Electricity Pricing Forecasting: A Comparison of MLP, TimesFM, and Clustered MLP

Hongyu Li

August 28, 2025

Background

In the ERCOT electricity market, real-time prices play a central role in coordinating supply and demand. These prices are determined through the Security-Constrained Economic Dispatch (SCED) process, which calculates the marginal cost of delivering electricity to each node on the grid every five minutes. The result is a set of Locational Marginal Prices (LMPs) that reflect system constraints, load conditions, and generation costs in near real time. Among the different SCED outputs, the System Lambda — which approximates the marginal system-wide price — serves as a useful, simplified representation of the full SCED LMP signal.

The project focuses on time-series forecasting of SCED (Security-Constrained Economic Dispatch) prices, an essential component in the economic operation of electricity markets. The SCED system helps balance electricity supply and demand by determining the optimal generation schedule and setting prices accordingly.

The real-world dataset used for this project provided crucial insights into the SCED price predictions. SCED LMP (Locational Marginal Pricing) was able to generate a profit of $95,000, reflecting the actual pricing behavior in the energy market. Similarly, SCED System Lambda, which is a system used to approximate SCED prices, generated a profit of $90,000, showing that it closely mirrors actual SCED prices. However, DAP System Lambda showed much lower profitability at only $45,000. Given these results, SCED System Lambda was chosen as the benchmark for approximating actual data in this project. The goal of the project is to build machine learning models that can predict SCED prices to maximize profit and assist decision-makers in the energy market.

Data Processing

The first step in the data processing pipeline was aligning SCED and DAP data, as they had different time resolutions: SCED was recorded at 5-minute intervals, while DAP was hourly. To address this, DAP data was interpolated to match the 5-minute intervals of SCED. Next, all features (DAP, solar, wind, natural gas, load) were normalized using StandardScaler, and SCED values were normalized by applying their mean and standard deviation.

The data was then prepared for modeling using a sliding window approach, with a 24-hour window (288 time steps) used to predict the next 1 hour (12 time steps) or 8 hours (96 time steps) of SCED. The dataset was split into training (2019-2021), evaluation (2022), and test (2023) sets. The test set starts from the second date in 2023 to utilize the first date as input. For MLP models, the input data was flattened from a 3D array to a 2D array, while for LSTM models, the data was kept in its 3D form to capture sequential dependencies.

While MLP and LSTM models used a standard train-test split approach, TimesFM operated differently by processing the entire dataset continuously. TimesFM used a 7-day window to predict the next 10 hours without needing to split the data into training and testing sets, which allowed it to leverage all available data and capture longer-term trends in SCED prices.

Methods

MLP

MLP Overview

The Multi-Layer Perceptron (MLP) is a type of neural network that consists of multiple fully connected layers. It uses non-linear activation functions, such as ReLU, in the hidden layers to model complex relationships. For this project, the MLP was used to predict SCED values over 1-hour (12 time steps) or 8-hour (96 time steps) horizons, using a sliding window of 24 hours (288 time steps) of historical data as input. Features included DAP, solar, wind, natural gas, and load.

MLP Architecture

The Multi-Layer Perceptron (MLP) model used in this study was optimized through grid search across several hyperparameters, including the number of neurons per layer, learning rate, hidden layer configurations, and batch size. The model's input consisted of 24 hours of historical data (288 time steps) for each feature, with 2 key features (SCED and DAP), resulting in an input size of num_samples $\times$ 576 (where $576 = 2 \times 288$). The output size varied based on the prediction horizon: 12 time steps for 1-hour predictions or 96 time steps for 8-hour predictions.

Hyperparameters tested included:

- **Number of Neurons**: 128, 256, 512.
- **Learning Rate**: 0.0001, 0.0005, 0.001.
- **Hidden Layers**: Configurations such as [128, 128], [256, 256], [512, 512], and [256, 256, 256].
- **Epochs**: 30, 35, 40.
- **Batch Size**: 32.

After evaluating all combinations, the best configuration was found to be an MLP with 3 hidden layers of 256 neurons, a learning rate of 0.001, and a batch size of 32. This model achieved significant performance, producing a profit of 47K in SCED price predictions. These hyperparameters were later applied in multi-feature and clustered MLP models, improving prediction accuracy and enhancing forecasting performance. The grid search process ensured the model was well-tuned for optimal SCED price prediction.

