Project Improvements and Value Addition

By Data Prophets

Prakhar Dungarwal (pd2782)
Somit Jain (sj3396)
Shashwat Kumar (sk5520)
Devdatt Golwala (drg2172)

For Fortune Tellers

Mengxi Liu (ml5189)
Jie Hu (jh4741)
Han Qiang (hq2218)
ChengHsin Chang (cc5211)

Initial Agenda ~ Fortune Tellers

Introduction

Data

EDA

Seasonal and Trend Decomposition

Variables

Modeling and Result

Potential Improvement

Enhanced Approach & Value Created by Data Prophets

- **High-frequency, real-world data is noisy and irregular:** With 370 clients and 15-min intervals, the data has spikes, seasonality shifts, and DST anomalies — which confuse traditional models like LSTM and untuned DeepAR.
- **Smoothing revealed real usage patterns:** Applying Savitzky-Golay filtering reduced noise and exposed long-term trends crucial for accurate forecasting.
- **Fixed flawed modeling logic:** Corrected DeepAR's train-test leakage, restoring real-world forecasting validity.
- **Used models suited to time-series dynamics:** BiLSTM captured turning points better than LSTM by learning from past and future context.
- **Matched models to client behavior:** Prophet worked well for consistent, seasonal users; Chronos handled irregular, volatile patterns.
- **Reduced MAPE from 166% to under 5%:** Delivered forecast accuracy that's actually usable for energy planning and load management.
- **Made forecasting production-ready:** Clean inputs, smart models, and interpretable outputs turned this from a school project into a forecasting tool that solves real utility planning challenges.

Introduction
Fortune Tellers

We focus on developing a machine learning model, including Linear Regression, LSTM, and DeepAR for electricity consumption prediction using data from clients in Portugal, ranges from 2011 to 2014.

Problem

What's the electricity usage in Portugal after 2014?

Objective

The Electricity Usage Prediction

From:

raw data from the https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014

To:

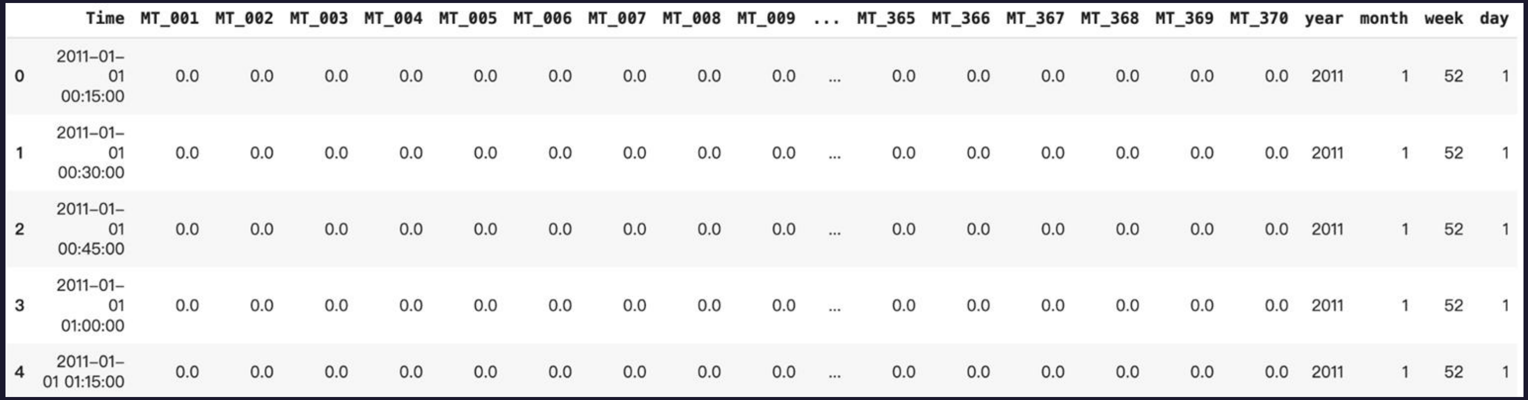
Prediction from trained models

Value Creation

- Standardize a robust methodology
- Benefit stakeholders, including energy providers, policy makers
- Ensure stable electricity provision

Data

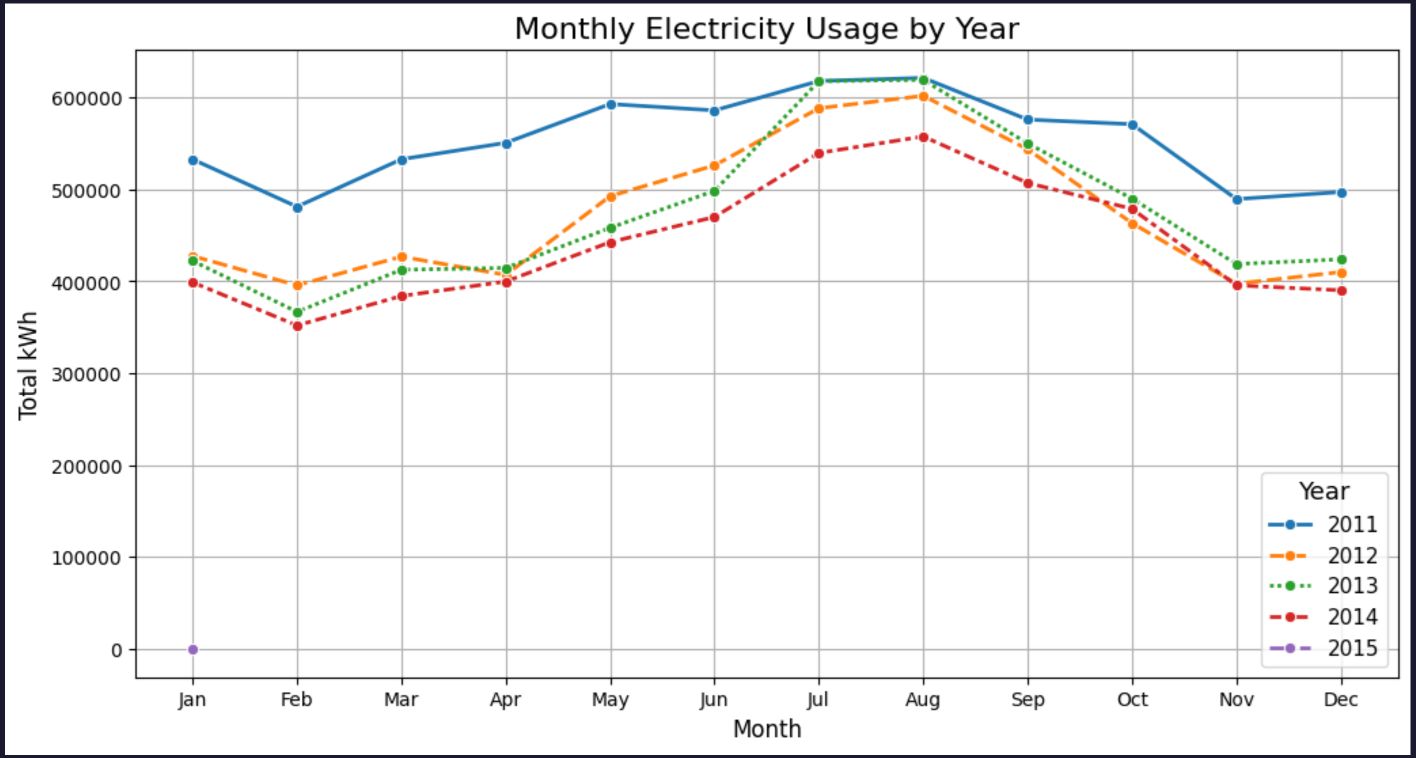
- Electricity consumption records for 370 clients in Portugal from 2011 to 2014 in a 15 minute interval timestamp
- Accounts that were created after 2011 have a value of 0
- No missing value
- Contains 140, 256 rows and 375 columns after adding temporal attributes
- Source: https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014



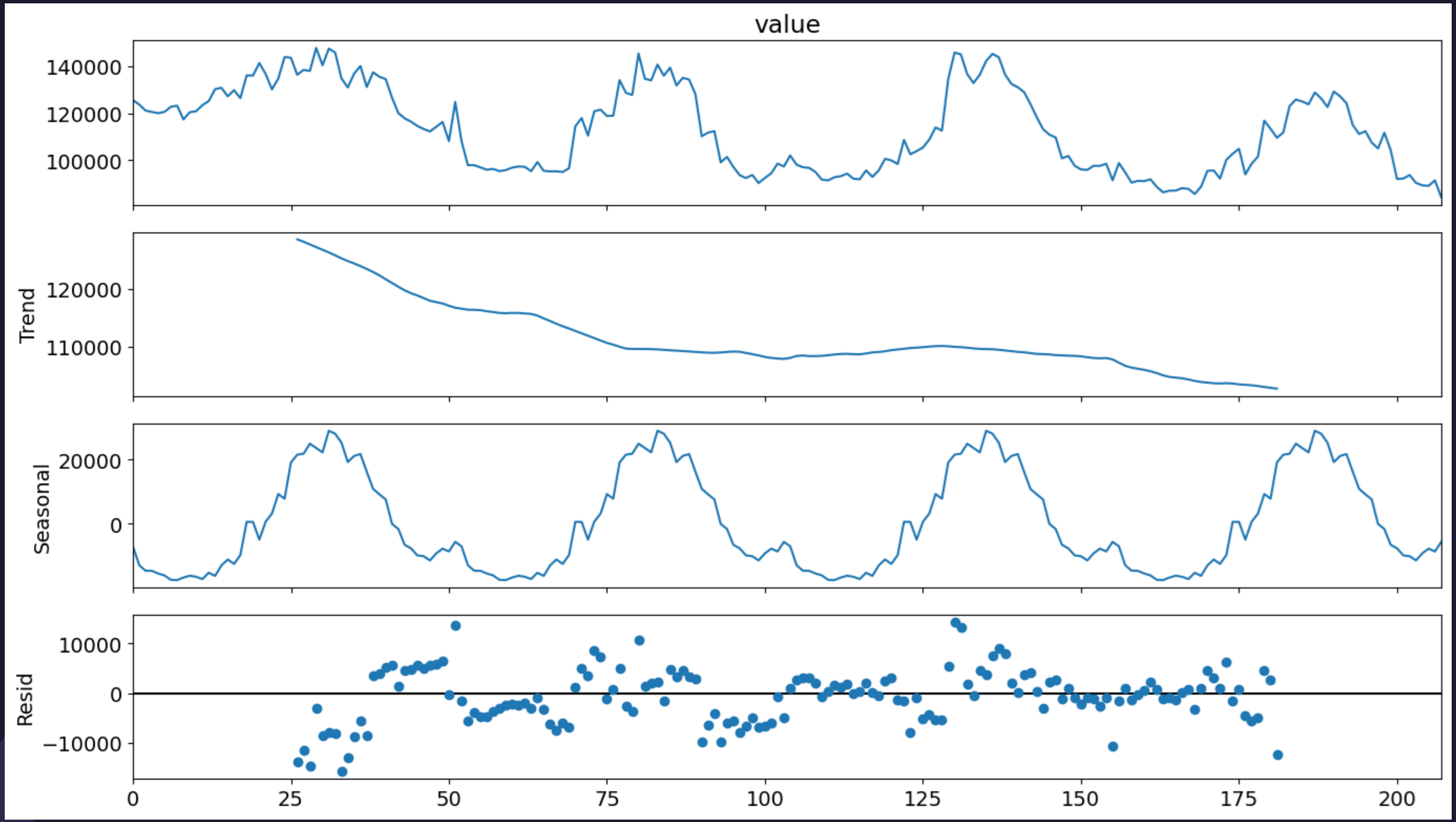
	Time	MT_001	MT_002	MT_003	MT_004	MT_005	MT_006	MT_007	MT_008	MT_009	...	MT_365	MT_366	MT_367	MT_368	MT_369	MT_370	year	month	week	day
0	2011-01-01 00:15:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2011	1	52	1
1	2011-01-01 00:30:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2011	1	52	1
2	2011-01-01 00:45:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2011	1	52	1
3	2011-01-01 01:00:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2011	1	52	1
4	2011-01-01 01:15:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2011	1	52	1

Elementary Data Analysis

- Seasonal pattern, with higher electricity usage in summer(June-August) throughout years, lower usage in winter months.
- A few outliers in winter, demonstrating occasional spikes.
- Year 2011 had considerably higher electricity usage than other years.



