Unsupervised Anomaly Detection with Autoencoders

CSE6748 – Applied Analytics Practicum Fall 2019
Sponsored by United Technologies Corporation

Submitted by Peter Chen (cchen376) - Data Scientist
Reviewed by Wim Verleyen – Associate Director of Data Science

Project Purpose

Current Challenges

The purpose of this project is to conduct R&D to improve existing anomaly detection approaches for machine and manufacturing data. The goal is to apply these techniques to UTC machines such as Aircraft Engines. Currently, anomaly detection for these machines consist primarily of domain-expert defined thresholds and simple statistical analysis of aggregate data. This is a time intensive process and does not scale well to complex system with many sensors and non-stationary data. Therefore, it is our goal to develop a more robust technique that utilizes modern advances in Machine Learning which are well suited for large complex datasets.

Examples of anomalies in machine data

- Vibration failure
- Mechanical stall
- Component liberation
- Fire and electrical failures

Downsides of traditional techniques

- Lots of time needed to define thresholds
- Unable to detect contextual anomalies
- Prone to missing anomalies when there are a lot of sensors
- Difficulty visualizing multivariable sensors
- Prone to missing anomalies when the data is non-stationary - changing contexts
- Lack of labeled anomalies (extreme class imbalance) makes it difficult to set accurate thresholds

New approach

A new approach to anomaly detection will be tested that incorporates **unsupervised algorithms** that are trained solely on non-failure/nominal data. The goal is to build a robust representation of nominal behavior that can be used to detect whether new data points deviate from nominal conditions. These unsupervised models aim to overcome the limitations of the traditional approaches to be able to overcome extreme class imbalance, non-stationary data, and contextual outliers. The methodology developed has a two-part process.

1. Train and Autoencoder to accurate reconstruct nominal data
2. Utilize traditional techniques on autoencoder reconstruction error to detection anomalies
    - Unsupervised Model, Probability based methods, Supervised model

Autoencoder Overview

Introduction

Autoencoders utilize neural networks for the task of **unsupervised representation learning.** Specifically, Autoencoders aim to model the intrinsic structure within the dataset to learn a compressed representation of the original input signals. To achieve this, Autoencoders are trained to predict the input signals with a bottleneck constraint enforced such that model is forced to encode the data into a lower dimensional representation (known as the code) before decoding it back into the original signal while minimizing reconstruction error. For anomaly detection, the underlying assumption is that the structure of the non-failure/nominal system is different than the structure of a failed system. Therefore, the reconstruction error will be amplified when the model is given data that is anomalous.

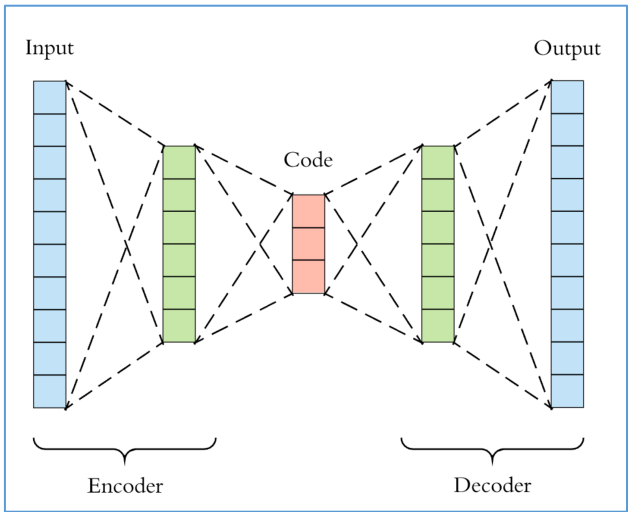


Figure 1: autoencoder diagram

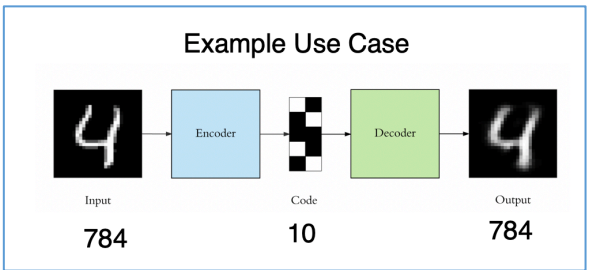


Figure 2: Example use-case for autoencoder

Anomaly Detection with Autoencoders

Autoencoders can be used to detect anomalies based on the magnitude of the reconstruction error.

1. Train model to predict inputs on non-failure data
2. Set statistical threshold for nominal behavior based on training data's reconstruction error
3. Test model on data with both non-failures and failures
4. Calculate the reconstruction error for test data and classify as anomalous based on error

As mentioned above, there are many ways to analyze the reconstruction error. The experiments below will attempt to develop a robust technique that is good at classifying anomalies.