MLP Results

Grid search results for hidden layer [256, 256, 256] are shown in Table 1. After experimenting with various hyperparameters, the best configuration was found to be an MLP with 128 input neurons, 0.0001 learning rate, 3 hidden layers of 256 neurons each, 30 Epochs with a batch size of 32, achieving a profit of 47K averaging 2 rounds. This model was trained on the 24-hour window of past data to predict future SCED prices, and it was later applied to multi-feature and clustered prediction tasks.

Round	n_neurons	Learning Rate	Hidden Layers	Epochs	Batch Size	Profit
0	128	0.0001	256, 256, 256	30	32	48779.67478
1	128	0.0001	256, 256, 256	30	32	44648.81318
0	128	0.0001	256, 256, 256	30	64	31612.29684
1	128	0.0001	256, 256, 256	30	64	42945.96879
0	128	0.0001	256, 256, 256	35	32	32293.67971
1	128	0.0001	256, 256, 256	35	32	38287.26708
0	128	0.0001	256, 256, 256	35	64	45217.00050
1	128	0.0001	256, 256, 256	35	64	43265.50989
0	128	0.0001	256, 256, 256	40	32	42897.53013
1	128	0.0001	256, 256, 256	40	32	36942.77746
0	128	0.0001	256, 256, 256	40	64	38010.72213

Table 1: MLP Model Hyperparameters and Results

Conclusion

The MLP model provided reliable SCED price predictions, yielding a profit of 47K. It demonstrated the model's ability to capture time-series patterns, although further optimization could improve its accuracy, especially with the addition of other features or clustering techniques.

LSTM

The Long Short-Term Memory (LSTM) model was also tested for this task due to its ability to model long-term dependencies. However, despite its potential advantages, the LSTM model did not perform well on this dataset, primarily due to extended training times and

insufficient predictive accuracy. Although LSTM is known for capturing sequential dependencies, its performance was below that of the MLP model in this case, making it less suitable for the given forecasting task.

TimesFM Model

TimesFM Overview

TimesFM is a time-series forecasting model that utilizes a factorization machine framework to capture interactions between time steps in the data. Unlike MLP and LSTM, TimesFM is specifically designed to handle both short-term and long-term dependencies by factoring in interactions between different time steps. In this project, TimesFM was used to predict 10 hours of future SCED values based on the past 7 days (2016 time steps) of data, enabling the model to account for periodic behaviors and trends over a longer horizon.

One of the advantages of TimesFM over MLP and LSTM is that it doesn't require a traditional training-validation-test split. Instead, TimesFM is trained continuously on the entire dataset, adapting to the latest trends and capturing both immediate and long-term patterns in SCED pricing.

Loading The Model

The model is loaded using the TimesFm class from the timesfm library. It is pre-trained and retrieved from a checkpoint stored in the Hugging Face repository (google/timesfm-2.0-500m-pytorch), which is either cached or downloaded locally when accessed for the first time. The model configuration is specified through the TimesFmHparams, where key parameters are defined: the backend is set to GPU for faster computations, the batch size is set to 512 samples per batch, the horizon length determines the number of future time steps to forecast, the num layers defines the model's depth (50 layers in this case), and the context length specifies the number of historical time steps used for making predictions. Additionally, positional embeddings are set to False, indicating that no extra positional encoding is applied. Example in Figure 1.

TimesFM Single Feature Predictions

2023 data is split into 7-day windows and each 7-day window is passed into the model to make predictions for the next 10 hours.

Initially, TimesFM was used to predict SCED values solely based on the SCED data itself, but the results were suboptimal, with a profit of only 27K. We then shifted to predicting the difference between SCED and DAP (Day-Ahead Price), which produced a significant improvement, yielding a profit of 70K.

