DLBS Mini-Challenge Report: Energy Consumption Forecasting

1. General Approach

This report details the process of analyzing energy consumption data and developing forecasting models. The primary tools used were Python, Pandas for data manipulation, Matplotlib/Seaborn for visualization, Statsmodels for baseline statistical modeling, and PyTorch for deep learning models (LSTMs). The work is documented in two Jupyter notebooks: `DLBS_MC.ipynb` (covering data preparation, EDA, and baseline modeling) and `DLBS_Deep_learning_runs.ipynb` (covering LSTM model development, training, and evaluation).

2. Results Summary

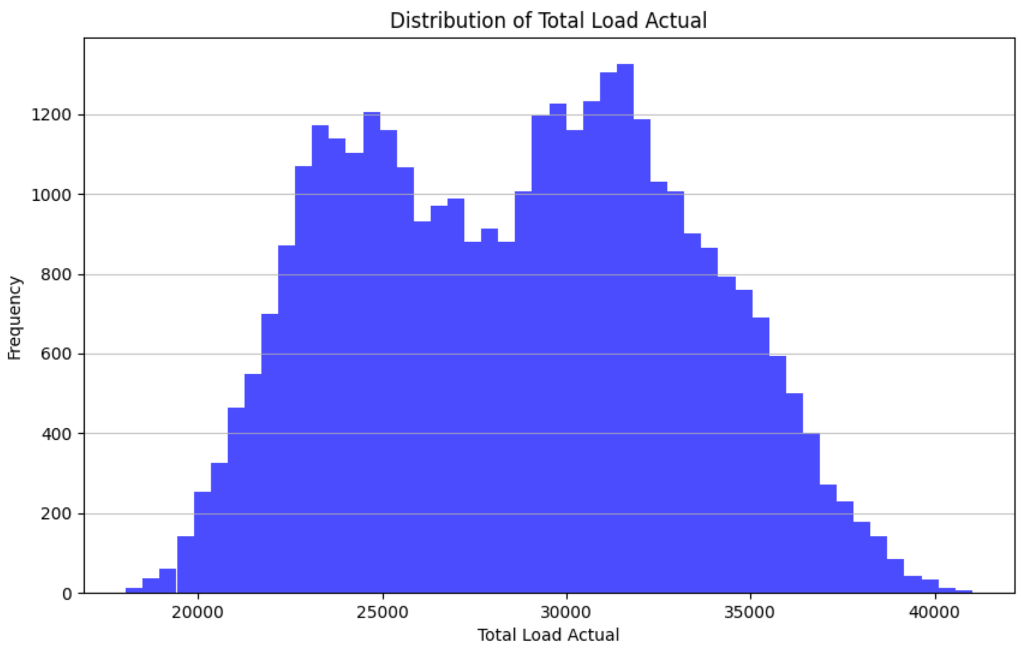
The research aimed to forecast energy consumption using historical energy load and weather data. The data comprised hourly energy consumption and weather features over a single year. Initial analysis revealed daily and seasonal patterns in energy use. An ARIMA model was established as a baseline, which showed limited accuracy, predicting just a flat line. Subsequently, LSTM neural networks were explored. A univariate LSTM using only past energy load showed decent performance but didn't surpass the original dataset's forecast. A multivariate LSTM incorporating selected weather and engineered time based features improved predictions, achieving a MAPE below 1% and outperforming the original forecast. However, a multivariate LSTM with an extensive feature set performed very poorly, highlighting the importance of feature selection. The best performing model was the multivariate LSTM with a reduced feature set.

2.1 Data Analysis

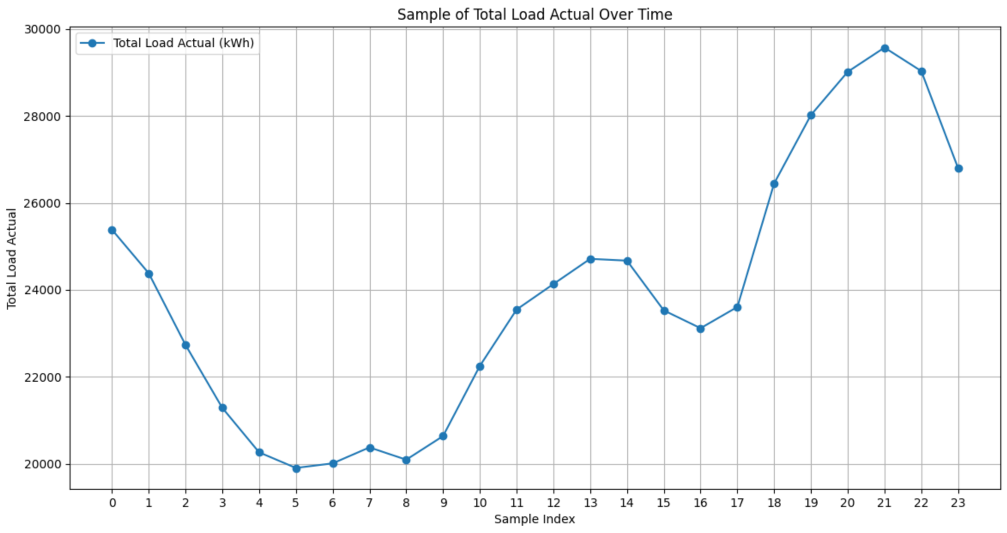
Following the Karpathy and Lones guidelines, the data analysis involved several steps:

