Router Anomaly detection: Pipeline & Model Design

This document describes the data pipeline, feature engineering, scaling , sequence generation and modelling aspects of the pipeline.

Quick Note:

Both the resources and logs folders are present inside the src folder for simple imports

Quick Summary:

Input: The pipeline takes a csv input per router (Max and mean metric value) calculates the rest of the features.

Output: A forecasting model that predicts the next time stamps for max and mean metric value for each router and classifies based on the residuals

To run the testing:

Python test.py –routernumber 0

(Change the number of router number (0-9) for different testing)

Test.py shows 1) General forecast 2) Anomaly detection result (Both in graphs)

Key Pipeline Components:

1) Data import: `src/data_preparation.py::import_dataset` and `import_test_dataset` - Imports the CSV file for both the training and testing phase. Can be used for inference as well
2) Feature Engineering: `src/data_preparation.py::compute_features` - Creates multiple features from max and mean metric value to forecast the next time stamps
3) Data Upsampling: `src/data_preparation.py::upsample_per_interface` - Upsamples the frequency of the datapoint to 1 minute since less number of data points were present
4) Outlier Imputation: `src/data_preparation.py::impute_outliers_series` and `impute_outliers_group` - Takes a group of points from a single router and performs winsorizing using IQR range to cap the extreme values
5) Encoding: `src/data_preparation.py::encoding_router` - Used for encoding the router names to integer for efficient embedding
6) Sequence Builder: `src/data_preparation.py::building_lagged_sequences` - Builds time lagged sequences that can be used for training and testing. The current window used is 6 and it can be changed in config.py
7) Scaling: `src/data_preparation.py::scale_data` - Scales the data based on robust scaler to make sure that extreme values that does not skew the scaling process
8) Modeling : `src/model.py::build_model` - Compiles and builds the tensorflow LSTM model
9) Hyper parameter tuning: 'src/Hyperparameter_tuning.ipynb ' – Performs hyperparameter tuning for the LSTM model using Random search. Package used is keras tuner.

Model Design Rationale:

1) The task was treated as a forecasting + unsupervised classification problem due to unclear labels

2) An embedding layer for the LSTM model was used to adapt a global model for different baseline behavior between routers. This also helps the model condition the temporal dependencies based on the router and learn effectively.
3) The prediction is made for both max and mean since they both capture short immediate fluctuations and long over lasting fluctuations as well. Max covers – peak and Mean covers typical load
4) Per router split and scaling was performed as each routers were heterogeneous and to avoid bleeding scale information across routers.
5) Winsoring + robust scaler was used to make sure extreme outliers doesn't affect scaling and modelling of LSTM with patterns
6) The Dbscan was used to detect the anomalies that doesn't belong with the distribution. It acts supervised and doesn't need a thresholding for classification unlike some unsupervised techniques.