TimesFM Multi Feature Predictions

The model also explored using covariates to enhance predictive performance. There are four types of covariates: dynamic numerical, dynamic categorical, static numerical, and static categorical. For this project, dynamic numerical covariates such as DAP, solar, wind,

```python
# Loading the timesfm-2.0 checkpoint:
# For Torch
tfm = timesfm.TimesFm(
      hparams=timesfm.TimesFmHparams(
          backend="gpu",
          per_core_batch_size=512,
          horizon_len=look_ahead,
          num_layers=50,
          use_positional_embedding=False,
          context_len=look_back,
      ),
      checkpoint=timesfm.TimesFmCheckpoint(
          huggingface_repo_id="google/timesfm-2.0-500m-pytorch"),
  )
```

Figure 1: Loading TimesFM

```
{
    'country': ['FR', 'FR'],
    'inputs': [[53.48, 51.93, 48.76, 42.27, 38.41], [48.76, 42.27, 38.41, 35.72, 32.66]],
    'gen_forecast': [[76905.0, 75492.0, 74394.0, 72639.0, 69347.0, 67960.0, 67564.0], [74394.0,
72639.0, 69347.0, 67960.0, 67564.0, 67277.0, 67019.0]],
    'week_day': [[3, 3, 3, 3, 3, 3, 3], [3, 3, 3, 3, 3, 3, 3]],
    'outputs': [[35.72, 32.66], [32.83, 30.06]],
}
```

Figure 2: Covariates Example

natural gas, coal, hydro, and load were used. Covariates data includes both historical data and data in the horizon window to be able to make predictions, see example in Figure 2.

Two modes of prediction were considered:

- **Option 1:** First, TimesFM forecasts SCED, then a linear model is used to regress the residuals on the covariates ("timesfm + xreg").
- **Option 2:** A linear model is fit on the time series and covariates first, followed by forecasting the residuals using TimesFM ("xreg + timesfm").

In this project, Option 1 was employed, where SCED predictions were made first, and the residuals were modeled using other features (DAP, solar, wind, and load). This allowed TimesFM to refine its predictions by incorporating the influence of these external covariates.

The inclusion of covariates progressively improved the model's results. Using only SCED and DAP as covariates resulted in a profit of 56K, and adding solar and wind increased the profit to 60K. The best results were achieved by predicting the difference between SCED and DAP, along with all available covariates, yielding a profit of 67K to 69K. However, the highest profit of 70K was obtained by solely using the difference between SCED and DAP for prediction.

In conclusion, while TimesFM showed potential when using SCED alone, incorporating the difference between SCED and DAP along with external covariates yielded the most accurate predictions, achieving a profit of 70K. This demonstrated the effectiveness of using TimesFM with covariates for time-series forecasting in electricity pricing.

Predictions	Dynamic Numerical Covariates	Profits
SCED	None	27K
Diff (SCED - DAP)	None	70K
DAP	SCED	56K
SCED	DAP	58K
Diff (SCED - DAP)	DAP	63K
SCED	DAP, load	59K
SCED	DAP, load, solar, wind	59K
SCED	DAP, coal, hydro, nuclear, storage, solar, wind, natural gas, other, load	60K
Diff (SCED - DAP)	load	69K
Diff (SCED - DAP)	DAP, load	68K
Diff (SCED - DAP)	DAP, coal, hydro, nuclear, storage, solar, wind, natural gas, other, load	67K

Table 2: TimesFM Results with Covariates and Profit

TimesFM with Finetuning Models

To improve the forecasting accuracy of TimesFM, we implemented a two-stage pipeline in which the raw predictions from the pretrained TimesFM model are passed through supervised learning models for correction. Specifically, we explored using a Multi-Layer Perceptron (MLP) and a Convolutional Neural Network (CNN) to finetune the 96-step predictions produced by TimesFM. These downstream models are trained to incorporate structured covariates alongside the base predictions to learn and apply correction patterns.

Base TimesFM Setup

TimesFM was used as a base model to generate initial predictions for real-time prices (RTP) in the ERCOT market. Instead of predicting RTP directly, the model predicts the residual between RTP and day-ahead prices (DAP), defined as `residual = RTP - DAP`. The model takes a 7-day window of historical residuals as input and predicts the next 10 hours (i.e., 120 time steps at 5-minute resolution) of residuals.

The input data is preprocessed by taking the difference between RTP and DAP to obtain the residual. This residual time series is then passed into the TimesFM model using Hugging Face's implementation. No additional covariates such as wind, solar, load, or gas are included in the base model. The output from TimesFM is a forecast of future residuals, which are later added back to the corresponding DAP values to recover the final RTP predictions.