Dataset and Metric

Source: https://arxiv.org/pdf/1809.10717.pdf

The dataset chosen is a real-world dataset from a typical paper manufacturing machine (see example image below). The dataset comes from 61 sensor readings on the machine that are sampled at 2-minute intervals. There is a binary variable (y) which indicates failure events. Failure events represents breakages in the paper which can take several hours to resolve. The primary objective is to predict when these failure events will occur.

Figure 3: Example of Paper Mill Machine

Dataset features

- Time series of Paper Mill sensors and failures. A failure represents a paper break which takes over 1 hour to resolve
- 61 sensor parameters and 1 binary failure variable
- 18,396 records over the course of 1 month - sampled every 2 minutes
- 124 failures and 18,274 non-failures - large class imbalance

Example of sensor data – red line represents time of failures (124 total)

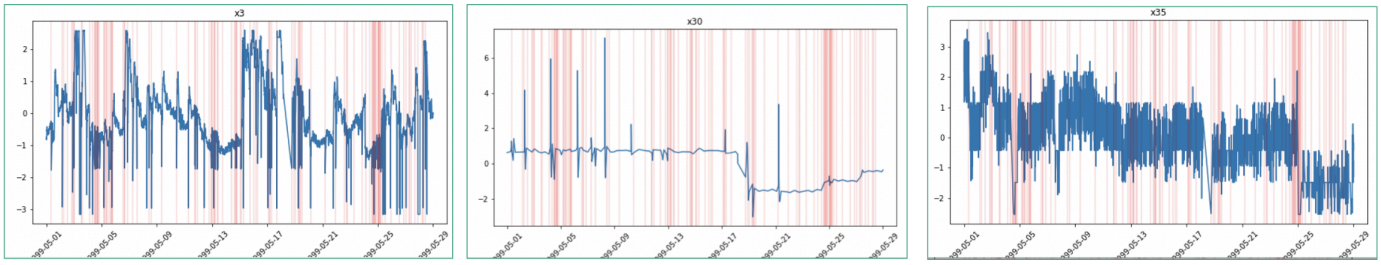


Figure 4: Time Series plots of 3 sensors in Paper Mill Dataset

Discussion

This dataset is a good fit for unsupervised anomaly detection because there is significant class imbalance (99.5% vs .5%) and supervised learning does not perform well. In the paper, the best supervised XGBoost model has a F1 score of .11 which leaves a lot of room for improvement. Additionally, the failures are somewhat randomly spread throughout the time series which means that traditional time series methods will most likely not work well.

Additionally, this dataset is quite similar to datasets that are internal to UTC. The machines that UTC makes output lots of sensor data in a time series manner. Failures are rare so class imbalance is common at UTC.

Data Preprocessing

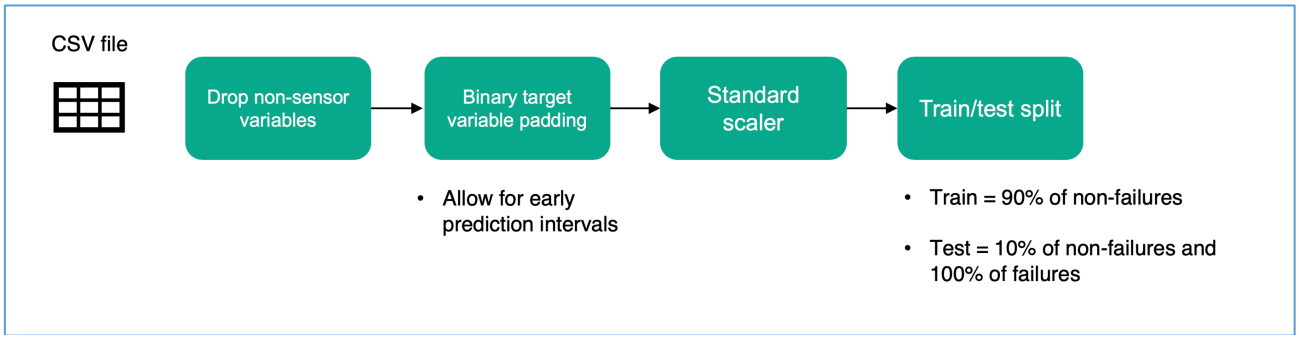


Figure 5: Preprocessing Pipeline for Paper Mill Dataset

The paper mill dataset will require processing (see above) to enable the autoencoder to be trained. The steps are outlined below.

