Machine Learning for Power Demand & Generation

Prediction of the Modern Grid

Nick Khormaei

Carson Lansdowne

Jordan Hendricks

Abstract

Accurately predicting electricity supply and demand is crucial for ensuring grid stability, optimizing resource allocation, and reducing operational costs. Our project aims to explore the performance of classical and machine learning models for power demand forecasting and power generation (realized load) prediction. We compare the models through common metrics for predicting three time-scale windows: next-day, three day, and six day forecasts. Additionally, we experimented with data generation and price prediction using select models.

Introduction/Background

Throughout this report we will explore using both classical time series models and machine learning models to predict short-term power demands using historical data on the ERCOT grid. The ERCOT grid is divided into eight geographic regions, and the data used in this report includes both regional and system-wide information. Demand load forecasting is an important

aspect of the grid: the more accurate the prediction of the demand, the better distributors can plan for load generation and a more stable grid.

Among the features analyzed, regional average temperatures, natural gas production, and coal/lignite generation showed the strongest correlation with actual grid load. Conversely, variables such as wind and solar generation, along with day-ahead and real-time market prices, demonstrated weaker correlations. To ensure high data fidelity, we collected hourly data spanning three years, from 2022 to 2024, and performed linear interpolation to fill in missing data and improve model performance.

We aim to explore the performance of models such as ARIMA, SARIMAX, Prophet, TTMs, LSTM, and XGBoost and determine their effectiveness of predicting short-term power demands. We will compare these models using metrics such as mean-squared error, R-squared, and root-mean squared error among others over next-day, three-day, and six-day forecast periods. We also explore the results of these models in practical terms, by utilizing our dataset to explore the market spread that our load and price predictions would allow us to capture in the day-ahead-market. This analysis is expected to contribute to optimizing grid operations and enhancing energy resource management, ultimately leading to improved grid stability and reduced operational costs.

System Design

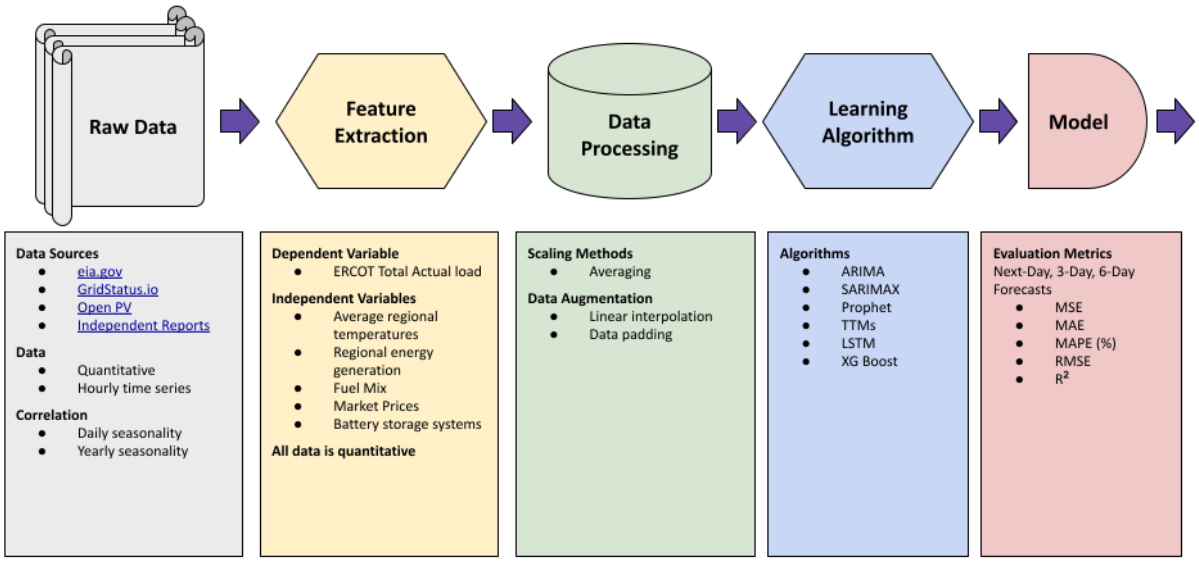


Figure 1: System Design Workflow

Methodology

Data Description and Processing

Our data is a time series matrix with hourly data from the ERCOT grid from the beginning of 2022 to the end of 2024. It includes 46 independent variables that include average regional temperatures, regional energy generation, fuel mix, market prices, and battery storage system development. A correlation matrix was used to determine which exogenous variables were most correlated to the ERCOT Total Actual Load, the dependent variable.

