Big G Truck project

Torque Titans

Yichuan, Dollada, Zach

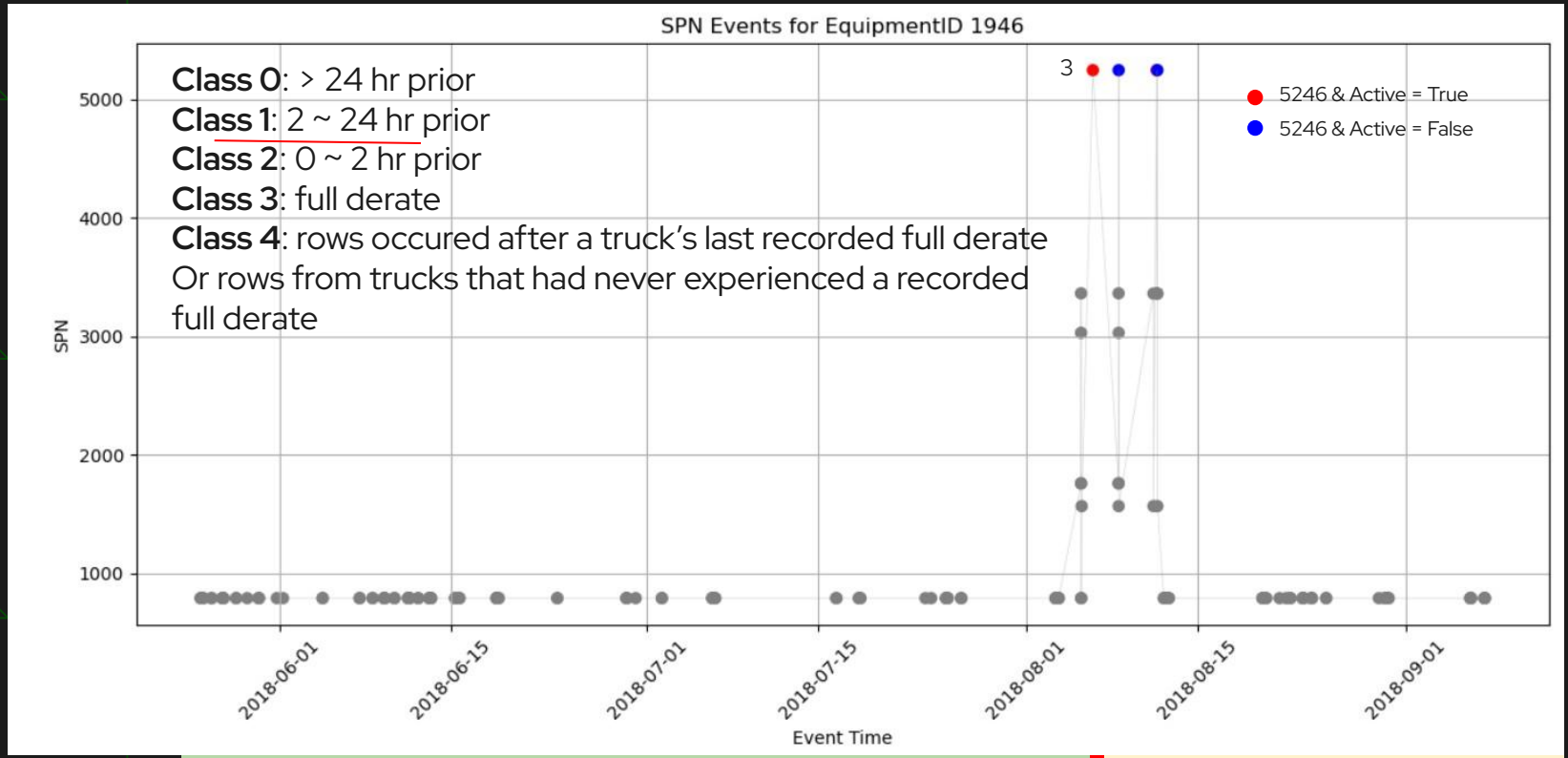
The Problem

- Heavy duty vehicles occasionally run into a full derate
- Full derates happen when the internal computer forces a slow down or service of the vehicle.
- Derates are expensive.
- Is it possible to predict a full derate for a period of time before it happens to avoid extra expenses.

Project Overview

- Clean Data for predictor variables
- Impute/Encode missing values
- Feature choice/Class balancing
- Apply XGBoost model to make predictions

Exploratory Data Analysis



Cleaning Data

Remove Confounding Data

- Removed error logs produced around repair shop areas within a 1 mile radius.
- Disregarded rows where SPN 5246 has active status is FALSE

Section Time Series Data

Sliced the time series data into time interval groups.
These groups are:

- 0 hrs (moment of full derate)
- Less than 2 hours before full derate
- Between 2 and 24 hours before
- Larger than 24 hours

Impute/Encode Missing Values

Use of models to replace missing values with potential real values.