The model is run in evaluation mode without training, using pretrained weights from the Hugging Face TimesFM checkpoint. Each batch consists of a 7-day rolling window (2016 time steps) to predict the 10-hour residual. The batch is shifted step-by-step across the test period to generate rolling predictions.

MLP-based Finetuning

The MLP model takes as input the 10-hour residual predictions generated by TimesFM along with a set of covariates. These covariates include the past 24 hours (288 time steps) of SCED price, DAP price, load, and fuel mix features, flattened into a single vector. The goal is to learn a mapping from this expanded feature space to corrected residual predictions. The output of the MLP is then added back to the DAP price to produce the final SCED

prediction. The model consists of three fully connected layers with ReLU activations. This setup achieved a final test-time arbitrage profit of approximately $50,000.

CNN-based Finetuning

The CNN model also operates on the 10-hour TimesFM residual predictions, with the same covariates as the MLP model. These features are passed in their 2D format (time $\times$ feature) into a convolutional architecture, which captures local temporal patterns through sliding filters. The CNN consists of two 1D convolutional layers followed by a fully connected output layer. Similar to the MLP, the predicted residuals are added back to the DAP price, and the model achieved a comparable final profit of around $30,000.

Normalized Data for TimesFM

In addition to using raw residuals, we explored the effect of applying normalization to the TimesFM inputs. Two normalization techniques were tested: z-score normalization and Gaussian normalization, both applied to the input features before feeding them into the base TimesFM model (without any second-phase models like MLP or CNN). Both methods yield an arbitrage profit of approximately $70,000, similar to unnormalized TimesFM baseline.

MLP Finetuning on Normalized TimesFM

We also trained MLP models on top of TimesFM outputs generated from normalized inputs. That is, TimesFM was first trained on z-score or Gaussian-normalized inputs, and the resulting residual predictions were passed to the MLP model. With z-score normalization, the TimesFM + MLP pipeline achieved a profit of $42,800, while Gaussian normalization yielded $47,300. This suggests that although normalization helps the base TimesFM model, its benefits are less consistent when used in conjunction with second-phase MLP models.

Clustered MLP

Clustering for Time-Series Forecasting

Clustering is used to segment the data into distinct groups where the properties within each group are similar, but the properties between the groups differ. This technique is especially useful when we want to predict outcomes based on groups of similar observations, as it helps to create models tailored to specific patterns. In this case, we applied K-means clustering to the dataset, creating five clusters that each represent different operational scenarios or market conditions. We aim to create a model for each cluster to make more accurate predictions.

Cluster Overview

The clusters identified represent different operational conditions in electricity markets. For example, Cluster 0 corresponds to off-peak hours with stable renewable generation, Cluster 1 captures rare events with extreme price spikes, and Cluster 2 represents mild generation and load during early morning hours. Cluster 3 is characterized by peak hours with high wind and

natural gas generation, while Cluster 4 captures midday hours with high solar production. These clusters were identified based on the SCED (System Control and Economic Dispatch) price and other covariates such as wind, solar, and natural gas generation, which help define the market conditions during each time period.

Training the Clustered MLP Model

Once the clusters were identified, we used a continuous 24-hour sliding window to create input sequences for training. The key difference from the initial MLP models is that the target label for each observation belongs to its respective cluster. In other words, we trained separate models for each cluster while using continuous data from the previous 24 hours to predict the SCED price for that time. This approach allowed us to use the past 24 hours of data (e.g., DAP, solar, wind, and load) for each time step, and predict the price for the next time step. This way, the model could adapt to the specific conditions of each cluster, leveraging the historical patterns and relationships inherent to each group.

The training process for each cluster involved splitting the data into training, validation, and test sets, just like in previous models. After training the model on each cluster's specific data, the results were evaluated based on prediction accuracy and compared to the baseline performance of the model without clustering.

Results

The clustering approach significantly improved the predictive performance of the model. By training separate models for each cluster, the results showed an improvement in the SCED price prediction accuracy, achieving a 53K result compared to 48K without the clustering approach. This demonstrates that clustering can help improve forecasting by tailoring models to the specific patterns and behaviors found in different segments of the data.

In conclusion, clustering allows for more accurate predictions by accounting for the inherent differences in market behavior across different operational periods, ultimately leading to better performance in forecasting SCED prices.

