Seas the Data - Technical Summary

This project represents an early milestone in my transition from a high-performance computing (HPC) and scientific software background into the world of data science. I've spent nearly two decades writing complex simulation and modelling tools—primarily in C++—for engineering and physics domains. Over the past year, I've focused intensely on upskilling in Python, statistics, machine learning, and practical data science workflows. While I'm still early on this particular path, I've built strong foundations and am progressing rapidly. This submission reflects my learning curve, but also my ability to reason about modelling, work systematically, and deliver reproducible code. I ask that you evaluate it not just on what it is, but on what it suggests I can grow into.

Rather than use the provided template and evolve it slightly, I opted to build my own forecasting engine from scratch. The model is based on four distinct signal components: (1) sub-day patterns (15-minute resolution), (2) weekly rhythms (day-of-week effects), (3) annual trends (seasonal shifts by week-of-year), and (4) a linear trend extrapolated from historical yearly aggregates. These were separated, normalised, and recombined to produce fine-grained predictions. I also implemented test/train separation based on a split-date approach, and evaluated forecast accuracy by computing residuals and uncertainty metrics (MAE, RMSE, and STD) using the test set.

The resulting model was able to produce high-resolution forecasts that respected both daily and seasonal dynamics. I then extended this into a full-year projection and included uncertainty bands to help visualise the reliability of the prediction. These confidence intervals were based on the standard deviation of residuals from the test data, applied symmetrically to the projected values.

However, one important technique I overlooked was `TimeSeriesSplit`. My approach used a fixed historical window for training and tested against future data, which simulates a realistic forecasting context but doesn't provide the same insight into model stability across multiple temporal slices. In retrospect, adopting `TimeSeriesSplit` or walk-forward validation would have allowed more robust generalisation error estimation. This was a missed opportunity and a valuable lesson.

Despite that, I'm proud of how well the core ideas performed. The model was modular, interpretable, and extensible. The separation of temporal effects gave flexibility in tuning and visualisation. It doesn't rely on black-box machine learning and instead captures domain-relevant behaviour in a way that business users can understand. I also focused on clarity, reproducibility, and structured logging—practices I bring from my HPC background.

This work reflects both my curiosity and my discipline. I didn't have all the answers, but I dug deep into the structure of the data and built a layered model that made sense to me. I hope that, more than just evaluating the finished tool, you'll see the kind of developer and data scientist I'm becoming—and consider the potential I offer if given the chance to grow within a collaborative

and rigorous team.