When reviewing the raw data, there were often missing timesteps, requiring the development of data augmentation methods to 'expand' the dataset and identify missing data points. Once found, linear interpolation was applied to each independent variable to fill in the missing data.

Approach (Models)

ARIMA

ARIMA is a classical time series model that combines autoregression (AR), differencing (I), and a moving average (MA) to model non-stationary data. It was chosen to establish a baseline for forecasting ERCOT grid load using a simple, classical time series approach. Since ARIMA models rely solely on the past values of the dependent variable, they are well-suited for understanding how effectively a linear, autoregressive model can capture short-term trends. Given its relatively fast training time and low computational requirements, ARIMA provides a practical starting point for comparison against more complex models.

Despite its limitations in handling seasonal patterns, ARIMA was included to evaluate its effectiveness when paired with data smoothing techniques. Since the total load data showed daily periodic patterns, the data was averaged in six-hour intervals to help reduce noise and improve short-term predictive accuracy. First-order differencing was applied to address non-stationarity, but some residual seasonality persisted, which can be seen in the ACF plot's periodic spikes. This experiment helped highlight the model's strengths in short-term forecasting while demonstrating its limitations with seasonal data.

Statsmodels, SciKitLearn & Pandas libraries were implemented to utilize the ARIMA framework. Matplotlib was used to visualize model predictions and ACF/PACF plots to better understand the model's performance.

SARIMAX

SARIMAX was chosen due to its ability to handle both seasonal patterns and external regressors, making it well-suited for the ERCOT grid load data. The dataset exhibited strong seasonal trends on both daily and yearly scales. Given our data's strong seasonality, SARIMAX effectively captured the daily periodicity present in the hourly data.

Unlike ARIMA, SARIMAX leverages both the dependent variable's historical values and correlated exogenous variables. Based on the correlation matrix, Far West average temperature and natural gas production showed the strongest correlation with ERCOT's total load, making them ideal regressors. This allowed SARIMAX to better account for fluctuations in power demand influenced by external factors. Similar to ARIMA, the ACF and PACF plots demonstrated seasonality through periodic spikes and nonuniformity. However, by incorporating seasonal terms into the model, SARIMAX effectively modeled these recurring patterns.

Statsmodels, SciKitLearn & Pandas libraries were implemented to utilize the SARIMAX framework. Matplotlib was used to visualize model predictions, ACF/PACF, and seasonal ACF/PACF plots to better understand the model's performance.

Prophet

Prophet is Meta's flagship time series analysis and prediction model, open-sourced in the late 2010s. It aims to capture seasonal effects at multiple "human" levels & historical trend changes.

Prophet is an additive regression model with four primary components, per Meta:

1. Piecewise growth curve trend, detected by changepoints
2. Yearly seasonal component modeled using Fourier Series
3. Weekly seasonal component with dummy variables
4. Holidays

There is obvious applicability of such a model for projecting out the future time-series of load data. Seasonal decomposition shows annual, seasonal, and daily patterns that this model should be apt to capture. Unfortunately, Prophet is commonly implemented as a univariate forecasting model, although it does have features for data generation using the forecasting methodology at arbitrary intervals. We leverage this use-case for data generation of gapped data, as well as adding in a regressor – natural gas generation – to the forecast model, to experiment with expanding the model into effectively a multivariate function.

Additionally, the Prophet model results were expanded to capture best-case profit spread based on Day-Ahead (DAM) Price points and realized future load & spot prices.

SciKitLearn, Pandas, & Prophet libraries were implemented to leverage the Prophet framework, including Prophet's built-in plotting tool and Stan, the statistical language that drives the fast results of the modeling tool.

