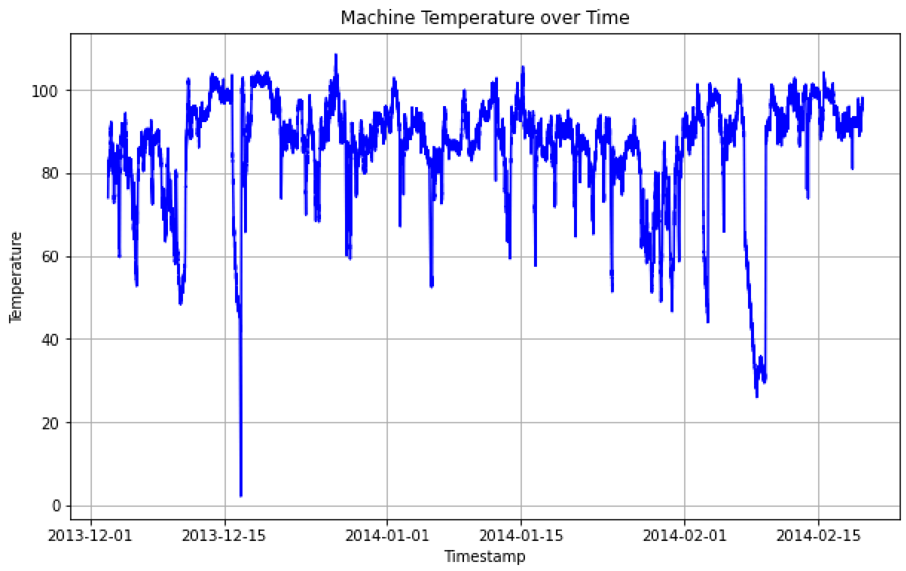
Autoencoders for Anomaly Detection

“machine_temperature_system_failure.csv" dataset in realKnownCause offers a comprehensive set of anomalies from different domains, making it a suitable choice for anomaly detection. The data is highly relevant in practical applications. Anomalies such as planned shutdowns, difficult-to-detect issues, and catastrophic failures are common in industrial settings, making the dataset suitable for real-world anomaly detection evaluation.

1. **Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise?**
   a. The timestamp column indicates the date and time of each reading, while the value column contains the temperature sensor readings.
   b. The dataset consists of 22,695 entries and 2 variables.
   c. The timestamp variable is in datetime format, allowing for easy manipulation and analysis of temporal data.
   d. The value variable is in float format, representing the temperature readings.
   e. Upon initial inspection using the describe() method, we find that the temperature readings range from a minimum of 2.08°C to a maximum of 108.51°C. The mean temperature reading is approximately 85.93°C, with a standard deviation of 13.75°C.
   f. There are no missing values in the dataset, as confirmed by the isnull().sum() method. This indicates that the dataset is complete and ready for analysis.



2. **Describe the details of your autoencoder models, including the layers, activation functions, and any specific configurations employed.**

    a. Basic Autoencoder Model:
        i. Layers:
            1. Input layer: Dense layer with 64 units and ReLU activation function
            2. Encoder layer: Dense layer with latent_dim (3 in this case) units and ReLU activation function
            3. Decoder layer: Dense layer with 64 units and ReLU activation function
            4. Output layer: Dense layer with window_size units and linear activation function
            5. Activation functions: ReLU for all hidden layers, linear for the output layer

        ii. Loss function: Mean Squared Error (MSE)
        iii. Optimizer: Adam

    b. Deep Autoencoder Model:
        i. Layers:
            1. Input layer: Dense layer with 128 units and ReLU activation function
            2. Hidden layers: Dense layers with 64 units and ReLU activation function
            3. Encoder layer: Dense layer with latent_dim (3 in this case) units and ReLU activation function
            4. Decoder layer: Dense layer with 64 units and ReLU activation function
            5. Output layer: Dense layer with window_size units and linear activation function

        ii. Activation functions: ReLU for all hidden layers, linear for the output layerLoss function:
        iii. Mean Squared Error (MSE)Optimizer: Adam.

    c. Denoising Autoencoder Model:
        i. Layers:
            1. Input layer: GaussianNoise layer with noise level of 0.1
            2. Hidden layers: Dense layers with 64 units and ReLU activation function

3. Encoder layer: Dense layer with latent_dim (3 in this case) units and ReLU activation function
4. Decoder layer: Dense layer with 64 units and ReLU activation function
5. Output layer: Dense layer with window_size units and linear activation function

ii. Activation functions: ReLU for all hidden layers, linear for the output layerLoss function:

iii. Mean Squared Error (MSE)Optimizer: Adam

d. Dense Autoencoder Model

i. Layers:

1. Input layer: Dense layer with 128 units and ReLU activation function
2. Hidden layer 1: Dense layer with 64 units and ReLU activation function
3. Latent layer: Dense layer with latent_dim (3 in this case) units and ReLU activation function
4. Hidden layer 2: Dense layer with 64 units and ReLU activation function
5. Output layer: Dense layer with window_size units and linear activation function

ii. Activation functions: ReLU for all hidden layers, Linear activation function for the output layer

iii. Mean Squared Error (MSE)Optimizer: Adam

3. **Discuss the results and provide relevant graphs:**
    a. **Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.**

        i. Base Autoencoder:
            1. Training Loss: 16.1166
            2. Validation Loss: 17.4068
            3. Testing Loss: 15.1434

        ii. Deep Autoencoder:
            1. Training Loss: 8.2760
            2. Validation Loss: 7.8601
            3. Testing Loss: 7.3279

        iii. Denoising Autoencoder:
            1. Training Loss: 36.0010
            2. Validation Loss: 37.4299
            3. Testing Loss: 35.6720

        iv. Dense Autoencoder:
            1. Training Loss: 36.0010
            2. Validation Loss: 37.4299
            3. Testing Loss: 7633.3153

b. **Plot the training and validation accuracy over time (epochs).**

(Answer to this question is not valid as accuracy is not the measure)

Accuracy:

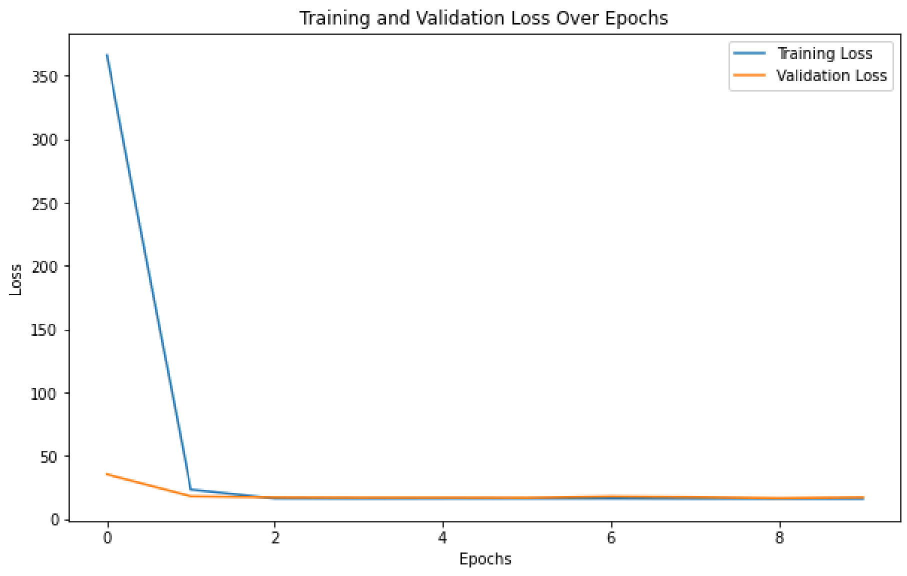
Autoencoders are typically not evaluated using accuracy metrics because they are unsupervised models that learn to reconstruct their input data rather than predict specific labels or categories. Therefore, their performance is typically evaluated based on how well they reconstruct the input data, as indicated by the loss values.

But we have included the code that calculates the accuracy of the dense autoencoder model by comparing the true labels with the predicted labels and taking the mean of the comparison results which comes out to be 1.

c. **Plot the training and validation loss over time (epochs).**

i. Base Autoencoder:

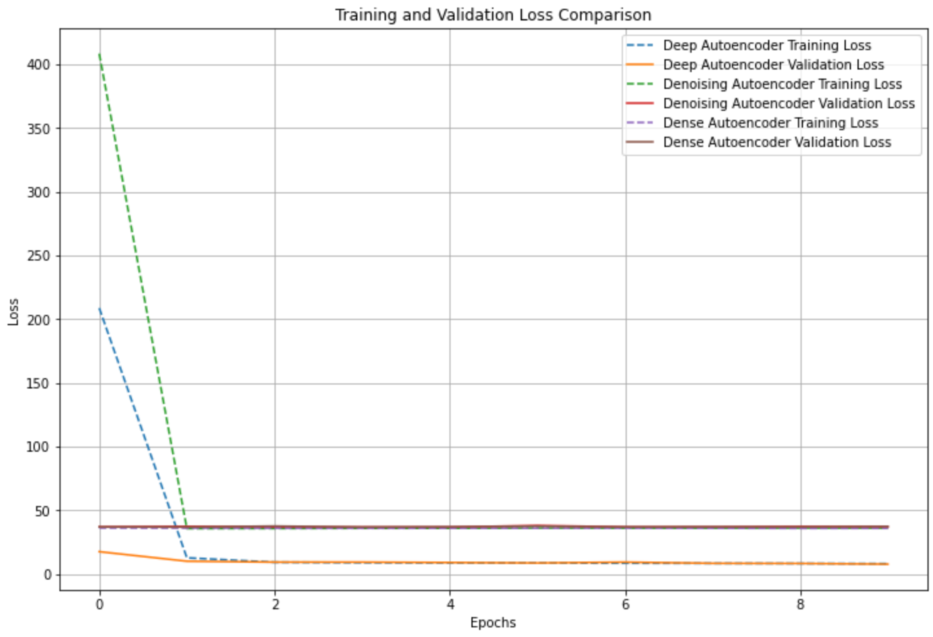
1. The training loss decreases significantly from the first epoch (366.3406) to the last epoch (16.1166), indicating that the model is learning to reconstruct the input data more accurately over time.
2. The validation loss is consistently lower than the training loss, which is a positive indicator that the model is not overly biased towards the training data and is learning to generalize.



The overall trend indicates that the base encoder model is effectively learning the underlying patterns in the input data and improving its ability to reconstruct the data with higher fidelity as training progresses.

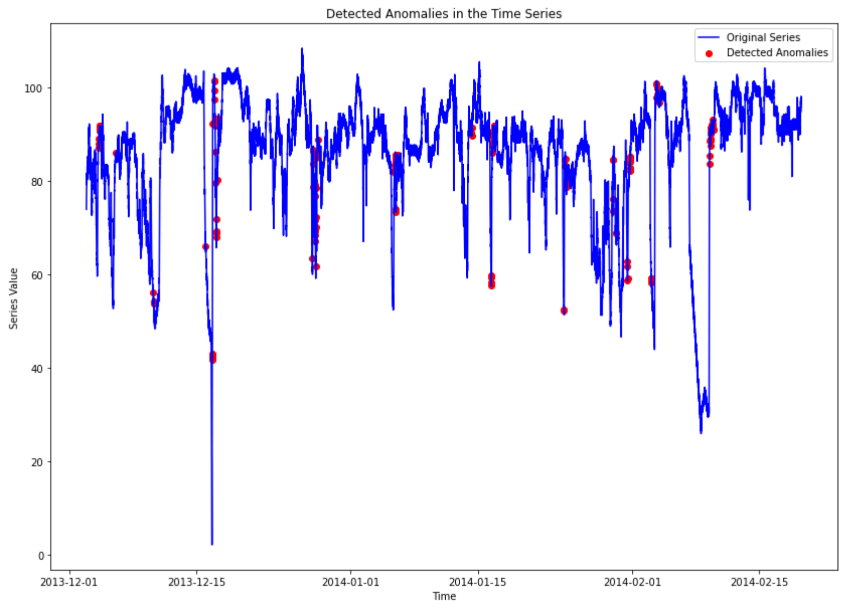
ii. Deep Autoencoder, Denoising Autoencoder and Dense Autoencoder

1. Deep Autoencoder:
    a. The deep autoencoder's training loss decreases from 208.8713 to 8.2760 over 10 epochs, showing significant improvement in data reconstruction.
    b. Validation loss decreases as well, indicating the model's ability to generalize to unseen data.
    c. Fluctuations in loss values suggest potential for further hyperparameter tuning for better convergence.
2. Dense Autoencoder:
    a. The dense autoencoder's training and validation loss follow a similar pattern as the denoising autoencoder, starting at 408.5215 and 37.0830, respectively, and decreasing to 36.1352 and 37.1998 over 10 epochs.
    b. However, the dense autoencoder's loss values are significantly higher, indicating potential issues with the model architecture or training process.
3. Denoising Autoencoder:
    a. The denoising autoencoder's training loss starts at 408.5215 and decreases to 36.1352 over 10 epochs, while the validation loss starts at 37.0830 and decreases to 37.1998.
    b. The model shows some fluctuations in loss values, indicating potential for further training or hyperparameter tuning.