- **Visual Data Inspection**: Initial plots of the energy load (distribution, time series sample) and weather features (time series samples, boxplots) were created. This helped to quickly identify patterns, such as the bimodal distribution of energy usage, daily peaks/troughs, and the nature of weather variables (e.g., sporadic rainfall vs. continuous temperature changes).

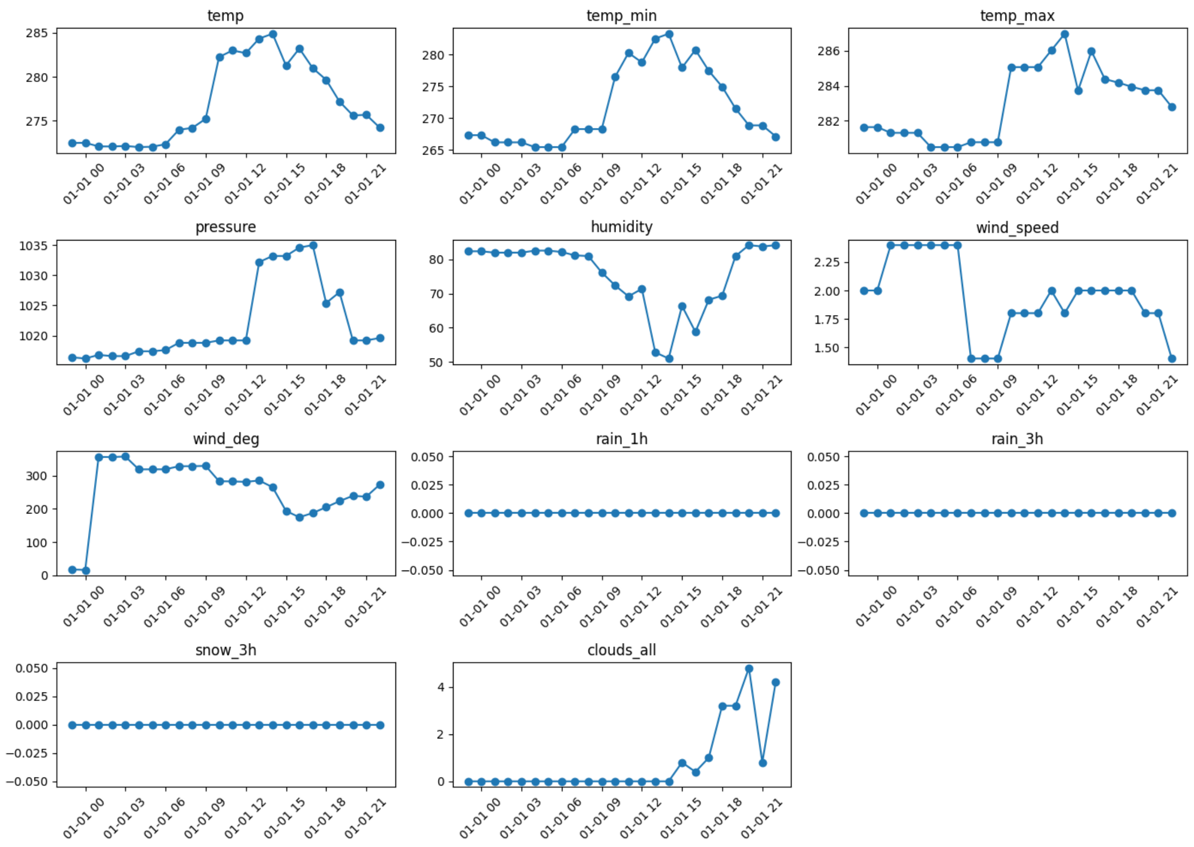
- Distribution of Total Load Actual:



