UNIVERSITY of ROCHESTER

Vnomics-1
Final Presentation

Vnomics
A Revolution In Vehicle Economics Through Technology


Daniel He


Matthew Taruno


James Sastrawan


Linzan Ye


Vatsal Agarwal

BUSINESS CONTEXT

Core Issue

1. DPF failure is costly.
2. Sudden Breakdown

GOAL

Build a model, based on data science techniques, to predict DPF failures in trucks at least two weeks in advance

DRIVING BEHAVIOR

1. Idling
2. Shifting
3. Accelerating
4. Speeding

TRIP CONTEXT

1. Load
2. Truck Config
3. Environment
4. Speeding

REQUIREMENTS

I

Produce a model to accurately predict DPF failure in trucks by establishing early indicators of such failure

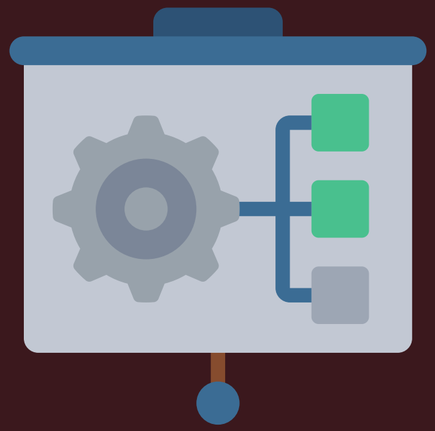
II

Alert customers of potential need for regeneration ahead of time to avoid costly roadside breakdowns

III

Ideally, a truck with any failure indicators should not be missed, so the model should have a high recall score.

Project Deliverables

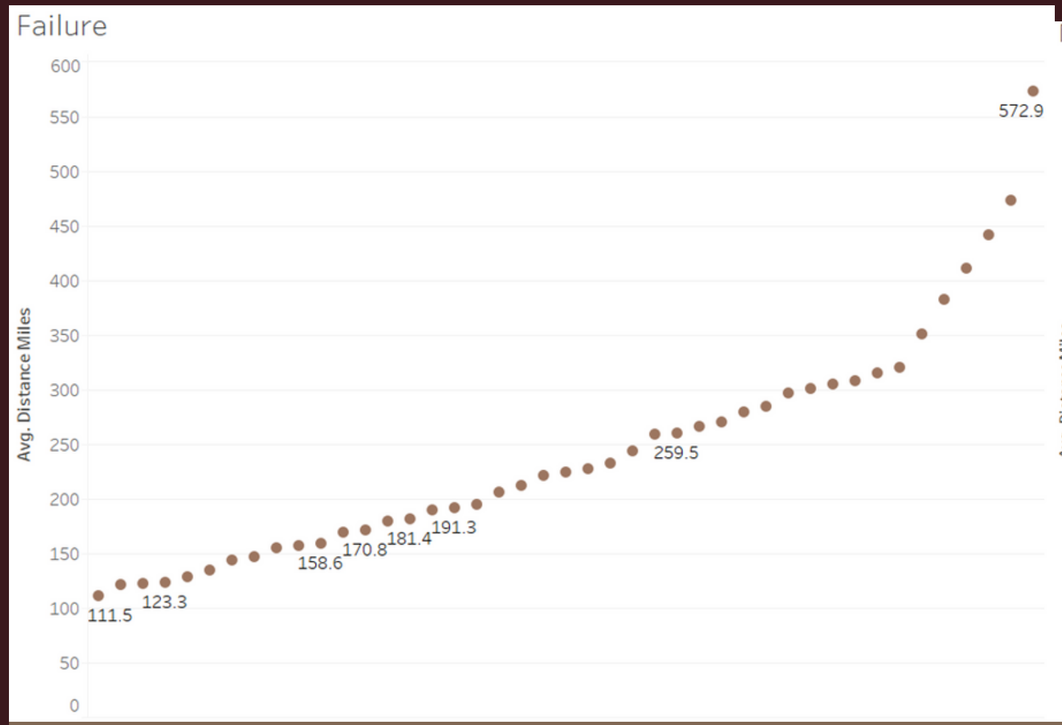


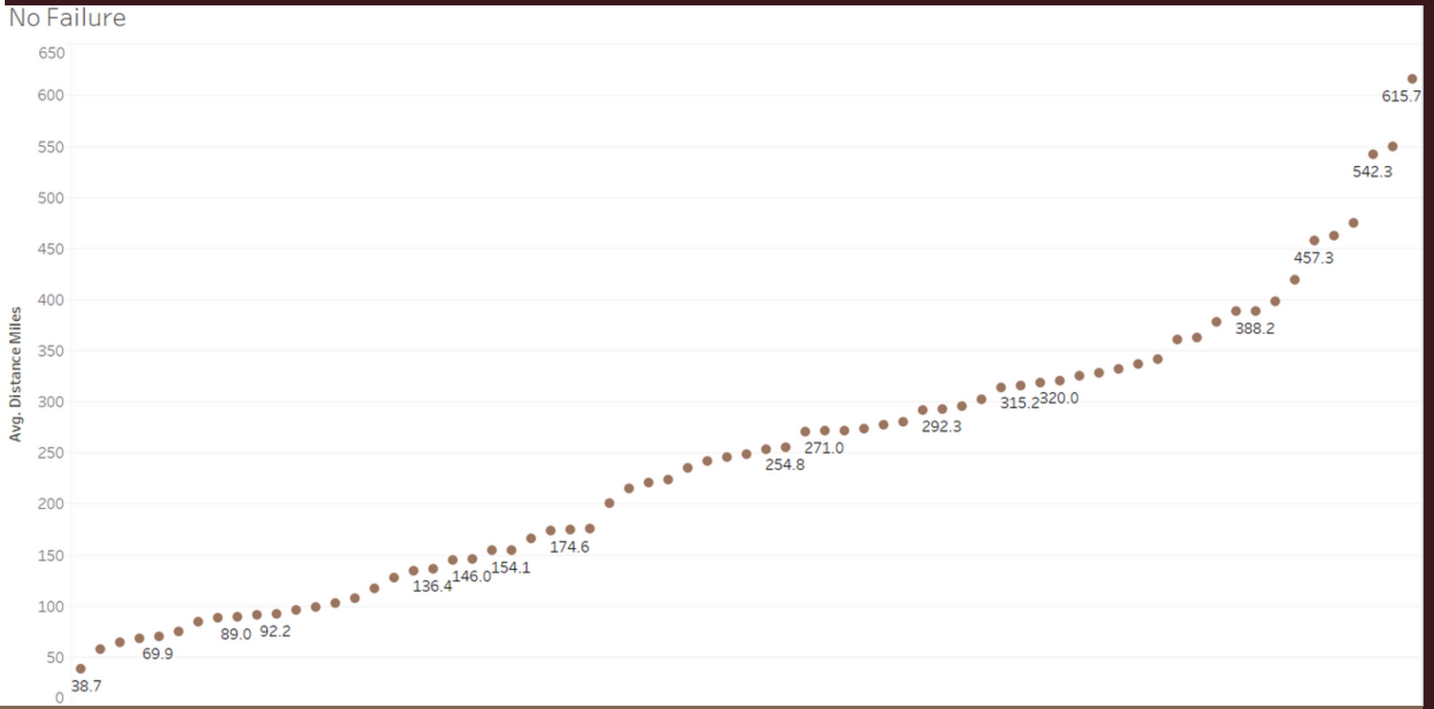
1. Baseline model and iteration over Models to improve performance