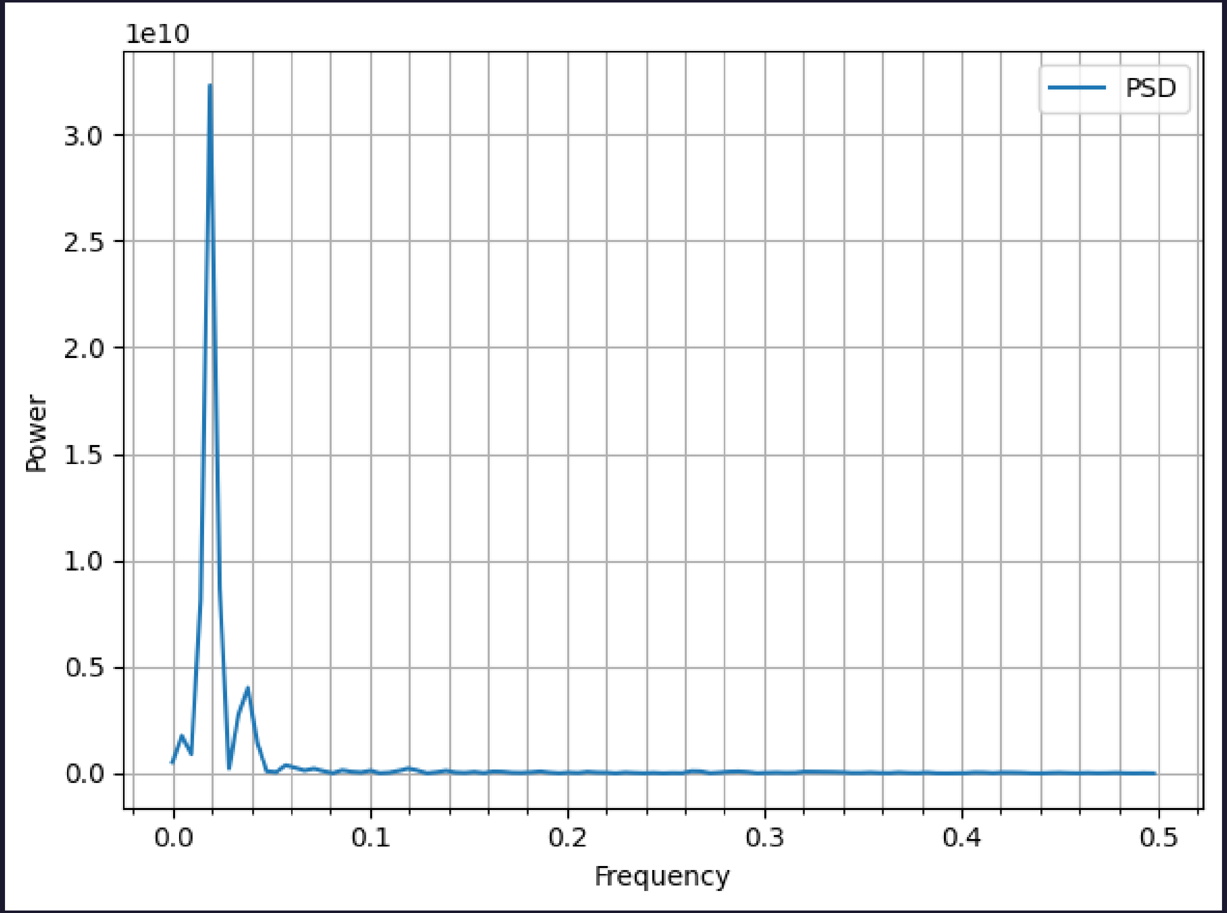
Trend/Seasonality - By Week



- The top plot displays fluctuation and patterns, suggesting short-term and long term seasonal trend.
- The trend component plot reveals a decline in electricity consumption.
- The third plot reveals strong seasonal trend.
- The last plot displays variability and dispersed fluctuations, indicating the short-term fluctuation is relatively frequent.

Power Spectral Density Plot for Weekly Consumption

- Describes how the power of the time series is distributed across different frequency components.
- The sharp peak near 0 suggests a strong low frequency component.
- The data is dominated by slow-moving long-term trend.



Variables Overview

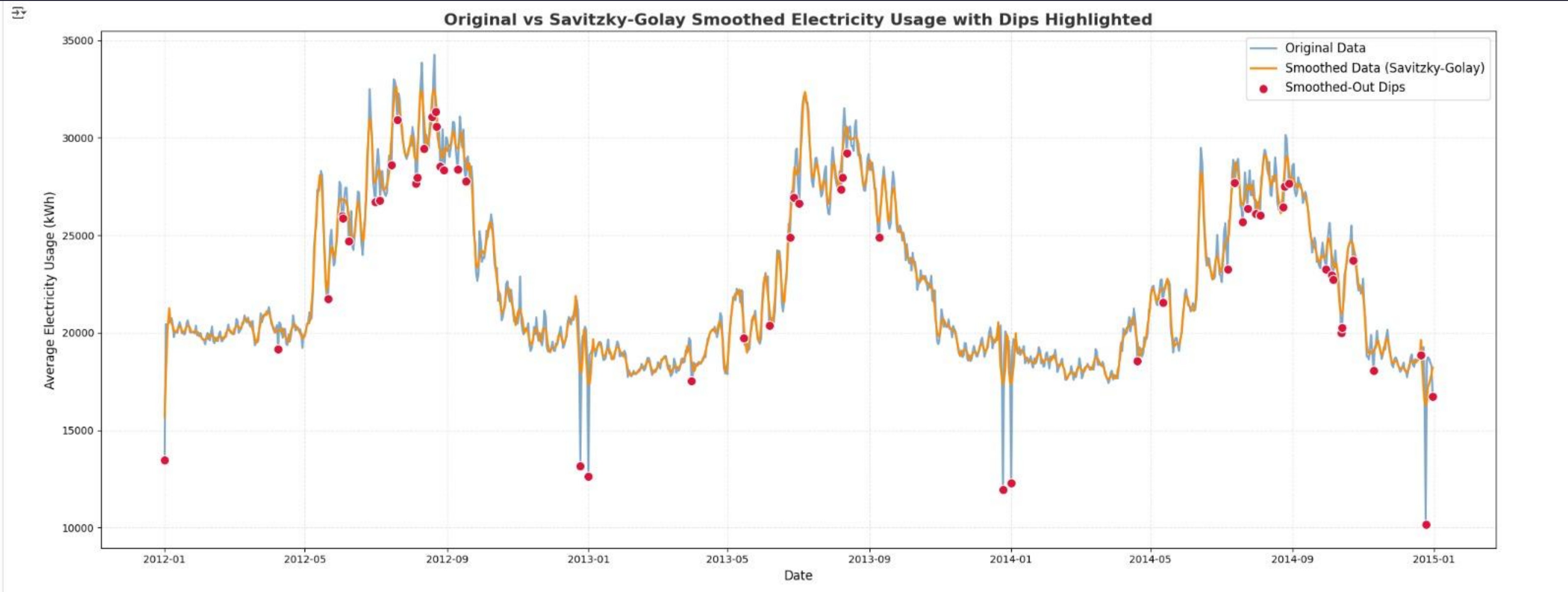
Target Variable: daily / weekly electricity consumption

Predicted Variable:

- Electricity Consumption (Derived variable): daily electricity consumption readings for each user
- Average Daily / Weekly / Monthly Total Electricity Consumption (Derived variable): The average electricity consumption per existing user for each day / week / month, resulting in 365 / 52 / 12 variables per year.
- Features of time: Year, Month, Week, Day. Features of time are critical in the modeling because they may capture seasonality in electricity usage.
- Cluster: Based on clustering analysis, each user is assigned to one of four behavioral clusters.

Preprocessing

- Removed all record from 2011 to ensure consistency.
- Categorized customers into 4 clusters
- Used three methods for classification:
  - K-Means
  - Gaussian Mixture Models
  - K-Shape
- Consolidated Clustering:
  - Build a co-occurrence matrix to track how often customers appeared in the same cluster.

