Machine Learning Exercise Task: Weather Forecasting Documentation

1. ML Task Overview

For this task, two models are employed: Multivariate ARIMA and LSTM, *in order to forecast the last 48 hours of the dataset.* Both models are suitable for seasonal data.

ARIMA (AutoRegressive Integrated Moving Average) is a statistical method used for time series forecasting. ARIMA consists of three main components: Autoregression (AR), which uses past values of the variable to predict future values; Differencing (I), which transforms the time series to make it stationary and Moving Average (MA), which uses past forecast errors to predict future values.

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) designed to process sequential data by maintaining an internal memory. LSTMs can capture long-term dependencies in the data, learn patterns and relationships within sequential data, making them particularly useful for time series forecasting.

The following steps are followed:

1. **Data Cleaning**

    - Each column is plotted for visual inspectio purposes.
    - Data are smothed with a rolling window of 120 points (5 days). Smoothing can increase the efficiency of a forecasting model, as it can reveal underlying patterns and trends, as well as reduce noise.
    - No singificant outliers are observed. Noise can be reduced with smoothing, as mentioned above.
    - Remove "Apparent Temperature" due to its high correlation (99%) with "Temperature", potentially causing data leakage.
    - Replace missing data by linearly interpolating. Given the minimal proportion (0.8%) of missing values in the dataset, this method is efficient.
      Address missing data of categorical columns by replacing them with the mode of each column.
    - Data points occurring after 2016-11-01 are remove as they appear artificial.

2. **Feature Generation and Selection**

    - Convert categorical columns to numeric using label encoding to facilitate model training.
    - Select columns with an absolute correlation coefficient greater than 0.4, while removing columns with zero variance.

3. **Fine-Tuning**

    - Utilize pmdarima's stepwise search to identify optimal parameters for the ARIMA model, optimizing forecasting accuracy. Pmdarima library also deals with non-stanionary timeseries.

- LSTM is tuned by visual inspection.

4. **Forecasting**

   - Employ Multivariate ARIMA to address seasonal data patterns, enabling the capture of seasonal trends. The features selected based on correlation (as explained in section 2) are used as exogenous parameters in the model.
     Use 120 past days as training.
     Last 48 hours are excluded, so that the efficience of the model can be tested.

   - MSE between 48-hour test set and forecast is ~4%, showing that ARIMA is efficient.

   - LSTM is trained two times, using the entire dataset for training: 40 days as past window for 50 epochs and 90 days as past window for 10 epochs. Note that, in the last example, only 10 epochs are used only for visualization purposes. More epochs are needed, so that LSTM can capture the underlying patterns.

5. **Additional Notes – Conclusion**

ARIMA is suitable for short-term forecasting, however it is sensitive to outliers and unsual data patterns. Thus, smoothing of the dataset increased the efficience of the model.

LSTM on the other hand is a recurrent neural network, that can capture long-term temporal patterns and perform long-term forecasting. However, it requires computational power and fine-tuning can be time-consuming when run locally.

2. Database and Flask Application

The database is created with sqlite3 module within python.
A Flask application inserts forecast data derived from ARIMA and includes a GET and a POST endpoint.

3. Scripts

arima_.py: Runs ARIMA model, returns forecast values and saves the model locally.
create_db.py: creates database (called in app.py)
process_data_.py: includes all functions necessary for data cleaning, feature selection etc.
Executable file: app.py: Flask application that creates sqlite database (with create_db.py), runs arima_.py and inserts forecast values to database. Also includes GET and POST endpoints.

Additional scripts (executed seperately from app.py):
lstm_.py: Runs LSTM model and saves the model locally.
load_models_and_plot_.py: Loads both ARIMA and LSTM models and create plots, showing the efficiency of each model. Also creates correlation heatmap for all parameters.

4. Plots

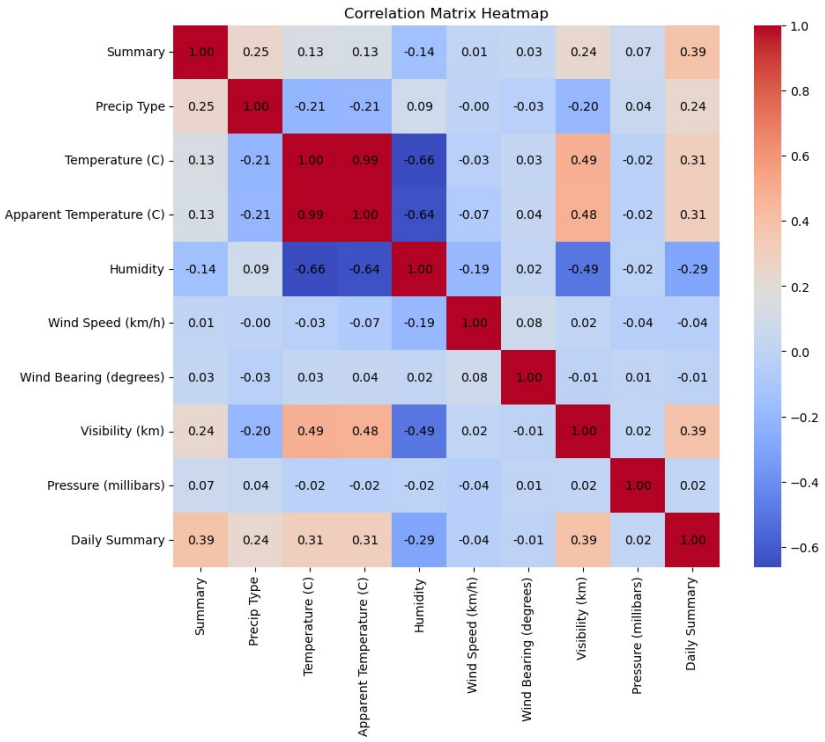

Illustration 1: Corellation matrix of all parameters

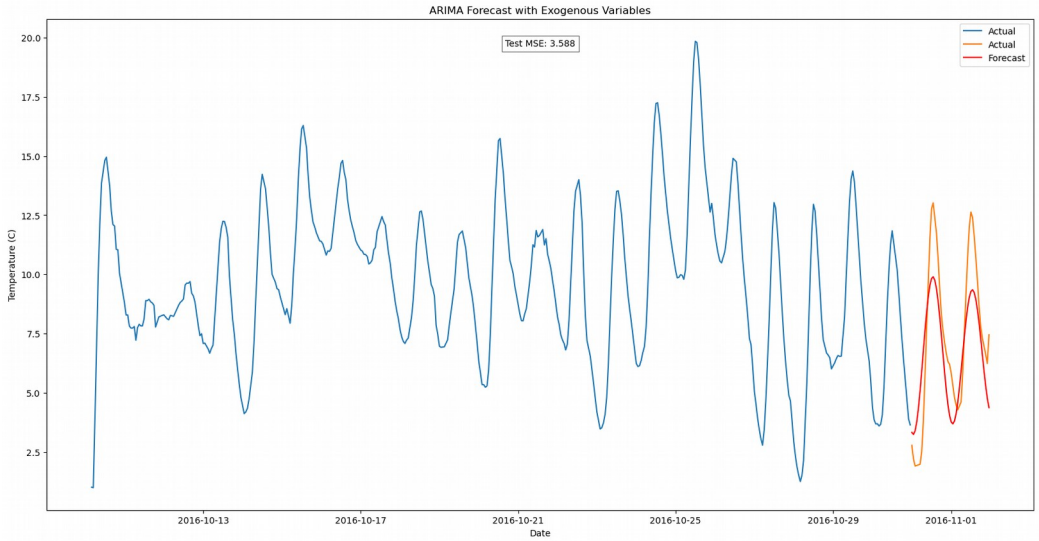

Illustration 2: ARIMA model

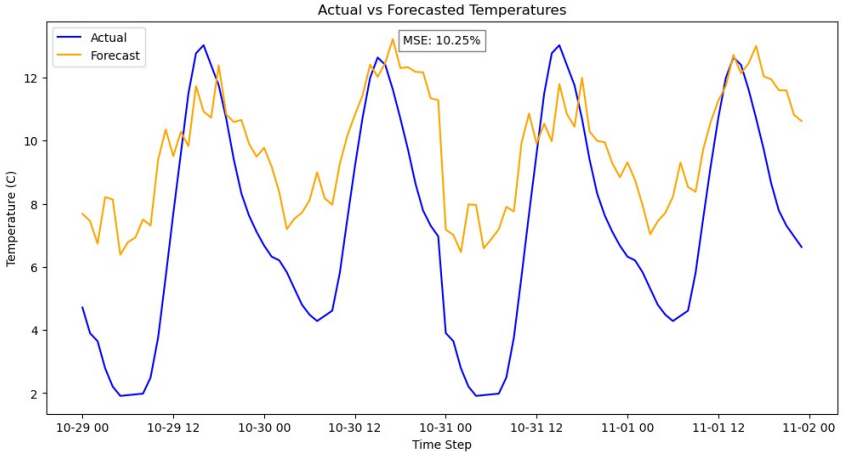

Illustration 3: LSTM with 40 days for training and 50 epochs

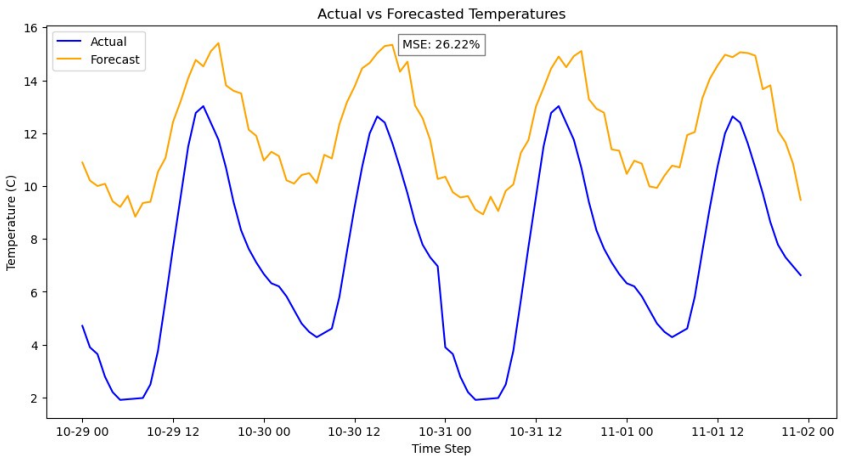

Illustration 4: LSTM with 90 days for training and 10 epochs

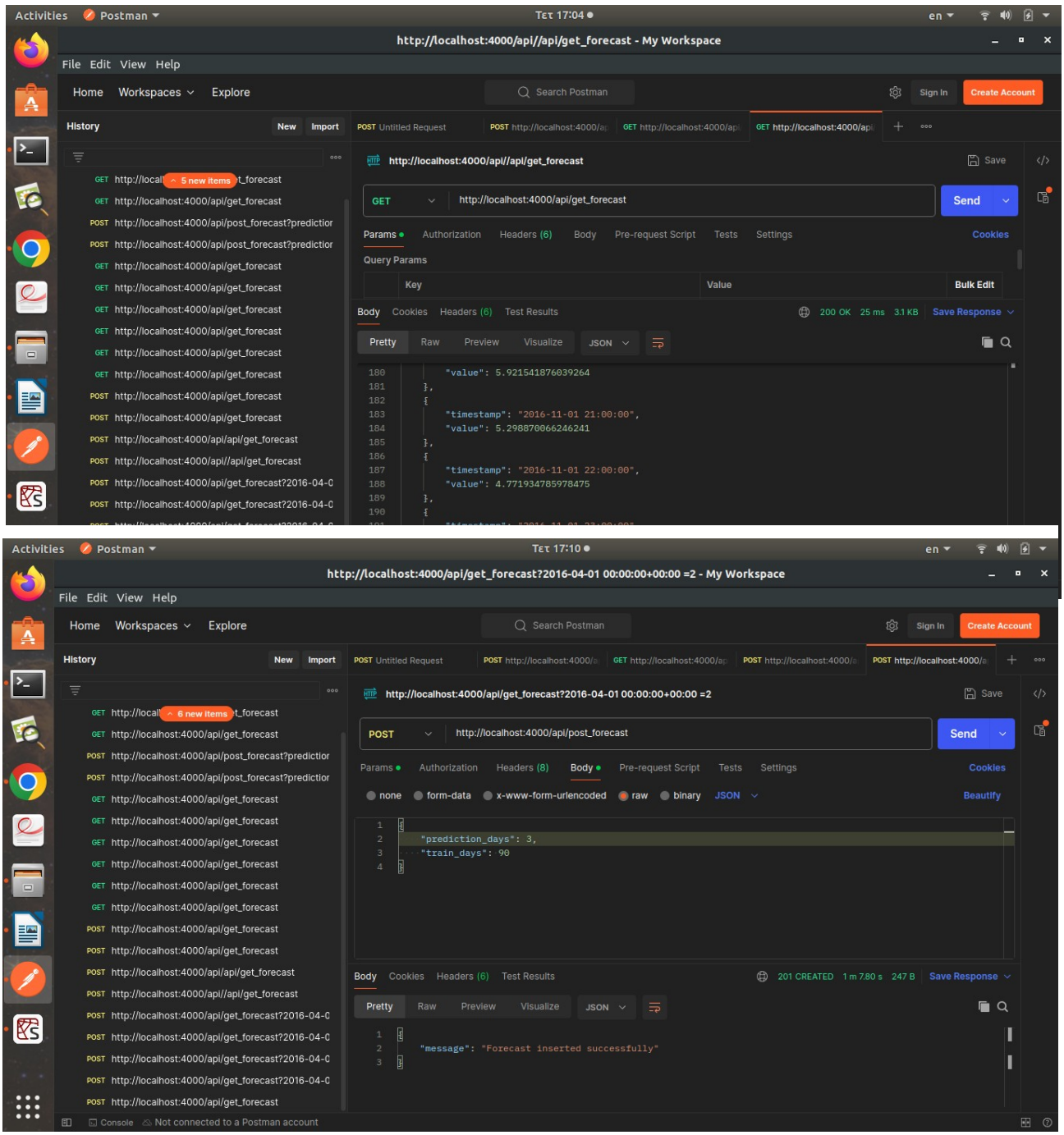

Illustration 6: Postman - POST endpoint

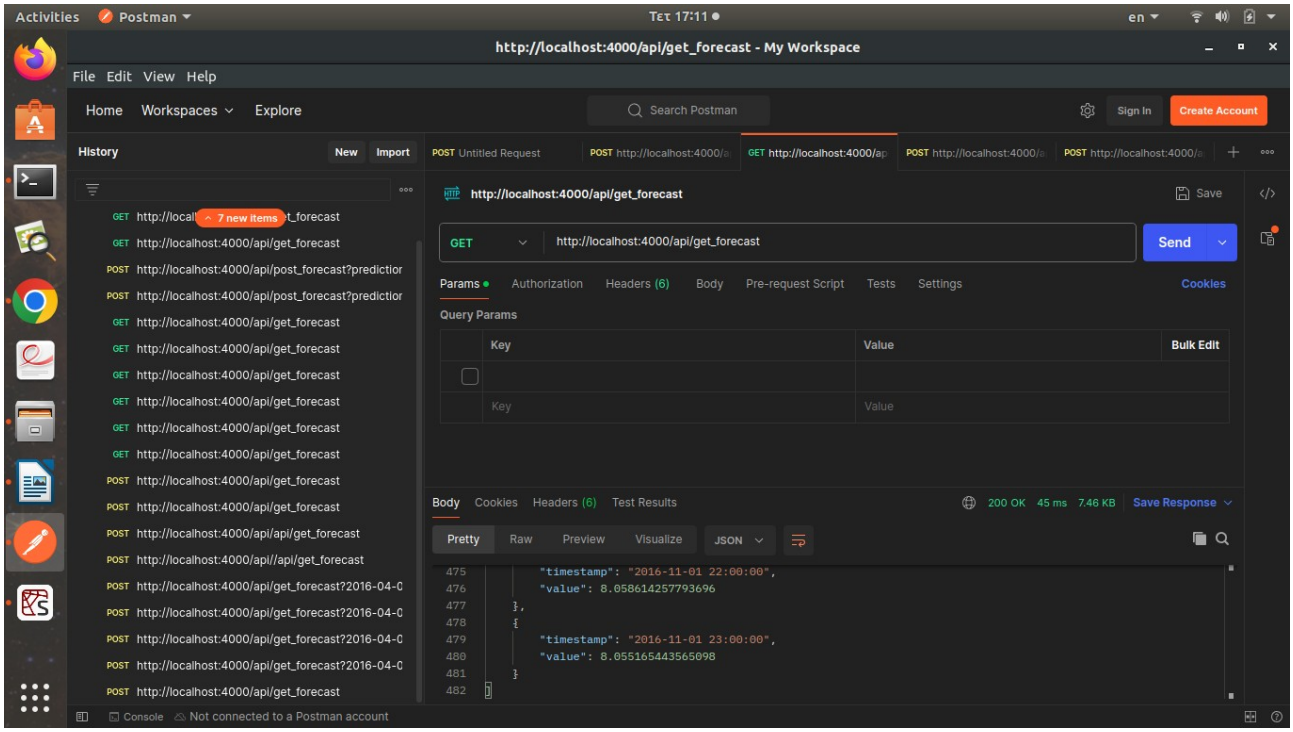

Illustration 7: Postman GET endpoint after POST