1. Drop non-sensor variables – Drop irrelevant columns
2. Binary Target Variable Padding – Allow for predictions of failure to occur 1 and 2 timestamps before occurring
3. Standard Scaler – Rescale columns so that mean is 0 and std is 1
4. Train Test Split – Only train on class 0 (non-failures) – unsupervised learning

Metric for Classification

The chosen metric is **Precision – Recall Area Under the Curve** aka **PR AUC** aka **Average Precision Score.** The reason why this metric is chosen is because it does not require a set threshold but instead, is able to evaluate how the model generally separates class 0 from class 1. Additionally, PR AUC or Average Precision Score is especially well suited to anomaly detection models where class 1 is significantly smaller than class 0. The ROC AUC is another popular metric to assess classification but is better suited when the classes are balanced. Therefore, PR AUC is used.

Other metrics that are monitored are **f1 score** and **ROC AUC.**

Methodology #1 – Simple reconstruction error aggregation

Simple aggregation involves training an autoencoder on nominal data, deploying the autoencoder on the test dataset, and using a simple aggregation of reconstruction error to classify anomalies.

The network topology is shown below for the auto-encoder. Additionally, the following parameters are utilized

Autoencoder Features

- **Code Library**: Keras/TensorFlow
- **Topology**: 32 - 16 -32
- **Layers**: Fully Connected Dense Layers
- **Input/output dimensions**: 61
- **Loss**: MSE
- **Epochs**: 50
- **Batch size**: 12
- **Activation Function**: RELU for hidden layers and liner for output layer

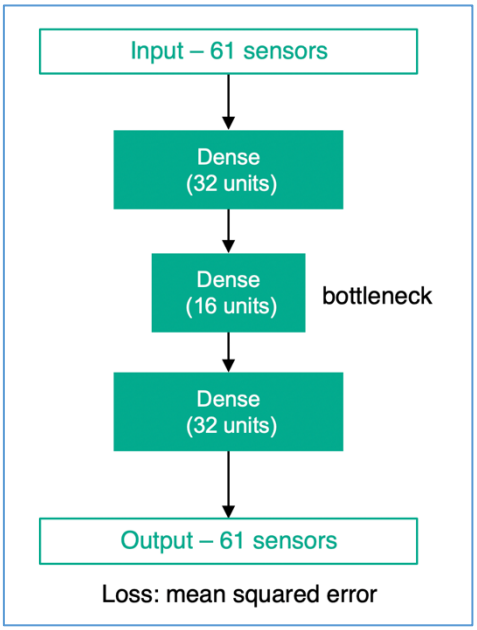


Figure 6: Network Topology for Autoencoder

Autoencoder summary and convergence

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 32)	1920
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 32)	544
dense_4 (Dense)	(None, 59)	1947

Total params: 4,939
Trainable params: 4,939
Non-trainable params: 0

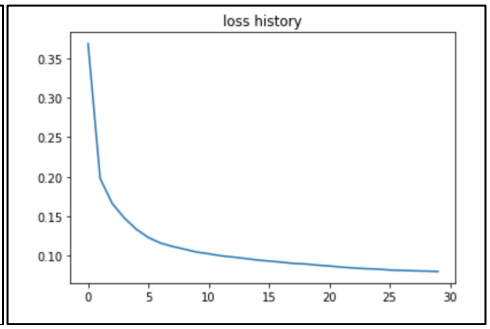


Figure 7: Autoencoder Summary and Loss History

The methodology for simple reconstruction error aggregation is shown below in figure 7. The method works as follows.

1. Train an autoencoder model on non-failure data to accurate reconstruct itself (see loss curve)
2. Make predictions on the test dataset which contains failures and determine the reconstruction error
3. Aggregate the reconstruction error using simple statistics (mean, std) and classify as anomaly based on error
4. Benchmark model based on F1 score, ROC AUC, and PR AUC

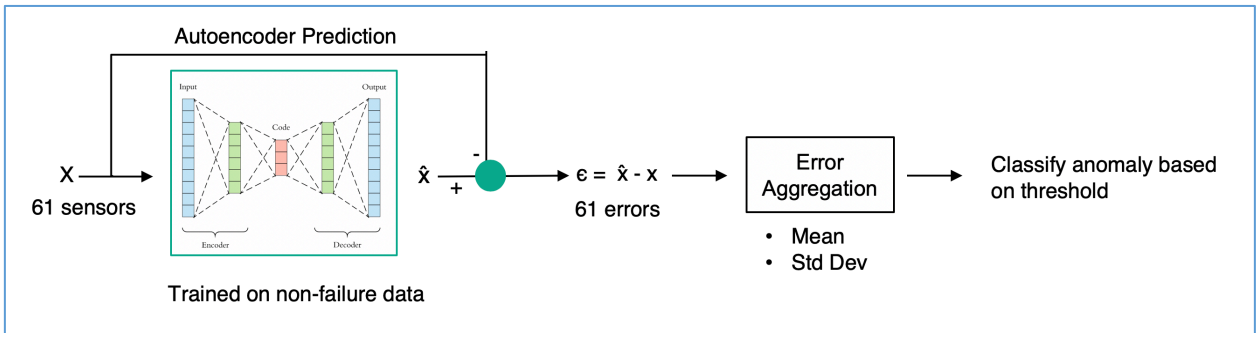


Figure 8: Simple Reconstruction error aggregation diagram

Results – Simple reconstruction error aggregation

Autoencoder Results vs XGBoost

Mean Aggregation – mean value of 61 sensor reconstruction errors

Std Dev Aggregation = standard deviation of 61 sensor reconstruction error

	XGBoost	Autoencoder Mean error	Autoencoder std error
F1 Score	.11	.391	.497
ROC AUC	Not provided	.68	.81
PR AUC	Not provided	.33	.40