2. Weekly Collaboration and Reports Tracking Progress

3. Final Model with High Recall Score

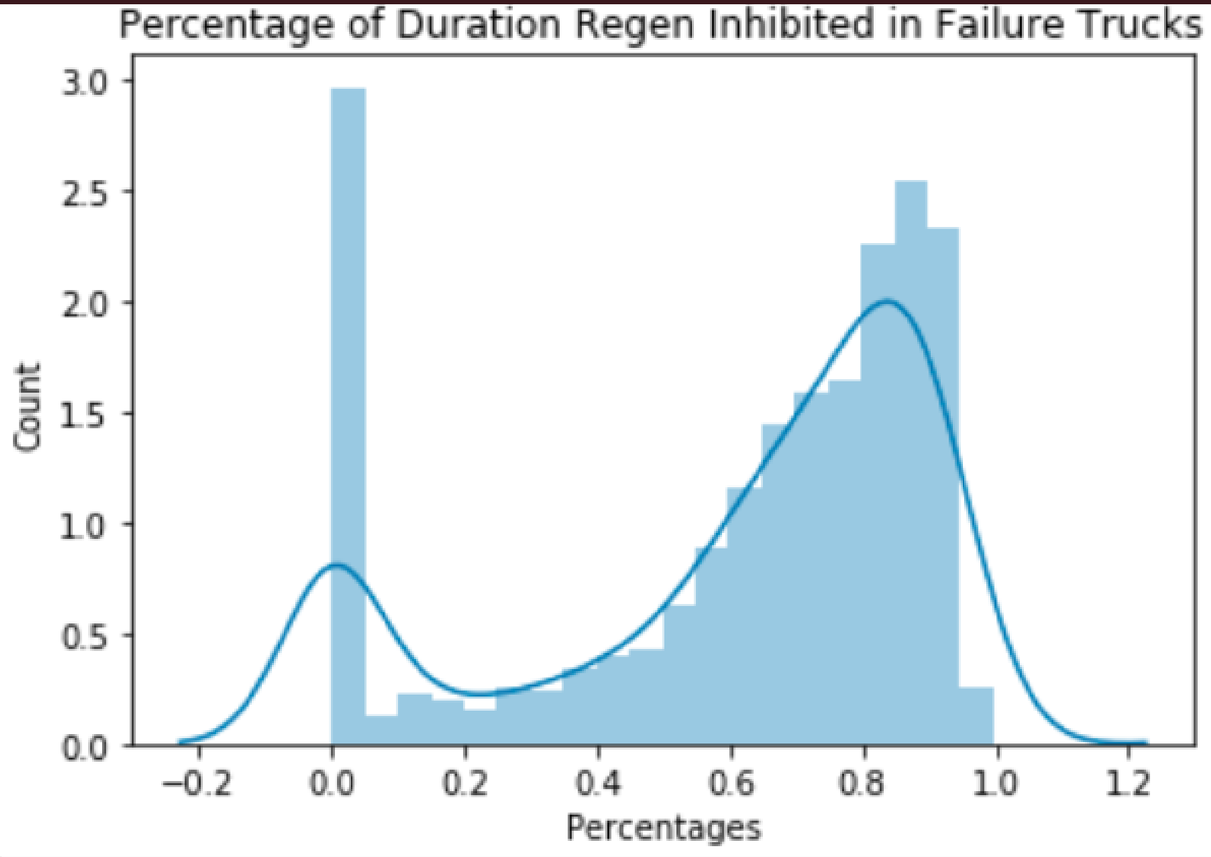
DISTRIBUTION OF TRUCK DISTANCE

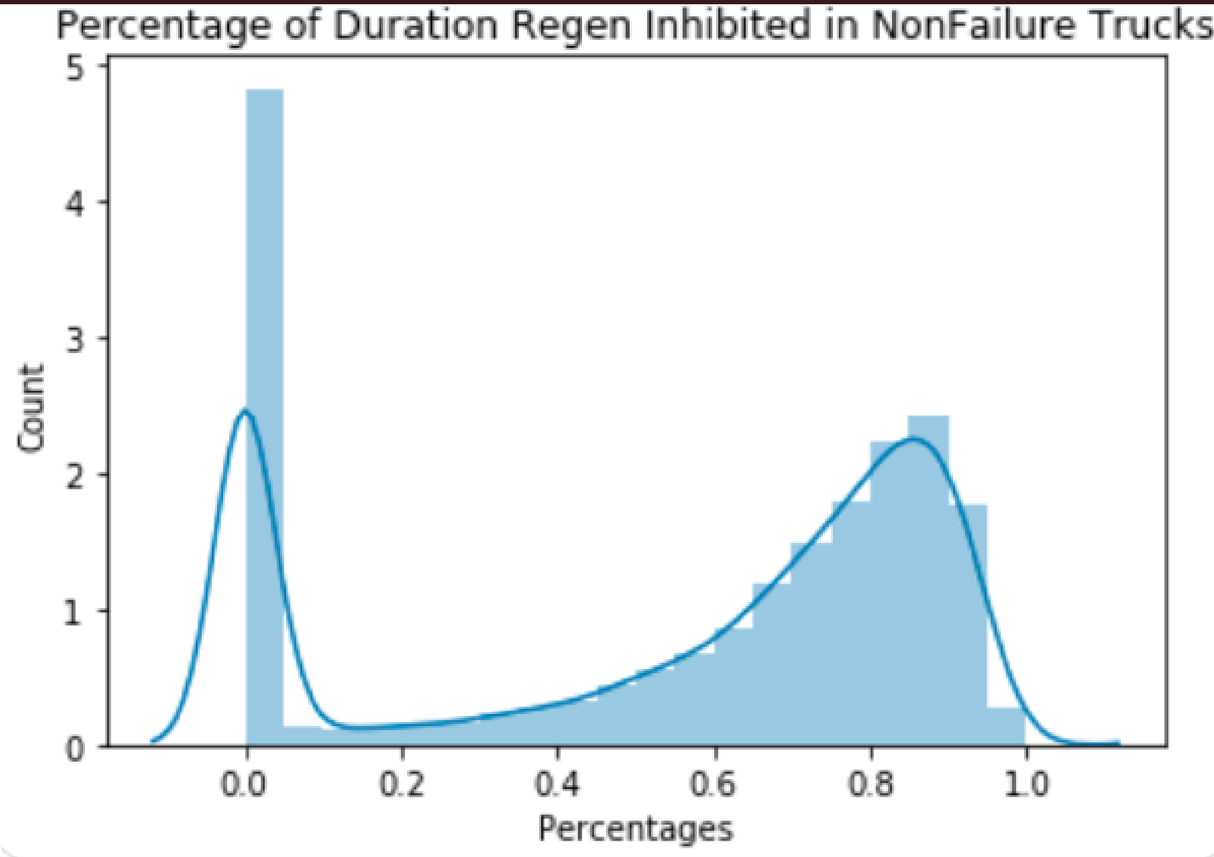




Each x-axis label is a single platform ID - we see that the average distance trucks travels rance from around 90 to 600 - which shows that the variance in the distances of trucks are significant

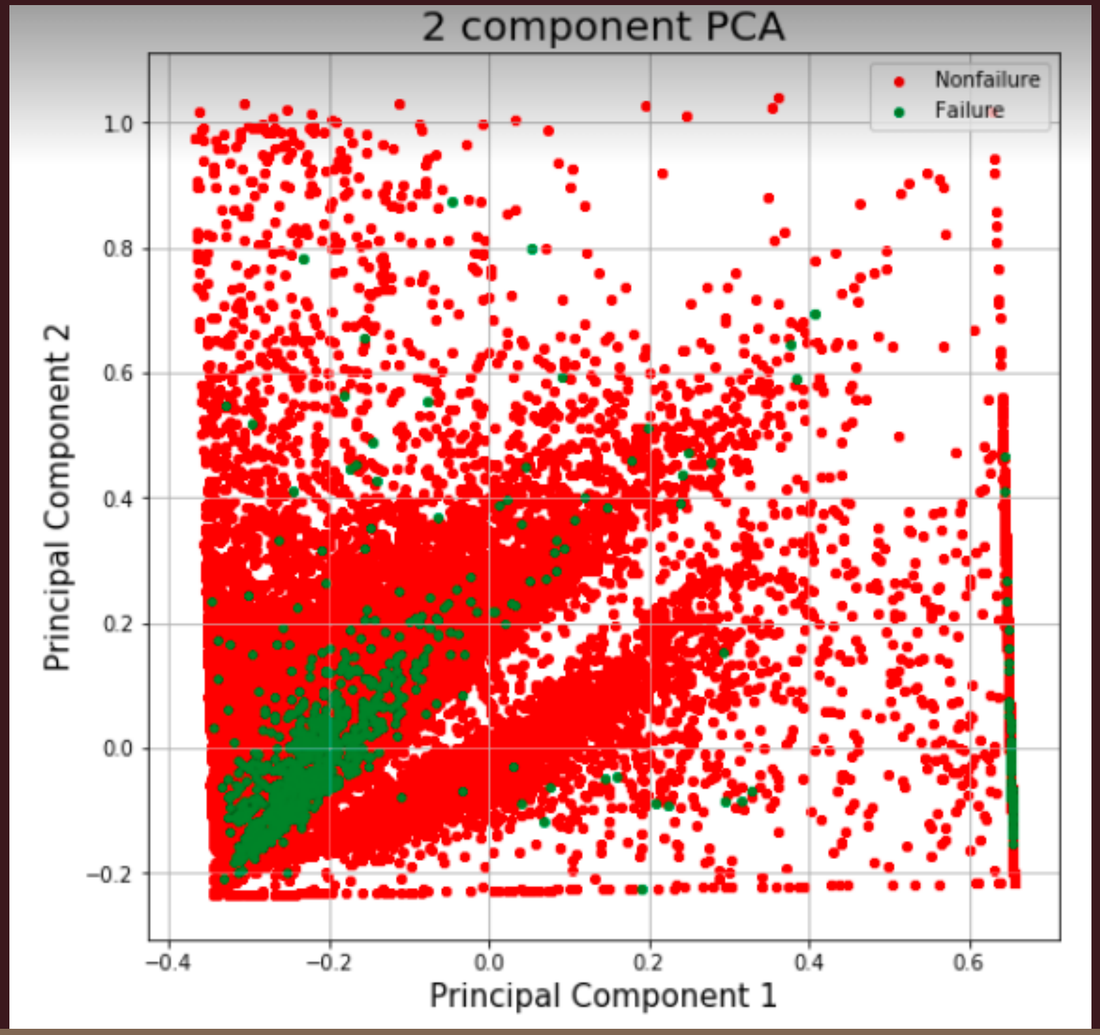
DURATION REGEN COMPARISON





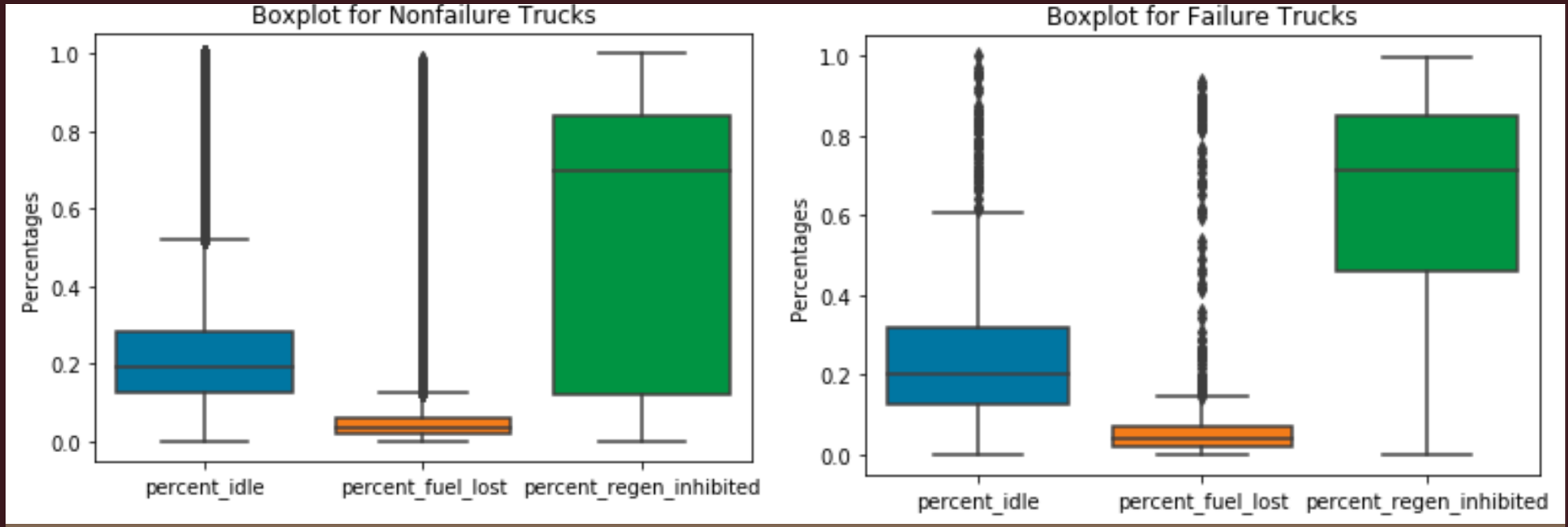
The nonfailure trucks has a stronger left skew suggesting the feature as a proxy for failure. We use this as one of our features we select later.

2 Component PCA



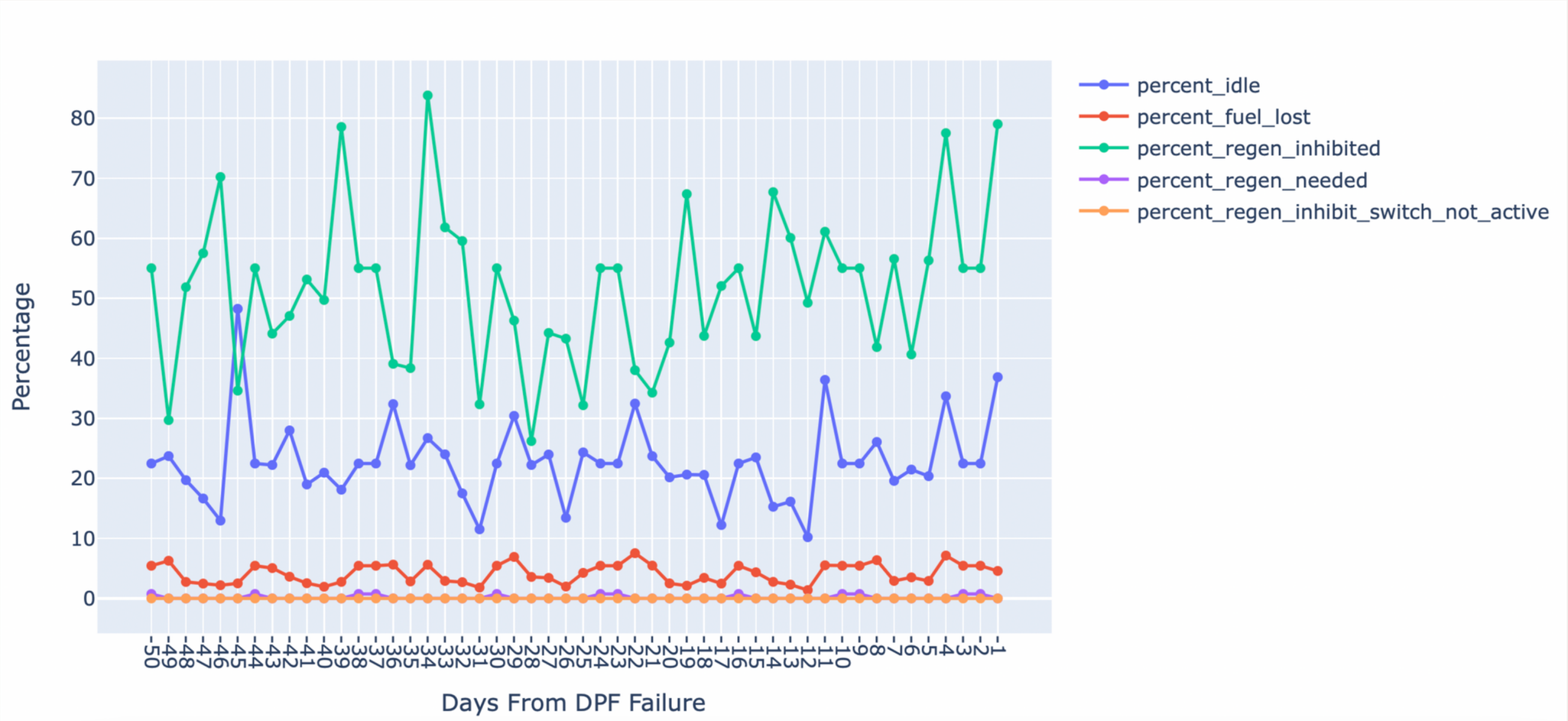
We attempted to run PCA on the original dataset for feature explainibility - the results were not optimal