TTMs

Tiny Time Mixers is a set of ~1M parameter pre-trained transformer models, specifically tooled for use forecasting time series. From the initial publication by Ekambaram et. al: *"Tiny Time Mixers (TTM), a compact model (starting from 1M parameters) with effective transfer learning capabilities, trained exclusively on public TS datasets. TTM, based on the light-weight TSMixer architecture, incorporates innovations like adaptive patching, diverse resolution sampling, and resolution prefix tuning to handle pre-training on varied dataset resolutions with minimal model capacity."*

TTM is a high-performance zero-shot framework for time series data in both the short and medium time frames. We selected this transformer model for two primary reasons. One, it's portability for multivariate analysis (initially, the approach was to train a Temporal Fusion Transformer from scratch on multivariate data, abandoned due to compute constraints). Two, it's visualization and packaging libraries. Many transformer models use "windowing" in their processing, which requires a specific format, often agnostic of dates; in order to preserve certain dates throughout the model, this model framework was selected.

We selected open-sourced models and frameworks provided by IBM Research through the Granite Series, using the model head sourced from the Hugging Face online repository. TTMs are built on the PyTorch library, so we utilized the torch library to format and build the input datasets. We also heavily relied on IBM's TSFM library, which packages and evaluates the time series dataset. Note, some of the functions in this repo are "monkey patched" to provide the interpretable functionality necessary for this report.

LSTM

LSTMs (Long Short-Term Memory networks) are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. Unlike traditional RNNs, which struggle with vanishing gradients, LSTMs utilize memory cells and gating mechanisms to selectively retain and forget information. In this project, LSTMs were employed to forecast ERCOT grid load by leveraging historical demand, temperature, and renewable generation data. Their ability to recognize complex temporal patterns makes them well-suited for time-series forecasting.

Given the fluctuating nature of energy demand, LSTMs were trained using historical sequences with different lookback periods (24-hour, 72-hour, and 144-hour windows) to evaluate their predictive accuracy over short- and medium-term horizons. The model was optimized using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to assess performance across different timeframes. While LSTMs are computationally intensive, their ability to model non-linear relationships and long-term dependencies makes them a powerful tool for energy forecasting.

The model was implemented using TensorFlow and Keras, with data preprocessing handled through Pandas and Scikit-Learn. Time-series visualizations were generated using Matplotlib to compare actual vs. predicted values, highlighting the model's strengths in capturing demand fluctuations while also exposing challenges in handling extreme load variations.

XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful ensemble learning algorithm that uses decision trees to model complex relationships in data. It was selected for ERCOT grid load forecasting due to its ability to handle non-linear patterns and feature interactions efficiently. Unlike traditional time-series models, XGBoost does not rely solely on past values but can incorporate additional explanatory variables such as temperature, renewable generation, and market prices to improve prediction accuracy.

To evaluate its forecasting capabilities, lagged features were created using historical demand data with lookback periods of 24, 72, and 144 hours. The model was trained to minimize Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), allowing for direct comparison with deep learning approaches like LSTMs. While XGBoost is not inherently designed for sequential dependencies, its ability to capture complex patterns and handle missing data makes it a strong candidate for short- to medium-term load forecasting.

The implementation utilized the XGBoost library in Python, with Scikit-Learn for preprocessing and feature engineering. Predictions were visualized using Matplotlib, showcasing the model's performance in capturing demand trends while highlighting areas where temporal dependencies may require additional modeling techniques, such as hybrid approaches combining tree-based methods with recurrent neural networks.

Evaluation

We evaluated all model's training and test data (where applicable, classical methods do not have the former) using baseline MAE, MSE, and R^2 metrics. MAE and MSE are useful for

demonstrating the scale-dependent errors of the model: the former is useful as it is interpretable in units of MWs, while MSE penalizes significant errors from the true value much more, which is often very important for power prediction – the grid often is most stressed by large swings in load. R^2 is informative of how our model compares to a baseline predicting the mean of the dataset. However, this metric is flawed in this use-case as the time-series is seasonal and exhibits nonstationary behavior.