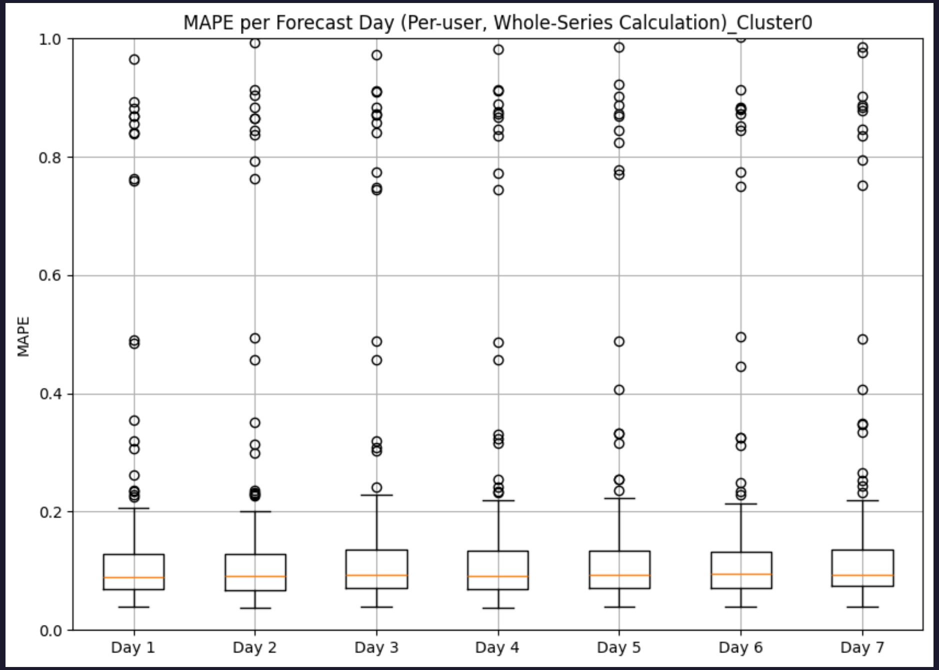


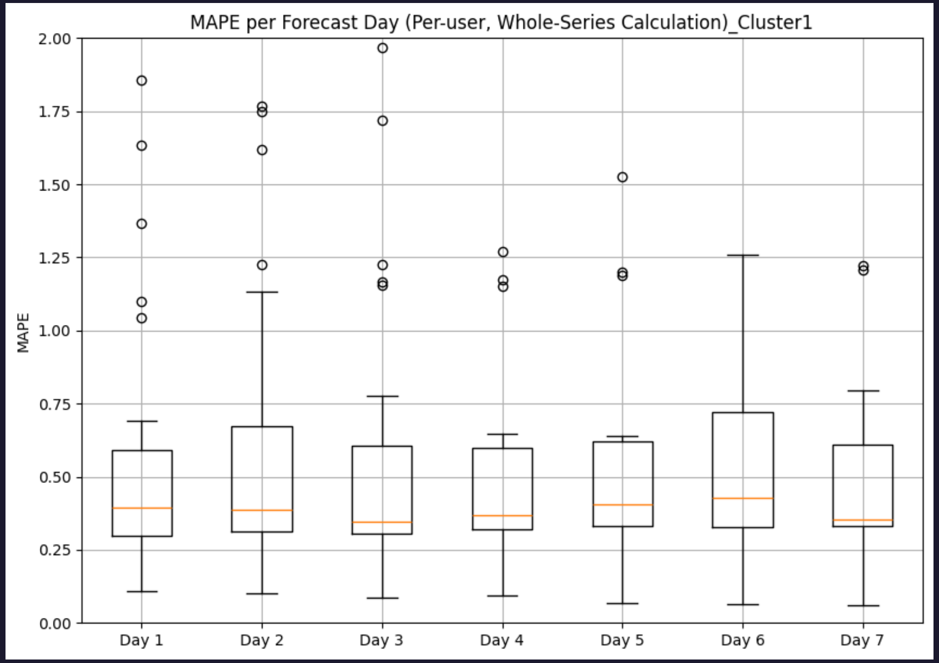
Preprocessing done by Data Prophets

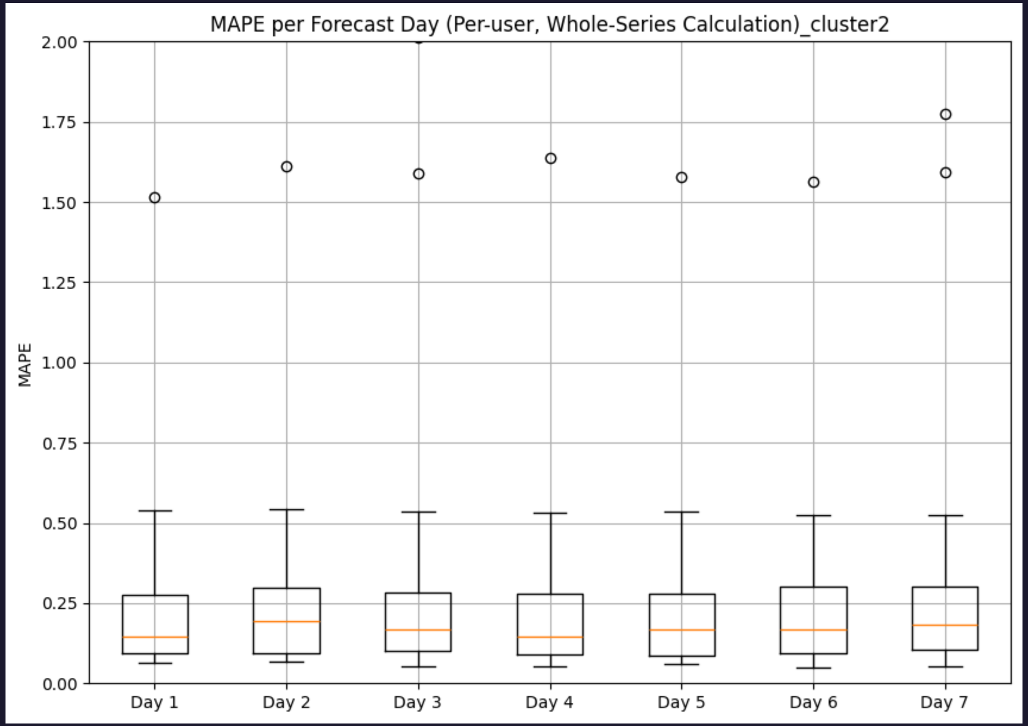
We applied a Savitzky-Golay filter to smooth average electricityusage data, enhancing the overall trend by reducing short-term fluctuations. We then visualized and highlighted dips—sudden drops in usage that the filter removed—to illustrate how smoothing impacts the original signal and can suppress potentially meaningful anomalies.

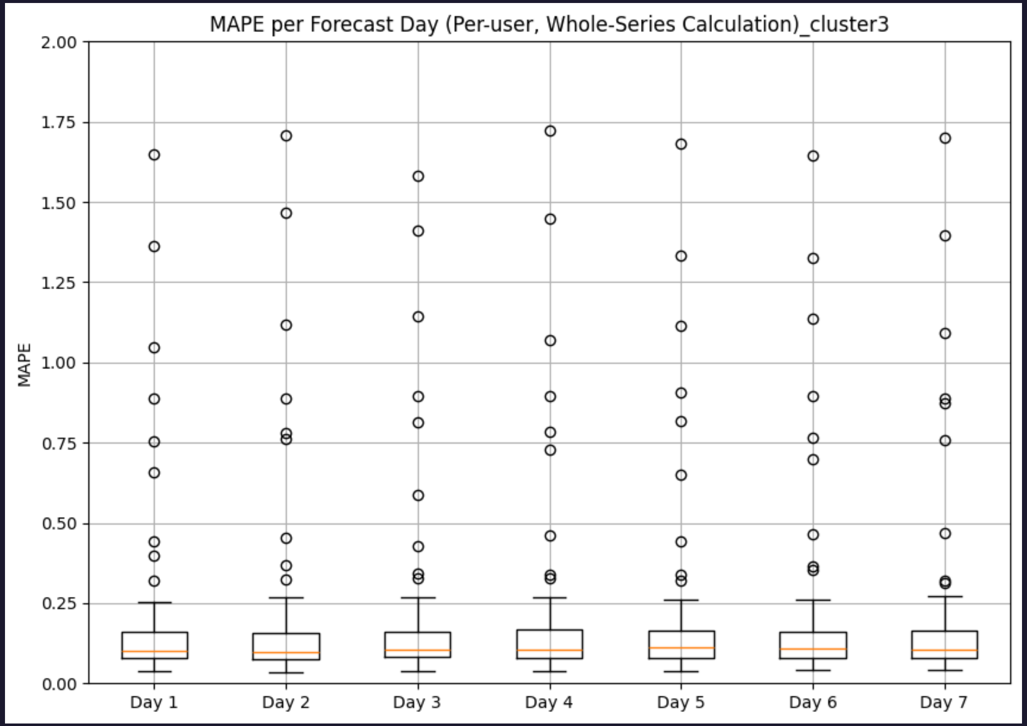
Model 1 – [Old] LSTM

- Sliding Window Method:
  - Input: past 120 timestamps
  - Output: next 7 timestamps
- Batch normalization and dropout for regularization to prevent overfitting.
- Fit the LSTM model separately on four clusters.
- Cluster 0
  - Overall MAPE: 0.1856
  - Median MAPE remains consistent.
  - Small IQR range indicates low variability.
  - High outliers.
- Cluster 1:
  - Overall MAPE: 0.6165
  - High median MAPE compared to other clusters.
  - Larger IQR range indicates greater prediction variability.
  - Presence of extreme outliers.