Missing Values + Balancing Classes

Initial Encoding

- Dropped columns with mostly missing data
- Create SPN + FMI column
- Target Encode 'spn_fmi', 'ecuSource', 'LampStatus', 'activeTransitionCount'

Scale & Impute

- Broadly apply Standard Scaler
- Impute values via HistGradientBoostingRegressor

Class Imbalance

- Favored over 24hr period heavily
- Used SMOTE to help balance the minority classes
- Added an extra number classifier to identify trucks that did not have a full derate in the data

XGBoost

Labeled the time intervals between 0 and 4

- 4 is an artificial number (9999999) to initiate a time delta column
  - (1) Created for entries from trucks that did not experience a recorded full derate
  - (2) entries that occured after a truck's last recorded full derate
- 3 references the time of a full derate
- 2 represents two hours before a derate
- **1 represents time between 2 and 24 hours prior to full derate**
- 0 represents time delta greater than 24 hours

Time Delta Category	Precision	Recall	f1	Support
0	0.01	0.05	0.02	2044
1	**0.05**	**0.49**	**0.09**	**292**
2	0.00	0.05	0.00	44
3	0.90	0.99	0.94	101
4	0.98	0.90	0.94	109538
Accuracy			0.88	112019
Macro avg	0.39	0.50	0.40	112019
Weighted avg	0.96	0.88	0.92	112019

ML Pipeline Construct

```
ct0 = ColumnTransformer(
        [('target_encoder', TargetEncoder(handle_unknown='ignore'), ['spn_fmi',
                                                                     'ecuSource',
                                                                     'LampStatus'
                                                                    ]
         )],
        remainder='passthrough')



xgbc_pipe = Pipeline(
    steps=[
        ('preprocessor0', ct0),
        ('StandardScaler', StandardScaler()),
        ('imputer', IterativeImputer(estimator = HistGradientBoostingRegressor(verbose=2, random_state=434), max_iter=1, random_state=343)),
        ('smote', SMOTE(random_state=344)),
        ('xgbc', XGBClassifier( enable_categorical=True,
                                eval_metric='mlogloss',
                                objective = 'multi:softmax',
                                num_class = 5,
                                device = "cuda",
                                random_state = 535
                              )
        )
    ]
).fit(X_train, y_train)

y_pred = xgbc_pipe.predict(X_test)
```

✓ 2m 32.0s

```
X = TrainPre2019.drop(columns = [   'Unnamed: 0',
                                    'Unnamed: 0.1',
                                    'RecordID',
                                    'ESS_Id',
                                    'actionDescription',
                                    'spn',
                                    'fmi',
                                    'faultValue',
                                    'MCTNumber',
                                    'Latitude',
                                    'Longitude',
                                    'is_fullderate',
                                    'is_fullderate_group',
                                    'time_to_next_SPN5246',
                                    'ServiceDistance',
                                    'EventTimeStamp',
                                    'eventDescription',
                                    'EquipmentID',
                                    'LocationTimeStamp',
                                    'EquipID_Index',
                                    'time_interval_to_SPN5246_class',
                                    'MCTNumber',
                                    'FaultId',
                                    'ecuSoftwareVersion',
                                    'ecuSerialNumber',
                                    'ecuModel',
                                    'ecuMake'
                                ]
                      )
y = TrainPre2019['time_interval_to_SPN5246_class']
```