We also include MAPE (Mean Average Percentage Error) for select models where scaling occurs to compare the relative error in performance between the models when normalized by the scale of the data. This is useful for select models where scaling is crucial.

RMSE is used for both models. It is similar to MSE but provides an error measurement in the same units as the data, allowing for better assessment of model fit and explainability.

Results

Training Set Results:

	ARIMA	**SARIMAX**	**Prophet (Univariate)**	**Prophet (Multivariate)**	**TTM (Scaled)**	**LSTM**	**XGBoost (scaled)**
MAE	2414	397	4415.7	4416.9	–	1872.2	-
MSE	12,796,289	781,823.6	35,304,171.7	35,313,991.8	0.35*	10,232,231	-
R^2	0.8868	0.9939	0.7169	8.597	–	.712	-
MAPE (%)	5.16	0.82	8.5933	0.7167	–	3.25	-

*See challenges

Test Set Results (3-day Forecast*):

	ARIMA	**SARIMAX**	**Prophet (Univariate)**	**Prophet (Multivariate)**	**TTM**	**LSTM**	**XGBoost**
MAE	2120	750	2389	2481	3291.3	2139.07	0.0362
MSE	7,197,149	911,993.8	7,848,410.3	9,715,179.3	16,495,005.3	7145.24	0.0610
R^2	0.4292	0.8747	0.1179	-0.0919	0.9643	.43	0.8174
MAPE (%)	4.89	1.77	5.30	5.46	5.5000	4.21	4.67

RMSE	2682.8	955	2801.5	3116.9	4061	1523	2543

*See other (One, Six Day) Forecast Results in PPT

ARIMA struggled when applied directly to hourly raw data due to its inability to account for seasonality. However, performance improved significantly after smoothing the data using six-hour averages. While first-order differencing helped reduce seasonality effects, the ACF and PACF plots still indicated persistent periodic patterns, limiting ARIMA's long-term forecasting capabilities. Despite its simplicity and fast computation, ARIMA's predictive performance remained limited, achieving an R^2 value of 0.43 for three-day forecasts and an RMSE of around 2,600.

In contrast, **SARIMAX** effectively handled the dataset's daily periodicity and delivered accurate predictions up to six days ahead. While more computationally intensive than ARIMA, SARIMAX effectively captured nonlinear patterns in the data. Through model tuning, the AR (P) and MA (Q) parameters were both set to two, with first-order differencing applied. For the seasonal terms, P and Q were set to two and one, respectively with seasonal differencing set to 1. Setting both seasonal parameters to 2 yielded similar but slightly worse results. A seasonal period of 24 effectively modeled daily periodicity, while attempting to model yearly seasonality with a period of 365 × 24 proved computationally excessive. While more computationally intensive, SARIMAX outperformed ARIMA in next day, three day, and six day forecasts. Its best forecast achieved R^2 value of 0.87 when forecasting up to three days ahead, with a relatively low root mean squared error (RMSE) of around 955.

Prophet adequately predicted the data across the one, three, and six day time horizons. However, the model is built to capture cyclic events and smoothes out the noise in the intraday time period. As a result, the sine-like waves capture the data within roughly 5% MAPE for all time horizons. This is unique to the other models trained, which generally differ in performance for different time horizons. We also found that the model does a superb job of capturing seasonal trends as well, though the "changepoints" that denote trend changes inferred in the model impute a slightly decreasing seasonal trend. Tuning the changepoints, we were unable to remove this "decreasing" trend and preserve model performance. The Prophet model was utilized to generate artificial data on the natural gas variable, which was found to be most correlated with the load. The model was then trained again on this additional independent regressor and resulted in slightly worse (less generalizable) performance over the three and six day time horizons.