Figure 9: Simple Aggregation Results Table

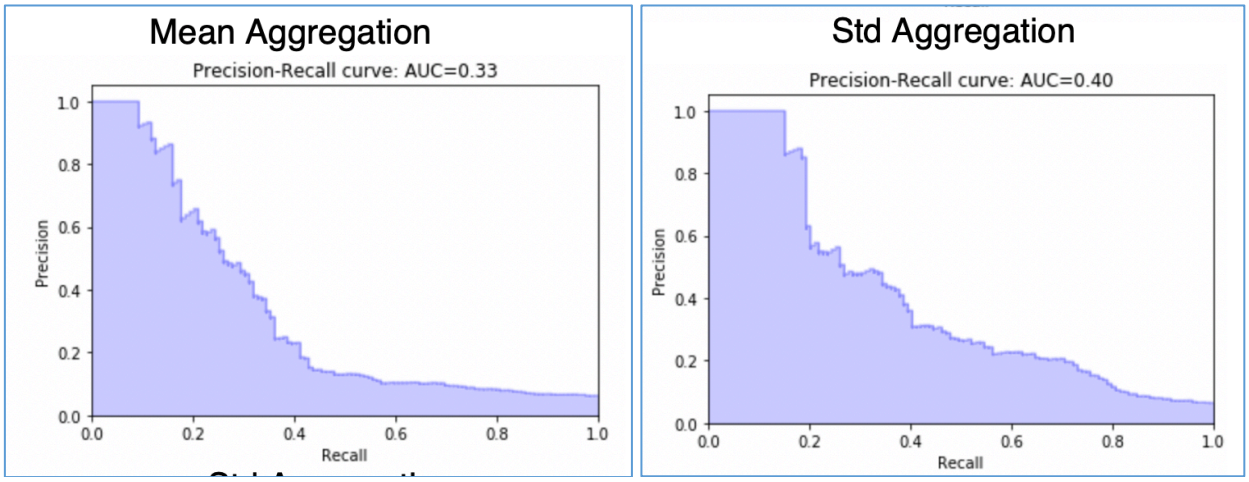


Figure 10: Precision Recall Curve for Simple Aggregation

Takeaways

- Both Autoencoder models performed significantly better than XGBoost in terms of classification (F1 score)
- Standard Deviation aggregation of error performed better than mean aggregation – this signifies that the spread of reconstruction error is important

Additional Studies for Simple Aggregation

Early Interval Study

A study is performed to determine how performance varies when predicting the failure N timestamps before. An early interval (N) of 1 through 3 is assessed with the results shown below.

	Early 1	Early 2	Early 3
F1 Score	.391	.147	.1
ROC AUC	.70	.60	.60
PR AUC	.33	.12	.10

Figure 11: Early Interval Study Results

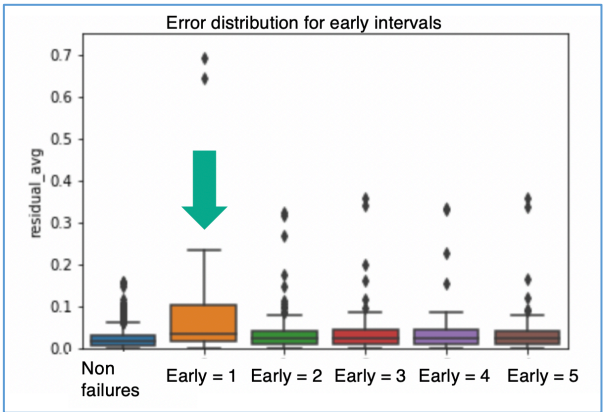


Figure 12: Early Interval Error Distribution

Takeaways

- Predictive capability deteriorates rapidly as early interval increases
- Failure signal is only clear 1 timestamp before occurrence. Failure happens very quickly rather than gradually

Feature Selection Study

A study was done to determine whether or not a subset of features would improve the performance of the model. The 61 sensor variables were down-selected to 15 via an XGBoost supervised learning model for failures. The top 15 features were determined by a ranking of feature importance. The results are as follows:

Hidden Layer Shape	All features - 59	Top 15 XGBoost Features
F1 Score	.391	.192
ROC AUC	.68	.66
PR AUC	.33	.28

Figure 13: Feature Selection Study Results Table

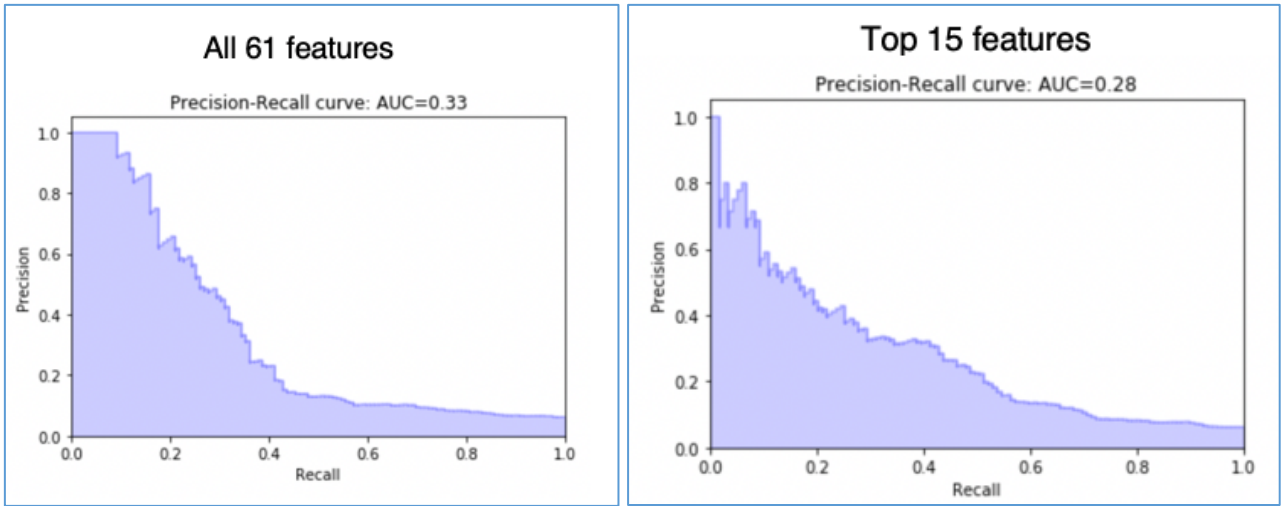


Figure 14: Precision Recall Curve for Feature selection study

Takeaways

- Feature selection did not seem to provide much benefit to the model
- Why this may be the case is because autoencoders already selecting the most important "features" or code in the bottleneck layer.

Topology Study

A study was done to determine whether the topology of the neural network impacts performance significantly. 3 topologies were assessed as seen below.

	Complex	Base Case	Simple
Topology	32-16-8-16-32	32-16-32	16
F1 Score	.368	.391	.257
ROC AUC	.68	.68	.65
PR AUC	.30	.33	.23

Figure 15: Topology Study Results

Takeaways

- 3-layer topology seems to work best. 5-layer tends to overfit and 1-layer tends to underfit
- It is evident that we should not target minimum reconstruction error but rather best separation of reconstruction error for failures vs non-failures. The autoencoder ideally should not be overfit to the trainset reconstruction error.

Methodology #2 – Gaussian model for Reconstruction Error

A Gaussian Model for reconstruction error is also evaluated. The method assumes that nominal sensor data can be modeled by a gaussian distribution. A gaussian is fit on the non-failure (training set) raw sensor values and training reconstruction errors. The probability density function (PDF) is assessed to determine outliers/anomalies of data points in the testing set.

Gaussian Model #1: Univariate gaussian model (no covariance)

1. Assume all variables (60) are normally distributed and independent (no covariance)
   a. $X_1 \sim N(\mu_1, \sigma_1) \ \ldots X_{60} \sim N(\mu_{60}, \ \sigma_{60})$

2. Fit normal distribution to training set sensor data (90% of non-failures) for each variable

3. Calculate pdf for test set (10% non-failures, 100% of failures)

4. Aggregate pdfs for the variables by taking mean and also the product of all pdf values
   a. $pdf_{mean} = mean(pdf_1, \ \ldots, pdf_{60})$
   b. $pdf_{product} = pdf_1 * pdf_2 \ * \ \ldots \ * pdf_{60}$

Gaussian Model #2: Multivariate gaussian Model

1. Assume all variables (60) can be fitted with a multivariate gaussian distribution
   a. $X \sim N(\mu, \Sigma)$
   b. $cov(X) = \ \Sigma$ is a 60 x 60 matrix

2. Fit normal distribution to training set sensor data (90% of non-failures) for each variable

3. Calculate pdf for test set (10% non-failures, 100% of failures)
   a. No need for aggregation since pdf is a single value that factors in all variables

Autoencoder reconstruction error Gaussian Model (using model #1 and #2 above)

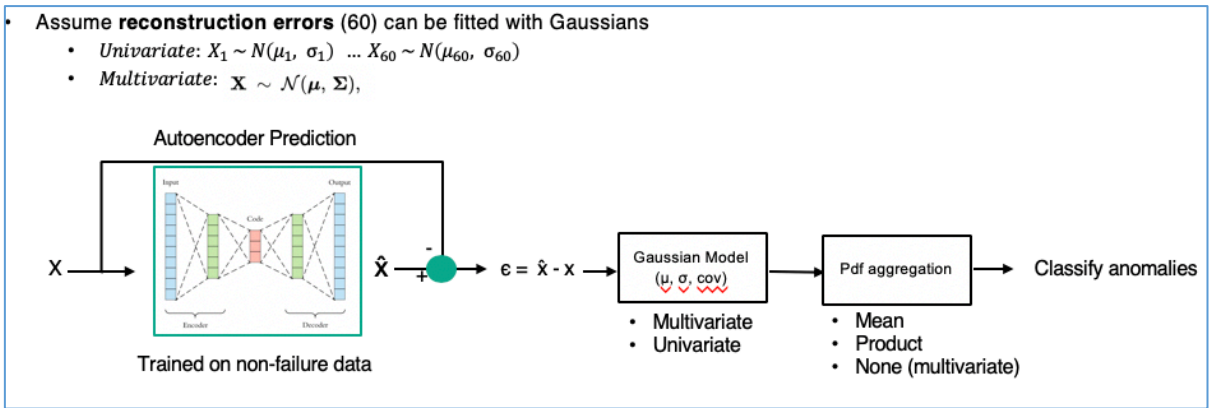


Figure 16: Autoencoder + Gaussian Model

Results – Gaussian model for Reconstruction Error

Results for raw sensor input

Early Interval	Autoencoder Std error	Autoencoder KNN	Univariate Pdf Mean	Univariate Pdf Product	Multivariate Pdf
ROC AUC	.73	.86	.53	.65	.68
PR AUC	.39	.62	.14	.22	.44

Figure 17: Gaussian Model Results on Sensor Inputs

Results for Autoencoder Reconstruction Error

Early Interval	Autoencoder Std error	Autoencoder KNN	Autoencoder Uni Pdf mean	Autoencoder Uni Pdf prod	Autoencoder Multi PDF
ROC AUC	.73	.86	.74	.66	.75
PR AUC	.39	.62	.39	.30	.41

Figure 18: Gaussian Model Results on Reconstruction Errors

Takeaways

- Gaussian Model underperformed autoencoder when using raw sensor inputs and matched autoencoder model when using reconstruction errors
- Raw sensor inputs are not truly gaussian since many of the sensors exhibit bi-modal distributions and have large skew in the distribution
- Reconstruction errors appear to be significantly closer to a gaussian distribution than the raw sensors. This is because the autoencoder is trained to optimize for MSE in which errors are normally distributed assuming that the model is able to fit the data well.

Methodology #3 –Unsupervised + Unsupervised

*limited details can be disclosed below due to company policy

This method aims to use 2 unsupervised techniques to improve performance in comparison to a simple autoencoder. The limitation of simple autoencoder with error aggregation can be attributed to 2 factors.

A new model is developed, and the results are shown below.

Methodology 3 Results

Early Interval = 1	Autoencoder Mean error	Autoencoder STD error	Methodology 3
F1 Score	.391	.497	.663
ROC AUC	.68	.81	.93
PR AUC	.33	.40	.69

Methodology #4 – Semi supervised

*limited details can be disclosed below due to company policy

This method aims to combine an unsupervised approach with a supervised approach to improve results from gaussian model and simple error aggregation model. See results below.

