README

1. Introduction:

As the name suggests, time series anomaly detection is the technique of identifying anomalies in a given time series. A collection of data points recorded at regular intervals is called a time series. These informational pieces may be recorded on an hourly, daily, weekly, monthly, quarterly, or annual basis. We can glean valuable information from the data, such as statistical traits and patterns, through time-series analysis. A key element of data observability systems is the ability to locate anomalous data points that can be seen as substantial spikes or decreases in a time series dataset. A crucial occurrence in your company or a potential opening to seize in order to influence new decisions can be indicated by anomalous data. For instance, in a financial system that is data-driven, the identification of anomalous patterns in transaction data may be a sign of possible fraud. The identification of anomalies in sensor data in a data-driven manufacturing system may point to a machine or production process issue that has to be fixed.

Finding abnormalities in time-dependent multivariate data is the aim of this study. In order to do this, a high significantly anticipated threshold is predicted, and it is then shown to be capable of self-adjusting in response to the dynamic input from the data observability system. A datapoint is considered an anomaly and an alert is triggered if it exceeds this level. To do this, we go forward by developing prediction models employing historical data in order to gauge and estimate the general common trend, seasonality, or cyclicity of the time series data. An ensemble model with improved performance will be created using Deep Learning, which has recently been proven to be quite effective.

2. System Requirements:

- Python: You need to have Python installed on your system. You can download Python from the official Python website and follow the installation instructions specific to your operating system.
- PyTorch: You need to install PyTorch, a deep learning framework, for the machine learning parts of your code. You can install PyTorch by visiting the official PyTorch website and following the installation instructions based on your system configuration.
- PyTorch Geometric: You need to install PyTorch Geometric, a library for deep learning on irregular input data such as graphs. You can install PyTorch Geometric by following the installation instructions available on the PyTorch Geometric GitHub repository
- SciPy: You need to install SciPy, a scientific computing library, for various statistical computations in your code. You can install SciPy by visiting the official SciPy website and following the installation instructions specific to your operating system.
- scikit-learn: You need to install scikit-learn, a machine learning library, for data pre-processing and evaluation metrics. You can install scikit-learn by visiting the official scikit-learn website and following the installation instructions specific to your operating system.
- NumPy: You need to install NumPy, a numerical computing library, for various numerical operations in your code. You can install NumPy by visiting the official NumPy website and following the installation instructions specific to your operating system.
- Pandas: You need to install Pandas, a data manipulation library, for handling data in tabular format. You can install Pandas by visiting the official Pandas website [28] and following the installation instructions specific to your operating system.
- Matplotlib and Seaborn: You need to install Matplotlib and Seaborn, plotting libraries, for data visualization in your code. You can install Matplotlib and Seaborn by visiting their official websites and following the installation instructions specific to your operating system.
- Prophet: You need to install Prophet, a time series forecasting library, for forecasting tasks in your code. You can install Prophet by visiting the Prophet GitHub repository and following the installation instructions provided there.
- Plotly: You need to install Plotly, a graphing library, for interactive visualizations. You can install Plotly by visiting the official Plotly website and following the installation instructions specific to your operating system.
- TensorFlow: You need to install TensorFlow, a deep learning framework, for some parts of your code. You can install TensorFlow by visiting the official TensorFlow website and following the installation instructions specific to your operating system.

- Keras: You need to install Keras, a high-level neural networks API, for building and training neural networks in your code. You can install Keras by visiting the official Keras website and following the installation instructions specific to your operating system.
- joblib: You need to install joblib, a library for saving and loading Python objects, for parallel execution in your code. You can install joblib by visiting the joblib GitHub repository and following the installation instructions provided there.

The notebook include the code for installing the required packages. However, if the versions don't match there might be an issue with importing the packages. In that case, the compatible versions should be installed separately.

3. Usage Instructions:

To run the code upload the notebook to Google colab and connect to runtime with GPU:

1. To the files section, upload the training and testing dataset in csv format or just a whole dataset that will be later split into train and test datasets accordingly. The datasets must be time series data with a date column (which is usually the first column in the dataset)
2. In the *main*() method, reset the variable *'dataset'*,'train_path' and *'test_path'* to the name of the dataset and the path to train and test dataset that you wish to use. If the dataset is not already split into test and train, give the path to the entire dataset to the 'train_path'. The program will split it as it executes. It is recommended to give the whole dataset instead of the train and test split datasets.
3. In case you have saved models for LSTM-NDT, create a folder in the files section and name it 'models'. Upload the saved models to this folder along with the errors.csv file. In *main()* method, update the *'config_lstmndt'* dictionary by changing the value of *'train'* to *False*. Or if you wish to train them on a dataset from scratch, set the 'train' value to *True*
4. In *main*() method the dataset name should be given to 'dataset' variable.
5. Dataset path should be given to 'train_path'
6. Click on runtime and click on *run all.* It may take a while because the program includes the code to install packages which takes around 45-60 mins. It might show warning you can ignore those.
7. When the *main*() cell is executed, there is an input prompt to give the name of the target column. Type the name of the target column as in the dataset csv file.
8. Next you are prompted to give the names of the regression columns. Specify the rest of the columns other than the target column separated by a comma.

9. The ensemble will process the data, train and test it. The best model among the pool will be used to detect anomalies. Finally, the results are logged to the output console. The output is explained in the next section

4. Output and Interpretation:

- The output displays the results of the ensemble model. The best model that has been chosen is displayed the number of point anomalies and collective anomalies along with the grouped anomaly indices are displayed.
- The MAE, MSE and MAPE scores corresponding to the best model are displayed as well.
- The saved models on the ensemble are found in the files section.

5. Performance and Limitations:

- The execution time is displayed at the end of the execution. It varies based on the size and dimension of the dataset. However, installing the packages might take 45 to 60 minutes.
- The LSTM-NDT model has a large runtime that may takes minimum of 60 minutes for a dataset of 6000 rows and 7 columns.
- On the other hand, the GDN and Prophet models have a short runtime which may range between a few seconds to hundreds.
- For proper training of the Prophet model the dataset must contain a date column in a date/datetime format.

6. Conclusion:

Automating the task of selecting a model for a dataset has proven to be challenging. A particular model may demonstrate good performance on one dataset but it might not generalize well for other datasets due variation in data characteristics. Our study employed the concept of ensemble learning for multivariate time series data using a range of models, including GDN, LSTM-NDT and Prophet to tackle this challenge. Ensemble learning uses multiple models, thereby capturing the advantages offered by a diverse pool of models. By combining the predictions of multiple models, ensemble methods are proven effective and improve overall performance across various datasets by combining the strengths of individual models. It effectively captured the advantage of a pool of models resulting in improved predictions across various datasets.

Furthermore, we can add more models if needed. Our findings underscore the potential of ensemble learning for automating model selection in time series analysis, enhancing prediction accuracy, and accommodating the complexities of real-world datasets. As research in ensemble learning continues to advance, further opportunities arise to optimize model selection processes and improve overall forecasting capabilities.