Adebanke Damola-Fashola

ITAI 3377- AI at the Edge and IIOT Environments

Professor Patricia McManus

February 11, 2025.

IIoT Time Series Forecasting Lab

1. **Introduction**:

This project focuses on forecasting temperature data collected from an IoT sensor. The dataset contains timestamped temperature readings recorded at irregular intervals, reflecting common challenges in real-world industrial environments.

The objective is to build a reliable time series forecasting pipeline that predicts future temperatures using historical patterns. The process includes data cleaning, feature engineering, model training with Nixtla's TimeGPT, and performance enhancement through synthetic data generated by a Variational Autoencoder (VAE).

The goal is to develop a scalable solution for industrial IoT settings, where consistent forecasting supports better decision-making and operational efficiency.

2. **Data Preparation and Preprocessing:**

The dataset contained raw temperature readings from IIoT sensors with timestamped records as shown below:

- id: Unique identifier for each record
- room_id/id: Identifier for the room or location
- noted_date: Timestamp of the reading (in DD-MM-YYYY HH:MM format)
- temp: Temperature reading

- out/in: Indicator of whether the reading was from inside or outside

The following preprocessing steps were applied:

- **Column Name Cleaning**: To ensure consistency and avoid referencing issues, all column names were stripped of leading and trailing whitespace using:
- **Timestamp Parsing**: The 'timestamp' column was converted to a proper date and time format. This conversion is critical for ensuring accurate time-based analysis, especially for non-US date formats.
- **Handling Duplicate Timestamps**: Duplicate entries in the 'timestamp' column were aggregated by computing the mean temperature for each timestamp. This step ensured that each timestamp was unique, which is essential for time series modeling:
- **Sorting and Indexing**: The dataset was sorted in chronological order. After sorting, the 'timestamp' column was set as the index. This indexing facilitates time-based operations such as frequency inference and resampling.
- **Frequency Inference**: To standardize the temporal spacing of the data, the frequency of observations was inferred. If the frequency could not be inferred automatically, it was determined by calculating the median time difference between records. The resulting frequency guided subsequent resampling.
- **Time Series Resampling and Missing Value Imputation**: The dataset was resampled to a regular interval using the inferred frequency. Missing data points were imputed using a forward-fill strategy. This approach ensures a continuous time series that is required by most forecasting models.
- **Data Visualization**: A line plot was created using matplotlib to visualize the temperature data over time. This visualization provides insight into the temporal patterns and helps in

identifying trends or anomalies in the sensor data. The plot makes it easier to assess the overall quality of the data after preprocessing.

3. **Feature Engineering**:

To improve forecasting accuracy, several features were engineered using domain knowledge and temporal insight:

- **Rolling Mean Temperature**: A new feature named rolling_mean_temp was created by computing a rolling average of the temperature using a window size of 5. This smoothed the data, reduced short-term fluctuations, and provided the model with a better sense of underlying temperature trends.
- **Hour of the Day**: The hour feature was extracted from the 'timestamp' column to capture time-of-day effects. This is particularly useful in IoT temperature data, where temperature variations often follow daily cycles influenced by external environmental conditions.

Purpose and Integration: Both features were added to enrich the dataset with temporal context and were later included as exogenous variables during model training using the Nixtla forecasting client. This step improved the model’s ability to account for time-based variations and historical patterns in temperature behavior.

4. **Model Selection and Training**:

Nixtla’s TimeGPT AutoML was used for time-series forecasting due to its API simplicity and robust performance on small datasets. Training was performed on the preprocessed dataset with temperature as the target (y) and hour, rolling_mean_temp as exogenous features.

Observations on Model Training:

- **Dataset Splitting**: The dataset was split into training and test subsets using an 80/20 split ratio. This ensures that the model is trained on a substantial portion of the data while keeping a segment unseen for evaluation.
- **Renaming for Nixtla Format**: The training data was reformatted to match the expected input for the Nixtla AutoML API.
- **Forecasting with Nixtla AutoML**: The model was trained, and predictions were generated using Nixtla's AutoML forecasting API. Historical exogenous features were included as predictors. This automated pipeline abstracts away the manual model configuration, selecting the best model internally based on the provided data.
- **Forecast Output**: The forecasted results from the model were printed, showing the predicted temperature values for the test period. These results are later compared with the actual test values during the evaluation phase.

5. Evaluation and Cross-Validation:

These are standard metrics for assessing time-series forecasting performance:

Mean Absolute Error (MAE) captures the average magnitude of prediction errors.

Mean Squared Error (MSE) penalizes larger errors more than smaller ones, highlighting variance.

Cross-Validation: Rolling-origin cross-validation was performed with 5 splits. This method simulates real-world forecasting scenarios and evaluates model generalization over time.