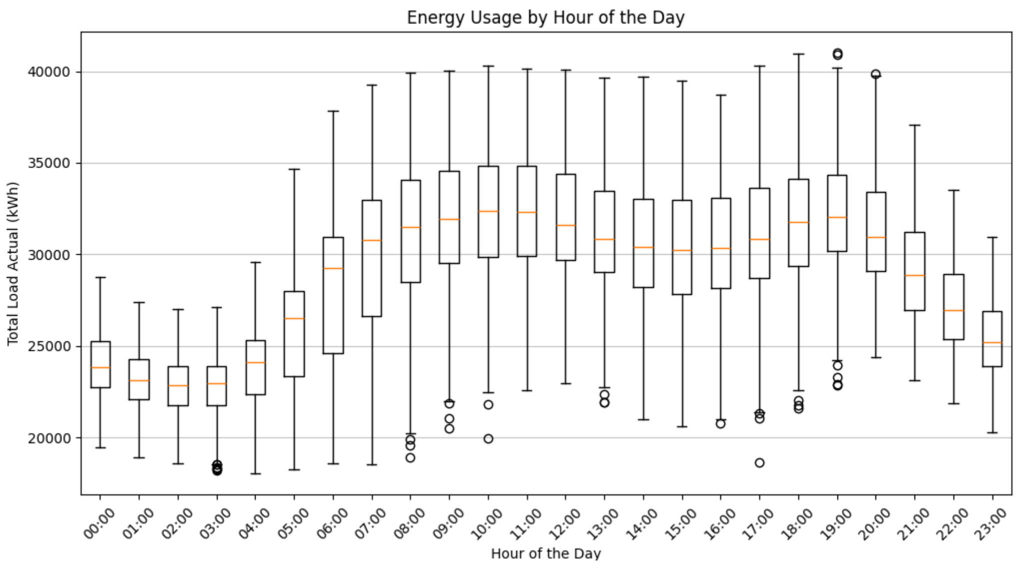
- Sample of Total Load Actual Over Time (First 24 hours):



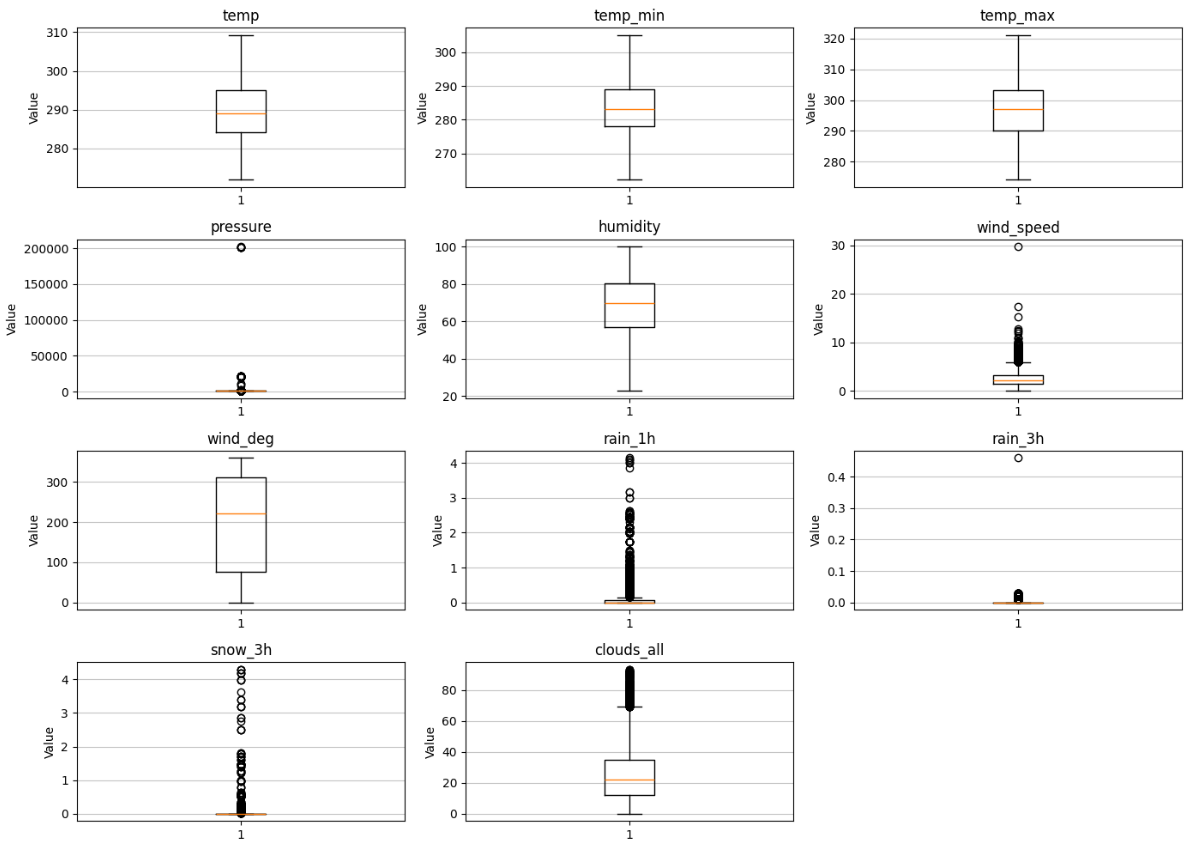
- Sample of Weather Features (First 24 hours):



- Energy Usage by Hour of the Day (Boxplot):



- Weather Features Boxplots:



- **Qualitative Data Understanding**: The data represents energy consumption in Spain, so incorporating Spanish holidays as a feature was a logical step. Understanding that weather patterns directly influence heating/cooling needs (and thus energy load) guided the inclusion and engineering of weather features.

