Time Series Forecasting and Optimization Module Plan

Kieran Tran
Forecasting & Optimization Lead

October 24, 2025

1 Purpose and Scope

The Time Series Forecasting and Optimization module provides predictive capabilities within the Smart Document & Decision Automation Platform. Its purpose is to model and forecast key business indicators—such as cash-flow, workload, and resource utilization—enabling data-driven decisions and proactive risk management.
The module consumes cleaned, structured data produced by upstream ETL and document-processing pipelines and outputs forecasts, uncertainty intervals, and optimization insights to downstream analytics dashboards.

- **Scope:** Develop and deploy time-series models (ARIMA, Prophet, LSTM) for cash-flow and workload forecasting; integrate with the risk-scoring module for scenario analysis. (Depending on the data, there may be a lot of noise in the measurement of the input, and so the use of uncertainty initialized HiPPO model may be applicable). Tool should also be able to provide different models for univariate and multivariate datasets. Currently, ARIMA is most popular however tech firms such as Google and Amazon have made the model more robust by implementing it in conjunction with BigQuery ML (Google Vertex AI Forecasting).
- **Boundaries:** This module does not perform OCR or data extraction; it relies on structured inputs from prior stages.
- **Success Metrics:** Forecast MAPE (Mean absolute percentage error) $\leq 10\%$,

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\sqrt{(\frac{A_t - P_t}{A_t})^2} \times 100\%$$

  and positive user feedback from pilot testing, especially in terms of readability and interpretation of data insights.

2 Deliverables

1. **Vertex AI Forecasting Dataset and Schema:**
   - Dataset hosted in BigQuery or Google Cloud Storage.
   - Schema includes time, identifier, target variable, and covariates.

- Data validation and quality report.

2. **Trained Forecasting Models:**
    - Multiple models trained via Vertex AI Forecasting.
    - Train and benchmark ARIMA, Prophet, and LSTM models. (These are current and reliable models used in the market, there are other advanced ones like UnHiPPo, but still in research phase, possible to implement this architecture from scratch in conjunction with the Vertex AI environment).
    - Configured with forecast horizon, context window, and holiday region.
3. **Evaluation and Backtesting:**
    - Metrics: MAE, RMSE, and MAPE.
    - Rolling and expanding window validation.
4. **Batch Inference Pipeline:**
    - Scheduled batch predictions stored in BigQuery.
    - Supports prediction intervals and versioned model metadata.
5. **API Layer (FastAPI on Cloud Run):**
    - `/forecast`: returns latest forecasts.
    - `/evaluate`: provides evaluation metrics.
    - `/detect-anomalies`: integrates Timeseries Insights API.
6. **MLOps and Monitoring:**
    - Infrastructure as Code (Cloud Build, Artifact Registry).
    - Observability: logging, tracing, and alerts via Cloud Logging.

3 Dataset Design

3.1 Required Columns

- **time_column:** Timestamp or date at regular intervals (e.g., daily, weekly).
- **time_series_id:** Unique identifier for each series (e.g., store or SKU).
- **target:** Value to forecast (e.g., sales, revenue).
- **Static features:** Attributes constant for each series (e.g., region, category).
- **Time-varying covariates:**
    - Known-in-advance: promotion calendars, planned prices, holidays.
    - Unknown-in-advance: observed weather, realized sales indicators.

3.2 Key Parameters

- **Granularity:** Time interval per observation (day, week, month).
- **Forecast horizon:** Number of future time steps to predict.
- **Context window:** Historical range used for learning.
- **Holiday region:** Region code for built-in holiday effects.
- **Backtesting:** Rolling and expanding window strategy.

4 Implementation Steps

4.1 Step 0: Prerequisites

(a) Create a new Google Cloud project and enable Vertex AI, BigQuery, and Cloud Storage.

(b) Assign IAM roles: `Vertex AI Admin`, `BigQuery Admin`.

(c) Install SDK: `gcloud init` and `pip install google-cloud-aiplatform`.

4.2 Step 1: Data Preparation

(a) Collect and clean historical time series data.

(b) Store data in BigQuery with correct schema.

(c) Ensure consistent granularity, no duplicates, and no data leakage.

4.3 Step 2: Dataset Creation

(a) In Vertex AI Console, navigate to Datasets → Create → Tabular → Forecasting.

(b) Import data from BigQuery and assign schema roles.