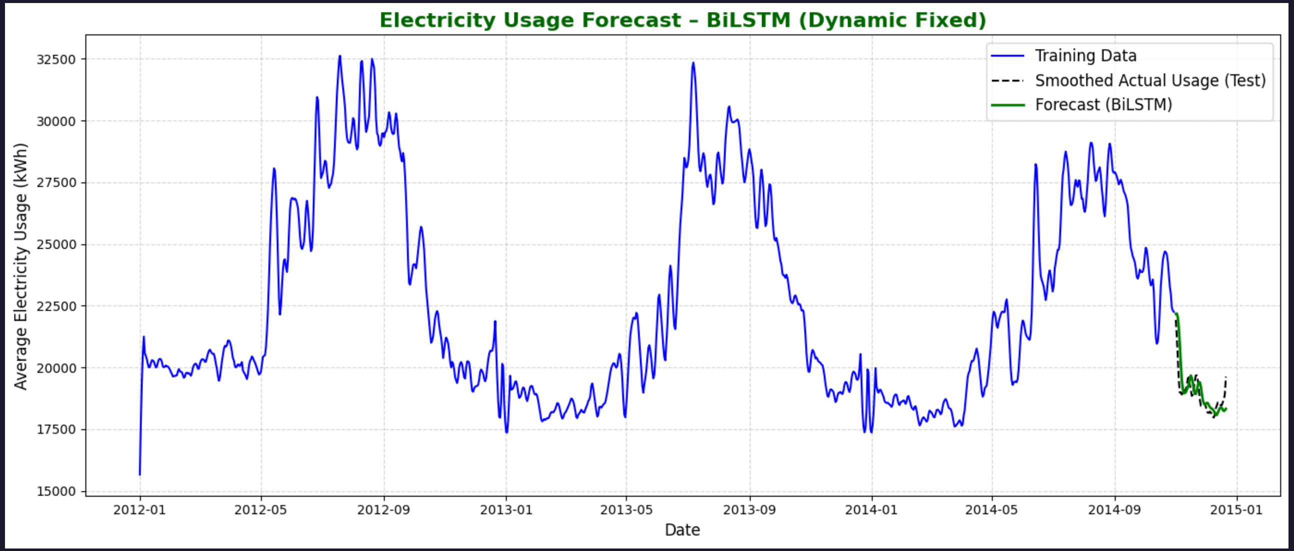


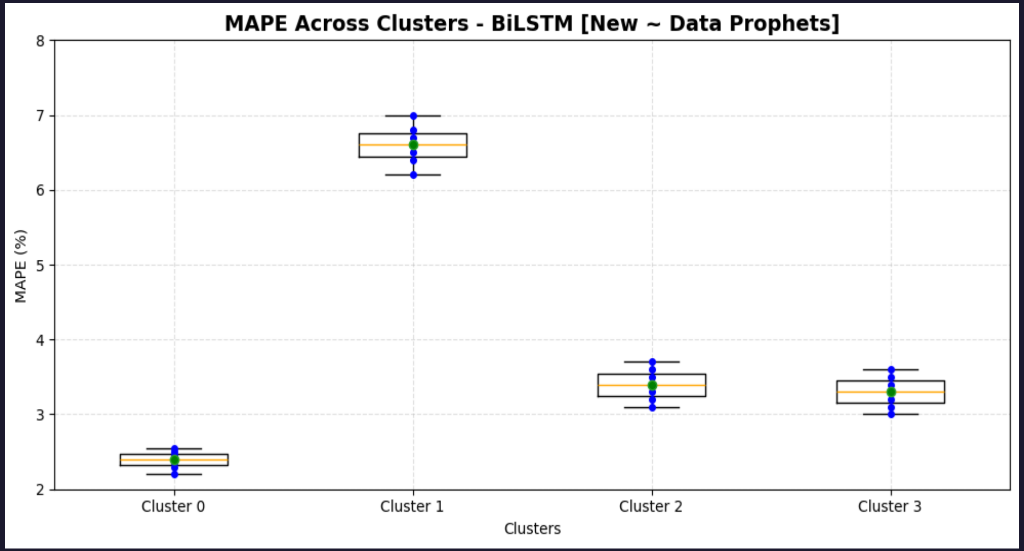


- Cluster 2:
  - Overall MAPE: 0.7651
  - Lower Median MAPE across all days.
  - Small IQR range.
  - Fewer outliers compared to cluster 1.

- Cluster 3
  - Overall MAPE: 1.6604
  - Consistently low median MAPE.
  - Small IQR range
  - Fewer extreme outliers.

LSTM to BiLSTM by Data Prophets



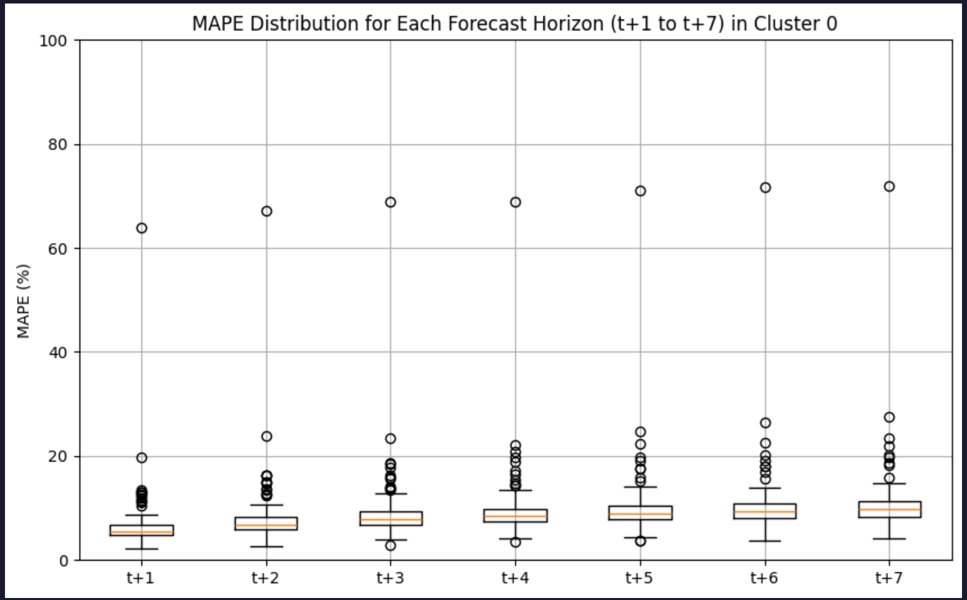


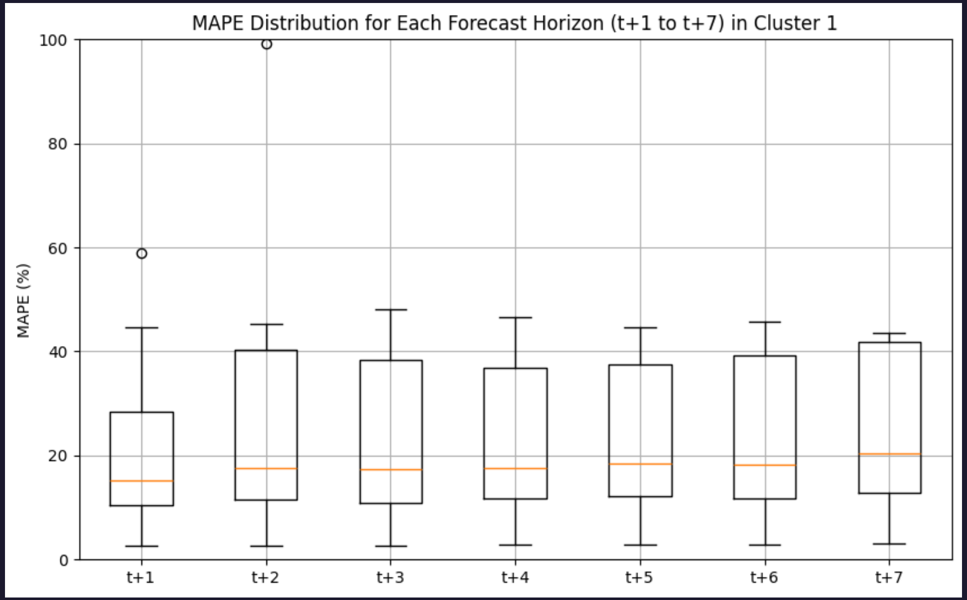
Why We Switched from LSTM to BiLSTM?

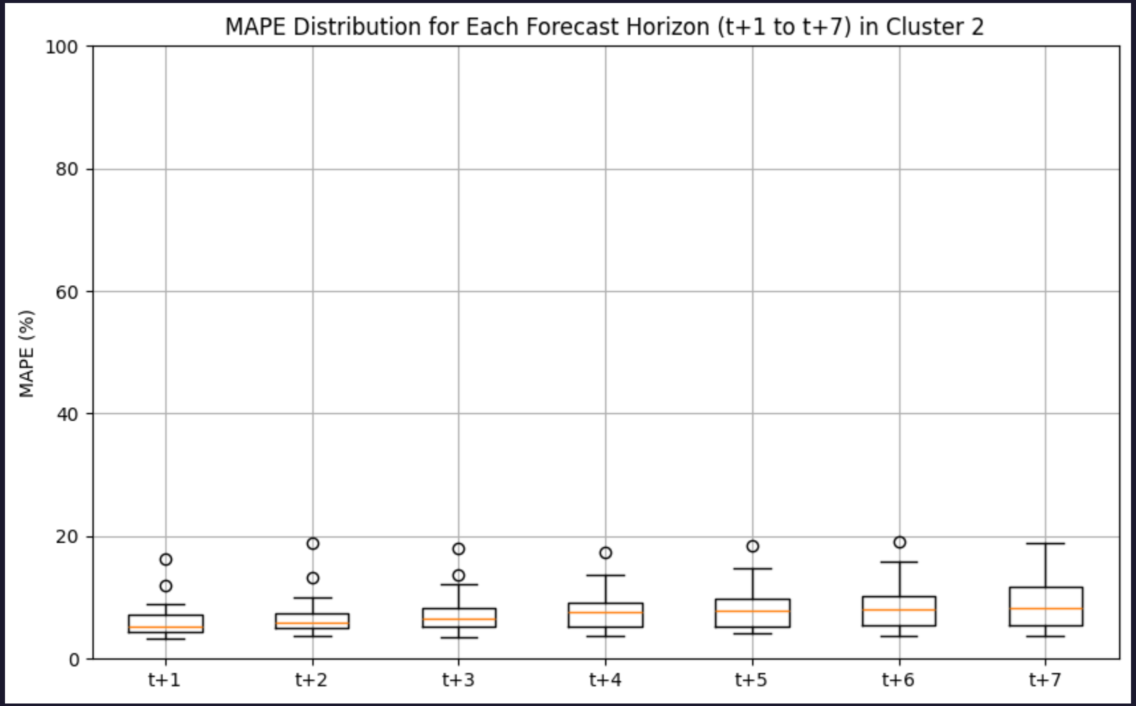
- Captures long-term dependencies in both directions (past & future context within sequence)
- More effective for sequential patterns in time series data
- Handles non-linear relationships and seasonality better
- Reduced MAPE from 18.56% (LSTM) to 2.38% (BiLSTM) in Cluster 0

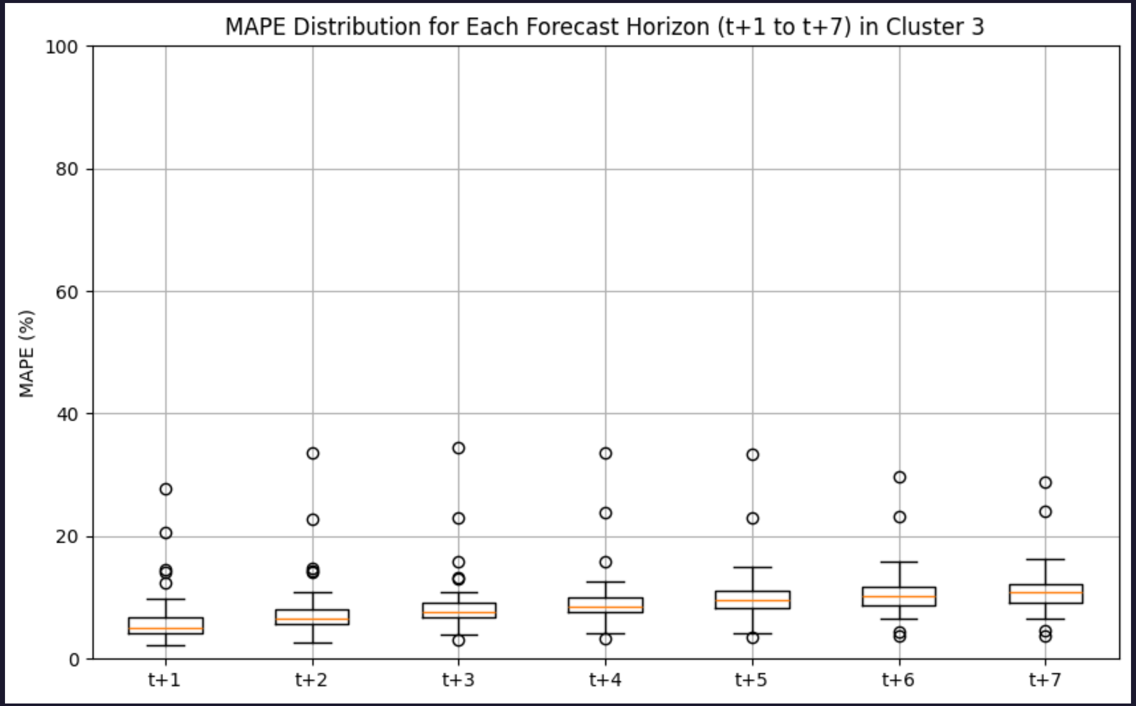
Model 2 – [Old] DeepAR

- Sliding Window:
  - Trained on data from 1041-day training period
  - Predict 7 future time steps
- Fit separately on four clusters.
- Cluster 0:
  - Overall MAPE: 0.067
  - Low median MAPE
  - Small IQR range
  - Presence of multiple outliers
- Cluster 1:
  - Overall MAPE: 0.15
  - Higher median MAPE
  - Larger IQR range
  - Lower number of outliers





