VT2023: IL2233 Project Technical Report

Time-Series Prediction and Anomaly Detection

Yujie Chen

Oct 15, 2023

Task 1. Time-series prediction with neural networks

1.1 Prediction with synthetic series using MLP, RNN, and LSTM

The code demonstrates the methodology for training and evaluating machine learning models (Multi-Layer Perceptron (MLP), Simple RNN, and LSTM) on different types of time series data. Here's a brief description of the methodology for each case:

1. MLP Model with Fake Data:

Data Generation: The code generates a synthetic time series dataset series and adds some noise to it. The data is reshaped into input-output pairs using the reshape function.

Data Splitting: It splits the data into training and testing sets using the data_splitting function. The testing dataset is 20% of the total data.

Model: An MLPRegressor model with a single hidden layer containing 10 neurons and the identity activation function is defined and trained.

Evaluation: The model is evaluated on both in-sample and out-of-sample data. Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) are calculated.

2. MLP Model with Noisy Sinusoidal Data:

Data Generation: This part generates a sinusoidal wave rsignal, adds random white noise, and reshapes the data into input-output pairs.

Data Splitting: Similar to the first case, it splits the data into training and testing sets.

Model: An MLPRegressor model with a single hidden layer containing 10 neurons and the hyperbolic tangent (tanh) activation function is defined and trained.

Evaluation: The model is evaluated on both in-sample and out-of-sample data. MSE and MAPE are calculated.

3. RNN and LSTM Models with Noisy Sinusoidal Data:

Data Generation: This section generates a sinusoidal wave rsignal and adds random white noise to it.

Data Preparation: The data is reshaped into input-output pairs, with input sequences of length 10 and an output length of 2.

Data Splitting: The data is split into training and testing sets.

Models: An RNN model is defined with a single hidden layer containing 128 neurons and a ReLU activation function. It's compiled with MSE loss and trained for 10 epochs.

An LSTM model is defined with a single hidden layer containing 128 neurons and a ReLU activation function. It's compiled with MSE loss and trained for 10 epochs.

Evaluation: The RNN and LSTM models are evaluated on both in-sample and out-of-sample data. MSE is calculated for their predictions, and the results are compared to the real values.

In all cases, MSE is used as a performance metric for evaluating the models' predictive accuracy. The number of epochs for training the MLP model and the RNN/LSTM models is set to 10. The testing dataset is 20% of the total data, and the models are trained on the remaining 80%. The models are then evaluated on their ability to predict both in-sample and out-of-sample data.

Questions:

(1) How have you designed the neural network for each series?

a. For the equal-difference series without noise, an MLP can be designed for one-step prediction. The MLP architecture typically consists of an input layer, a single hidden layer with 10 units with a linear activation function (identity).

b. For the equal-difference series with added white noise, an MLP with the same architecture as described above can be used for one-step prediction. It's necessary to control the amplitude of the noise series to achieve the desired signal-to-noise ratio.

c. For the deterministic series sampled from a sinusoidal wave, an RNN and an LSTM network can be designed for two-step prediction. The input vector size can be determined based on the desired model complexity and the characteristics of the data. RNNs and LSTMs are suitable for sequential data and can capture temporal dependencies in the data effectively.

d. For the stochastic series sampled from a sinusoidal wave with added random noise, similar to the deterministic series, an RNN and an LSTM can be designed for two-step prediction. The random noise amplitude can be controlled by adjusting the variance of the random variable representing the noise, similar to the approach in the second scenario.

(2) What hyper-parameters do you use in each case?

	Model	Units	Activation	Solver	Max_iter
1	MLP	a single hidden layer with 10 units	identity	adam	\
2	MLP	a single hidden layer with 10 units	tanh	adam	50000
3	RNN	128 units	relu	adam	\
	LSTM	128 units	relu	adam	\
4	RNN	128 units	relu	adam	\
	LSTM	128 units	relu	adam	\

(3) Can the neural network fit well to the specific series well? What are the accuracy

merits?

	Model	In-sample		Out-of-sample	
		MSE	**MAPE**	**MSE**	**MAPE**
1	MLP	0.0189	15.1	0.0295	17.2
2	MLP	0.0196	13.4	0.0145	10.3
3	RNN	1.45e-05	\	2.75e-05	\
	LSTM	2.59e-04	\	1.45e-05	\
4	RNN	0.0123	\	0.00276	\
	LSTM	0.0119	\	0.000817	\

Yes, the NN can fit well for all the series in general.

(4) How is the performance of LSTM in comparison with RNN? Is the LSTM outperforming the RNN in general?

For out-of-sample predictions, LSTM outperforms RNN with minor advantage. However, for in-sample predictions, the advantage of LSTM is weaker and even slightly worse than RNN.

The differences between in-sample and out-of-sample performance are natural for time-dependent data. In cases where the data has a temporal component, such as time series data, the patterns and relationships may change over time. Models trained on past data may struggle to accurately predict future data due to changes in underlying patterns.

1.2 Predict white noise, random walk, an ARMA process using neural networks

The code demonstrates the creation of LSTM models and their training and evaluation on three different types of time series data: white noise, a random walk, and a stationary series generated using an ARMA process. Here's a detailed description of the LSTM architecture employed in the study:

a. White Noise Model (model1):

LSTM Layer: The white noise model (model1) consists of a single LSTM layer with 128 units and a ReLU activation function. The input shape is defined as (input_len, 1), where input_len is set to 10. This means it takes a sequence of 10 time steps as input.

Output Layer: There is a single dense output layer with 1 unit (for predicting the next value in the sequence).

b. Random Walk Model (model2):

LSTM Layer: The random walk model (model2) utilizes a single LSTM layer with 256 units and a ReLU activation function. Similar to the white noise model, the input shape is (input_len, 1).

Output Layer: Just like the white noise model, it has a single dense output layer with 1 unit.

c. ARMA Process Model (model3):

LSTM Layer: The ARMA process model (model3) consists of a single LSTM layer with 128 units and a hyperbolic tangent (tanh) activation function. The input shape is specified as (input_len, 1). Output Layer: There is a single dense output layer with 1 unit, similar to the other models.

After training, the models are used to make in-sample predictions (on the test data), and the mean and standard deviation of the errors are calculated. The errors are computed as the squared differences between the true values and predicted values.

Finally, the models are used for out-of-sample forecasting. For each type of data, a single time step is forecasted using each respective model. The forecasting error for each model is computed as the squared difference between the true value and the forecasted value. The results for in-sample and out-of-sample errors are printed to the console.

Questions:

(1) How have you designed the neural network? What hyper-parameters do you choose? Give short motivation.

LSTM is chosen.

	Units	Activation	Solver	Epochs
1	128	relu	adam	50
2	256	relu	adam	50
3	128	tanh	adam	100

The tanh activation function is commonly used in LSTMs for modeling stationary time series generated by processes like ARMA(2, 2). It helps capture the stable mean and variance of the series by constraining the output range from -1 to 1. In contrast, the ReLU activation function, used in white-noise and random-walk series, allows for capturing non-linear relationships and activations without restricting the output range.

(2) Can the neural network fit well to the white noise series?

In-sample error 1: 1.0873267230050774
Out-of-sample error 1(MSE): 0.8522768516016946

Yes, it fits well.

(3) Can the neural network fit well to the random walk series?

In-sample error 2: 1.3751916470865406
Out-of-sample error 2: 0.8398182605576905

Yes, it fits well.

(4) Can the neural network fit well to the ARMA process? Why or Why not?

In-sample error 3: 1.1082318420910315
Out-of-sample error 3: 0.33892455678853595

Yes, it fits well. It has the lowest out-of-sample error, because it exhibits stable statistical properties over time. The LSTM, with its ability to capture long-term dependencies, can effectively learn and model the complex dynamics and patterns present in the stationary series.

1.3 Comparison with ARIMA-based modeling and prediction

The code demonstrates a methodology for time series prediction using different modeling techniques, including an MLP Regressor, Simple RNN, LSTM, and ARIMA, and evaluates their performance. Here's a brief description of the methodology:

1. **Data Generation:**

A synthetic time series series is generated by combining a Fibonacci series with Gaussian noise. This series is used as the dataset for the models.

2. **Data Reshaping:**

The reshape function is used to create input-output pairs. It takes a fixed number of time steps (n_steps) as input and reshapes the data into sequences of n_steps as features and the next data point as the target value.

3. **Model Training and Evaluation:**

Four different models are trained and evaluated on the time series data.

a. MLP Model (model1):

A Multi-Layer Perceptron (MLP) Regressor model is defined and trained with 10 hidden neurons, a ReLU activation function, and the Adam optimizer. It is trained for 5000 iterations.
The model is evaluated, and Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) are calculated for both in-sample and out-of-sample predictions.

b. Simple RNN Model (model2):

A Simple RNN model is defined with 128 neurons and a ReLU activation function. It has an output layer with a single unit.

The model is compiled with the Adam optimizer and trained for 200 epochs.

The model is evaluated, and MSE, MAE, and MAPE are calculated for in-sample predictions.

c. LSTM Model (model3):

An LSTM model is defined with 128 neurons and a ReLU activation function. It also has an output layer with a single unit.

The model is compiled with the Adam optimizer and trained for 100 epochs.

The model is evaluated, and MSE, MAE, and MAPE are calculated for in-sample predictions.

d. ARIMA Model:

An ARIMA model is used for time series prediction. The appropriate model order (p, d, q) is determined based on Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots.

The dataset is split into training and validation sets, and the ARIMA model is fitted to the training set with the determined order.

Predictions are made on the validation set using the ARIMA model.

4. Comparison and Evaluation:

The code calculates and compares the performance of the four different models in terms of MSE, MAE, and MAPE on both in-sample and out-of-sample data.

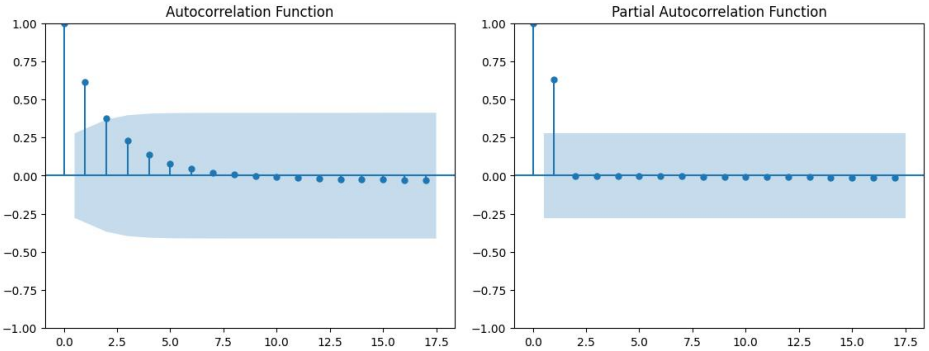
The results are printed to the console for analysis.

The methodology explores different modeling approaches, including classical time series modeling (ARIMA) and neural network-based models (MLP, RNN, LSTM), to predict a synthetic time series dataset and evaluates their performance using a variety of metrics.

Questions:

(1) How have you designed the neural networks? What hyper-parameters do you choose? Give short motivation.

	Units	Activation	Solver
MLP	10 units in a single hidden later	relu	adam
RNN	128	relu	adam
LSTM	128	relu	adam
ARIMA	p = 1, q = 7, d = 1		



(2) How have you trained your neural networks? Report the training epoch, learning rate, optimization algorithm.

epochs = 100, learining rate = 0.01, optimization algorithm = adam

(3) Can your MLP, RNN, LSTM networks fit well to the Fibonacci series? Which one is best?

Not very well. MLP is the best with a MAPE of 6.11e-08.

(4) Can your ARIMA model fit well to the Fibonacci series?

No, it perform poorly with a MAPE of 55.49 due to its linear nature and inability to capture the nonlinear and recursive relationship of the sequence.

(5) Which modeling approach, neural network based or ARIMA based, gives a better performance? Why? Discuss the pros and cons of different modeling approaches.

Neural network.

Neural network-based models generally perform better than ARIMA models for time series forecasting tasks due to their ability to capture nonlinear relationships, learn from data, and adapt to changing patterns. However, they come with the trade-offs of larger data requirements, computational complexity, and reduced interpretability compared to the more simplistic and interpretable nature of ARIMA models.

(6) Which model, ARIMA or NN, is more tolerant to noise? Increasing the noise ratio and report how the accuracy of the two models will be worsened.

ARIMA, it exhibits less variability in accuracy compared to neural networks.

ARIMA models are more tolerant to noise due to their ability to incorporate the differencing operation, which can help remove the noise component. By differencing the data, ARIMA models focus on modeling the underlying trend and seasonality patterns, making them less sensitive to short-term noise fluctuations.

Task 2. Decomposition-based anomaly detection

Anomaly identification in global land temperature changes

The code demonstrates a methodology for detecting anomalies or outliers in a time series dataset. Here's a brief description of the methodology used to produce these results:

1. Data Preparation:

The code reads a time series dataset from a CSV file named 'GlobalTemperatures.csv' using Pandas. It selects a specific range of rows and columns (rows 1560 to 3193 and columns 0 to 2) from the dataset. It converts the 'dt' column to a datetime format and sets it as the index of the DataFrame.

2. Manual Data Modification:

The code manually sets the temperature value for the date '1998-12-1' to 20 degrees. This is done by modifying the DataFrame directly.

3. Outlier Detection:

The code defines two functions for outlier detection:

calculate_zscore: This function calculates the Z-scores for the data by standardizing it. Z-scores indicate how many standard deviations a data point is from the mean.

identify_outliers_boxplot: This function identifies outliers using the interquartile range (IQR) method. It determines lower and upper bounds and flags data points that fall outside these bounds as outliers.

4. Time Series Decomposition:

The code uses the seasonal_decompose function from statsmodels.tsa.seasonal to perform time series decomposition. It decomposes the time series into its components: trend, seasonality, and residuals.

5. Visualization:

The code plots the original time series data along with its decomposed components (trend, seasonality, and residuals) using Matplotlib.

6. Anomaly Detection:

The code calculates Z-scores for the residual component obtained from the decomposition. It uses a predefined threshold (anomaly_threshold) to identify anomalies. Data points with absolute Z-scores greater than the threshold are considered anomalies.

Anomalies are stored in the 'anomalies' variable, and their indices and values are extracted.

7. Anomaly Visualization:

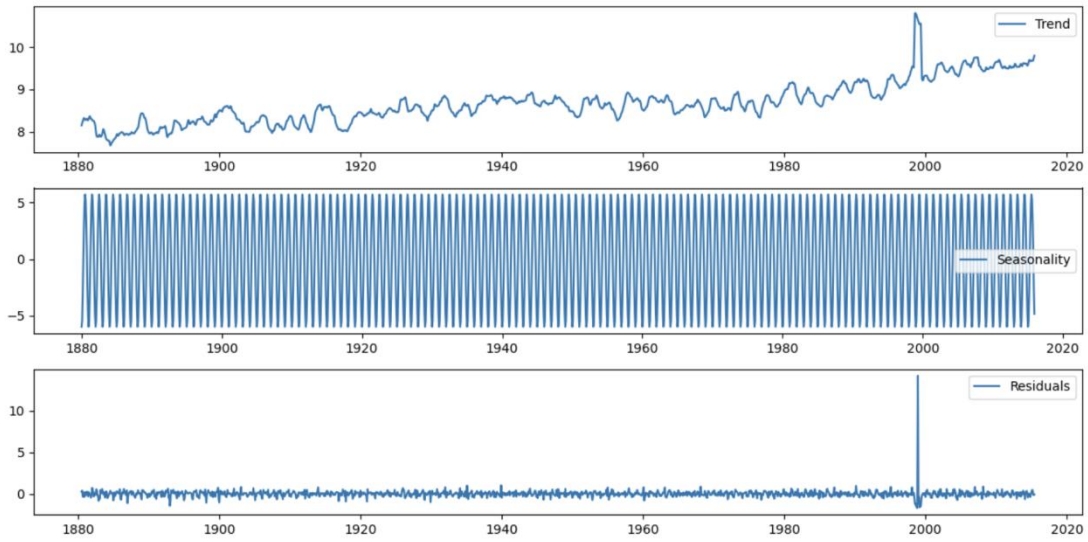
The code plots the original time series data and highlights the detected anomalies in red. It limits the time range to show anomalies that occurred between '1995-01-01' and '2000-01-01'.

The methodology demonstrates how to prepare, modify, decompose, and analyze a time series dataset to detect and visualize anomalies or outliers. It combines statistical methods, visualization, and manual data manipulation to gain insights into unusual data points in the time series.

Questions:

(1) Can the decomposition clearly separate the trend, season (constant period), and remainder components?

Yes, they are clearly separated.



(2) When decomposing the series, is there a general rule to determine which part belongs to a trend, a season, or a remainder? Or is it embedded in and thus dependent on each individual algorithm?

The determination of these components is dependent on the specific algorithm used in '*seasonal_decompose*'.