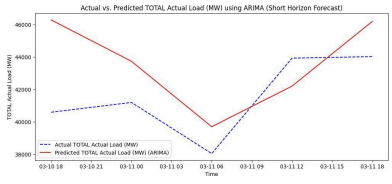
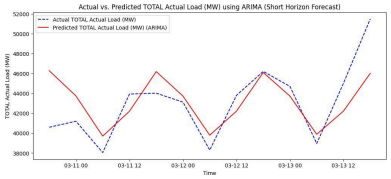
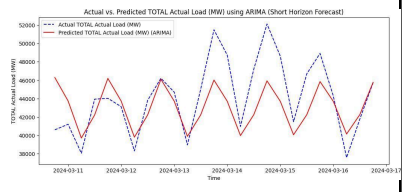
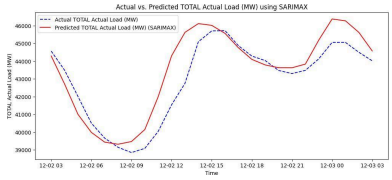
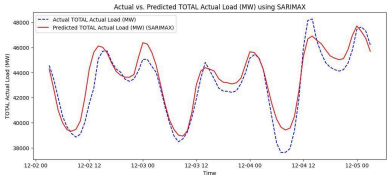
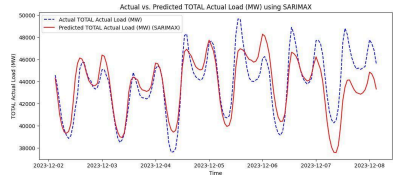
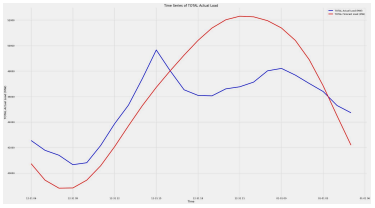
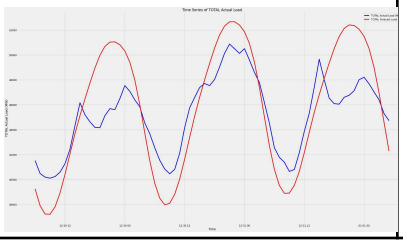
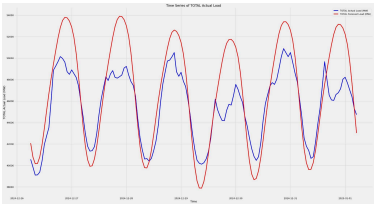
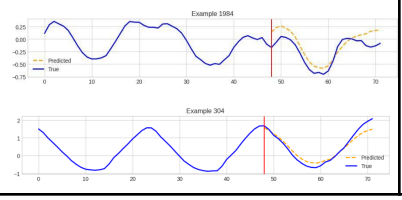
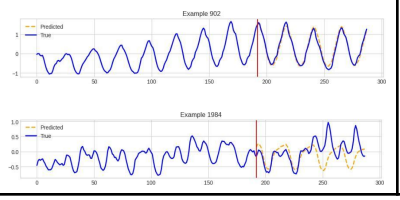
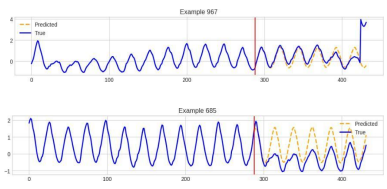
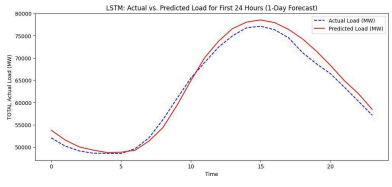
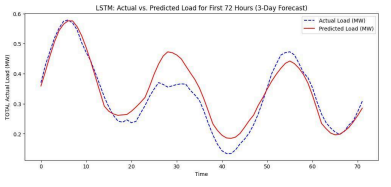
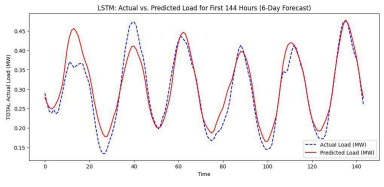
Tiny Time Mixers (TTMs) are a set of transformer models that are pre-trained on time series energy and other data with roughly 1M+ model parameters. The head of the model was trained