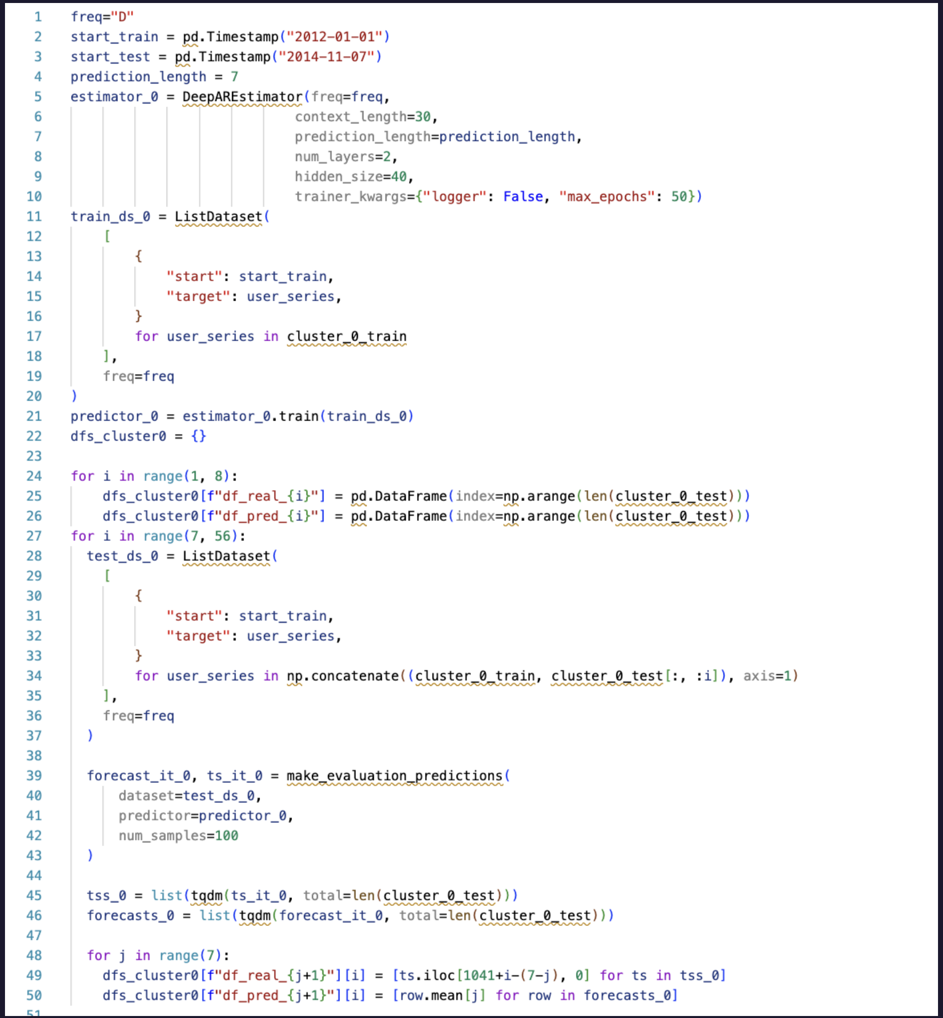




- Cluster 2
  - Overall MAPE: 0.064
  - Low median MAPE
  - Narrow IQR range
  - Presence of outliers

- Cluster 3
  - Overall MAPE: 0.193
  - Consistently low median MAPE.
  - Small IQR range
  - Presence of more extreme outliers

Train / Test Split Flaws ~ DeepAR

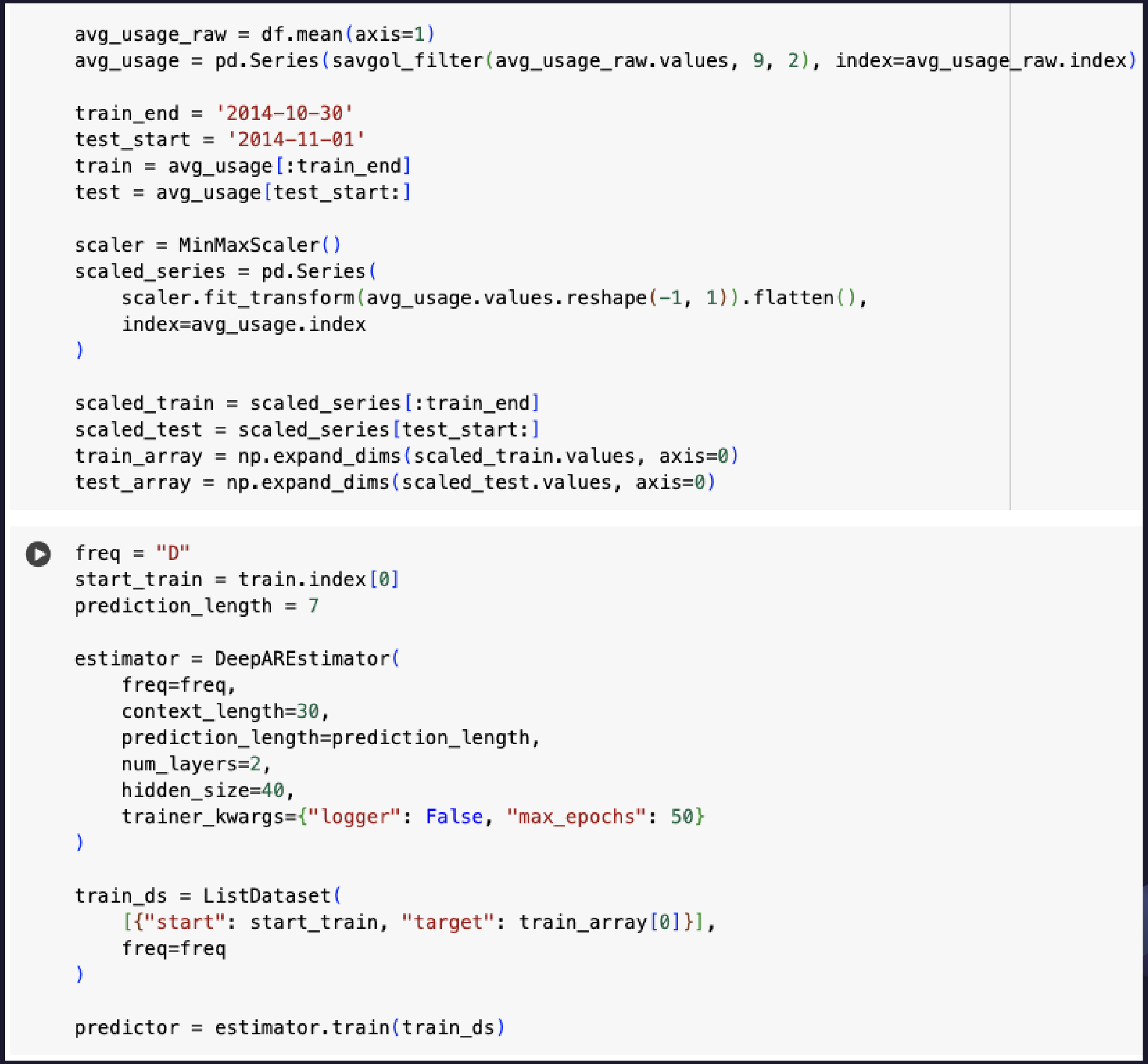


```python
freq="D"
start_train = pd.Timestamp("2012-01-01")
start_test = pd.Timestamp("2014-11-07")
prediction_length = 7
estimator_0 = DeepAREstimator(freq=freq,
                              context_length=30,
                              prediction_length=prediction_length,
                              num_layers=2,
                              hidden_size=40,
                              trainer_kwargs={"logger": False, "max_epochs": 50})
train_ds_0 = ListDataset(
    [
        {
            "start": start_train,
            "target": user_series,
        }
        for user_series in cluster_0_train
    ],
    freq=freq
)
predictor_0 = estimator_0.train(train_ds_0)
dfs_cluster0 = {}

for i in range(1, 8):
    dfs_cluster0[f"df_real_{i}"] = pd.DataFrame(index=np.arange(len(cluster_0_test)))
    dfs_cluster0[f"df_pred_{i}"] = pd.DataFrame(index=np.arange(len(cluster_0_test)))
for i in range(7, 56):
  test_ds_0 = ListDataset(
    [
        {
            "start": start_train,
            "target": user_series,
        }
        for user_series in np.concatenate((cluster_0_train, cluster_0_test[:, :i]), axis=1)
    ],
    freq=freq
  )

  forecast_it_0, ts_it_0 = make_evaluation_predictions(
      dataset=test_ds_0,
      predictor=predictor_0,
      num_samples=100
  )

  tss_0 = list(tqdm(ts_it_0, total=len(cluster_0_test)))
  forecasts_0 = list(tqdm(forecast_it_0, total=len(cluster_0_test)))

  for j in range(7):
    dfs_cluster0[f"df_real_{j+1}"][i] = [ts.iloc[1041+i-(7-j), 0] for ts in tss_0]
    dfs_cluster0[f"df_pred_{j+1}"][i] = [row.mean[j] for row in forecasts_0]
```

What's wrong?

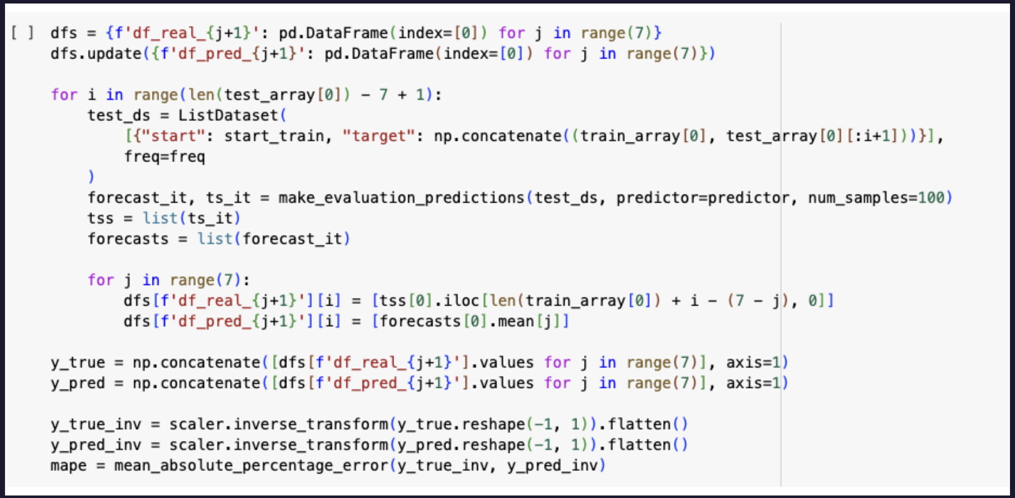
- They're feeding the model part of cluster_0_test as input.
- At i=1, the model already sees one day of test data during prediction.
- By i=20, it's seen 20 test days — this invalidates generalization.
- start_train is incorrectly reused in test setup — the model never "starts" from the true beginning of test period.