(3) Is there a growing tendency in the trend series?

Yes, there is.

Task 3. Prediction-based anomaly detection

3.1 Anomaly detection for uni-variate series with ARIMA

The provided code demonstrates a methodology for anomaly detection in a univariate time series using ARIMA (AutoRegressive Integrated Moving Average) modeling. Here's a brief description of the methodology used to produce these results:

1. **Data Import and Visualization:**

The code reads a univariate time series from a CSV file named 'anomaly.csv' using Pandas and sets the first column as the index. It then plots various visualizations of the time series using Matplotlib and Seaborn.

2. **Stationarity Test:**

The Augmented Dickey-Fuller (ADF) test is performed to check the stationarity of the time series. The ADF test statistic and p-value are printed. Additionally, differencing is applied to make the time series stationary, and another ADF test is performed on the differenced series.

3. **Statistical Summary and ACF/PACF Plots:**

A statistical summary of the differenced series is printed to provide descriptive statistics.

ACF and PACF plots are created to analyze the autocorrelation and partial autocorrelation between lagged values in the differenced time series.

The methodology combines data visualization, stationarity testing, differencing, and autocorrelation analysis to prepare the time series for ARIMA modeling. These steps are essential for identifying anomalies or outliers in time series data and serve as the initial data exploration and preprocessing steps.

Questions:

(1) Since this task uses a different approach (prediction-based anomaly detection) from Task 2 which uses decomposition for anomaly detection, describe what the differences of the two methods are?

Prediction-based anomaly detection focuses on forecasting future values and identifying anomalies based on deviations from predicted values. Decomposition-based anomaly detection involves decomposing the time series into components and identifying anomalies based on irregularities in residuals.

(2) Do they achieve the same results? Why or Why not?

No.

They have different focuses and approaches. Prediction-based methods prioritize accurate forecasting and detect anomalies based on deviations from predicted values. Decomposition-based methods isolate components of the time series and identify anomalies based on irregularities in residuals. These methods have varying sensitivities to different types of anomalies and rely on different assumptions about the data. Additionally, the choice of models and parameters can also impact the results obtained from each method.

(3) Given the anomaly ratio of 2%, what is the value of z-score?

5.039e-17

3.2 Anomaly detection in ECG signals with LSTM

(1) How have you designed the neural network? What hyper-parameters do you choose? Why?

64 units, optimizer = adam, epochs = 10.

Because the models perform well with those hyper-parameter.

(3) Can the neural network fit well to the ECG signals? What is the accuracy of your model?

Input	Bi-variate	Uni-variate

Size	MSE	MAPE	MSE	MAPE
4	0.0273	44.4	0.000308	5.34
8	0.0268	42.9	0.000292	5.23
16	0.0271	43.6	0.000330	5.63

Yes, it fits well.

(4) How much is the influence of the input vector size on the prediction accuracy?

When input size = 8, the model has the best performance.

(5) How many epochs do you use for training your LSTMs in order to achieve good accuracy? How much is the learning rate? What is your training optimization algorithm (e.g. SGD, Adam etc.)?

epochs = 10, learning rate = 0.01, alg = adam.

(6) Which way of treating the time series data gives better accuracy: two uni-variate series or one bi-variate series? Why?

Two uni-variate series is better.

The two variables are not strongly related. By modeling them separately, the models can better capture the unique patterns and behaviors exhibited by each variable.

Task 4. Clustering-based Anomaly detection

Anomaly detection in a bivariate series

The provided code demonstrates a methodology for clustering-based anomaly detection in a bivariate time series. Here's a brief description of the methodology used to produce these results:

1. Data Generation and Visualization:

Two bivariate series, X1 and X2, are generated with different means and variances. The data is visualized with both line and scatter plots.

2. Clustering-Based Anomaly Detection:

The data from X1 and X2 are concatenated into a single dataset.

K-Means is applied to the data using the KMeans function from scikit-learn. The number of clusters (num_clusters) is set to 4, and each data point is assigned a cluster label. Cluster labels are printed.

The distances from each data point to the cluster centroids are calculated, and the minimum distance for each point is obtained.

An outlier ratio is set to 5%, and the number of outliers is calculated based on the ratio.

Data points with the largest distances (outliers) are identified by sorting the minimum distances, and the data is split into normal and anomalous subsets.

