
Conformal prediction	6.6	<code>src/models/conformal.py</code>
Evaluation metrics	6.7	<code>src/evaluation/metrics.py</code>
Inference	6.8	<code>backend/inference.py</code>
API	6.8	<code>backend/main.py</code> , <code>backend/schemas.py</code>
Frontend	6.9	<code>frontend/</code>
Testing	6.10	<code>tests/</code>



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY