Oren Moreno

ITAI 3377

May 6, 2025

Professor Patricia McManus

L06 IIoT Time Series Forecasting

Introduction – Overview of the dataset and problem statement.

This project focused on predicting future temperatures from an Industrial IoT (IIoT) sensor using its past data. Being able to forecast these temperatures is important for things like spotting equipment issues early, running industrial processes more efficiently, and managing energy. The data came from a Kaggle dataset (koheimuramatsu/iot-temperature-forecasting), with temperature readings taken over several months in 2018. I used Python tools like Pandas for handling the data, Scikit-learn for some preparation steps, Nixtla for the actual time series forecasting, and TensorFlow/Keras to build a Variational Autoencoder (VAE) to create extra training data.

Data Preparation – Preprocessing steps taken.

Getting the data ready was a big first step. After loading the temperature readings, I converted the date information into a proper datetime format and sorted everything by time. A major task was dealing with a lot of duplicate timestamps – more than half the data was repeated! I removed these, which cleaned up the data but also created gaps, making the timeline irregular. The cleaned data (about 28,000 unique readings) was then split into 80% for training the model and 20% for testing it. I also scaled the temperature values to be between 0 and 1 using a MinMaxScaler, which helps the neural network models learn better. Because Nixtla needs data with a perfectly regular time interval (about one reading every minute), I had to "regularize" my training data. I created a complete minute-by-minute timeline and filled in my actual readings, using forward and backward fill to plug the gaps left by the duplicate removal. This gave Nixtla a smooth, consistent dataset to work with.

Model Selection & Training – Justification of the chosen model (Nixtla or Wolfram).

I chose Nixtla for forecasting because it's specifically designed for time series and offers advanced models like TimeGPT. I felt this would give better results than using a general-purpose machine learning model. My first model was trained using Nixtla's timegpt-1-long-horizon with just the regularized, scaled temperature data. This initial run successfully produced a forecast.

Feature Engineering & Evaluation – Key findings and model performance.

To try and improve the predictions, I added two features: the hour of the day (to catch daily patterns) and temp_lag_1 (the temperature from the previous minute, since temperatures often don't change suddenly). After adding these features, I re-split the data and re-scaled the temperatures. Then, I prepared the data for Nixtla again, making sure to regularize both the historical data (with features) and the future feature values.

The model trained with these features had a Mean Absolute Error (MAE) of about 1.94 and a Root Mean Squared Error (RMSE) of about 3.04 on the test set. Looking at the plot, the model followed the general temperature trends pretty well, though an error of around 3 degrees Celsius might be more or less acceptable depending on the specific use case. I thought about using rolling-origin cross-validation for a more thorough test, but my dataset was a bit too small for the setup I planned.

Generative Models – Impact of synthetic data on forecasting.

Next, I experimented with a Variational Autoencoder (VAE), a type of neural network, to create more training data. The idea was that if the VAE could learn what normal temperature patterns look like, it could generate new, realistic data. I trained the VAE on short sequences of the scaled temperature data. After some trial and error with its setup, the VAE trained well and was able to generate over 500,000 new synthetic data points that looked similar to the real ones. I then added this synthetic data to my original training data and retrained the Nixtla model (this time without the extra hour and lag features, for simplicity). However, a problem came up during evaluation: the Nixtla model, having seen all this extra data, started its forecast *after* the end of the synthetic data period (which was much later than my original test set). This meant I couldn't directly compare its predictions to my actual test data to see if the augmentation helped, because the timestamps didn't match up.

Individual Reflection

This project was a deep dive into practical time series forecasting. The biggest lesson was how much work data preparation can be, especially dealing with messy real-world data like the duplicate timestamps. Making the data regular for Nixtla was a key technical step. Building and debugging the VAE was also challenging but taught me a lot about neural networks. The most interesting part was seeing how a tool's specific behavior—like Nixtla's forecast start time—can impact how you evaluate your experiments. It showed that understanding your tools is just as important as understanding the algorithms themselves. While I couldn't definitively prove the VAE data helped the final forecast in this setup, I learned a lot about each part of the process.