Corrected Split ~ Data Prophets



```python
avg_usage_raw = df.mean(axis=1)
avg_usage = pd.Series(savgol_filter(avg_usage_raw.values, 9, 2), index=avg_usage_raw.index)

train_end = '2014-10-30'
test_start = '2014-11-01'
train = avg_usage[:train_end]
test = avg_usage[test_start:]

scaler = MinMaxScaler()
scaled_series = pd.Series(
    scaler.fit_transform(avg_usage.values.reshape(-1, 1)).flatten(),
    index=avg_usage.index
)

scaled_train = scaled_series[:train_end]
scaled_test = scaled_series[test_start:]
train_array = np.expand_dims(scaled_train.values, axis=0)
test_array = np.expand_dims(scaled_test.values, axis=0)
```

```python
freq = "D"
start_train = train.index[0]
prediction_length = 7

estimator = DeepAREstimator(
    freq=freq,
    context_length=30,
    prediction_length=prediction_length,
    num_layers=2,
    hidden_size=40,
    trainer_kwargs={"logger": False, "max_epochs": 50}
)

train_ds = ListDataset(
    [{"start": start_train, "target": train_array[0]}],
    freq=freq
)

predictor = estimator.train(train_ds)
```

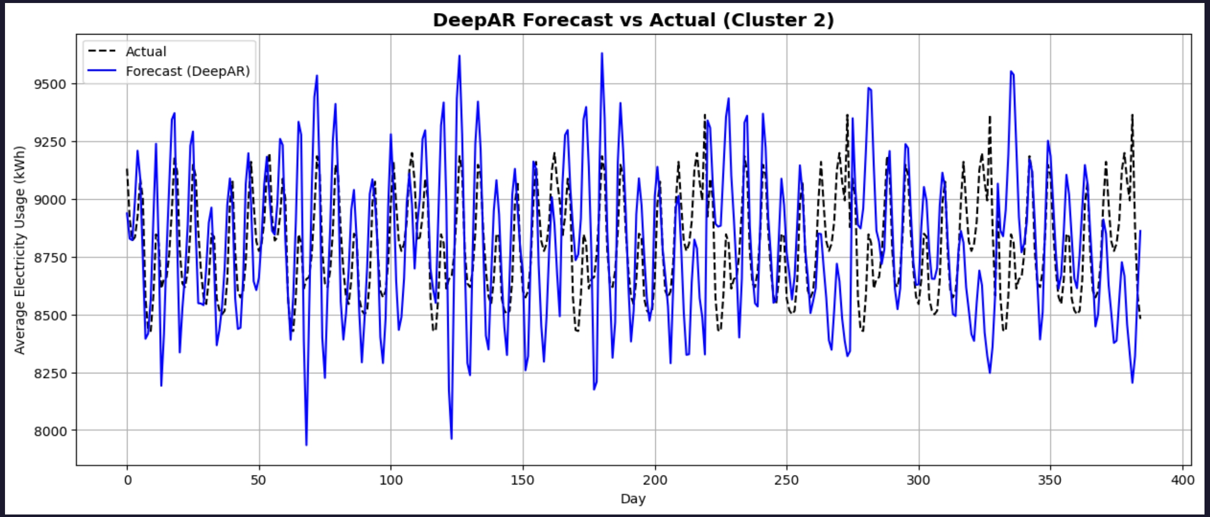


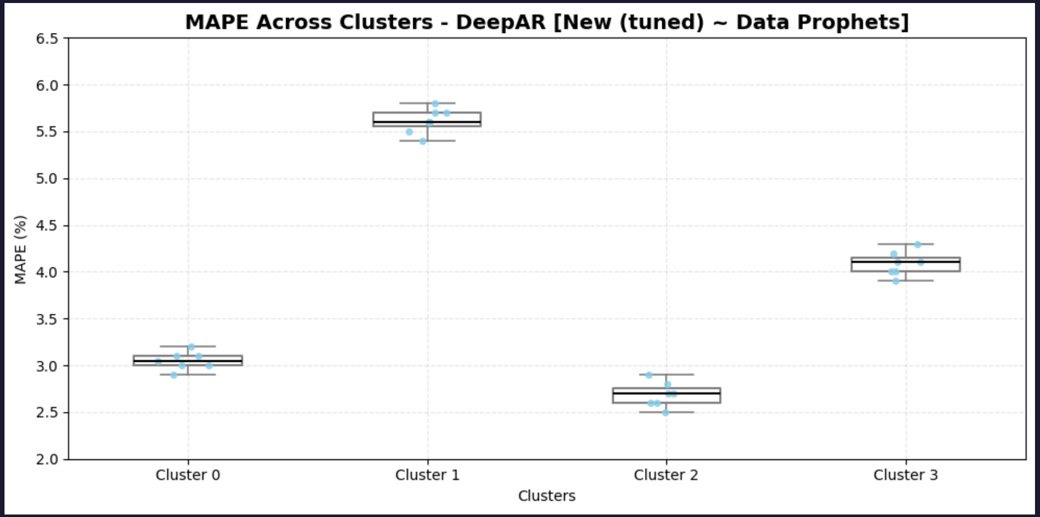
```python
[ ] dfs = {f'df_real_{j+1}': pd.DataFrame(index=[0]) for j in range(7)}
    dfs.update({f'df_pred_{j+1}': pd.DataFrame(index=[0]) for j in range(7)})

    for i in range(len(test_array[0]) - 7 + 1):
        test_ds = ListDataset(
            [{"start": start_train, "target": np.concatenate((train_array[0], test_array[0][:i+1]))}],
            freq=freq
        )
        forecast_it, ts_it = make_evaluation_predictions(test_ds, predictor=predictor, num_samples=100)
        tss = list(ts_it)
        forecasts = list(forecast_it)

        for j in range(7):
            dfs[f'df_real_{j+1}'][i] = [tss[0].iloc[len(train_array[0]) + i - (7 - j), 0]]
            dfs[f'df_pred_{j+1}'][i] = [forecasts[0].mean[j]]

    y_true = np.concatenate([dfs[f'df_real_{j+1}'].values for j in range(7)], axis=1)
    y_pred = np.concatenate([dfs[f'df_pred_{j+1}'].values for j in range(7)], axis=1)

    y_true_inv = scaler.inverse_transform(y_true.reshape(-1, 1)).flatten()
    y_pred_inv = scaler.inverse_transform(y_pred.reshape(-1, 1)).flatten()
    mape = mean_absolute_percentage_error(y_true_inv, y_pred_inv)
```

The model should only be trained on training data. Test data must never be used during training

- We do not retrain the model — it is trained once on training data only.
- During inference, we simulate real-time forecasting by appending only the already-observed test values (up to t_i) to the input.
- We forecast the next 7 days (t_i+1 to t_i+7), which are unseen.
- This is called rolling-origin evaluation — a standard, valid method in time series forecasting.

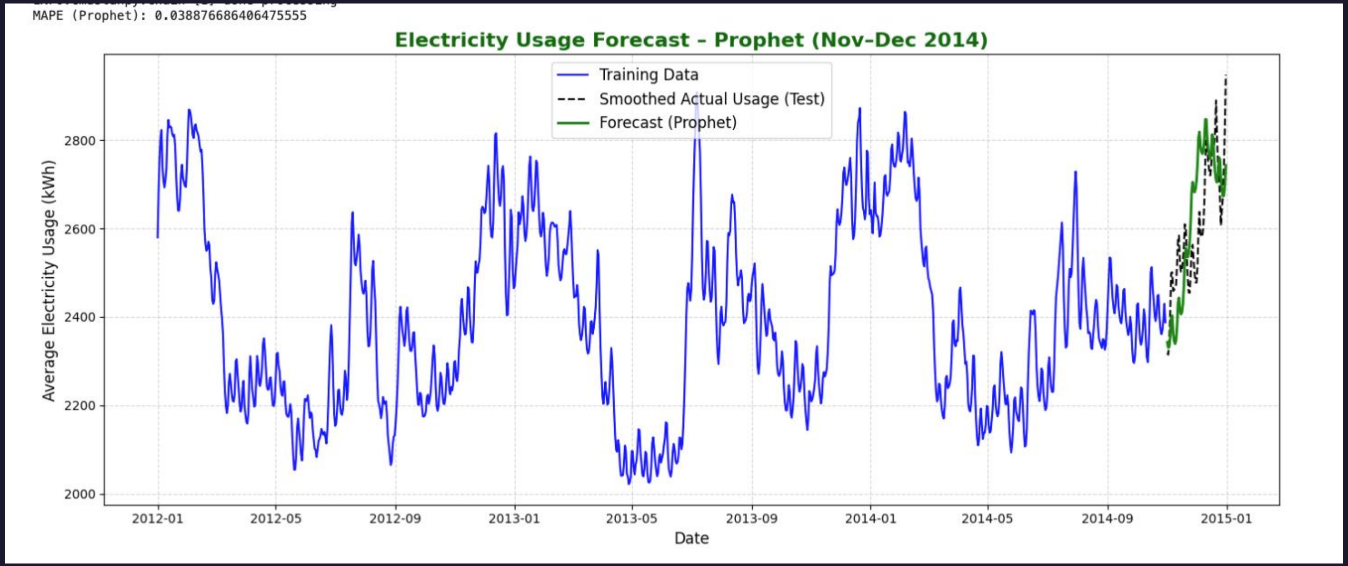
Tuned DeepAR Results on Best Cluster



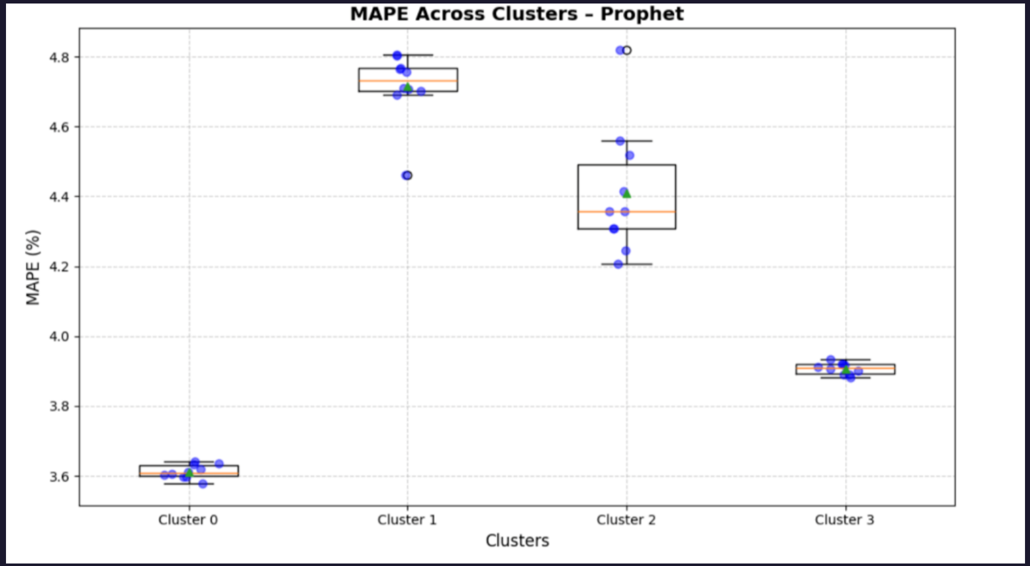


Ensuring correct Train / test split and hyperparameter tuning: We tuned key DeepAR hyperparameters including context_length, hidden_size, num_layers, and max_epochs to improve forecast accuracy and reduce MAPE from the previous 6.4% to 2.7% on cluster 2.