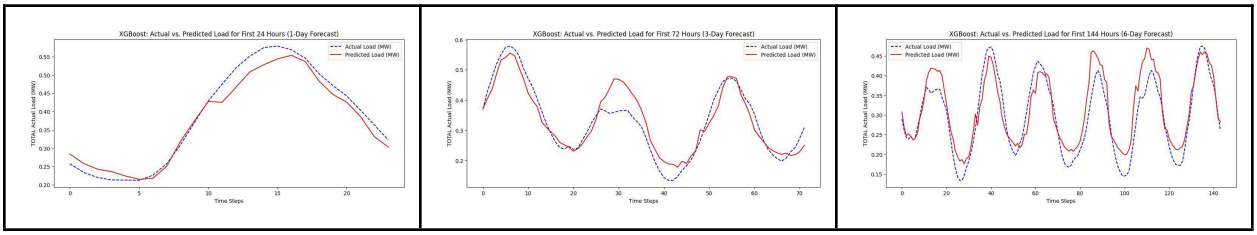
on all independent variables (excluding regional load and price) of our generated dataset, with target variables as the total load and real-time price. Though we performed only zero-shot training, the model framework allows for easy assessment of multiple test datasets from the input dataset (built on the torch framework). As a result, we were able to visualize the generalizability with multiple plots of this transformer model predictions for one, three, and six day forecasts. We found a scaled MAE of 0.11, or roughly 3000W, and a significantly higher MSE than other models. The poor performance of the TTM seems due to its inability to capture day-to-day swings on longer time horizons - the one day prediction improves significantly, though is still quite prone to large errors in the aggregate (across multiple test sets), i.e. the MSE is significant.

LSTMs (Long Short-Term Memory networks) demonstrated strong performance in capturing short-term dependencies in the ERCOT grid load forecasting task. By leveraging memory cells, LSTMs effectively retain information from previous time steps, making them particularly useful for modeling sequential energy demand patterns. The model was trained with varying input window lengths (one, three, and six-day sequences), and its performance was evaluated using MAE, RMSE, and R^2 metrics. While LSTM performed well in next-day forecasting, achieving an R^2 value of 0.31, its accuracy declined for longer horizons, likely due to error accumulation over time. The model effectively learned intraday fluctuations but struggled to generalize across multiple days, particularly for six-day forecasts, where RMSE values increased significantly. Despite these limitations, LSTMs outperformed ARIMA and Prophet in capturing non-linear trends and provided a more flexible architecture for multivariate forecasting.

XGBoost (Extreme Gradient Boosting) emerged as a competitive alternative by efficiently modeling complex relationships in the data through decision tree ensembles. Unlike LSTM, which learns patterns sequentially, XGBoost applies gradient boosting to iteratively improve weak learners, making it robust to noise and feature interactions. The model was trained on lagged features and key exogenous variables, including regional energy generation and natural gas prices. XGBoost achieved strong predictive performance, particularly for short and medium-term forecasts, with an R^2 value of 0.85 for one-day predictions and 0.81 for three-day forecasts. Its ability to capture feature interactions enabled it to maintain accuracy over longer horizons better than LSTM, though performance slightly degraded for six-day predictions. The model's efficiency, fast inference speed, and relatively low computational requirements made it a practical choice for real-time energy demand forecasting.

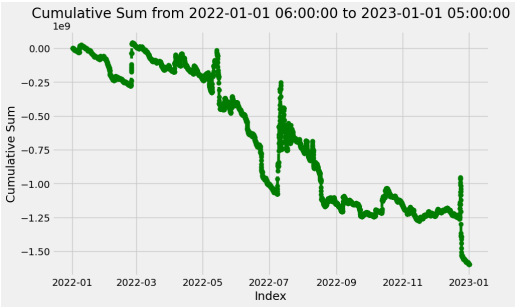
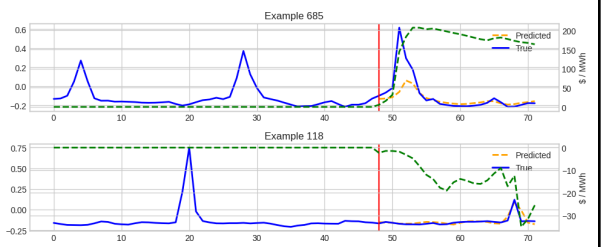
Plots