- **Quantitative Data Analysis**: This involved checking for NaN values (found in `total load actual` and imputed), aggregating weather data from multiple cities to a single representative series, and performing seasonal decomposition to statistically confirm the observed seasonality. Feature engineering (lags, cyclical features) was a key part of quantitatively preparing data for modeling.

2.2 Baseline Setup

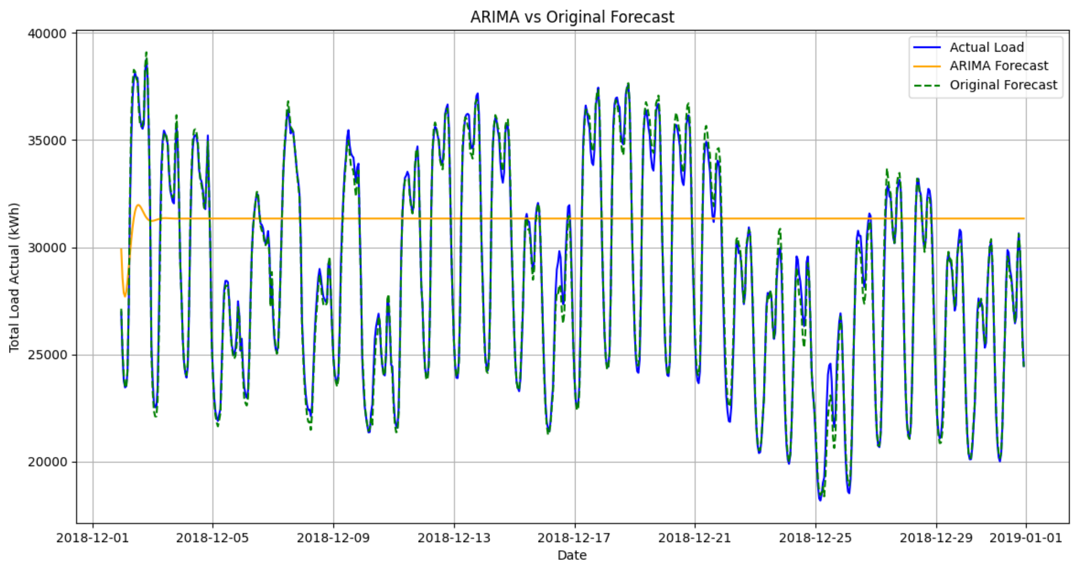
An ARIMA (Autoregressive Integrated Moving Average) model was chosen as the baseline. This is a standard and relatively simple statistical model for time series forecasting, making it a good benchmark before moving to more complex deep learning approaches.

- **Model Choice & Parameters**: An ARIMA(2,1,2) model was selected after some initial experimentation and running a quick grid search. The (p,d,q) parameters represent the order of autoregression, differencing, and moving average components, respectively. The model was trained on the `total load actual` data after feature engineering, using the last 720 hours (30 days) as the test set.

- **Evaluation Metrics**: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) were used to evaluate the baseline and subsequent models. MAPE is particularly useful for understanding the error in relative terms.

- **Analysis & Visualization**: The ARIMA model's predictions were plotted against the actual test data and the original forecast provided in the dataset.

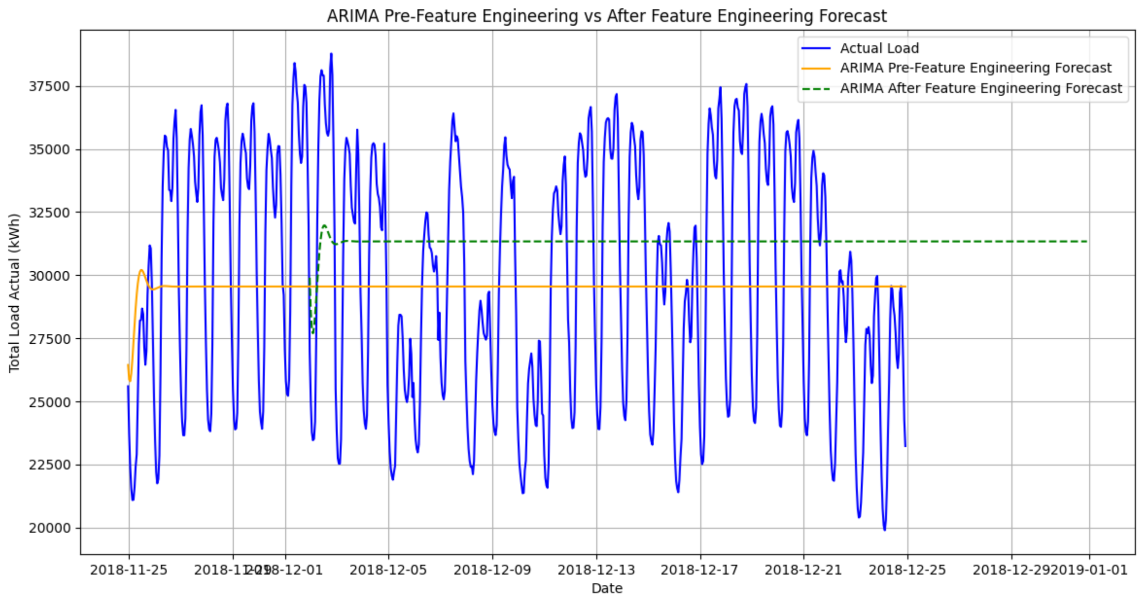
ARIMA vs. Original Forecast Plot:



The ARIMA(2,1,2) model, after a few initial steps, tends to predict a relatively flat line. This is a common issue with some statistical models when faced with complex time series, leading to suboptimal forecast accuracy compared to the original dataset's forecast. This behavior suggests the model fails to capture longer-term dynamic patterns effectively.

An ARIMA model was also trained on the data *before* extensive feature engineering to see if the added complexity was detrimental to this simpler model type.

ARIMA Pre- vs. Post-Feature Engineering Plot:



The comparison shows that the ARIMA model on data after feature engineering (which includes many lags of the target variable itself) performs slightly better than the one on data before such extensive lag creation, but both still exhibit the flat-lining behavior. This indicates that the inherent limitations of ARIMA for this specific dataset are not overcome by the feature engineering applied.

Following Karpathy, one selected aspect is **"get a feel for the data"**: The baseline setup, even with its limitations, provided a quantitative feel for the difficulty of the forecasting task and the performance level to beat. The original dataset's forecast was significantly better than the simple ARIMA, setting a higher practical benchmark.

2.3 Overfitting

For the deep learning models (LSTMs), the process started with an attempt to overfit a small subset of the data, as recommended by Karpathy, to ensure the model architecture was capable of learning.

- **Model Architecture Choice**: An LSTM architecture was chosen due to its suitability for sequence data and ability to capture long-term dependencies. The initial model consisted of an LSTM layer followed by a dense output layer.
- **Overfitting Process**: A small subset of the training data (first 30 days) was used. The univariate LSTM model (predicting `total load actual` from its past values) was trained for a high number of epochs (100) on this small dataset with a simple configuration (e.g., 50 hidden units, 1 layer, batch size 16). The goal was to achieve very low training loss, even if validation loss (on this tiny dataset) was poor, to confirm the model could memorize the data.
- **Incremental Complexity & Solving Learning Issues**: After confirming basic learning, the model was scaled to the full dataset. Issues like choosing appropriate sequence lengths, batch sizes, and learning rates were addressed during the hyperparameter tuning phase (Section 3.5). The `DLBS_Deep_learning_runs.ipynb` notebook shows the progression from a simple LSTM to more complex configurations and eventually to multivariate LSTMs.

2.4 Regularization

Regularization techniques were applied to the LSTM models to improve generalization and prevent overfitting on the full training set, aiming to gain validation accuracy by potentially sacrificing some training accuracy.

- **Dropout**: Dropout with a probability of 0.2 was introduced in the LSTM layers (if multiple layers) and before the final dense layer. This technique randomly deactivates a fraction of neurons during training, preventing complex co-adaptations.
- **Early Stopping**: This was a primary regularization method. Training was monitored on a validation set, and if the validation loss did not improve for a specified number of epochs (`patience`), training was halted, and the model weights from the epoch with the best validation loss were restored. This prevents the model from continuing to train into an overfit state.

The best model was selected based on its performance (primarily MAPE and RMSE) on the validation set during the hyperparameter tuning phase, which implicitly considered the effectiveness of these regularization techniques. The reduced multivariate LSTM, which included dropout and benefited from early stopping, was chosen as the best performing model.

2.5 Tuning

Hyperparameter tuning was performed for the univariate LSTM model to find an optimal set of parameters before extending to multivariate models. A grid search approach was implemented.

- **Grid Search**: A predefined grid of hyperparameters was searched. The parameters included `sequence_length`, `hidden_size`, `num_layers`, `dropout_prob`, `learning_rate`, `batch_size`, `optimizer`, and `loss_function`.
- **Process**: For each combination of hyperparameters, a model was trained and evaluated on the validation set. The combination yielding the best validation loss was considered optimal. The results of this search were saved in `lstm_grid_search_results.json` and a summary CSV `grid_search_detailed_results.csv`. *(Self-reflection: The grid search was computationally intensive but provided a systematic way to explore the hyperparameter space. The best parameters found for the univariate case (e.g., sequence length of 168 hours (7 days), 128 hidden units, 2 layers, 0.0 dropout, 0.001 learning rate, Adam optimizer, MSE loss) formed a strong basis for configuring the subsequent multivariate models, though some parameters like dropout were adjusted for the multivariate case based on further observations.)*

2.6 Optional: Squeeze the Juice Out of It

This step was not explicitly performed as a separate phase beyond the hyperparameter tuning and selection of the best multivariate model. However, the transition from univariate to multivariate LSTMs, and the feature selection involved in the "reduced multivariate LSTM", can be seen as an effort to "squeeze the juice" by incorporating more relevant information (weather features) and a more complex model structure where appropriate.