3. Visualization of Anomalies:

Scatter plots and line plots are generated to visualize the detected anomalies.

4. SOM for Anomaly Detection:

Data is normalized by subtracting the mean and dividing by the standard deviation.

A 2x1 SOM is trained with the data. The quantization_errors for each data point are calculated.

An outlier ratio is set to 5%. Outliers are identified by sorting the quantization_errors, and the data is split into normal and anomalous subsets.

Overall, this methodology combines K-Means and SOM for detecting anomalies in bivariate time series data. It includes data generation, visualization, clustering, and visualization of detected anomalies. The choice of clustering and SOM parameters may vary depending on the specific dataset and application.

Questions:

(1) How do you set the number of clusters? Why?

I tried with 2, 3, 4, and the result of anomalies detection barely have any differences.

(2) Which distance metric do you use? Are there other distance metrics which might be useful for this task?

Euclidean distance.

Dynamic Time Warping could also work. The DTW distance takes into account the temporal alignment between two time series by warping one series along the time axis to align it with the other series, while Euclidean distance only represent the direct linear distance between two data points. It measures the minimum cumulative distance between corresponding points in the two time series, allowing for local temporal shifts and distortions.

(3) Do the two different clustering methods (K-means and SOM) achieve the same results? Discuss why or why not.

No, they are not the same.

K-means and SOM use different algorithms for clustering. K-means is a partition-based clustering algorithm that assigns each data point to the nearest cluster centroid, while SOM is a type of unsupervised artificial neural network that organizes the data into a low-dimensional grid. The inherent differences in their algorithms can lead to variations in outlier detection results.

Task 5. Summary

5.1 Two classes of approaches to perform time-series modeling and prediction

1. Describe the two approaches in your own words.

The statistical learning approach (ARIMA) is a traditional method that focuses on capturing linear dependencies and short-term patterns in stationary time series data, while the deep learning approach (Neural Networks) is more flexible and excels in capturing complex nonlinear relationships and long-term dependencies.

2. Compare the two approaches and discuss their strength and weakness.

Listed in Q3.

3. List key points of the two approaches and key pros and cons in a table shown below (Table 2).

ARIMA

a. Assumption:

ARIMA assumes that the time series is stationary, meaning that the mean, variance, and autocorrelation structure do not change over time.

b. Modeling & Prediction:

ARIMA modeling involves identifying the optimal values for the three parameters: p (autoregressive order), d (order of differencing), and q (moving average order).
The model is trained on historical data and uses the AR, MA, and differencing components to forecast future values.

c. Strengths:

ARIMA is effective for capturing linear dependencies and short-term patterns in time series data. It is interpretable and provides insights into the impact of lagged values and differencing on the forecasted values. It performs well when the data exhibits clear trends and seasonality.

d. Weaknesses:

ARIMA may not handle complex nonlinear relationships well. It assumes stationarity, which may not hold for some real-world time series data. ARIMA may struggle with long-term forecasts and capturing sudden changes or outliers in the data.

Neural Network (NN) Model:

a. Assumption:

Neural networks are more flexible and do not require strict assumptions about the data distribution or stationarity.

b. Modeling & Prediction:

NN modeling involves designing the network architecture, including the number of layers,

neurons, and activation functions. The model is trained using historical data, adjusting the weights and biases to minimize the prediction error. NNs can capture long-term dependencies and learn from sequences of data using memory cells like LSTM.

c. Strengths:

Neural networks can capture complex nonlinear patterns and relationships in time series data. They are capable of handling long-term dependencies and making accurate predictions even in the presence of noisy data. NNs can learn from large-scale data and adapt to changing patterns over time.

d. Weaknesses:

Neural networks are often considered "black box" models, making it challenging to interpret the internal workings and understand the contributing factors to predictions. They require a significant amount of training data and computational resources. Overfitting is a potential issue, where the model may perform well on the training data but struggle to generalize to new data.

5.2 Three different anomaly detection methods

(1) Describe the three different methods.

(2) Compare and discuss the strength and weakness of the three methods.

(3) List key points of the approaches and key pros and cons in a table below (Table 3).