BOXPLOT FOR FAILURE AND NONFAILURE



We see that failure trucks have a narrower inter quartile range for percent_regen_inhibited, but they mostly have similar distributions

Key Characteristics 50 Days Before a Failure



Visualizing the data for Platform ID 300490 50 days before a failure

EDA's Main Insights

Notable differences between failure and nonfailure trucks

Things can be explained from a relatively small feature set

We found clear candidates for features that we shouldn't use

A few features had no difference between nonfailure vs failure

Obvious class imbalance problem at hand

Some clear outliers for certain features

Feature Creation

1. Raw numbers → Percentages

Before: Duration_mins, mins_idle

After: percent_of_time_idle

- Makes all entries on the same scale

2. Existing → New

Fuels Used, Miles Traveled

Fuels used per mile

- Different way to aggregate the data

Model Methods

1. Linear:
   - Random Forest for Feature Importances
   - PCA for Anomaly Detection

2. Nonlinear:
   - Autoencoders
   - TS Fresh Time Series Algorithms and Feature Transformations

Narrowing Down to Two Features

Created a random forest model to determine the most relevant features to distinguish between the failure and nonfailure classes

Class imbalance problem can be fixed

15 days before service date = Failure
15-30 days after service date = Nonfailure

Leads to a balanced dataset

Top 5 Important Features

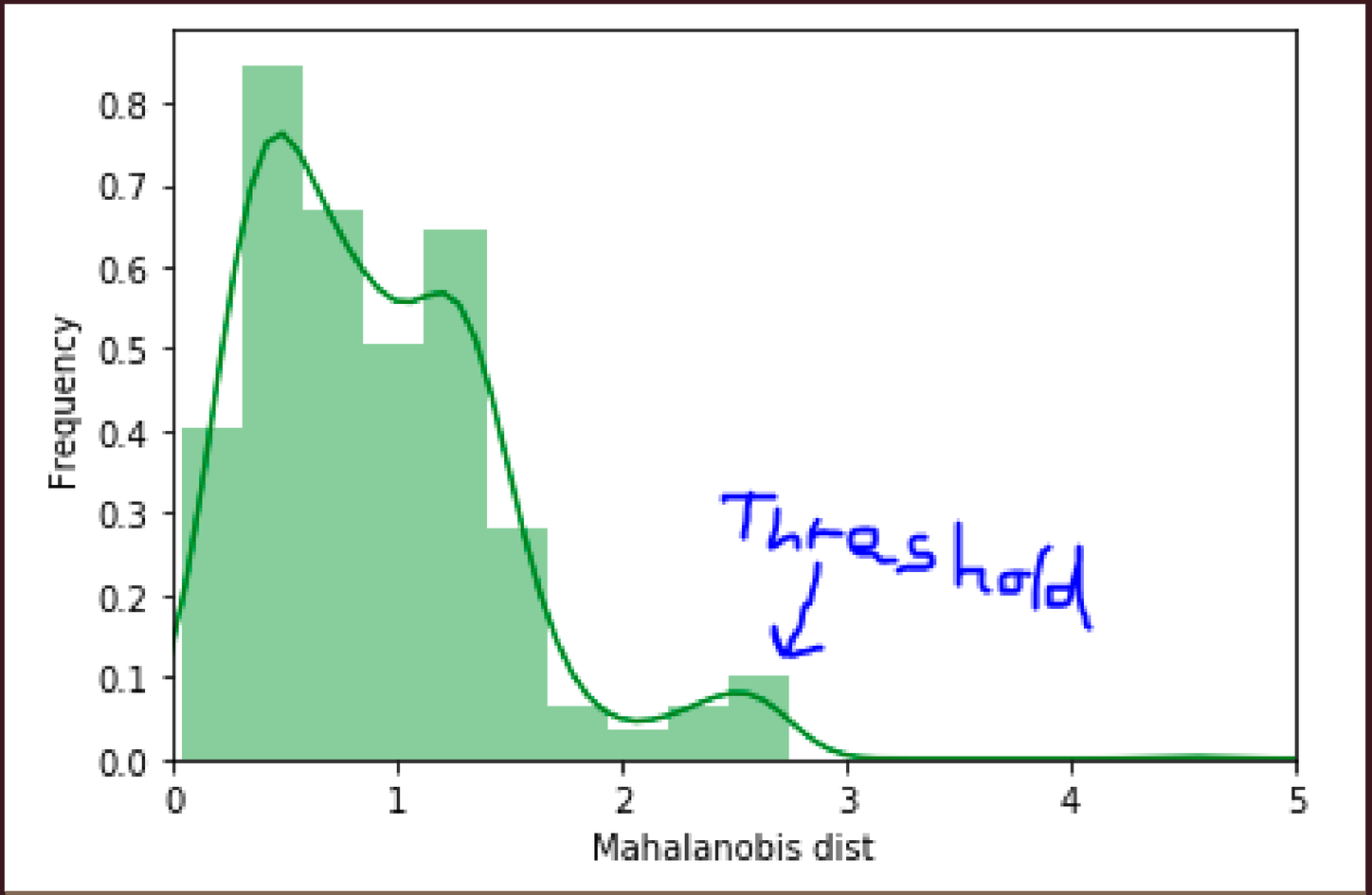
- percent_fuel_lost
- fuels_used_per_mile
- miles_per_minutes
- mins_idle_per_mile
- percent_regen_inhibited

PCA for Anomaly Detection

1. Reduce the original feature set to top 2 Principal Components
2. Use only nonfailure trucks to calculate the covariance matrix of the normal class
3. Calculate Mahalanobis distance for each of the failure trucks to the normal class
4. If distance above the threshold, it would be marked as an anomaly
5. Repeat the process with varying starting features

percent_fuel_lost, fuels_used_per_mile, miles_per_minutes, mins_idle_per_mile, percent_regen_inhibited

Distribution of Mahalanobis Distances



The threshold is 3 standard deviations from the mean, so from this histogram, it is evident the threshold would be around 2.8

Big Picture Pipeline

Simple Imputer

Raw Data Ingest → Percent Columns → **Preprocessing Pipeline**

15 Days Before Label

Schema Validation

Big Picture Pipeline

PCA
Visualizations
Random Forest

Narrow Down to Two Features

percent_regen_inhibited
percent_fuel_lost

ts_fresh

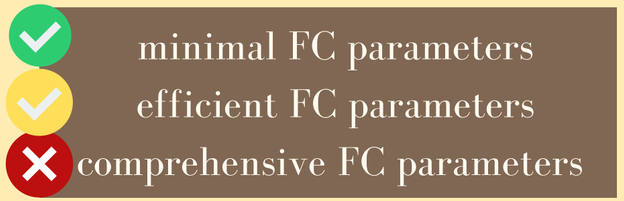
Transforming to Rolling Window Dataset

shift by 1
window by 20

shift by 25
window by 50

shift by 1
window by 50

ts_fresh

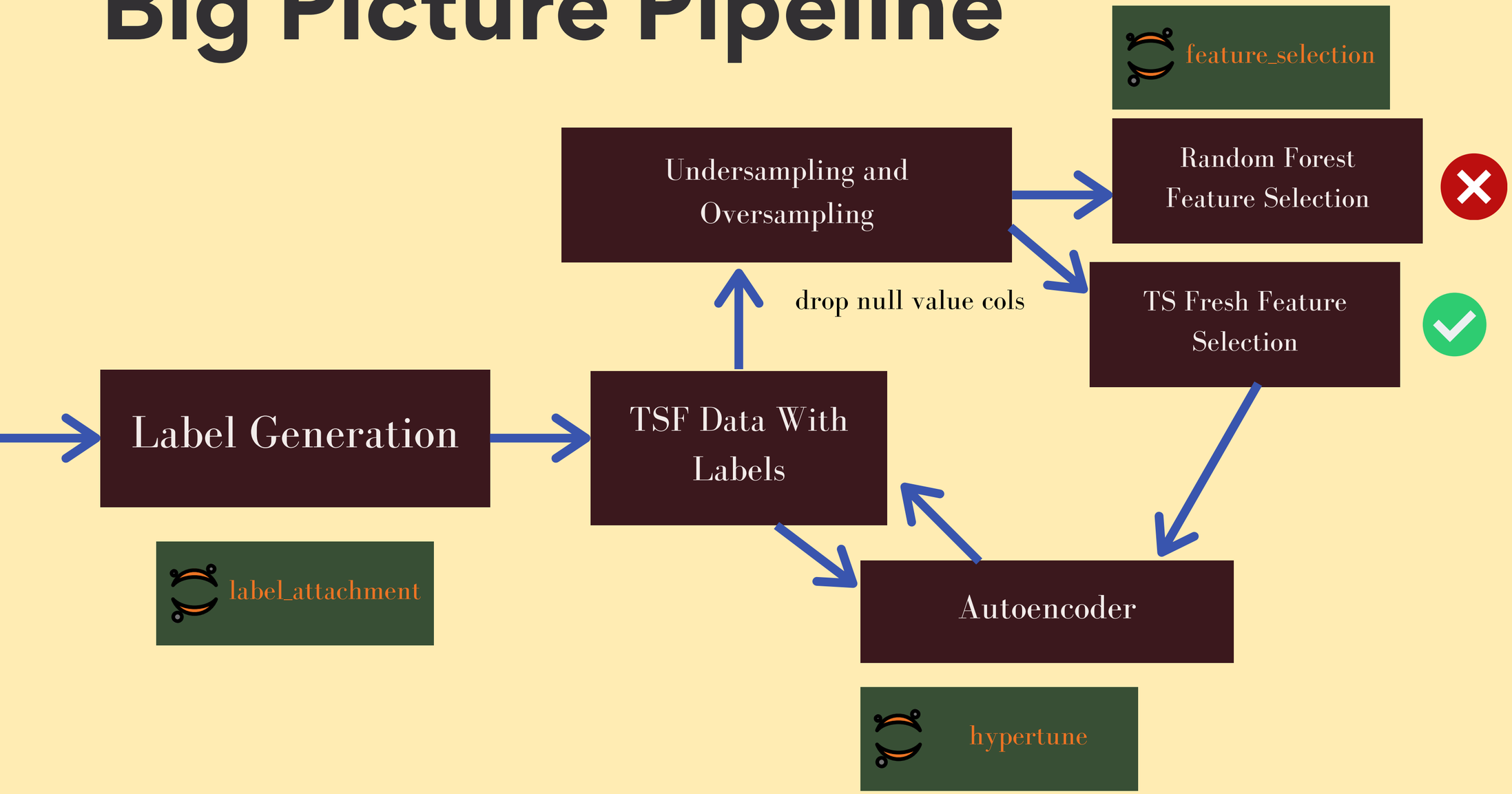


TS Fresh Extract Features

extract_features on
rolled window dataset

Hurdles: Runtime, parallelization,
data formatting, data wrangling,
seamless integration, etc.