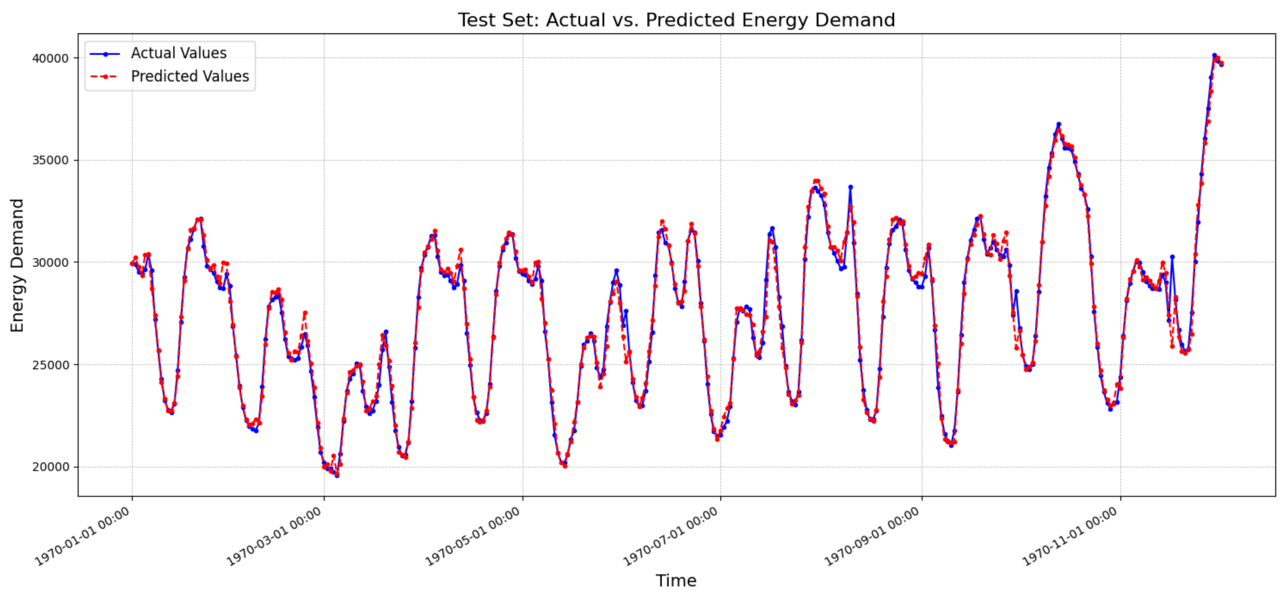
Training longer was implicitly handled by early stopping; the model trained as long as it was improving on the validation set. Ensemble models were not explored due to time constraints.

2.7 Key Results

The primary goal was to develop a model that could accurately forecast hourly energy consumption. The journey from a simple statistical baseline to more complex deep learning models yielded several key insights and results:

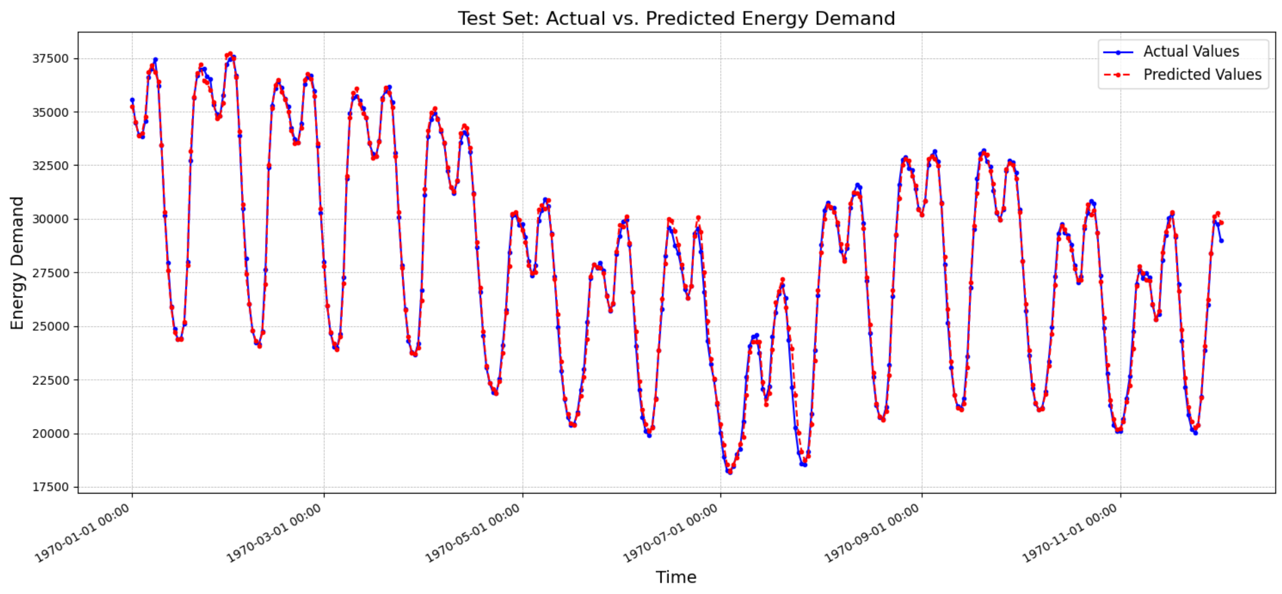
1. **Baseline Performance**: The ARIMA(2,1,2) model, while simple to implement, provided a relatively poor forecast with a test MAE of 4593.91 kWh, RMSE of 5488.19 kWh, and MAPE of 17.97%. Its tendency to flat-line predictions for longer horizons made it unsuitable for reliable forecasting in this context. The original forecast provided with the dataset was significantly better (MAE: 304.16, RMSE: 415.12, MAPE: 1.08%), setting a high benchmark.