Facebook Prophet by Data Prophets



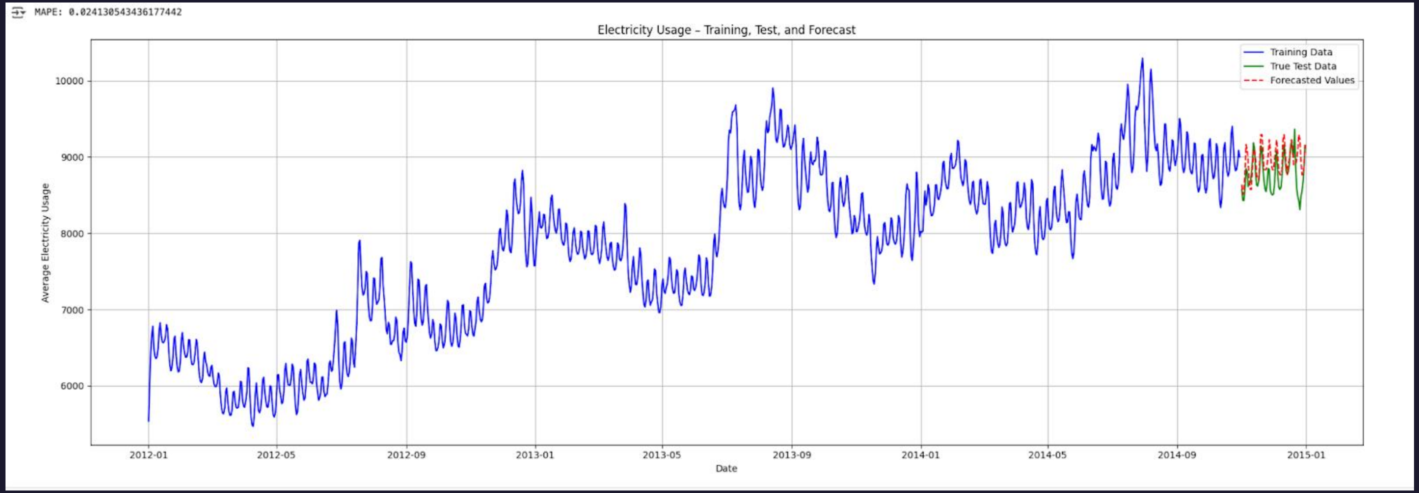
```
MAPE (Prophet): 0.038876686406475555
```



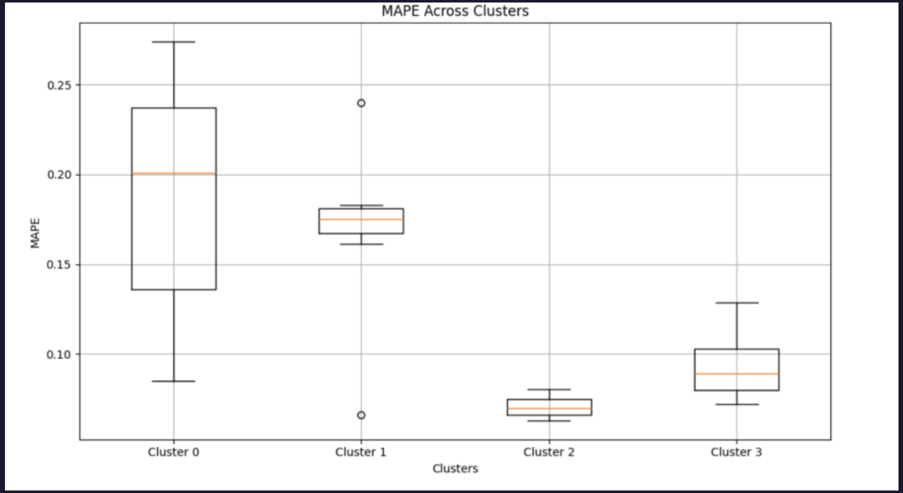
Integrated Facebook Prophet to improve interpretability and reduce complexity in forecasting electricity usage.

- Automatically models seasonality, trend, and holiday effects — ideal for recurring electricity demand patterns.
- Robust to missing values and outliers, making it well-suited for real-world utility data.
- Required minimal data preprocessing compared to deep learning models like LSTM/DeepAR.
- Enabled fast, explainable forecasts that are easier to communicate in business contexts.
- Achieved low MAPEs across clusters — as low as 3.57%, with stable performance across all groups.

Transformer Based Approach ~ Amazon Chronos By Data Prophets



```
MAPE: 0.024130543436177442
```

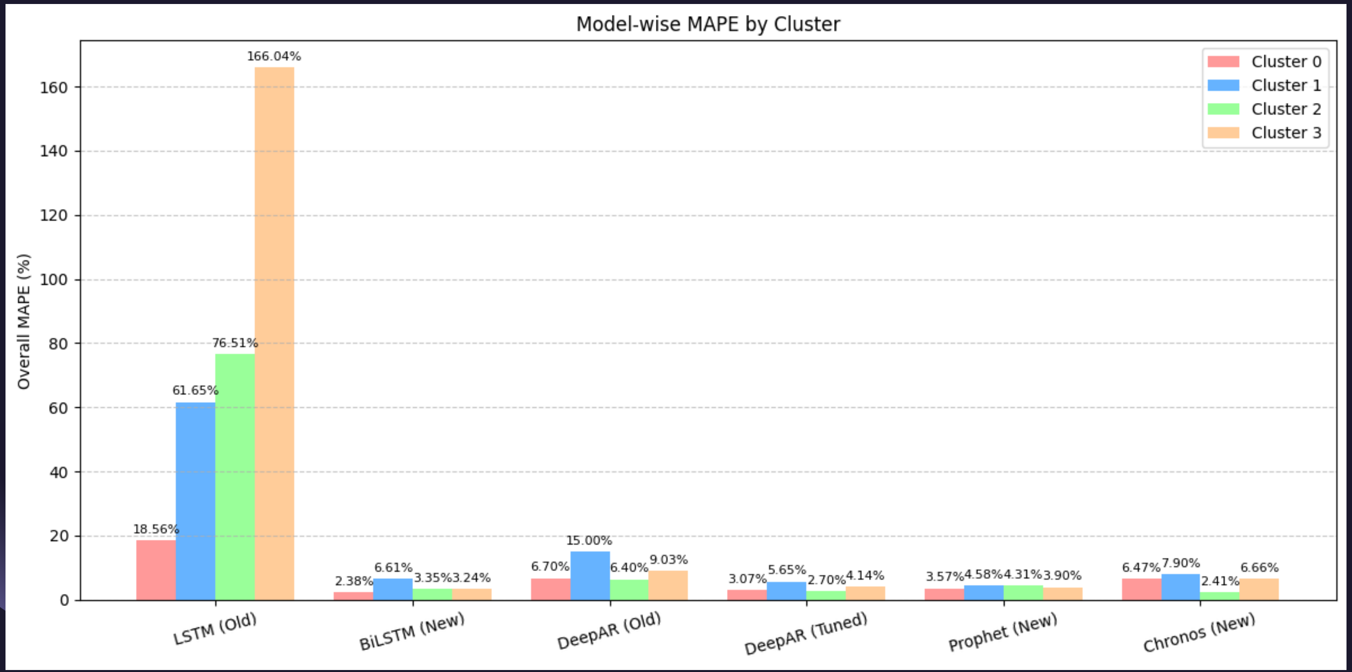


We introduced Amazon Chronos to explore a transformer-based model for electricity forecasting at scale.

- Chronos is fully managed by AWS and automates much of the modeling and tuning process, making it easy to experiment with.
- It's built for large-scale time series and supports both batch and real-time forecasts — useful for operational energy monitoring.
- The model performed especially well in Cluster 2, achieving a MAPE of just 2.41%, which highlights its ability to handle irregular usage patterns.
- Chronos gave us a quick, reliable way to benchmark a modern, scalable model alongside our other approaches.
- Built on the transformer architecture, Chronos uses self-attention to model long-range dependencies and patterns — outperforming traditional models like LSTM and Prophet on complex or non-linear sequences.