Big Picture Pipeline



TS Fresh Transformations

9 Minimal

74 Comprehensive

72 Efficient

52 Medium

sum_values
median mean
length
standard_deviation
variance
root_mean_square
maximum
minimum

variance_larger_than_standard_deviation
has_duplicate_max
has_duplicate_min
has_duplicate
sum_values
abs_energy
mean_abs_change
mean_change
mean_second_derivative_central
absolute_sum_of_changes

last_location_of_maximum
percentage_of_reoccurring_values_to_all_values
percentage_of_reoccurring_datapoints_to_all_data points sum_of_reoccurring_values
sum_of_reoccurring_data_points
ratio_value_number_to_time_series_length
maximum minimum benford_correlation
time_reversal_asymmetry_statistic c3 cid_ce
symmetry_looking large_standard_deviation
quantile autocorrelation agg_autocorrelation
partial_autocorrelation number_cwt_peaks

TS Fresh Transformations

9 Minimal

74 Comprehensive

72 Efficient

Too simplistic

Runtime too high
Memory timeout

Runtime around 2h
Memory timeout

Note: Cluster is capped at 30GB memory

52 Medium

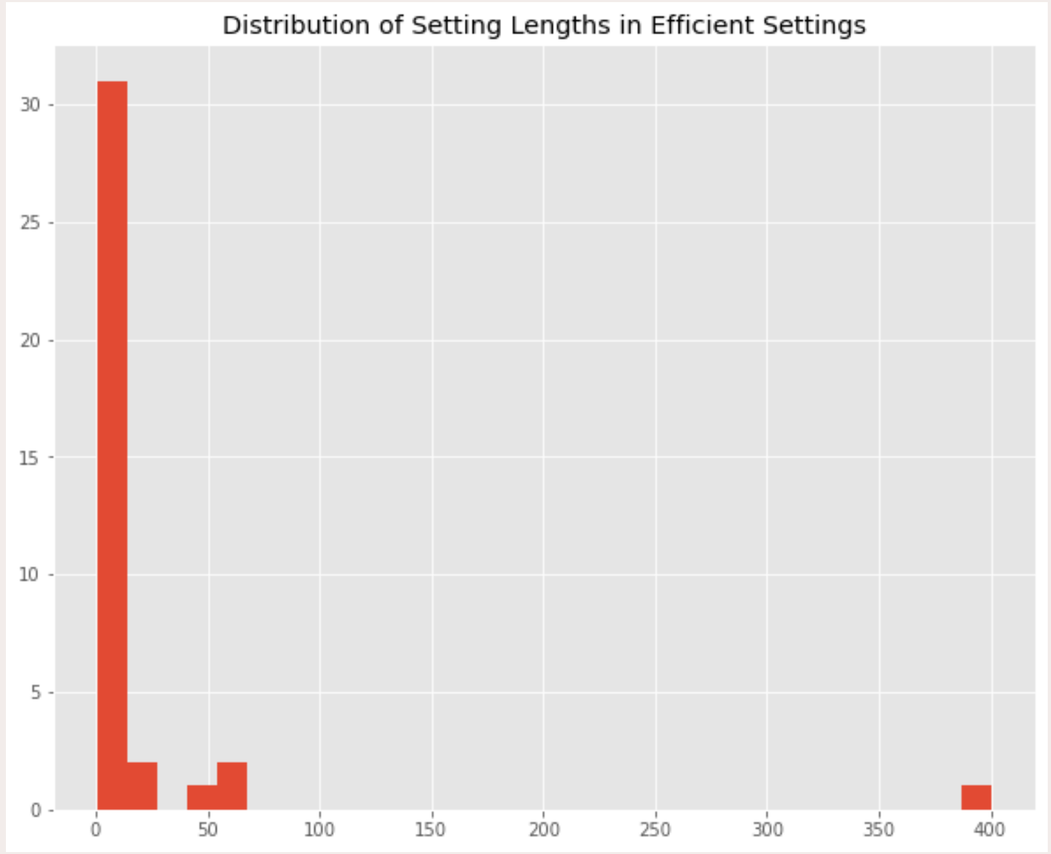
```
X_tsf_20_efficient = extract_features(df_rolled.drop("vehicle_serviced", axis = 1),
        column_id="id", column_sort="time", default_fc_parameters = efficient_settings)

Cancel  Running command...
Feature Extraction:  20%|          | 2/10 [21:00<1:56:49, 876.20s/it]
```

```
Cmd 71

X_tsf_20_medium = extract_features(df_rolled.drop("vehicle_serviced", axis = 1),
        column_id="id", column_sort="time", default_fc_parameters = keep_settings)

Cancel  Running command...
Feature Extraction:  40%|          | 4/10 [20:55<34:52, 348.69s/it]
```



Removed all instances where the transformation parameter setting dictionary value (sub transformations) exceeds 5

Tried to parallelize with Dask or make code Spark aware

Class Imbalance

Initial Dataset Distribution:
100,000 non failures
1380 failures

After Transformation

0.0909%

of examples are a DPF failure

97887:89

Undersampling

33%

of examples are a DPF failure

Synthetic Minority Oversampling Technique (SMOTE)

50%

of examples are a DPF failure

imblearn

Runs and Parameter Configurations

Class Balance Data

- Balanced
- Unbalanced

TSF Window Size and Overlap

- w: 50 s: 25
- w: 20 s: 20
- w: 20 s: 1

RF

- n_estimators

TS Fresh Transformations

- Minimal
- Comprehensive
- Efficient
- Medium

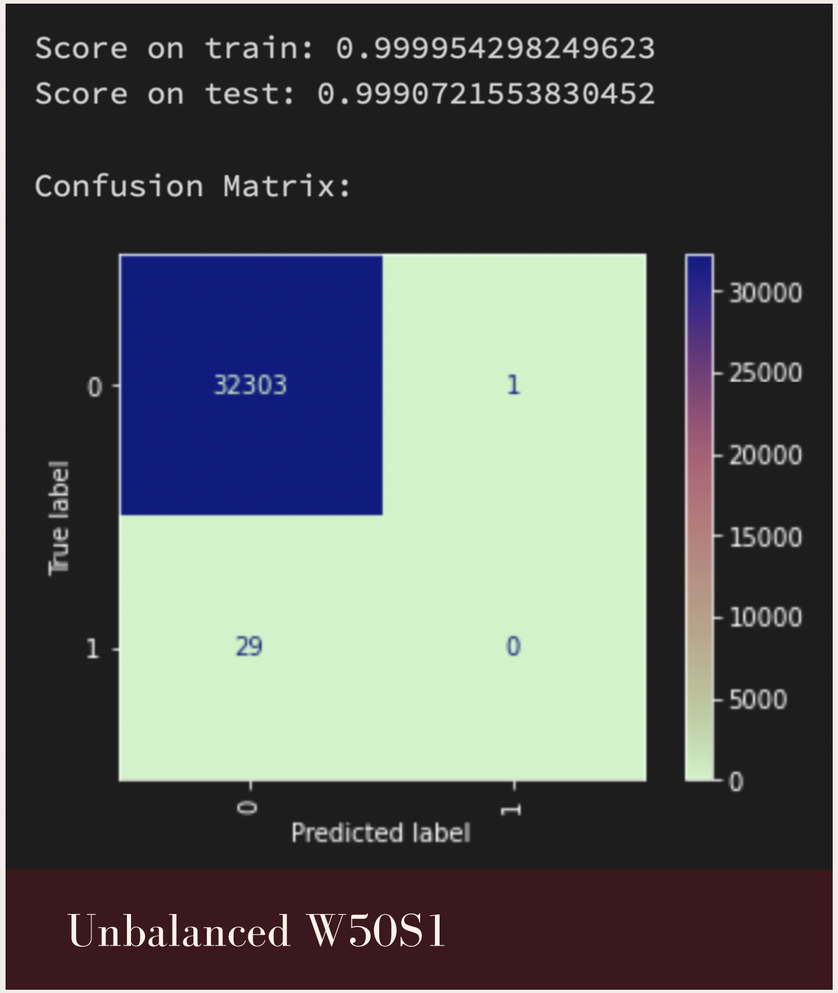
Autoencoder Parameters

- Autoencoder Window Size
- Early Stopping
- Learning Rate
- Dropout

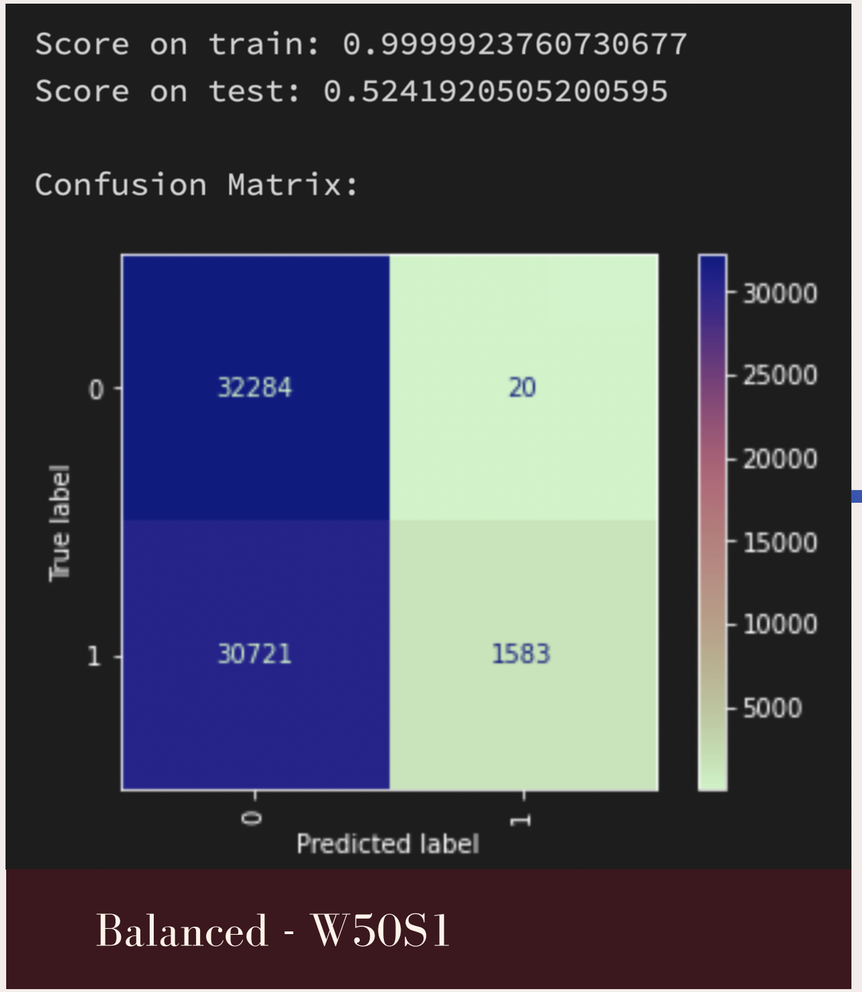
Original Data

- Features to Keep
- Normalization
- Imputation

Random Forest Feature Selection



Unbalanced W50S1



Balanced - W50S1

	Feature	Feature Importance
8	F1__minimum	0.083907
0	F1__sum_values	0.081044
16	F2__maximum	0.077756
2	F1__mean	0.076788
1	F1__median	0.075476
6	F1__root_mean_square	0.073237
17	F2__minimum	0.070889
10	F2__median	0.069379
11	F2__mean	0.057002
9	F2__sum_values	0.055843
7	F1__maximum	0.054174
4	F1__standard_deviation	0.053665
5	F1__variance	0.053024
14	F2__variance	0.041555
15	F2__root_mean_square	0.039783
13	F2__standard_deviation	0.036481
12	F2__length	0.000000
3	F1__length	0.000000

Balanced - W50S1

TS Fresh Feature Selection Results

SMOTE
Oversampled Data

tsfresh.feature_selection
.calculate_relevance_table

drop features with NA

Undersampled Data

Benjamini Hochberg procedure

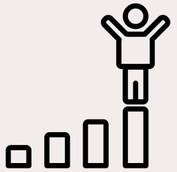
H0: The feature is not relevant and should not be added

H1: The feature is relevant and should be kept

Relevance Table

feature	feature	type	p_value	relevant	rank
F1__has_duplicate_max	F1__has_duplicate_max	real	0.000000	True	1
F1__last_location_of_maximum	F1__last_location_of_maximum	real	0.000000	True	2
F1__median	F1__median	real	0.000000	True	3
F1__longest_strike_above_mean	F1__longest_strike_above_mean	real	0.000000	True	4
F1__root_mean_square	F1__root_mean_square	real	0.000000	True	5
...	...	...	...	...	...
F1__value_count__value_1	F1__value_count__value_1	binary	0.000122	True	101
F1__range_count__max_1__min_-1	F1__range_count__max_1__min_-1	binary	0.000122	True	102
F2__value_count__value_1	F2__value_count__value_1	binary	0.000244	True	103
F2__range_count__max_1__min_-1	F2__range_count__max_1__min_-1	binary	0.000244	True	104
F2__time_reversal_asymmetry_statistic__lag_3	F2__time_reversal_asymmetry_statistic__lag_3	real	0.000375	True	105

105 rows × 5 columns



top 33 features had a p_value of 0!