2. **Univariate LSTM Improvement**: The best univariate LSTM model, after hyperparameter tuning, showed a marked improvement over the ARIMA baseline. It achieved a test MAE of 370.85 kWh, RMSE of 505.52 kWh, and MAPE of 1.33%. While better than ARIMA, it did not surpass the original dataset's forecast. **Univariate LSTM Predictions:**



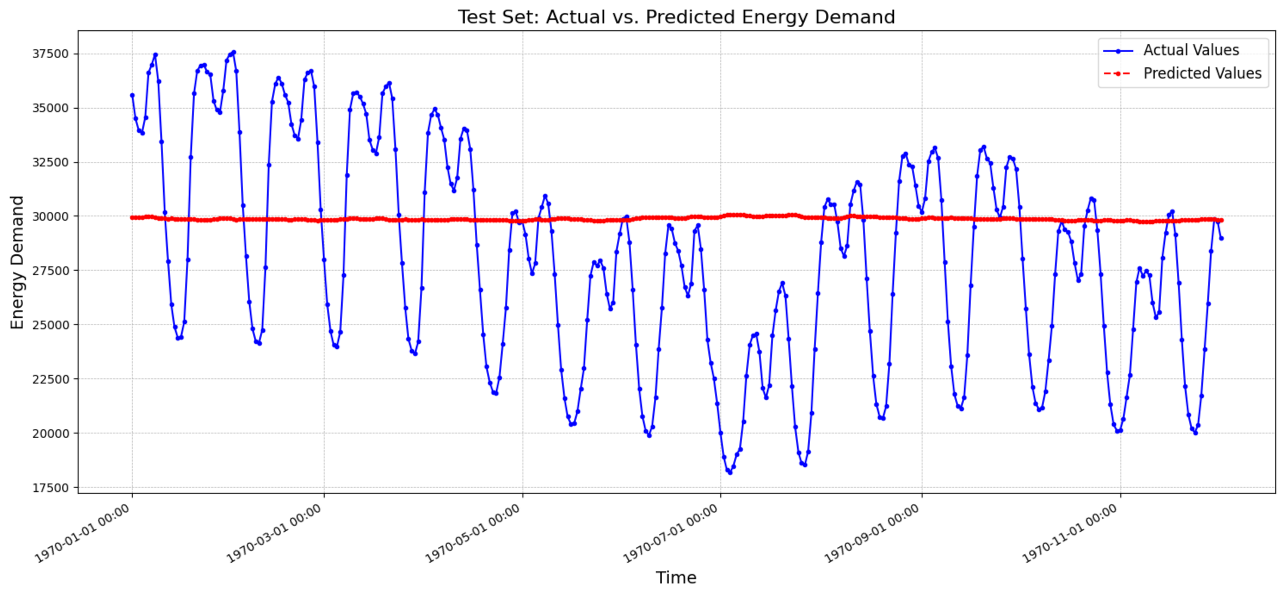
This plot shows the univariate LSTM capturing the overall daily patterns but struggling with some sharper peaks and troughs, and its MAPE is slightly higher than the original forecast.

3. **Reduced Multivariate LSTM - Best Performance**: The most significant result came from the multivariate LSTM using a selected set of features (key weather variables, time-based features, and lags). This model achieved a test MAE of 244.86 kWh, RMSE of 346.58 kWh, and an excellent MAPE of 0.87%. This performance surpassed the original dataset's forecast, demonstrating the value of incorporating relevant exogenous features. **Reduced Multivariate LSTM Predictions:**



The predictions from this model track the actual values very closely, including many of_the more volatile movements that the univariate model missed. The low MAPE indicates high accuracy. An example of a good prediction would be its ability to capture sharp morning ramp-ups in demand. A poorer (though still generally good) output might be a slight under or overestimation during extreme, less frequent peaks.