4.4 Step 3: Model Training

(a) Configure parameters:

  - Granularity (e.g., day)
  - Forecast horizon (e.g., 30)
  - Context window (e.g., 90)
  - Holiday region (e.g., SG)

(b) Train model with AutoML Forecasting.

(c) Review training logs and feature importance.

4.5 Step 4: Evaluation and Backtesting

(a) Review Vertex metrics: MAE, RMSE, MAPE.

(b) Conduct custom rolling backtests for multiple cutoffs.

4.6 Step 5: Batch Predictions

(a) Configure Vertex batch prediction job to write outputs to BigQuery.

(b) Include quantile outputs (e.g., 0.1, 0.5, 0.9).

(c) Schedule runs using Cloud Scheduler.

4.7 Step 6: API Deployment (FastAPI on Cloud Run)

- **/forecast:** Query BigQuery for latest predictions by series ID.
- **/evaluate:** Return stored metrics.
- **/detect-anomalies:** Call Timeseries Insights API for spike and trend detection.

4.8 Step 7: Monitoring and Operations

- Use Cloud Logging and OpenTelemetry for request tracing.
- Create alert policies for job failures and API downtime.
- Secure endpoints with OAuth2 and store secrets in Cloud Secret Manager.

5 Suggested Schedule

Week 1: **Project Setup:** Initialize GCP environment and services.

Week 2: **Data Preparation:** Upload and verify dataset in BigQuery.

Week 3: **Model Training:** Run baseline Vertex AI Forecasting job.

Week 4: **Evaluation:** Tune hyperparameters and select final model.

Week 5: **Batch Inference:** Schedule Vertex batch prediction to BigQuery.

Week 6: **API Integration:** Build and deploy FastAPI service on Cloud Run. (Check on this with Rishika and Natalie)

Week 7: **Finalisation:** Add monitoring, IAM tightening (Identity and Access Management), and documentation.

6 Key Definitions

- **Horizon:** Future time steps to forecast (e.g., 30 days).
- **Context window:** Historical data range used for training.
- **Granularity:** Frequency of time steps (daily, weekly, etc.).
- **Known-in-advance covariates:** Features with future values known at prediction time.

7 Dependencies and Collaboration

- **Data Inputs:** Cleaned structured data from Natalie's ETL pipelines.
- **Risk Scores:** Inputs from anomaly detection and risk modules (Kieran & Aryan).
- **Backend Integration:** API coordination with Rishika's backend services.
- **Dashboard:** Visualization integration with Natalie & Rishika.

8 Resource Needs

- GPU/High-memory CPU instance for LSTM training is we are implementing deep learning techniques.
- Money for Google Vertex AI account, but we do get $300 free credit for 90 days upon opening a new account.

9 Risks and Mitigations

Risk	Impact	Explanation and Mitigation Strategy
Limited or Sparse Historical Data	High	Insufficient historical data may prevent the model from capturing seasonal or cyclical patterns, leading to unreliable forecasts. To mitigate this, synthetic data augmentation and public benchmark datasets can be used to expand the training base. Vertex AI's built-in data preprocessing (such as automatic aggregation and imputation) can also help fill gaps and smooth incomplete series.
Model Drift After Deployment	High	Over time, real-world data distributions may change due to new policies, market conditions, or operational practices, causing model drift and degraded accuracy. Mitigation includes implementing continuous monitoring with drift detection metrics, scheduling regular retraining through Vertex AI Pipelines, and tracking performance trends over time.
Overfitting on Small Datasets	Medium	When datasets are small, models may memorize noise rather than learn general patterns, resulting in poor generalization to unseen data. Mitigation strategies include applying k-fold cross-validation, regularization, and early stopping, as well as enabling Vertex AutoML's built-in hyperparameter optimization and model ensembling to balance bias and variance.
High Compute Cost During Experimentation	Medium	Training large forecasting models or running multiple hyperparameter searches can quickly escalate cloud costs, especially on GPU-backed instances. To manage costs, use preemptible GPU/CPU instances, monitor budgets with Google Cloud Billing alerts, and leverage Vertex AI Workbench for efficient, notebook-based experimentation. Batch jobs should be scheduled during low-cost periods when possible.
Forecast Interpretability	Low	Complex models such as deep neural networks can be difficult to interpret, which may reduce stakeholder trust in predictions. To address this, implement explainability tools such as SHAP (SHapley Additive exPlanations) values and sensitivity analysis using Vertex Explainable AI, providing insights into feature importance and model reasoning.