Next Day Forecast	3-Day Forecast	6-Day Forecast
ARIMA		
		
SARIMAX		
		
Prophet		
		
TTM (Test Set 1&2)		
		
LSTM		
		
XG Boost		



Real-World Assessment

Prophet and Tiny Time Mixer were additionally assessed in terms of the Day-ahead-market spread they would allow us to capture. Prophet load forecasting was used to assume total market capture to bound best case profit spread, while TTM was retrained on the data with Real-time price forecasting as a target. Both models underperformed and it is noteworthy that the RT price and total load have a very weak correlation (0.2).

Prophet - Cumulative Day-Ahead Profit (best case)	TTM - Cumulative 3-Day ahead profit
	

Challenges and Solutions

When reviewing the raw data, there were often missing time steps and incomplete hourly records over the three year period. To address this, developed functions to iterate through each dataset and expand them to account for every hour from 2022 to 2024. This was challenging because different datasets accounted for daylight savings time (DST) differently. To overcome this, these timestamps were translated to Coordinated Universal Time (UTC) which served as a constant frame of reference. Furthermore, 2024 was a leap year which required the functions to account for an extra day. All time-related operations, including additions, translations, and calendar consistency, were managed using the pandas date-time format.

Once the dataset was expanded, there were still missing values in the dataset. To handle these gaps, we implemented methods to fill in the missing values using linear interpolation. This approach provided a simple yet effective means to maintain the continuity of the data, ensuring that subsequent analyses and model training could proceed without interruption.

We initially planned to utilize a transformer framework for time series analysis – TFT, or Temporal Fusion Transformer – but ran into significant constraints with compute. The framework runs on PyTorch and we wanted to train the transformer network on multivariate data, at least two or three times the size of the univariate dataset. Because the constraints for this model became too great, we pivoted to utilizing a pre-trained model, TTM, for the transformer network. Because this model is trained on time series datasets, including energy analysis datasets, it trains within the compute constraints we had (~20 GB of RAM) and results in a decently performing model over the assessed time frames (see the results section). However, the documentation for the TTMs proved opaque for descaling the data with HuggingFace's trainer framework; were only able to produce scaled MSE for the training set.

Extensions and Future Work

As a potential extension, we plan to explore the use of Generative Adversarial Networks (GANs) for data imputation. Unlike linear interpolation, which assumes a linear progression between data points, GANs could model complex, non-linear relationships in the data. This approach would better capture the underlying dynamics of the time series, leading to improved accuracy in forecasting and a more robust handling of missing values in future iterations of the project.

Additionally, the results of the transformer network seem to be particularly poor. It could be worth examining the framework or pretrained model further to assess a fundamental disconnect or misapplication of the windowing methodology utilized for this time series. The data is scaled when processed, so popping the hood on the scaling utilities may reveal such a disconnect as well. Additionally, performing a fit on the data to find hyperparameters finely tuned would be a proper next step.

Conclusion

Ultimately, we found the results of several of these methods to be noteworthy. SARIMAX did a particularly good job of learning daily and multi-day noise. XGBoost did a good job capturing this noisy component of the data as well, though it is not explainable. Our other classical methods explored, particularly TTM, do a good job of capturing cyclicality in the time series data, but do not properly model day-to-day variations in the total load on the grid. Ultimately, we find that none of these models are tuned or explainable enough to use for practical purposes such as load planning and price.

References

Data

https://www.gridstatus.io/live
https://www.eia.gov/electricity/wholesalemarkets/ercot.php
https://modoenergy.com/research/ercot-battery-energy-storage-buildout-projection-july-2024-12-gw-18-gw-2025-approval-energize-synchronize-interconnection-queue
https://www.ercot.com/gridinfo/resource/2023

Models

https://huggingface.co/
https://www.ibm.com/granite/docs/models/time-series/
https://facebook.github.io/prophet/docs/quick_start.html
https://github.com/ibm-granite/granite-tsfm
https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html
https://www.statsmodels.org/stable/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html
https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html
https://xgboost.readthedocs.io/en/stable/