Name: Adrian Zevenster Position: ML Engineer

GitHub Repo: https://github.com/adrianzevenster/TimeSeries_AnomalyAnalysis

# 1 Dervico - Time Series Analysis

The aim is to build a classification model which can predict values that are anomalistic from Time Series data - provided in the 'training.csv', and 'testing.csv' files. Datasets contain flags indicating values where known anomalies have been observed.

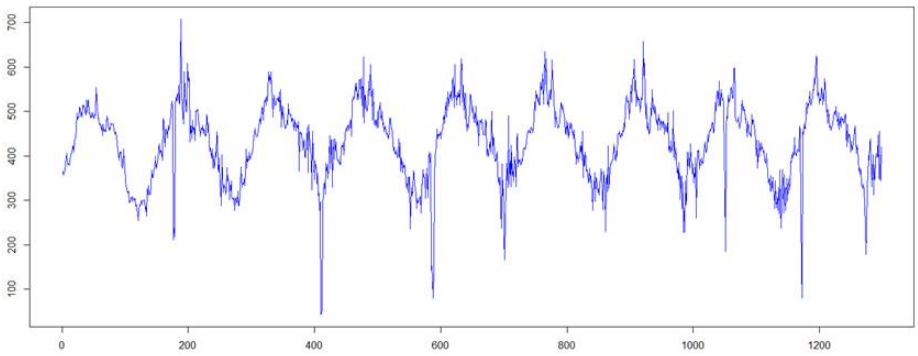



Figure 1: Dervico Sample Image

This task is divided into the following:

Step 0 : Database Creation

Step 1 : Data Preprocessing and Cleaning

Step 2 : Model Creation

Step 3 : Performance Evaluation

Step 5 : Conclusion

Step 6 : Exploration and Suggestions

# 2 Methods and Discussion

This section provides an overview of the methodologies implemented in the *'Dervico.ipynb'* Jupyter notebook. It includes a discussion of the key files within the repository, emphasizing their relevance to our objectives. Additionally, the section delves into the rationale behind the chosen strategies, offering insights into the decision-making process that shaped our approach."

## 2.1 Step 0: Database Creation

Database with 2 separate tables is created named *'training_raw'*, and *'testing_raw'*. This is to ensure data quality and for validation as data wrangling may lead to biased data sources. Database creation scripts are included in GitHub main repository under *'CreateDatabase-Imports.sql'*

While determining the counts of anomaly classes the following was observed:

| EventDate | TrainingCount | TestingCount |
|---|---|---|
| 2020-08-19 | 2 | 0 |
| 2020-09-19 | 2 | 0 |
| 2020-09-22 | 59 | 0 |
| 2020-09-28 | 65 | 0 |
| 2020-10-04 | 4 | 0 |
| 2020-10-14 | 2 | 0 |
| 2020-10-16 | 34 | 0 |
| 2020-11-13 | 2 | 0 |
| 2020-12-16 | 100 | 0 |
| 2020-12-17 | 50 | 0 |
| 2020-12-19 | 3 | 0 |
| 2021-01-01 | 3 | 0 |
| 2021-01-11 | 4 | 0 |
| 2021-01-23 | 2 | 0 |
| 2021-02-12 | 0 | 5 |
| 2021-03-08 | 0 | 3 |
| 2021-03-18 | 0 | 3 |
| 2021-03-27 | 0 | 3 |
| 2021-03-29 | 0 | 3 |
| 2021-03-30 | 0 | 5 |
| 2021-04-02 | 0 | 3 |
| 2021-04-03 | 0 | 3 |

Table 1: Anomaly Classes Count Per Timestamp for Training and Testing Datasets

## 2.2 Step 1: Preprocessing and Cleaning

We have started the Jupyter script *Dervico.ipynb* with some basic data formatting conversions to ensure datatypes.

We then move on to creating a visual presentation of both training and testing data highlighting where anomaly classes have been observed, presented in Fig 2. From this, we observe that known anomalies were all on the lower extrema of the times series data presented for both training and testing. These data points are somewhat sporadic which could affect model performance without more features to indicate the nature of anomalies.

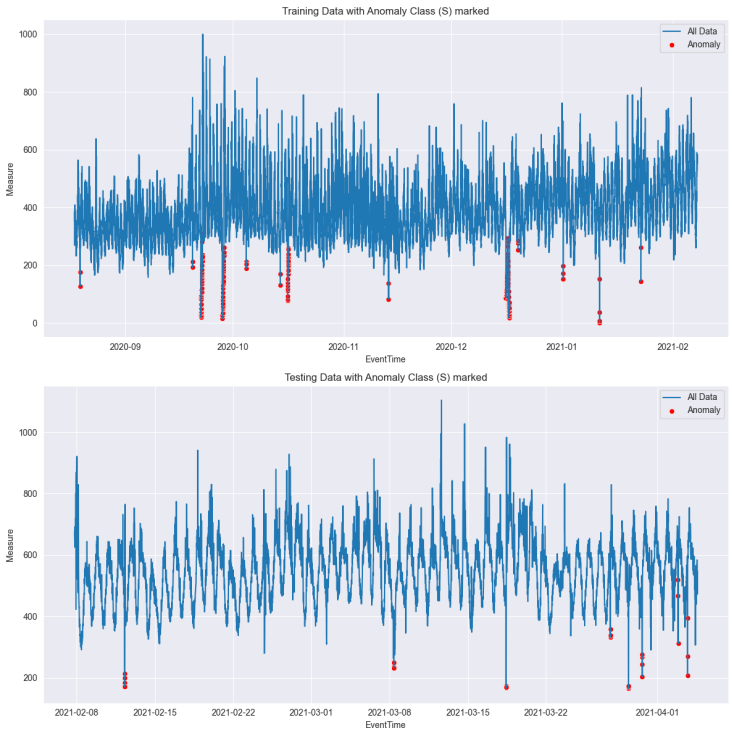



Figure 2: Illustration of observed anomalies on training and testing data

## 2.3 Step 2: Model Creation

For this classification task, we will utilize Support Vector Machines (SVM) and Random Forests (RF). Presente in *Dervico.ipynb* notebook

SVM is utilized due to the accuracy of classification tasks, however, have are difficult to interpret. With RF more interpretable, however, performance is hindered with low-dimensionality datasets.

On the first interaction, both models are trained on both training and testing sets to evaluate performance. Classification tasks are typically evaluated by: Precision, Recall, F1-scores - which are a balance between Precision and Recall.

The following results were obtained on predictions made:

Scores in Table 2 2 are calculated by True Positive (TP) - Normal, True Negative (TN) - Anomaly, False Positive, False Negative (FN). The metrics for both SVM and RF are presented in Fig 3 3

## 2.4 Step 3: Performance Evaluation

As depicted in figure 3 3 and table 2 2. Both models were considerably accurate in predicted *Normal* classes, with the Random Forest slight edge. The same Normal class performance was observed on the testing data with both SVM

Table 2: Classification Reports for SVM and Random Forest Models

| Model | Metric | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|---|
| SVM Training | Normal | 1.00 | 1.00 | 1.00 | 24668 |
| | Anomaly | 0.97 | 0.71 | 0.82 | 332 |
| | Macro Avg | 0.98 | 0.85 | 0.91 | 25000 |
| | Weighted Avg | 1.00 | 1.00 | 1.00 | 25000 |
| SVM Testing | Normal | 1.00 | 1.00 | 1.00 | 7983 |
| | Anomaly | 1.00 | 0.07 | 0.13 | 28 |
| | Macro Avg | 1.00 | 0.54 | 0.57 | 8011 |
| | Weighted Avg | 1.00 | 1.00 | 1.00 | 8011 |
| Random Forest Training | Normal | 1.00 | 1.00 | 1.00 | 24668 |
| | Anomaly | 0.94 | 0.75 | 0.84 | 332 |
| | Macro Avg | 0.97 | 0.88 | 0.92 | 25000 |
| | Weighted Avg | 1.00 | 1.00 | 1.00 | 25000 |
| Random Forest Testing | Normal | 1.00 | 1.00 | 1.00 | 7983 |
| | Anomaly | 1.00 | 0.21 | 0.35 | 28 |
| | Macro Avg | 1.00 | 0.61 | 0.68 | 8011 |
| | Weighted Avg | 1.00 | 1.00 | 1.00 | 8011 |

and RF classifiers. However, on testing data, the RF model outperformed the SVM model considerably in predicting anomaly classes.

The ROC AUC curve is used as the classification performance evaluation, presented in figure 4.

ROC Curves illustrate the poor performance in testing data when classifying anomalies using the SVM model. However, the perfect ROC curve might indicate overfitting training data when using RF models.

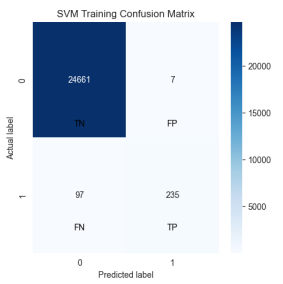
# 3 Conclusion and Suggestions

Depending on the risk of increasing classification anomalies the RF model presents a viable option for predicting anomaly classes. However, both models had considerable True Negative and False Positive Classifications. This means that values that normal values were predicted to be anomalies and anomaly values were predicted to be normal.

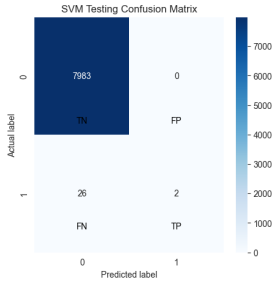
From figure **??**. we can illustrate the actual anomalies vs predicted anomalies for both SVM and RF models. We observe that the SVM model missed anomalies with lower Measured values that the RF model was able to identify.

Concerns remain about the sporadic nature of anomaly labels. Typically on time series data values deviating from the upper bound should indicate an anomaly in itself depending on the nature of the use case. Values that were flagged as anomalies were not consistent, thus without more features viable performance may remain undesirable.

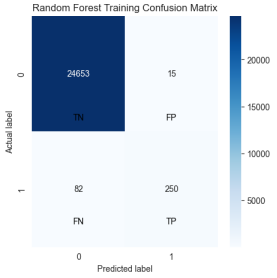
To address concerns and remediate a solution. We have suggested using



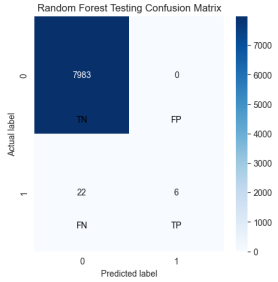

(a) SVM Training Classifications




(b) SVM Training Classifications




(c) RF Training Classification




(d) RF Testing Classifications

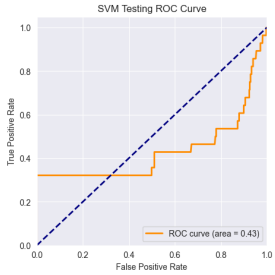
Figure 3: Confusion Matrix Illustrating Prediction from SVM and RF model

Z-scores to identify values deviating dramatically from the mean on the upper and lower bounds. This is presented in *Dervico_Z_Score.ipynb*.

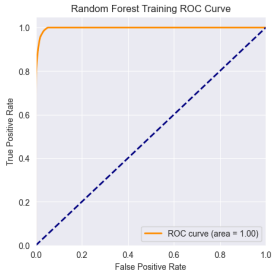
Here we identify and simulate bounds anomalies that would be predicted with current values flagged as anomalies. To address the sporadic values a simulation of optimal thresholds based on Z-scores is identified and. See the below illustration of the threshold varying on which RF model was also trained, where a perfect prediction score was achieved.

Please see illustrated in figure 6 boundaries and threshold values, which are available in *Dervico_Z_Score.ipynb*

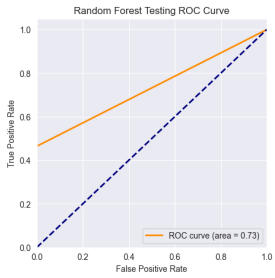

(a) SVM Training Performance




(b) SVM Testing Performance

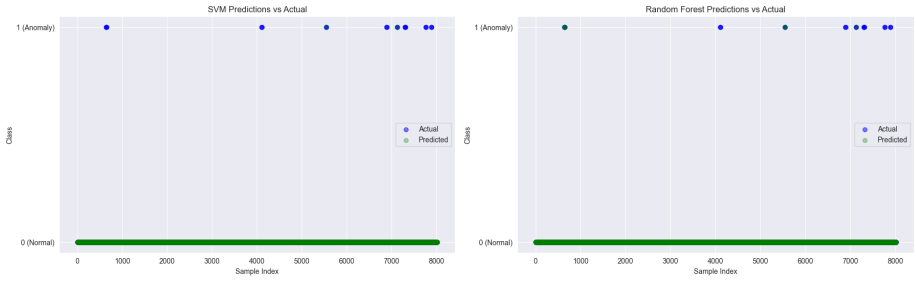



(c) RF Training Classification




(d) RF Testing Classifications

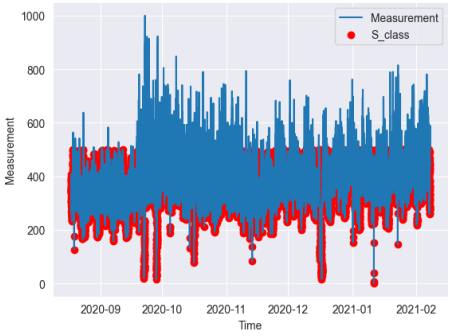
Figure 4: ROC Curves Depicting Classification Performance for RF and SVM Models



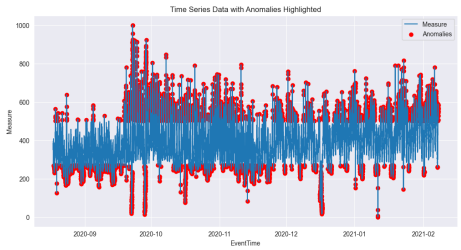

(a) SVM Predictions vs Actual

(b) RF Predictions vs Actual

Figure 5: Actual vs Predicted Anomalies by RF and SVM models




(a) Z-Score without adjustments to anomalies




(b) Z-Score bounds with optimal thresholds

Figure 6: Z-Score based Anomaly Preditions