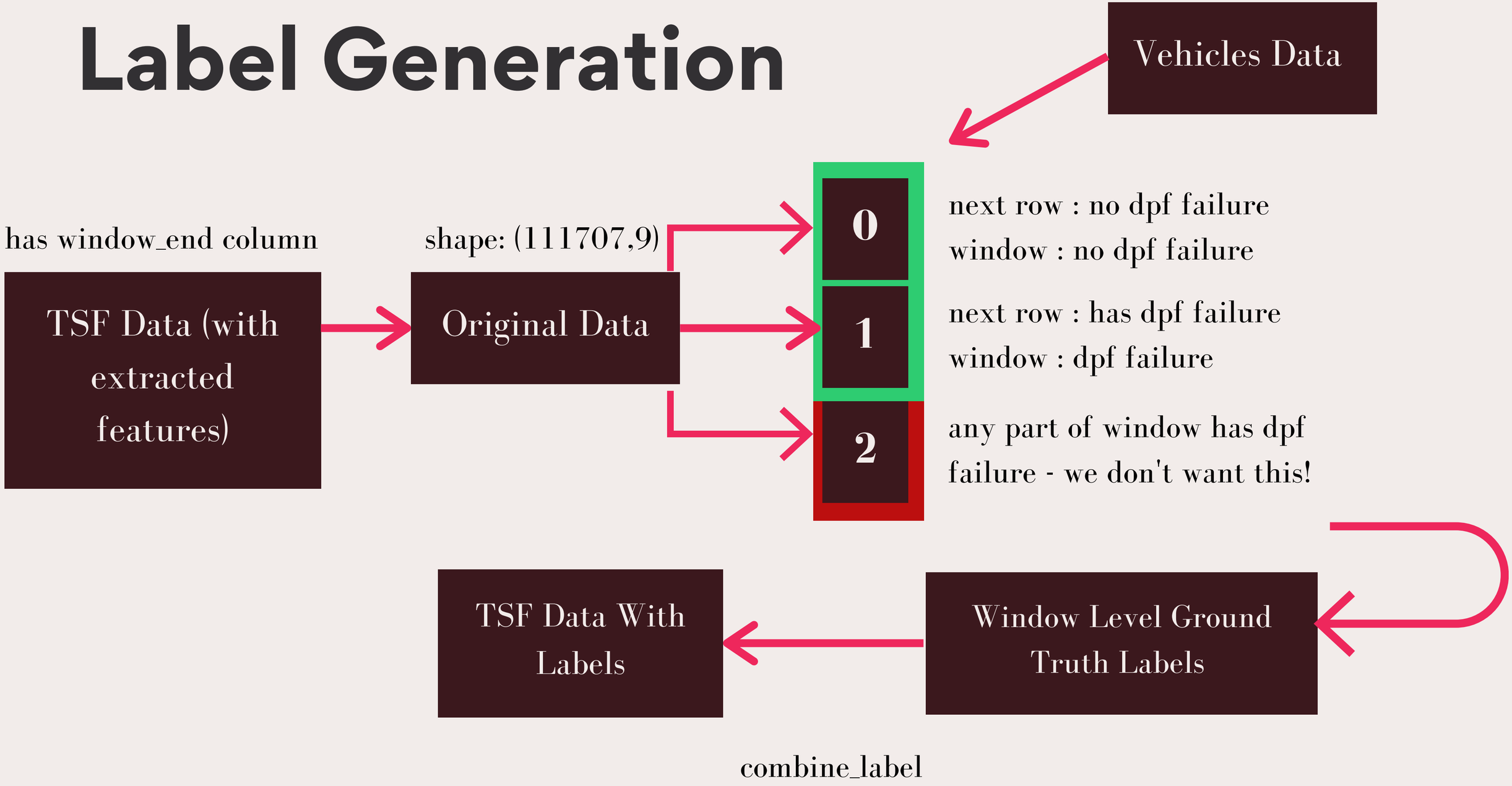
feed top 20 to autoencoder

Relevance Table

feature	feature	type	p_value	relevant
F1__spkt_welch_density__coeff_2	F1__spkt_welch_density__coeff_2	real	0.057590	False
F1__ratio_value_number_to_time_series_length	F1__ratio_value_number_to_time_series_length	real	0.058556	False
F1__variation_coefficient	F1__variation_coefficient	real	0.060909	False
F1__percentage_of_reoccurring_datapoints_to_all_datapoints	F1__percentage_of_reoccurring_datapoints_to_al...	real	0.062433	False
F1__percentage_of_reoccurring_values_to_all_values	F1__percentage_of_reoccurring_values_to_all_va...	real	0.064571	False
...	...	...	...	...
F2__range_count__max_0__min_1000000000000.0	F2__range_count__max_0__min_1000000000000.0	constant	NaN	False
F2__range_count__max_1000000000000.0__min_0	F2__range_count__max_1000000000000.0__min_0	constant	NaN	False
F2__number_crossing_m__m_-1	F2__number_crossing_m__m_-1	constant	NaN	False
F2__number_crossing_m__m_1	F2__number_crossing_m__m_1	constant	NaN	False
F2__count_above__t_0	F2__count_above__t_0	constant	NaN	False

132 rows × 4 columns

Label Generation

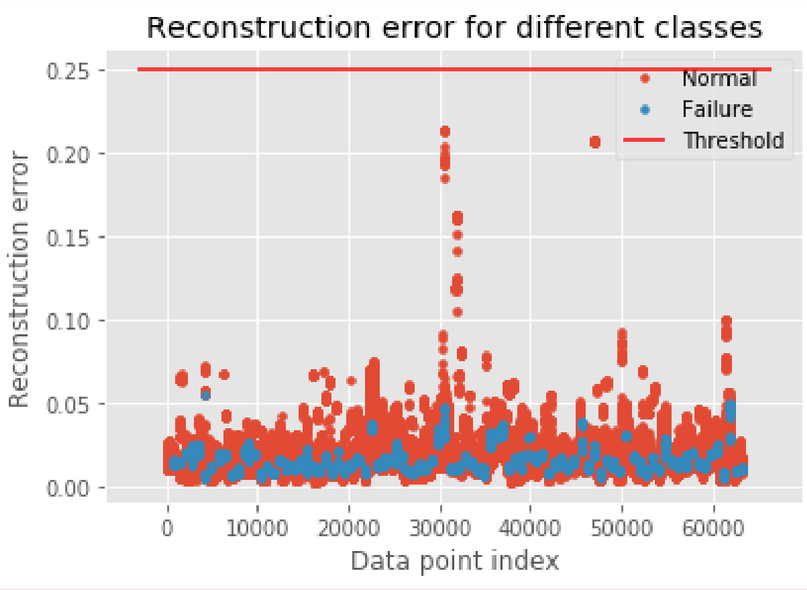


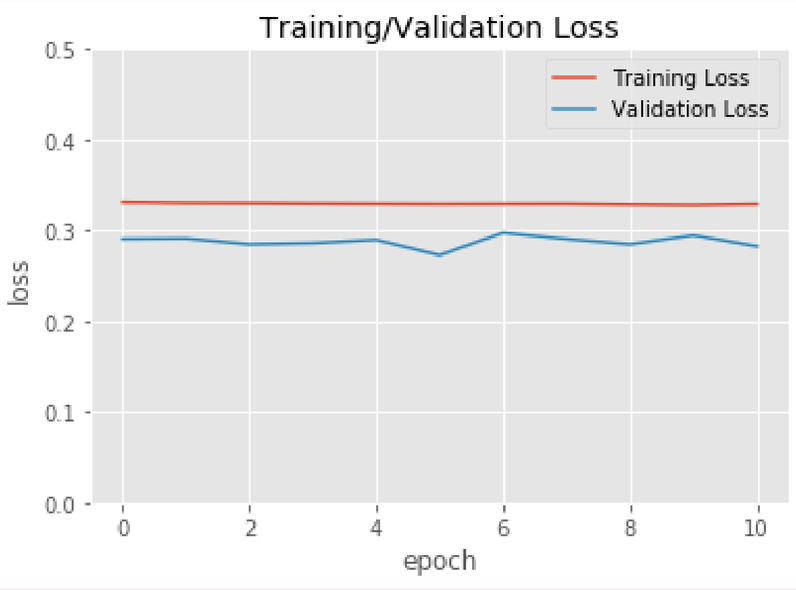
Autoencoder

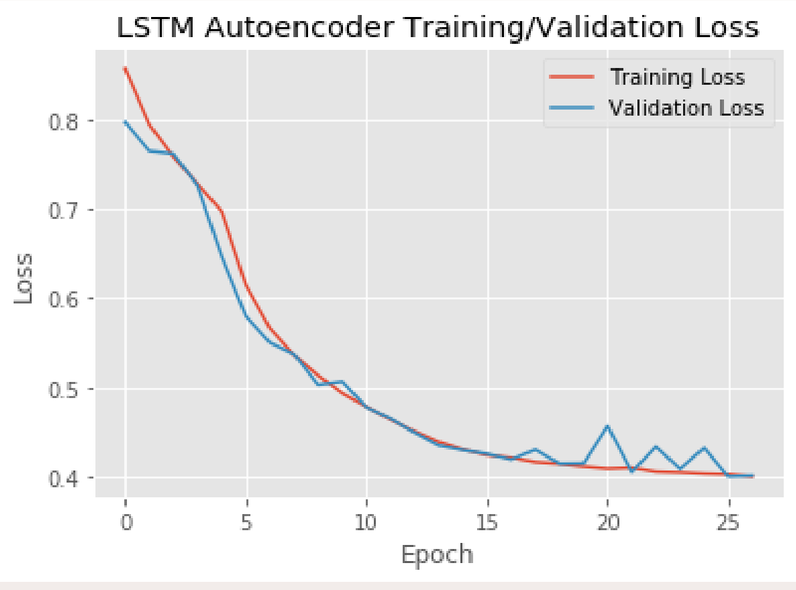
Dense Layers

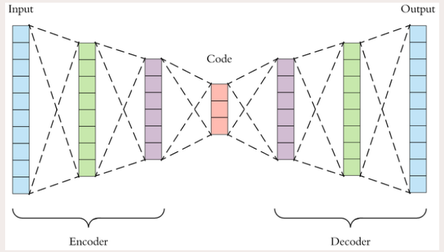
Convolutional Autoencoder

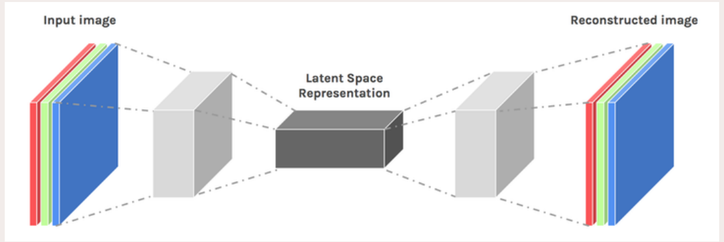
LSTM Autoencoder











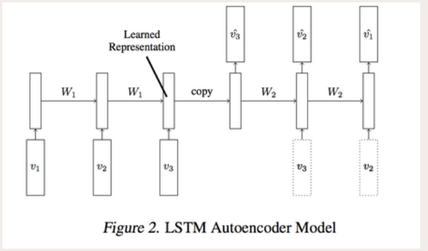
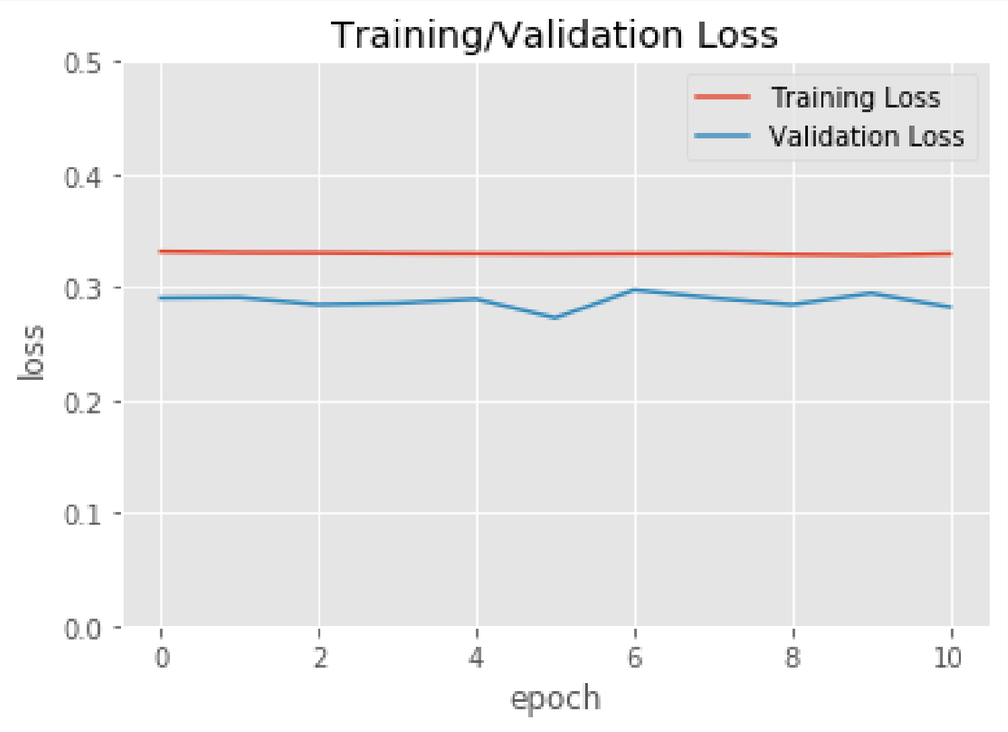