1. Density-based Anomaly Detection:

a. How Anomaly is Detected:

Density-based anomaly detection methods identify anomalies based on the density of data points in the time series. Anomalies are typically defined as points that fall in low-density regions or have significantly different density compared to the majority of data points. Density estimation techniques such as Gaussian mixture models or kernel density estimation are commonly used to estimate the density of the data.

b. Anomalies Assumed:

Density-based methods assume that anomalies are characterized by their deviation from the typical density distribution of the data. Anomalies are expected to exhibit lower density values or occupy sparsely populated regions compared to normal data points.

c. Strengths:

Can detect local anomalies and outliers that do not conform to the expected density distribution.

Does not require assumptions about data distribution or stationarity.

Effective in detecting anomalies in multi-dimensional and complex data.

d. Weaknesses:

Sensitive to parameter selection, such as the bandwidth or kernel function used in density

estimation.

Performance can be affected by the curse of dimensionality in high-dimensional data.

May struggle with identifying anomalies in regions of similar densities or handling varying densities across different data subsets.

2. Decomposition-based Anomaly Detection:

a. How Anomaly is Detected:

Decomposition-based anomaly detection involves decomposing the time series into components such as trend, seasonality, and residuals. Anomalies are detected by identifying irregularities or unexpected patterns in the residuals, which represent the part of the data not accounted for by the other components. Various techniques, such as STL (Seasonal and Trend decomposition using Loess) or moving averages, can be used for decomposition.

b. Anomalies Assumed:

Decomposition-based methods assume that anomalies disrupt the expected patterns within the time series, particularly in the residuals. Anomalies can manifest as sudden changes, level shifts, local deviations, or abnormal fluctuations in the residuals.

c. Strengths:

Effective in capturing anomalies related to irregular patterns, shifts in seasonality, and unexpected variations in the data.

Provides insights into different components of the time series, facilitating anomaly interpretation.

Can handle different types of anomalies, including point anomalies and contextual anomalies.

d. Weaknesses:

May struggle with capturing anomalies that are not reflected in the residuals or when the decomposition assumptions do not hold.

Sensitivity to parameter selection, such as the window size or the choice of decomposition technique.

Performance can be affected by the presence of noise or outliers in the data.

3. Prediction-based Anomaly Detection:

a. How Anomaly is Detected:

Prediction-based anomaly detection involves forecasting future values based on historical data and identifying anomalies based on deviations from the predicted values. Anomalies are detected when the observed values significantly differ from what was predicted by the model. Methods such as ARIMA (AutoRegressive Integrated Moving Average) or LSTM (Long Short-Term Memory) can be used for prediction.

b. Anomalies Assumed:

Prediction-based methods assume that anomalies correspond to significant deviations or

outliers from the predicted values. Anomalies are expected to exhibit abnormal behavior that deviates from the underlying patterns captured by the forecasting model.

c. Strengths:

Effective in detecting anomalies that deviate significantly from the predicted values.

Can capture sudden changes, outliers, and irregular fluctuations in the time series.

Suitable for identifying anomalies with a noticeable impact on future predictions.

d. Weaknesses:

May struggle with complex nonlinear relationships or anomalies that do not follow the underlying patterns captured by the model.

Performance can be affected by model selection, parameter tuning, and assumptions about data stationarity.

Long-term forecasting and capturing anomalies that extend beyond the available historical data can be challenging.

4. Clustering-based Anomaly Detection:

a. How Anomaly is Detected:

Clustering-based anomaly detection involves grouping similar data points into clusters and identifying anomalies as data points that do not belong to any cluster or belong to small or dissimilar clusters. Anomalies are detected based on their distance or dissimilarity to the cluster centroids or based on metrics like cluster density or cohesion. Clustering algorithms like k-means, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), or Gaussian Mixture Models can be used.

b. Anomalies Assumed:

Clustering-based methods assume that anomalies exhibit dissimilarity or deviation from the majority of the data points or cluster patterns. Anomalies are expected to be isolated, distant, or distinct from the normal clusters.

c. Strengths:

Effective in identifying anomalies that deviate from the typical cluster patterns or do not belong to any cluster.

Can handle multi-dimensional data and detect global anomalies that affect multiple attributes or dimensions.

Suitable for detecting novel or unknown anomalies not explicitly defined by predefined rules or patterns.

d. Weaknesses:

Sensitive to the choice of clustering algorithm, distance metric, or parameter selection.

Performance can be affected by high-dimensional data or when anomalies are located within or close to normal clusters.

May struggle with detecting anomalies in sparsely populated or noisy regions, or when clusters overlap or have varying densities.