Method 4 Results

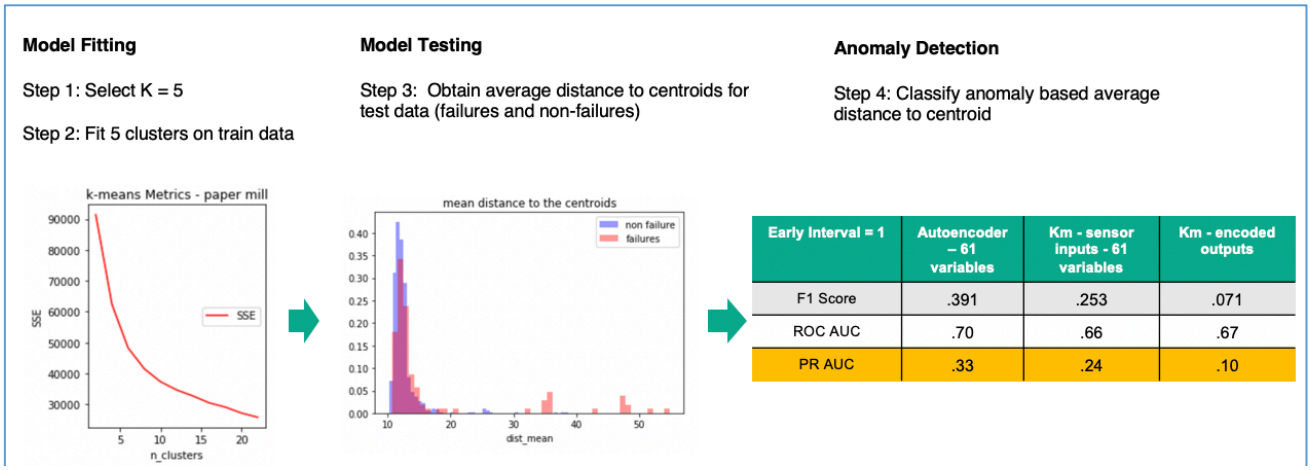
	Autoencoder Mean error	Autoencoder STD error	Method 3	Method 4
F1 Score	.391	.497	.663	.691
ROC AUC	.68	.81	.93	.86
PR AUC	.33	.40	.69	.65

Comparison to Clustering

The simple aggregation model is compared to a K means clustering anomaly detection algorithm. The K means clustering fits K clusters on the training data (non-failures) and uses the average distance to K cluster centroids as a way to separate failures and non-failures. This process is highlighted below in figure 21.

The K means clustering performs worse but serves as a baseline for performance. The reason why K means might be well suited is noted below:

- K means works best when it assumes that data is of similar density all throughout. This may not be the case for the paper mill data
- K means assumes that data is clustered around centroids in a spherical fashion. It does not account for data that has high covariance across different dimensions
- K means is ill-suited for data where failures are located within a cluster but isolated in a particular segment of the cluster.



Early Interval = 1	Autoencoder – 61 variables	Km - sensor inputs - 61 variables	Km - encoded outputs
F1 Score	.391	.253	.071
ROC AUC	.70	.66	.67
PR AUC	.33	.24	.10

Figure 19: K means clustering approach and results

Additional Clustering Studies

Further studies were done which involved clustering the top 15 features chosen by XGBoost along with clustering the encoded layer outputs from the trained autoencoder. These trials did not yield any models that were better than the base case autoencoder model.

	All 61 sensors		Top 15 sensors	
Early Interval = 1	Km - sensors	Km - encoded	Km - sensors	Km - encoded
F1 Score	.253	.05	.071	.027
ROC AUC	.66	.51	.60	.45
PR AUC	.24	.10	.13	.06

Figure 20: Additional Clustering Results

t-SNE Study

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction technique that is well suited for 2D and 3D visualization of high dimensional data. It has shown success in separating anomalies from nominal data in different domains and has been successfully in visualization images, text, and other forms of complex data that have high dimensions. An attempt was made to visualize the encodings of the autoencoders on the test set to see if anomalies would stand out. Unfortunately, t-SNE showed poor performance in separating failures from non-failures. Ideally the blue nominal points would be separated from the colored points (failures). This did not occur as seen below.