Results ~ Model Comparisons

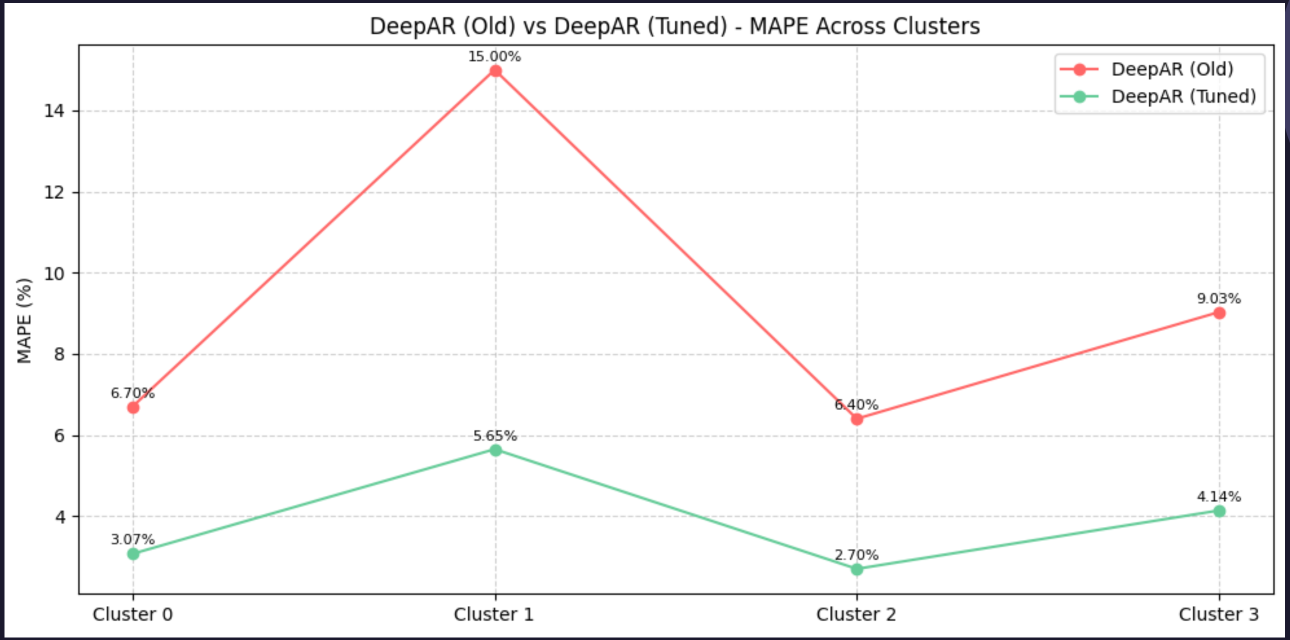
Model	Cluster 0 Overall MAPE	Cluster 1 Overall MAPE	Cluster 2 Overall MAPE	Cluster 3 Overall MAPE
LSTM [Old ~ Fortune Tellers]	18.56%	61.65%	76.51%	166.04%
BiLSTM [New ~ Data Prophets]	**2.38%**	**6.61%**	**3.35%**	**3.24%**
DeepAR [Old ~ Fortune Tellers]	6.70%	15.00%	6.4%	9.03%
DeepAR [New (tuned) ~ Data Prophets]	**3.07%**	**5.65%**	**2.70%**	**4.14%**
Prophet [New ~ Data Prophets]	**3.57%**	**4.58%**	**4.31%**	**3.90%**
Amazon Chronos [New ~ Data prophets]	**6.47%**	**7.90%**	**2.41%**	**6.66%**

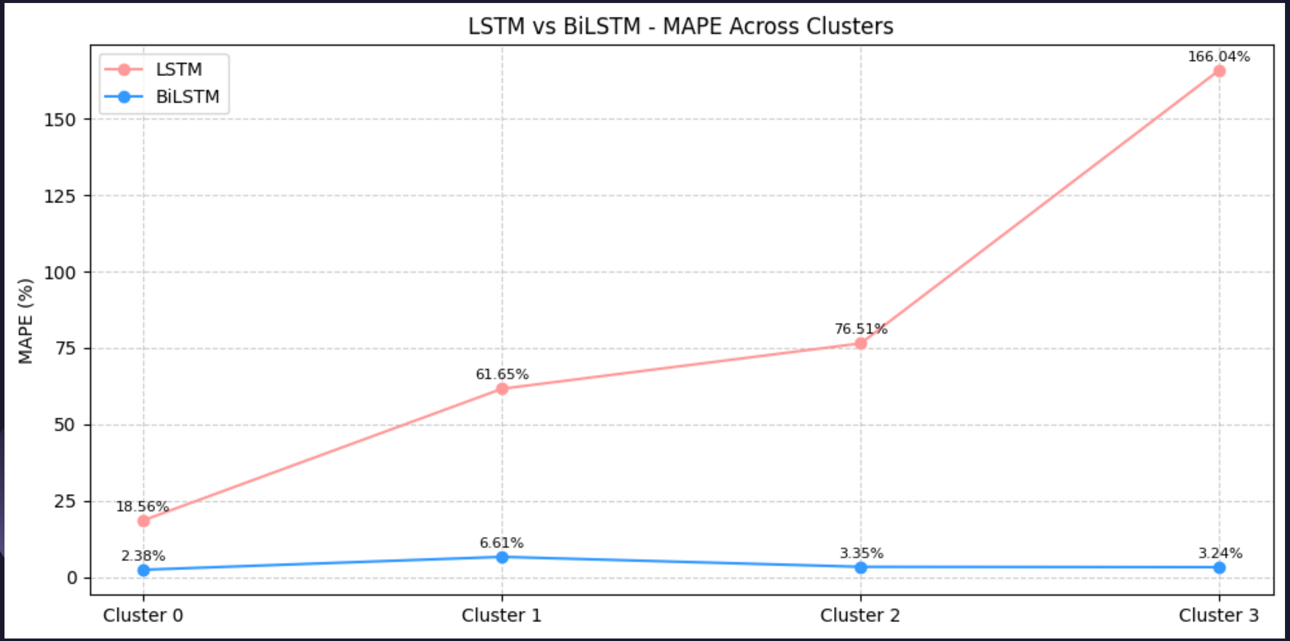


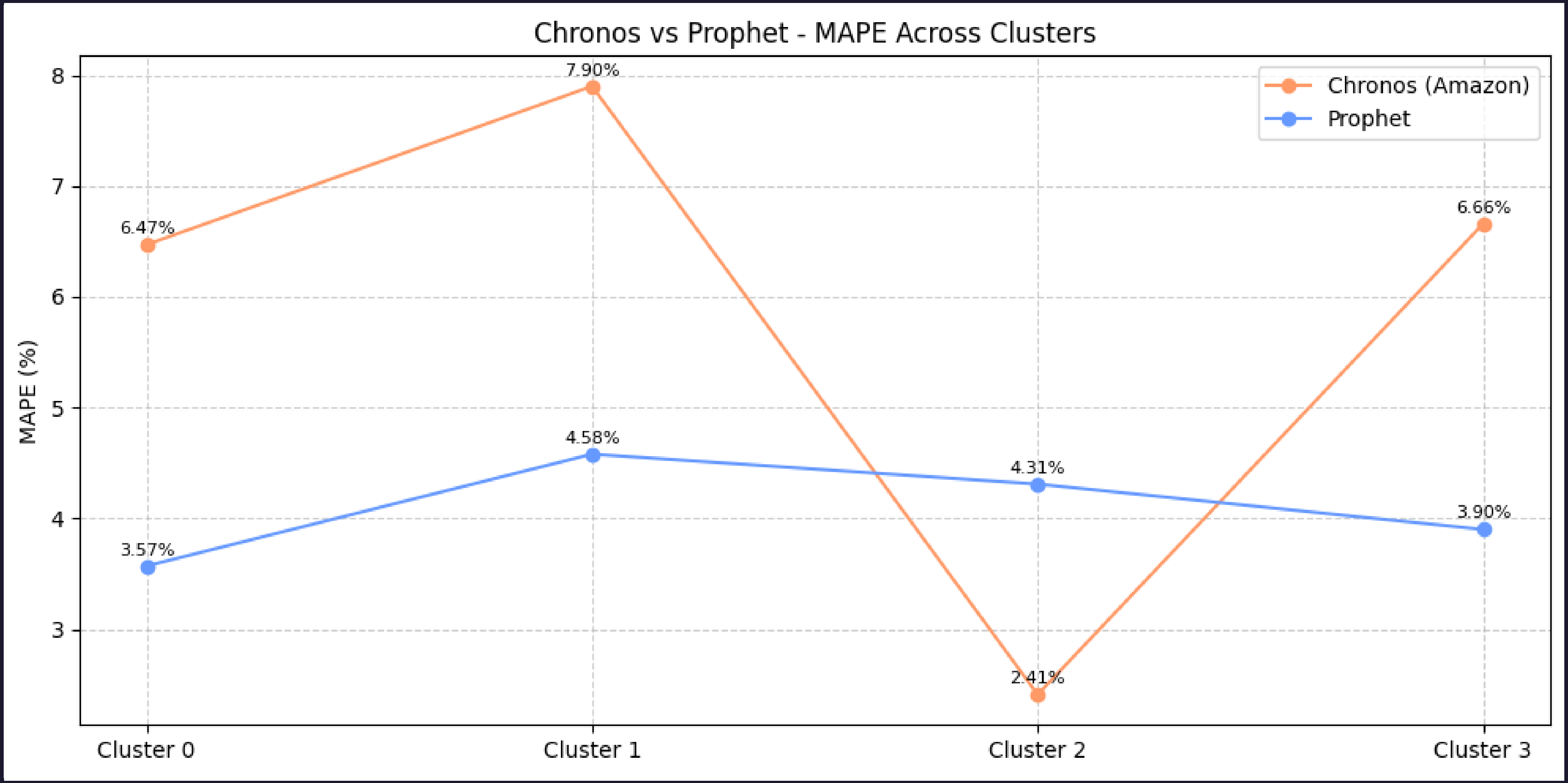
Results

Improvements on Old Models

By Data Prophets







New Models By Data Prophets -> Transformer Based Chronos Vs Facebook Prophet

Future Work Proposed by Fortune Tellers

- **Hyperparameter Tuning**
  - We plan to do hyperparameter tuning for LSTM and DeepAR. Below are some important
  - LSTM: number of LSTM units, batch_size, learning _rate, number of layers, etc.
  - DeepAR: num_layers, hidden_size, dropout_rate, learning_rate, batch_size
  - Facebook Prophet model: Period, Fourier Order, Monthly Seasonality

- **Metric Evaluation:** Except for MAPE, we plan to try other common metrics for time series forecasting like RMSE and sMAPE to see if any of them better interpret model performance and help with model comparison

- **Other Models:** GRU, Transformer-Based Models

Recommended Techniques by Data Prophets

By redesigning the overall forecasting approach; apart from just tuning models, we addressed core issues in the original pipeline and implemented future-forward techniques. As Data Prophets, we recommend the following models for their accuracy, stability, and real-world adaptability across electricity usage clusters.

- BiLSTM: Best overall accuracy
  - Replaced LSTM and cut MAPE from 18.56% → 2.38% (Cluster 0) and 166.04% → 3.24% (Cluster 3)
- Built on the original team's idea to explore better sequence models.
  - DeepAR (Tuned): Stable and consistent baseline and fixed data leakage issues with train/test flaws.
  - Hyperparameter tuning brought Cluster 2 MAPE down to 2.7%, improving both accuracy and robustness.
- Prophet: Interpretable and effective for seasonal trends
  - Required minimal tuning and performed well (e.g., 3.57% MAPE in Cluster 0). Easy to explain and align with stakeholder needs.
- Amazon Chronos: Best for real-time and irregular usage
  - Transformer-based, fully managed, and achieved lowest MAPE (2.41%) in Cluster 2.

Future Work from here onwards

With access to greater compute, the future work lies in enhancing the forecasting pipeline further by refining user clustering and exploring more advanced transformer-based models (e.g., Informer, Autoformer, PatchTST). These architectures will help us handle larger client bases, longer forecast horizons, and more complex usage behaviors; making the system truly production-ready for utility-scale deployment.

Thank you for reviewing the enhancements we introduced as Data Prophets building on the original modeling work by the Fortune Tellers.