Training and Validation Loss Comparison
Deep Autoencoder Training Loss
Deep Autoencoder Validation Loss
Denoising Autoencoder Training Loss
Denoising Autoencoder Validation Loss
Dense Autoencoder Training Loss
Dense Autoencoder Validation Loss
Loss
Epochs
0
50
100
150
200
250
300
350
400
0
2
4
6
8

d. **Generate a confusion matrix using the models predictions on the test set.**

We cannot generate a traditional confusion matrix for the autoencoder models' predictions on the test set because we do not have the ground truth labels for anomalies. However, we have visualized the anomalies detected by plotting them against the original time series data for Base Encoder:



This visualization can give us an idea of how well the models are detecting anomalies.

e. **Report any other evaluation metrics used to analyze the models performance on the test set.**
   i. Base Encoder:
      1. Mean Squared Error (MSE): 15.1434
      2. Root Mean Squared Error (RMSE): 3.8915
      3. R-squared (R2): 0.9189
   ii. Deep Autoencoder:
      1. Mean Squared Error (MSE): 7.3279
      2. Root Mean Squared Error (RMSE): 2.7070
      3. R-squared (R2): 0.9608
   iii. Denoising Autoencoder:
      1. Mean Squared Error (MSE): 35.6720
      2. Root Mean Squared Error (RMSE): 5.9726
      3. R-squared (R2): 0.8090
   iv. Dense Autoencoder:
      1. Mean Squared Error (MSE): 7633.3153
      2. Root Mean Squared Error (RMSE): 87.3688
      3. R-squared (R2): -39.8711

Based on the performance metrics of the four autoencoder models:

1. Base Encoder: This model achieved a relatively low MSE and RMSE, indicating good performance in reconstructing the input data. The R-squared value of 0.9189 indicates that the model explains 91.89% of the variance in the data, which is quite high.
2. Deep Autoencoder: The deep autoencoder performed even better than the base encoder, with a lower MSE and RMSE, and a higher R-squared value of 0.9608. This suggests that the deeper architecture of the model allowed it to capture more complex patterns in the data.
3. Denoising Autoencoder: The denoising autoencoder had a higher MSE and RMSE compared to the base and deep autoencoders, indicating a higher reconstruction error. However, the R-squared value of 0.8090 suggests that the model still explains a significant portion of the variance in the data.
4. Dense Autoencoder: The dense autoencoder performed the worst among the four models, with a very high MSE and RMSE, and a negative R-squared value. This indicates that the model's predictions are worse than simply using the mean value of the data as a prediction.

In conclusion, the deep autoencoder is the most effective among the models considered here for anomaly detection in this dataset. It achieves the lowest reconstruction error and highest explanatory power, suggesting that it can effectively capture the underlying patterns in the data.

4. **Discuss the strengths and limitations of using autoencoders for anomaly detection.**
    a. Strengths:
        i. Unsupervised Learning: Autoencoders can learn representations of data without the need for labeled examples, making them suitable for anomaly detection in situations where labeled data is scarce or unavailable.
        ii. Non-linearity: Autoencoders can model complex, non-linear relationships in data, allowing them to capture intricate patterns that may indicate anomalies.
        iii. Dimensionality Reduction: By learning a compressed representation of the input data, autoencoders can effectively reduce the dimensionality of the data, making it easier to detect anomalies in high-dimensional spaces.
        iv. Adaptability: Autoencoders can be adapted to different types of data and anomaly detection tasks by adjusting the architecture, activation functions, and other parameters.
        v. Feature Learning: Autoencoders can automatically learn relevant features from the data, which can be beneficial for detecting anomalies that may not be apparent in the original features.
    b. Limitations:
        i. Limited Reconstruction Ability: Autoencoders are trained to reconstruct normal data, so they may struggle to accurately reconstruct anomalous data, especially if the anomalies are significantly different from the normal patterns.
        ii. Sensitivity to Hyperparameters: The performance of autoencoders for anomaly detection can be sensitive to the choice of hyperparameters, such as the number of layers, units per layer, and learning rate, requiring careful tuning.
        iii. Difficulty in Interpreting Latent Representations: While the latent representations learned by autoencoders can be effective for anomaly detection, interpreting these representations and understanding why a particular instance is flagged as anomalous can be challenging.
        iv. Limited Generalization: Autoencoders may struggle to generalize well to unseen anomalies or data distributions that differ significantly from the training data, leading to false positives or missed anomalies.
        v. Computational Complexity: Training and evaluating autoencoder models can be computationally intensive, especially for large datasets

or complex architectures, which can limit their scalability in certain applications.