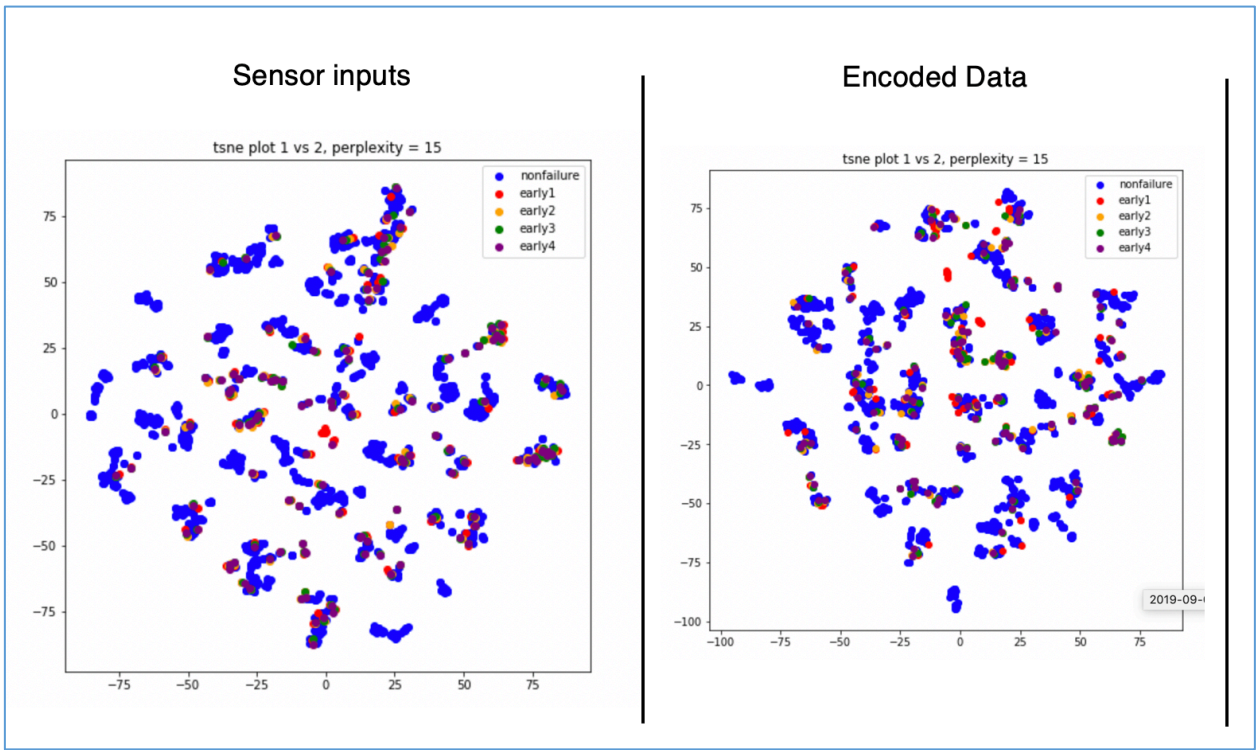


Figure 21: t-SNE visualizations

Ideally, TSNE would show separation between failures and non-failures. In other use-cases such as images and text, TSNEs has proven that it can aid in visualization of different clusters. See https://lvdmaaten.github.io/tsne/ for examples.

Comparison to other datasets and use-cases

1. Machine Failure Dataset
   https://bigml.com/user/czuriaga/gallery/dataset/587d062d49c4a16936000810
   - 9000 rows – 81 failures
   - 17 sensor variables
2. Credit Card Fraud Dataset - https://www.kaggle.com/mlg-ulb/creditcardfraud
   - 280K rows – 492 frauds
   - 28 anonymous variables
3. IBM IoT dataset - https://developer.ibm.com/patterns/predict-equipment-failure-using-iot-sensor-data/
   - 900 rows, 393 failures
   - 9 sensors
4. Credit Card Default Dataset -
   https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
   - 30000 rows, 6600 defaults
   - 25 payment variables

(PR AUC)	Credit Card Fraud Imbalance = 0.1%	Paper Mill Imbalance = 0.6%	Machine Failures Imbalance = 1%	Credit Card Default Imbalance = 20%	IBM IOT Imbalance = 40%
Logistic Regression	.60	.21	.38	.30	.96
Univariate Gaussian (mean pdf)	.59	.17	.21	.60	.83
Multivariate Gaussian	.50	.40	.57	.57	.80
AE + Mean error	.20	.33	.12	.49	.54
AE + Gaussian	.78	.40	.33	.55	.84
Methodology 3	.78	.69	.52	.52	.79
Methodology 4	.88	.65	.72	.68	.97

Figure 22: Results on other datasets

Discussion

Overall, the unsupervised anomaly detection models performed better than supervised approached when the class imbalance is very large. This makes logical sense as class imbalance makes supervised learning particularly challenging. Based on the results above Methodology 3 and Methodology 4 are the most promising methods for separating failures from non-failures using autoencoders.

Conclusions

- 3 promising method were developed. Method 3 and Method 4 are the most promising.
- There is value in seeing the autoencoder reconstruction error as features
- ML based anomaly detection models tend to work better than cluster/probability-based models for this class of problem
- Precision – Recall curve is useful in assessing how a model separates failures from non-failures in the presence of extreme class imbalance
- T-sne did not provide useful insights for sensor data failure separation
- Autoencoder should be tuned to maximize separation of failures from non-failures rather than simply minimizing reconstruction error

References

1. https://lvdmaaten.github.io/tsne/
2. https://www.jeremyjordan.me/autoencoders/
3. https://towardsdatascience.com/lstm-autoencoder-for-extreme-rare-event-classification-in-keras-ce209a224cfb
4. https://arxiv.org/pdf/1802.04431.pdf
5. https://www.sciencedirect.com/science/article/abs/pii/S0952197619301721
6. https://medium.com/@curiousily/credit-card-fraud-detection-using-autoencoders-in-keras-tensorflow-for-hackers-part-vii-20e0c85301bd
7. https://arxiv.org/abs/1811.05269
8. https://sites.cs.ucsb.edu/~bzong/doc/iclr18-dagmm.pdf
9. https://saketsathe.net/downloads/autoencode.pdf
10. http://proceedings.mlr.press/v95/guo18a/guo18a.pdf
11. https://cds.cern.ch/record/2209085/files/Outlier%20detection%20using%20autoencoders.%20Olga%20Lyudchick%20(NMS).pdf