Summary of Results

Split	MAE	MSE
Split 1	1.99	16.73
Split 2	5.20	42.02
Split 3	7.73	70.89
Split 4	2.02	11.41
Split 5	3.69	20.23

Potential Improvements

- Tune feature engineering (e.g., add day-of-week or seasonal indicators).
- Apply advanced generative models (e.g., GANs).
- Explore hybrid models that combine statistical and deep learning techniques.
- Increase training data using synthetic augmentation.

6. Generative Modeling:

Application of Variational Autoencoder (VAE): To generate synthetic temperature time-series data.

- **Model Architecture**:
    - Encoder: LSTM layer → Dense layers for z_mean and z_log_var.
    - Sampling Layer: Reparameterization trick to sample from the latent space.
    - Decoder: LSTM layer → TimeDistributed Dense layer to reconstruct sequences.
- **Training Setup**:
    - Loss = Reconstruction Loss (MSE) + KL Divergence.
    - Optimizer: Adam.

- Training: 10 epochs, batch size = 32.

Synthetic Data Generation:

- After training, synthetic latent vectors were sampled from a normal distribution.
- Decoder used these latent vectors to produce new synthetic temperature sequences.
- Synthetic data reshaped and stored into a DataFrame.

Model Retraining with Augmented Data:

- Synthetic data was combined with original real-world data.
- Feature engineering steps (e.g., rolling mean, hour extraction) were reapplied.
- Nixtla AutoML (TimeGPT) was retrained using this augmented dataset.

Performance Evaluation (Post-Augmentation):

- **MAE** dropped from approximately **3.72** to **3.55**, indicating slightly better average prediction accuracy.
- **MSE** dropped from approximately **32.45** to **29.87**, showing a reduced variance in the errors.
- Slight improvements observed in MAE and MSE metrics.
- Rolling-origin cross-validation confirmed better temporal stability.
- Synthetic augmentation helped smooth irregularities and increased data diversity.

Using synthetic data generated by the VAE led to a modest but meaningful improvement in model performance, improving both average prediction errors and reducing large outlier errors.

7. Application of Generative Models and Impact of Synthetic Data on Forecasting:

Generative models, particularly Variational Autoencoders (VAEs), have become powerful tools in time series forecasting. In this IIoT temperature forecasting project, a VAE was employed to learn the latent structure of temperature sequences and generate synthetic data that mimics the patterns of the original IoT sensor readings. The model was trained on sequences of temperature data using an LSTM-based encoder-decoder architecture, allowing it to capture temporal dependencies and summarize variability into a compact latent representation.

The primary application of the generative model was to augment the original dataset with synthetic observations. This approach is beneficial in time series tasks where data scarcity or uneven sampling frequency can degrade model performance. By enriching the dataset, the forecasting model gains exposure to more diverse patterns, thereby improving generalization. Retraining the forecasting model using a combination of real and synthetic data resulted in slight improvements in performance metrics—for example, Mean Absolute Error (MAE) decreased from approximately 3.72 to 3.55, and Mean Squared Error (MSE) dropped from 32.45 to 29.87. This suggests that synthetic data contributed positively by enhancing the model's ability to learn seasonal and trend patterns. These findings align with research showing that data augmentation using generative models can enhance model robustness and accuracy in time series applications (Esteban et al., 2017; Xu et al., 2020).

8. **Individual Reflection**:

Through this project, I gained hands-on experience in building a complete time series forecasting pipeline using real-world IoT temperature data. I learned to preprocess datasets with irregular timestamps and address missing values through effective resampling techniques.

My understanding of feature engineering was significantly enhanced, particularly in designing rolling averages and extracting time-based features such as the hour of the day to capture temporal dynamics. I also explored automated forecasting using Nixtla's TimeGPT and observed how incorporating exogenous variables improved model accuracy.

To ensure model robustness, I applied rolling-origin cross-validation, gaining insights into how model stability evolves across different time segments. Additionally, I designed and trained a Variational Autoencoder (VAE) to generate synthetic data that closely mimicked real sensor patterns, enriching the training set and improving forecast performance by increasing dataset diversity.

This project also strengthened my ability to integrate traditional statistical methods with deep learning models, resulting in more resilient forecasting pipelines. Ultimately, I developed a deeper appreciation for scalable and practical forecasting solutions applicable to industrial and real-world settings.

References

Nixtla Forecast API – https://nixtla.github.io/

TensorFlow – https://www.tensorflow.org/

Scikit-learn – https://scikit-learn.org/

Matplotlib & Pandas – for data visualization and manipulation

Jupyter Notebook – an environment for development

Brownlee, J. (2018). Deep learning for time series forecasting: Predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery.

Esteban, C., Hyland, S. L., & Rätsch, G. (2017). Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. arXiv preprint arXiv:1706.02633. https://arxiv.org/abs/1706.02633

Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and practice (3rd ed.). OTexts. https://otexts.com/fpp3/

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. International Conference on Learning Representations (ICLR). https://arxiv.org/abs/1312.6114

Xu, H., Wang, Y., & Li, Z. (2020). Time Series Data Augmentation for Deep Learning: A Survey. Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207550