Figure 2. LSTM Autoencoder Model

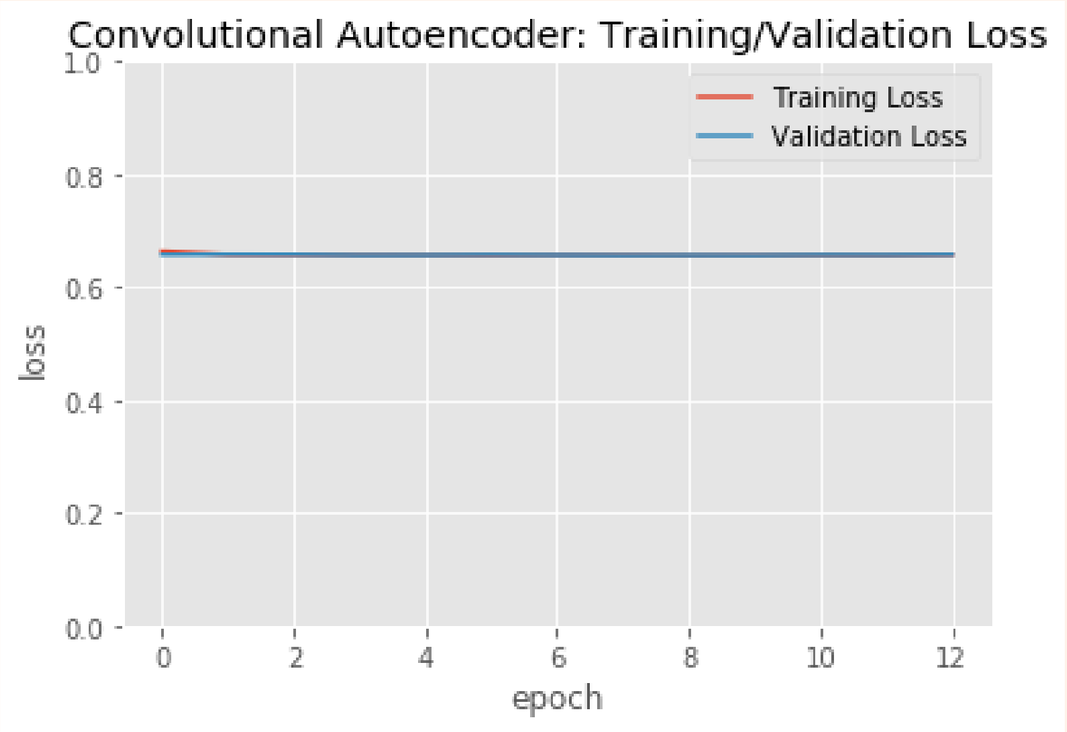
Loss

Convolutional Autoencoder:

Before



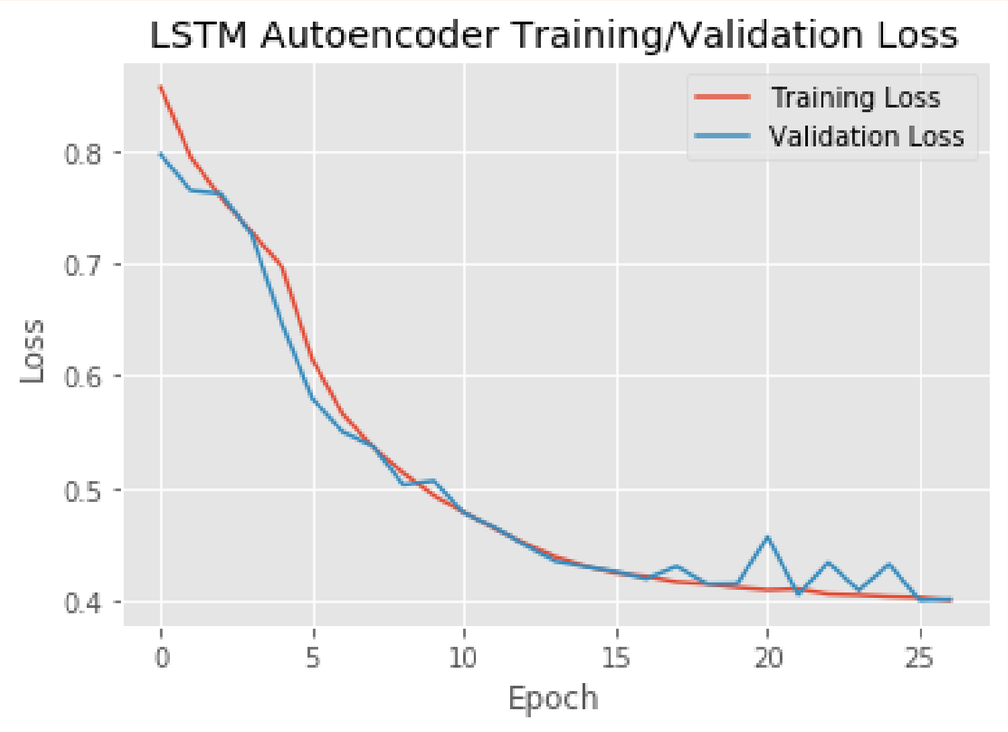
After



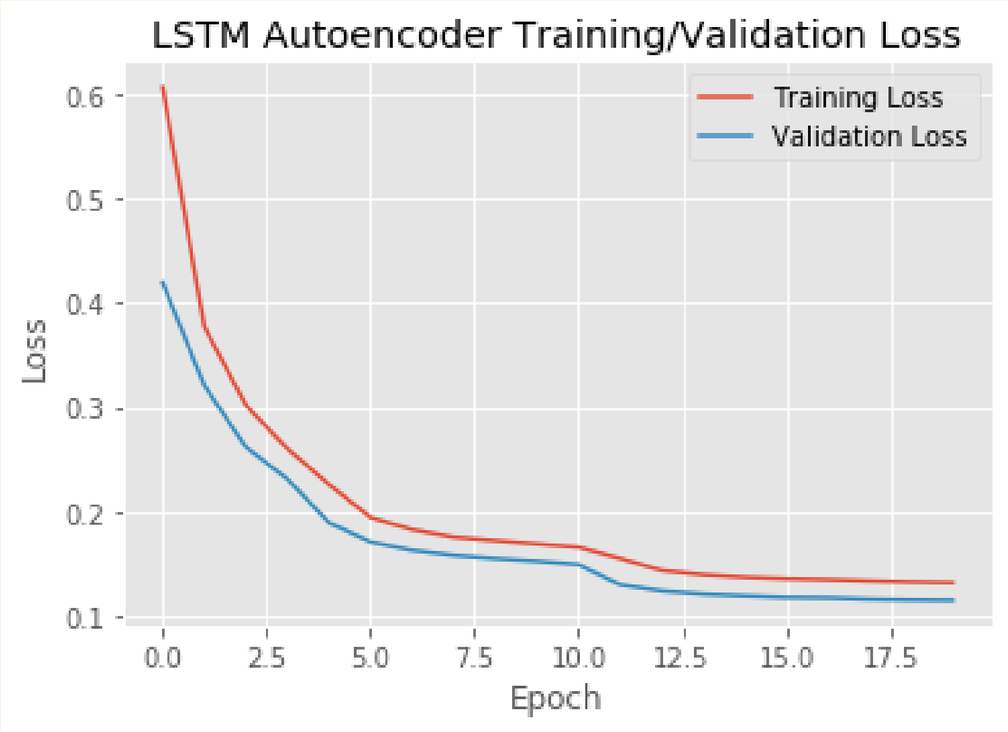
Loss

LSTM Autoencoder:

Before



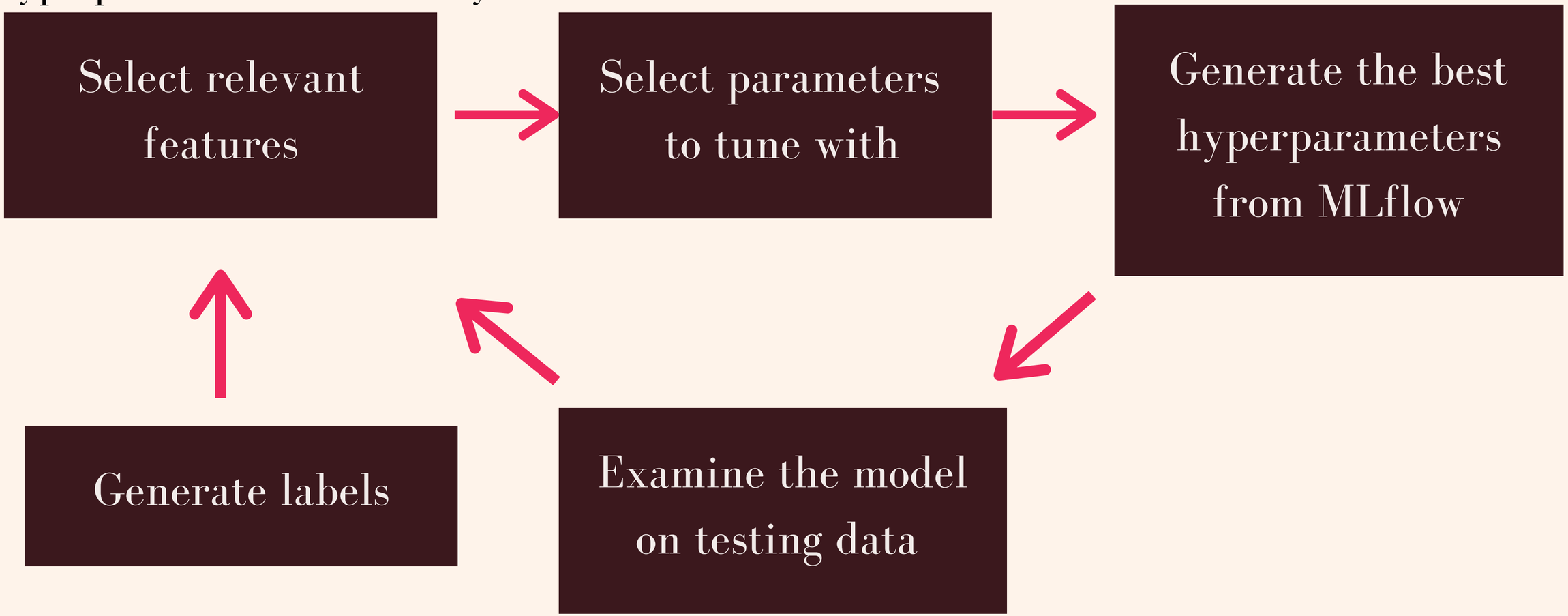
After



MLflow

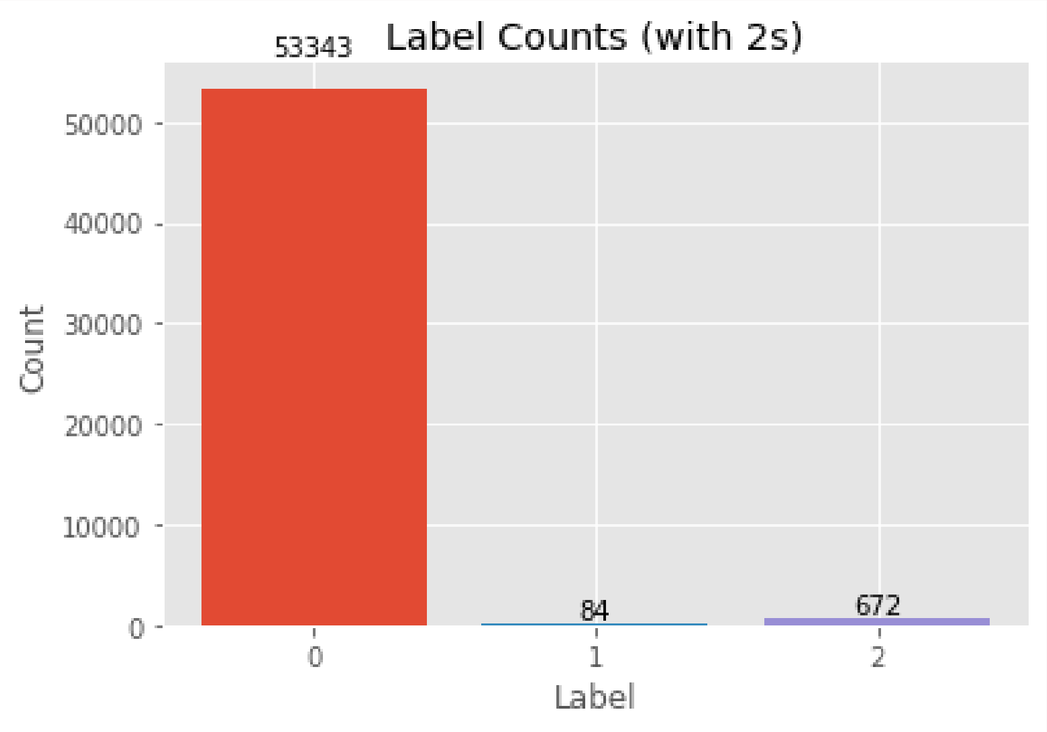
hyperparameter to "manually" tune

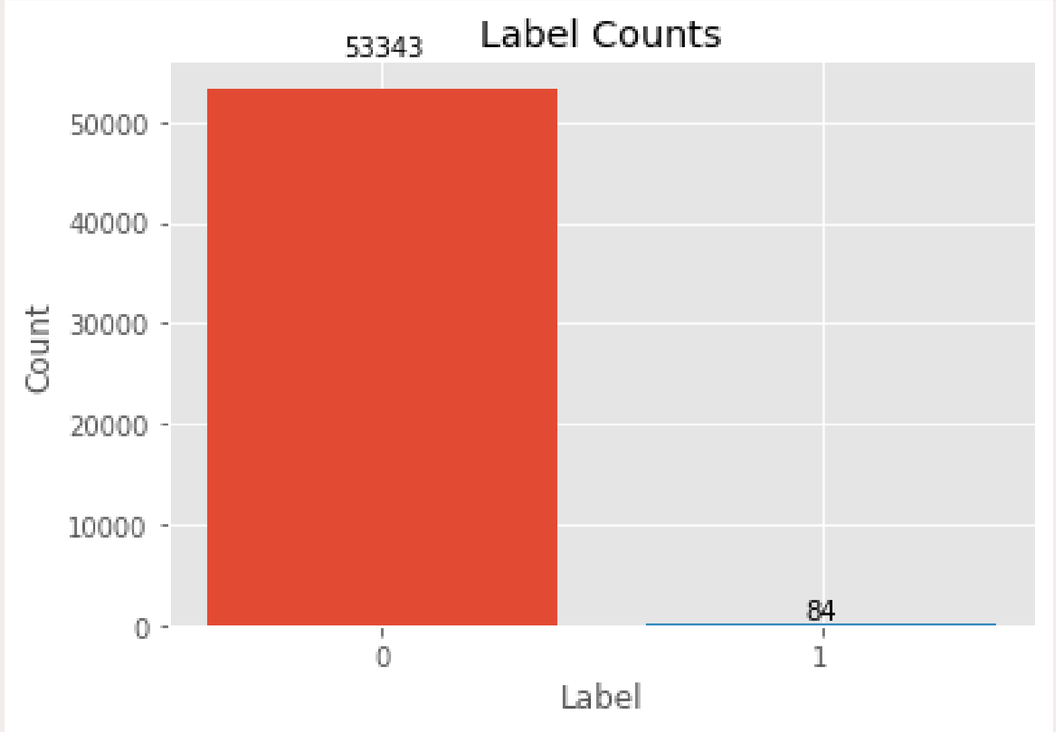
automated tuning with ML ops



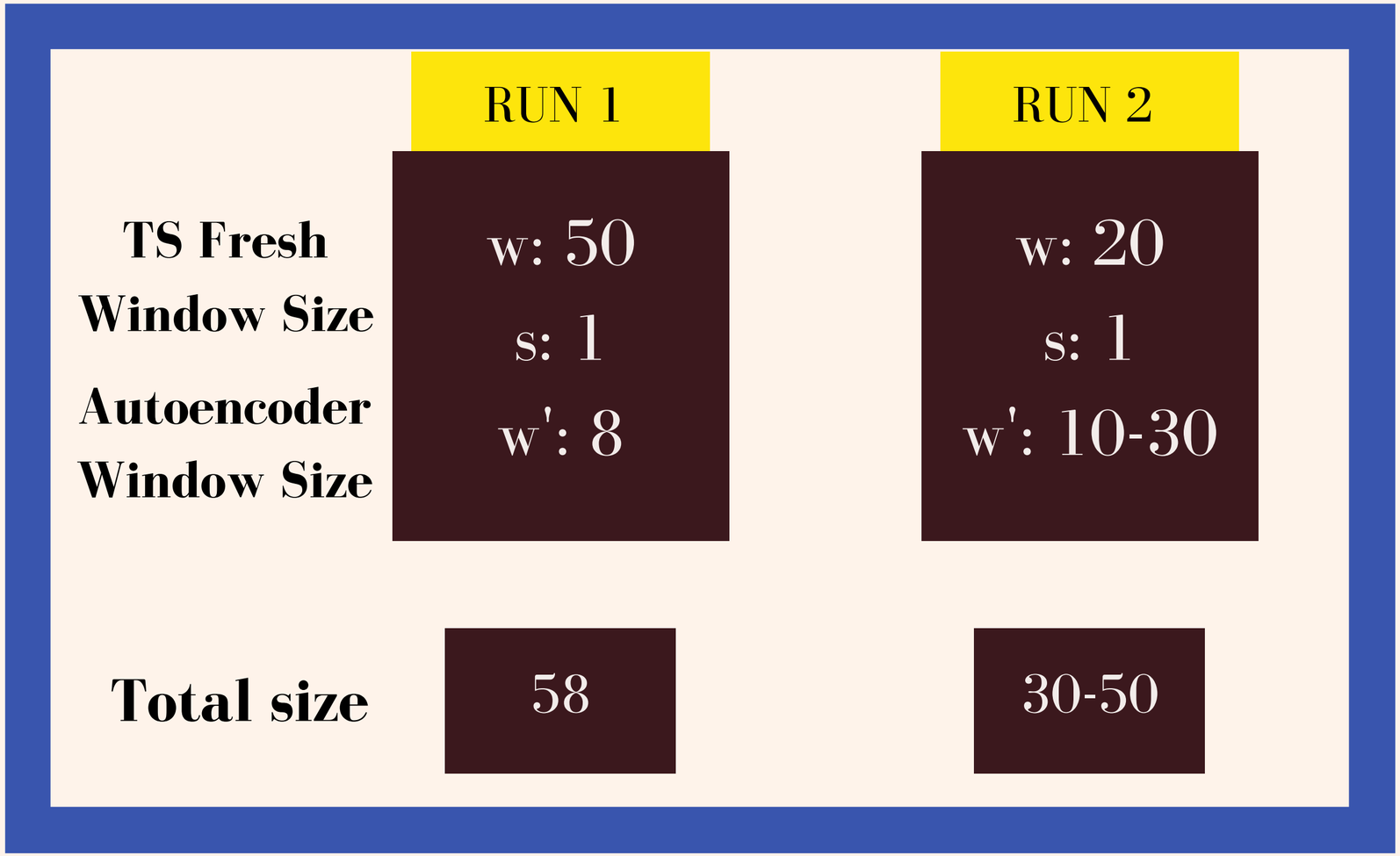
Autoencoder Labels

with a window size of 8





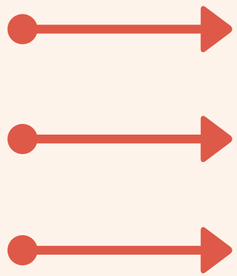
MLflow


RUN 1
RUN 2
TS Fresh Window Size
Autoencoder Window Size
w: 50
s: 1
w': 8
w: 20
s: 1
w': 10-30
Total size
58
30-50

MLflow

Parameters to tune with:

LSTM layer1 units
LSTM layer2 units
epochs
learning_rate
dropout rate
optimizer("Adadelta", "Adam")
window_size



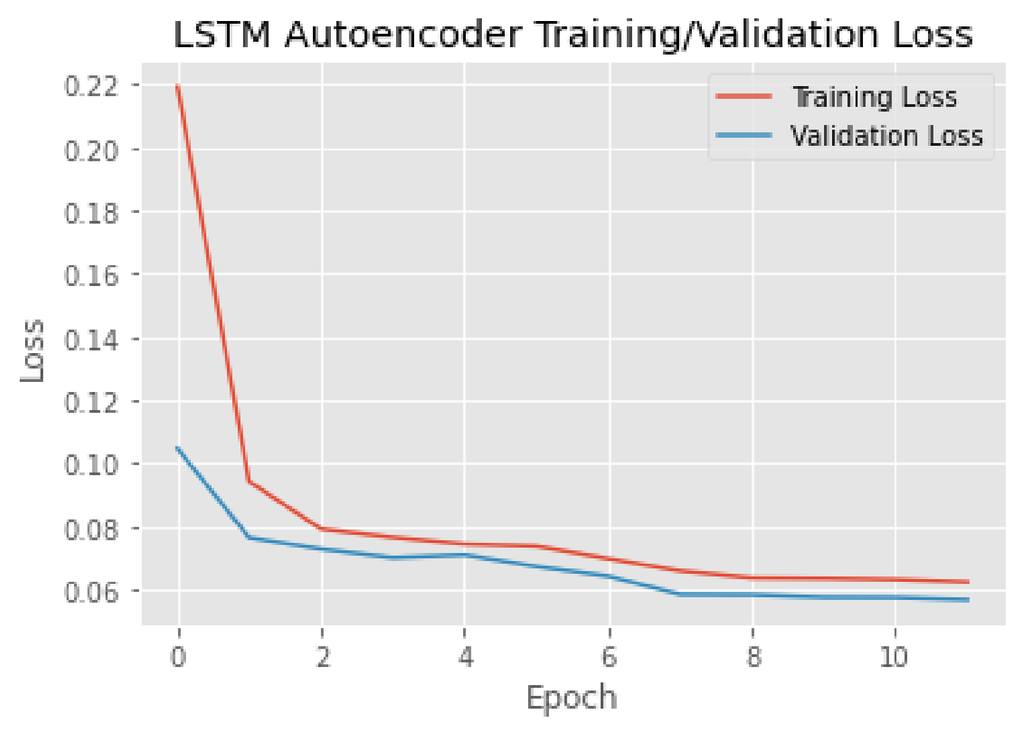
Example output

'dropout': 0.02243656795453065
'epochs': 96.0
'learning_rate': 0.006810209468199 95
'lstm_l1': 14.0
'lstm_l2': 8.0
'optimizer': 'Adam'
'window_size':10