4. **Full Multivariate LSTM - Overfitting/Information Overload**: In contrast, the multivariate LSTM trained on the *full* set of engineered features (including extensive lags for all weather variables, totaling 98 features) performed very poorly. Its predictions essentially became a flat line, similar to the ARIMA model but at a different level, with a very high MAPE (e.g., test MAE 2871.01, RMSE 3509.43, MAPE 10.43%). **Full Multivariate LSTM Predictions:**



> *This demonstrates a critical learning: more data/features are not always better. The model was likely overwhelmed by the high dimensionality and noise, failing to learn meaningful patterns.*

In summary, the reduced multivariate LSTM stands out as the most successful model, effectively leveraging both historical energy data and external factors to produce accurate forecasts. The importance of careful feature selection for complex models was a crucial takeaway.

3. Discussion

Answering the Research Question: The research question asked how deep learning methods can improve hourly electricity demand forecasting in Spain. The results provide a definitive answer: LSTM networks, when properly configured with relevant exogenous features, can significantly improve forecasting accuracy. The multivariate LSTM achieved 0.87% MAPE for hourly predictions, representing a substantial improvement over traditional statistical methods (ARIMA: 17.97% MAPE) and even surpassing the existing forecast benchmark (1.08% MAPE). This demonstrates that deep learning methods can indeed contribute meaningfully to improving hourly electricity demand forecasting in Spain.

Error Analysis: A MAE of approximately 245 kWh means that, on average, the prediction for any given hour is off by that amount. Given that average loads can be in the tens of thousands of kWh (e.g., 25000-30000 kWh), this absolute error is relatively small. The RMSE of 346 kWh, being slightly larger than MAE, suggests some instances of larger errors, but not excessively so. For a utility company, such errors would likely be manageable within typical operational buffers.

Pros, Cons, Opportunities, Risks:

- *Pros*: The LSTM approach is data driven and can capture complex non linear patterns that simpler models miss. It can adapt to changing consumption behaviors if retrained periodically. The inclusion of weather data makes it robust to weather induced demand changes.
- *Cons*: LSTMs are more complex to develop, tune, and require more computational resources than ARIMA models. They can be "black boxes," making it harder to understand the reasoning behind specific predictions. The need for careful feature engineering and selection is also a drawback.
- *Opportunities*: Highly accurate forecasts enable better grid stability, reduced operational costs (e.g., by optimizing generator dispatch), and more efficient energy trading. They can also support the integration of renewable energy sources by predicting demand more reliably.

- *Risks*: Poor forecasts can lead to grid instability, blackouts (As has happened in spain and portugal earlier this year), or financial losses. Over reliance on a complex model without robust monitoring and validation processes is a risk. The model's performance might degrade if underlying consumption patterns change drastically or if input data quality issues arise (e.g., sensor failures for weather data).

Baseline vs. Optimized Models: The optimized multivariate LSTM significantly outperformed the ARIMA baseline. The baseline served its purpose in highlighting the complexity of the task and providing a lower bound on performance. The LSTM's ability to model non-linearities and leverage exogenous variables was key to its superior performance.

Cases Working Well vs. Poorly: The best model (reduced multivariate LSTM) worked well in capturing typical daily and weekly cycles, including ramp-ups and ramp-downs. It also adapted well to variations presumably caused by weather changes. It might perform less well during highly unusual events not represented in the training data (e.g., unforeseen extreme weather events, national emergencies affecting energy use). The full multivariate LSTM performed poorly across the board, indicating a failure to learn rather than specific situational weaknesses.

Model and Data Sufficiency: The chosen LSTM models and the available data (energy load and weather) appear sufficient to answer the research question for short term forecasting. The quality of the data was generally good after initial cleaning. For longer term forecasting or predicting very specific events, more specialized data or models might be needed.

4. Reflection

This mini challenge gave me good hands on experience in developing deep learning based time series forecasting models. The primary conclusion for the specific research question is that LSTMs, particularly when enhanced with relevant exogenous features like weather data, can achieve high accuracy in energy load forecasting, surpassing simpler statistical models and even good quality existing forecasts.

What went well: The systematic approach, starting from EDA, establishing a baseline, and then incrementally building and evaluating LSTM models, was effective. The feature engineering, especially the creation of cyclical time features and appropriate lags, proved beneficial.

What I would do differently next time: I would allocate more time for exploring a wider range of hyperparameter optimization techniques beyond grid search, perhaps trying random search or Bayesian optimization, which can be more efficient. I would also investigate attention mechanisms within the LSTM architecture, as they might further improve performance by allowing the model to focus on more relevant past time steps or features. More rigorous statistical tests for comparing model performances would also be beneficial. And finally maybe further testing xLSTMs, as I tried to get them working but was having a lot of trouble early on, which lead me to just fall back onto regular LSTMs.

Changes to task specification: Personally I wouldnt change anything, maybe giving students a fallback predefined problem statement if they cant come up with anything themselves. But other than that, I feel everything was good.