PCA

Overview

To reduce the input dimensionality while retaining core information relevant to SCED prediction, we apply Principal Component Analysis (PCA) as a preprocessing step. The motivation is to compress the high-dimensional feature space into a smaller set of uncorrelated components, reducing the complexity for TimesFM while preserving the underlying structure of the data. This approach also allows us to test whether TimesFM can still perform well with compressed, decorrelated features.

Technical Setup

The PCA pipeline begins with normalizing all features, followed by scaling the SCED feature by a factor of 10 to emphasize its importance during dimensionality reduction. PCA is then

applied across all features from 2019 to 2023 to compute the projection basis. The full input is projected onto the top 11 principal components, which are then treated as time-series features and fed into TimesFM. After prediction, the output is reconstructed back into the original feature space using the inverse PCA transform. The SCED values are then scaled back (divided by 10) and denormalized for evaluation.

Results

We evaluate several PCA configurations by varying the number of components, normalization scope, prediction horizon, and prediction target (SCED vs. residual). Key results are summarized below:

PCA Training Data	SCED ×10	# Components	Prediction Target	Horizon	Profit (USD)
All data (2019–2023)	Yes	11	SCED	1-step	88.0K
Train only (before 2023)	Yes	11	SCED	1-step	88.36K
Train only (before 2023)	Yes	3	SCED	1-step	88.29K
Train only (before 2023)	Yes	3	SCED	10-hour	29.8K
Train only (before 2023)	Yes	3	Diff (SCED - DAP)	10-hour	68.8K
Train only (before 2023)	Yes	12	Diff (SCED - DAP)	10-hour	68.5K
Train only (before 2023)	No	9	Diff (SCED - DAP)	10-hour	51.5K

Table 3: PCA Forecasting Results Across Settings

These results highlight that using 11 components preserves performance, while predicting residuals rather than SCED significantly improves profits under multi-step forecasting.

Results and Conclusion

Our experiments evaluated multiple forecasting approaches, including MLP, Clustered MLP, TimesFM, and a PCA-enhanced TimesFM pipeline.

The MLP model achieved a profit of 47.3K when trained on normalized historical SCED and DAP values. Adding clustering improved performance: by training a separate MLP for each cluster of time series, the Clustered MLP model captured localized patterns more effectively and reached a higher profit of 53.1K.

TimesFM, a transformer-based model, outperformed both MLP and Clustered MLP. By predicting the difference between SCED and DAP prices using dynamic covariates like DAP and load, TimesFM achieved a maximum profit of 70K across a 10-hour forecasting window. This makes TimesFM the best-performing model under standard multi-step prediction settings.

We also explored a PCA-enhanced pipeline that projects all features from 2019–2023 into 11 principal components after normalization, with SCED scaled by a factor of 10 to increase its weight. These components were forecasted using TimesFM, and predictions were reconstructed back to the original space. This PCA setup achieved a profit of 88.36K, but only under a 1-step-ahead prediction setting. When evaluated on a full 10-hour prediction horizon, PCA-based forecasting achieved up to 68.8K in profit—comparable to but not exceeding TimesFM's best performance.

In summary, while PCA helps boost short-term prediction accuracy, it does not consistently improve long-horizon forecasts. TimesFM remains the most effective model overall for multi-hour SCED price forecasting.

Future Exploration

In future work, several directions for improvement and exploration can be considered to enhance the predictive power and flexibility of the models:

Incorporating Clusters as Categorical Features to the TimesFM Model

One avenue for improving the TimesFM model is incorporating clusters as categorical features. By introducing the cluster information, TimesFM can better differentiate between distinct market conditions, improving its forecasting accuracy. This can help the model adapt to the specific patterns within each cluster.

Fine-Tuning the Top Layers of the TimesFM Model

Another potential improvement is fine-tuning the top layers of the TimesFM model. This would allow the model to focus on critical features and capture more relevant nuances in the data. Fine-tuning these layers could optimize performance and better adapt to evolving trends in SCED prices.

Adding MLP or Other Models on Top of the TimesFM Model

A further exploration is stacking MLP or other models on top of TimesFM. By combining the strengths of both models, we can enhance the prediction accuracy. This hybrid approach could allow TimesFM to capture long-term trends, while the MLP model refines predictions by identifying intricate relationships within the data.