MLflow
results

With w=50+8

'**dropout**': 0.022436567954530 65,
'**epochs**': 96.0
'**learning_rate**': 0.006810209468199 95,
'**lstm_l1**': 14.0
'**lstm_l2**': 8.0
'**optimizer**': 'Adam'

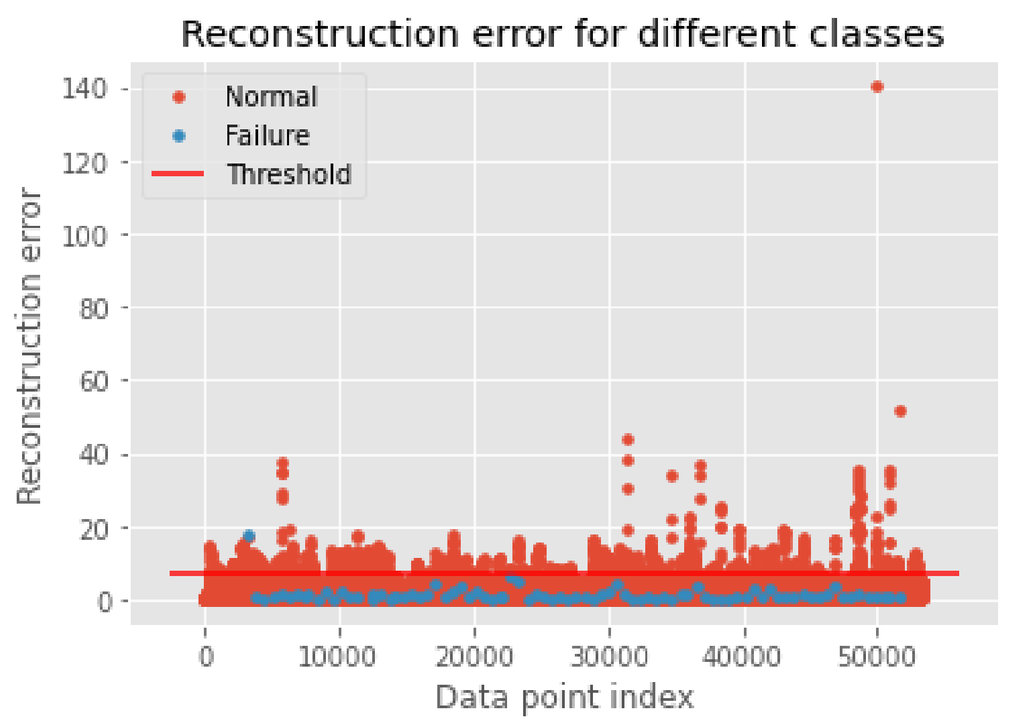


RUN 1

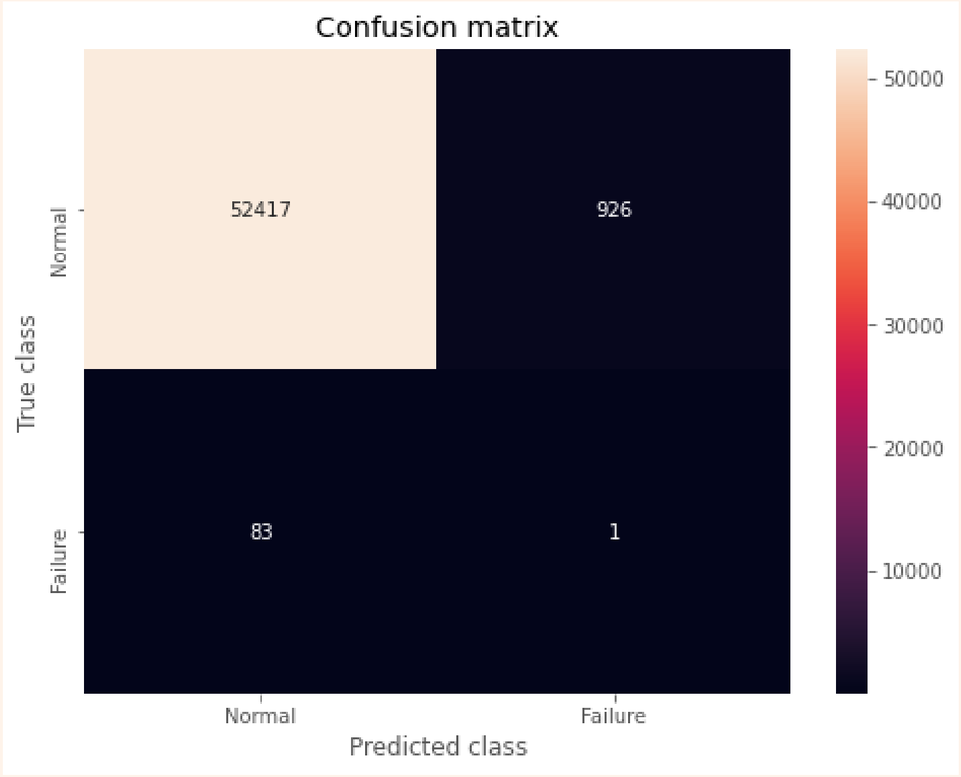
MLflow
results

With w=50+8

'dropout': 0.022436567954530 65
'epochs': 96.0
'learning_rate': 0.006810209468199 95
'lstm_l1': 14.0
'lstm_l2': 8.0
'optimizer': 'Adam'



The data is clustered together - bad sign

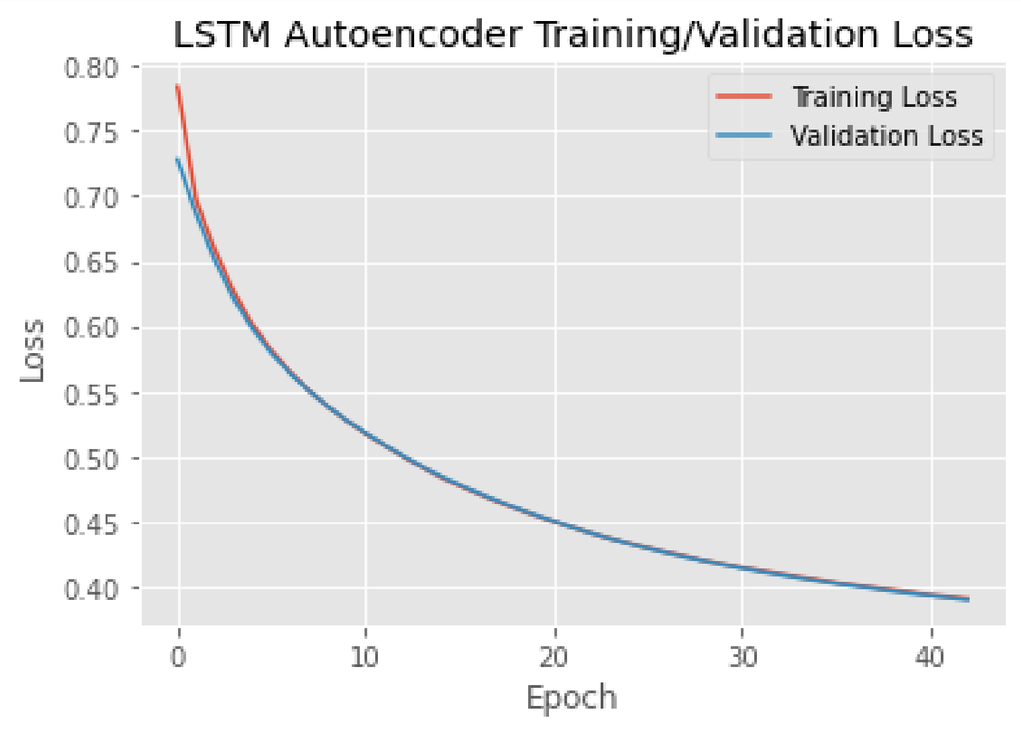


RUN 1

MLflow
results

With w=20+10 ~ 30

'dropout': 0.07031539323259835
'epochs': 55.0
'learning_rate': 0.08271996530819141
'lstm_l1': 32.0
'lstm_l2': 10.0
'optimizer': 'Adadelta'
'window_size': 10.0

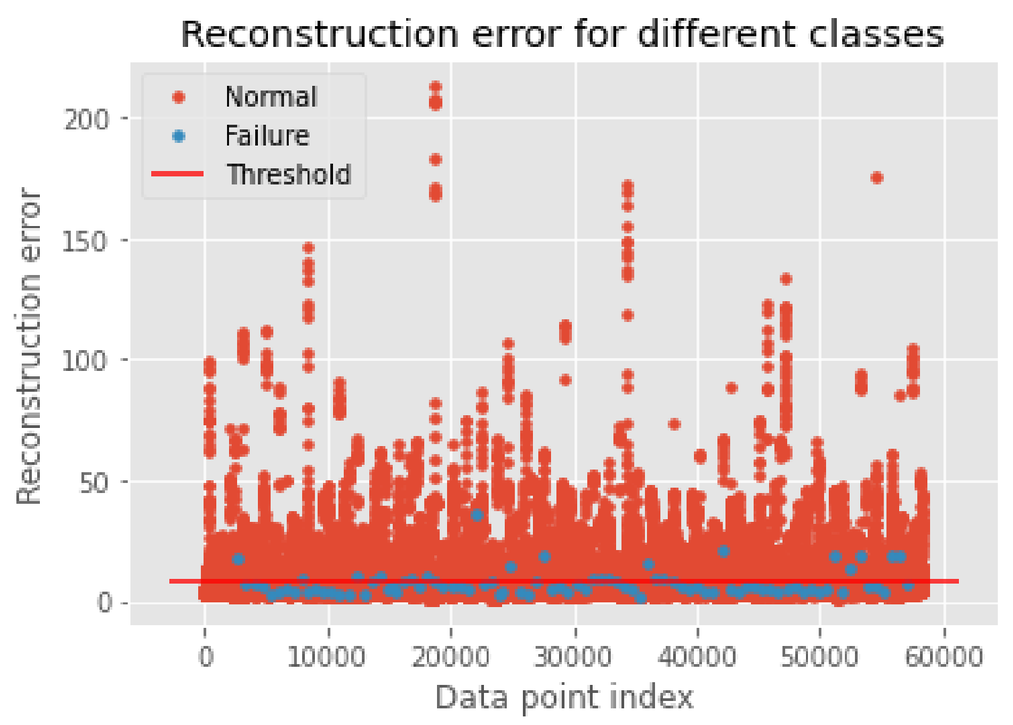


RUN 2

MLflow
results

With w=20+10 ~ 30

dropout': 0.07031539323259835
'epochs': 55.0
'learning_rate': 0.08271996530819141
'lstm_l1': 32.0
'lstm_l2': 10.0
'optimizer': 'Adadelta'
'window_size': 10.0



The data is clustered together - bad sign

RUN 2

Results and Conclusion

Lots of improvements with LSTM Autoencoder

Corrected TS Fresh Selection Method and Pipeline

Finished the complete pipeline (tuning and evaluation)

Non-failure and failure trucks MSE still clustered together

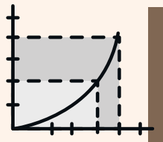
Current Best Recall Scores:

50% and 56%

for DPF nonfailure *for DPF failure*

Summary of Challenges

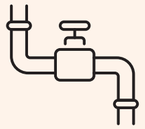
General



Learning Curve

Containerization

Code Productionalization



Pipeline has many moving parts

New Approaches Required

Local and cluster memory - runtime issues

Reproducibility

No Leaks in Pipeline

Next Steps

1. Continue hyperparameter tuning to optimize for recall with Hyper Opt and ML Flow
2. Deal with clustered reconstruction error through more feature engineering
3. Consider using the diagnostic data as a confounding factor
4. Further productionalize pipelines and code to ensure no leakage and have better deployment

Thank You for Listening!

Questions?