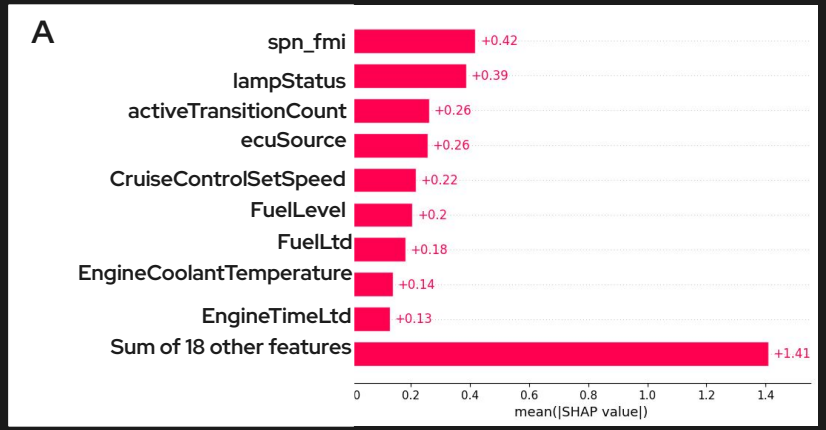
✓ 0.0s

External Testing/Validation, with Data from 2019+

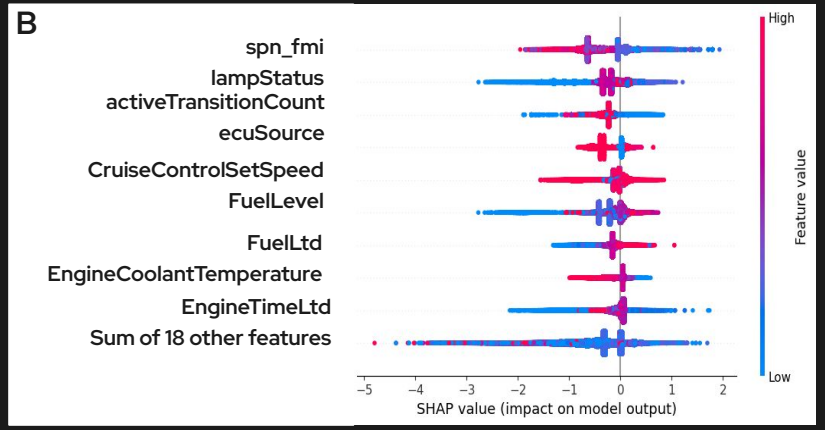
		precision	recall	f1-score	support
>24 Hrs: Cat	0	0.01	0.05	0.02	2044
>2~24 Hrs: Cat	1	0.05	0.49	0.09	292
>0~2 Hrs: Cat	2	0.00	0.05	0.00	44
Full Derate Encounter: Cat	3	0.90	0.99	0.94	101
No recorded Derate: Cat	4	0.98	0.90	0.94	109538
accuracy				0.88	112019
macro avg		0.39	0.50	0.40	112019
weighted avg		0.96	0.88	0.92	112019

*** Yielded Recall Scores Higher Than Imputation by XGB Imputer or Gradient Boosting Regressor ***

Feature Importance & Impact Using SHAP

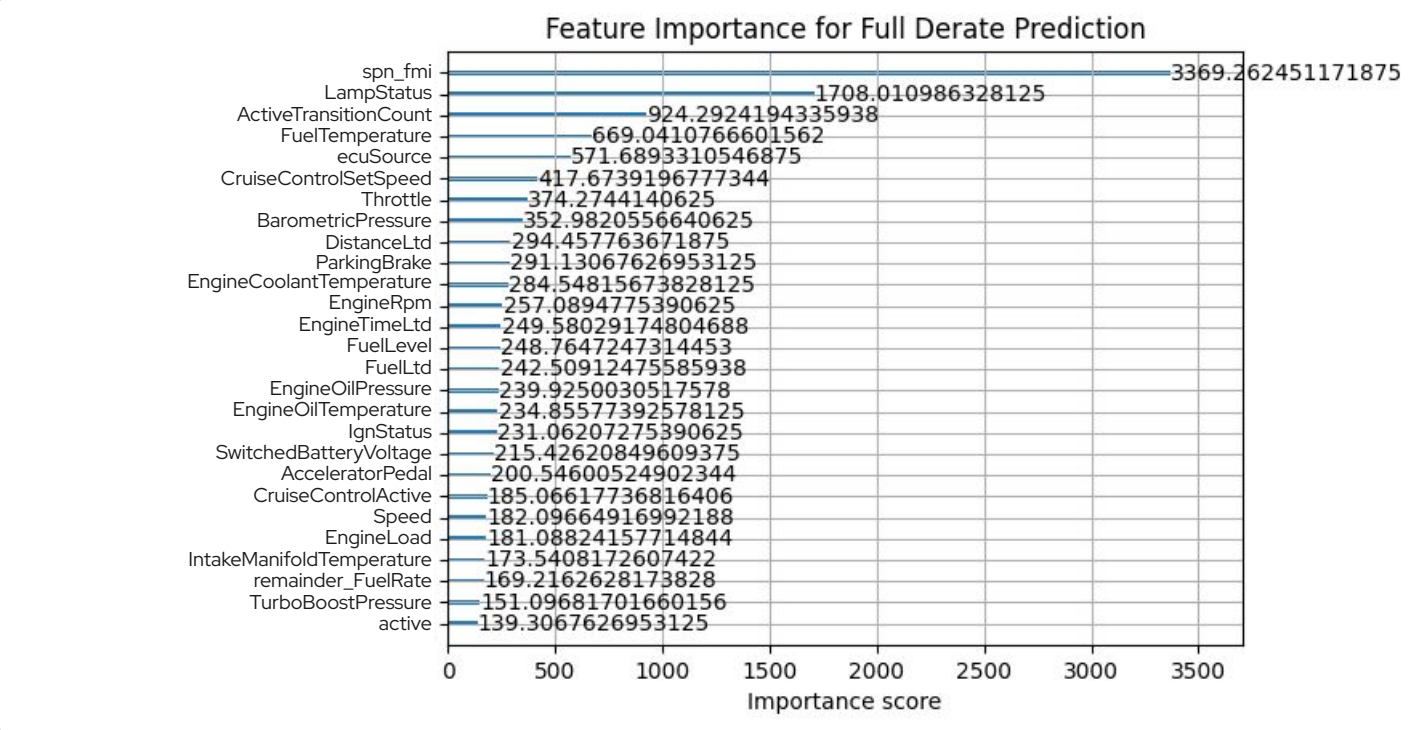


A. The average importance of each feature in the model

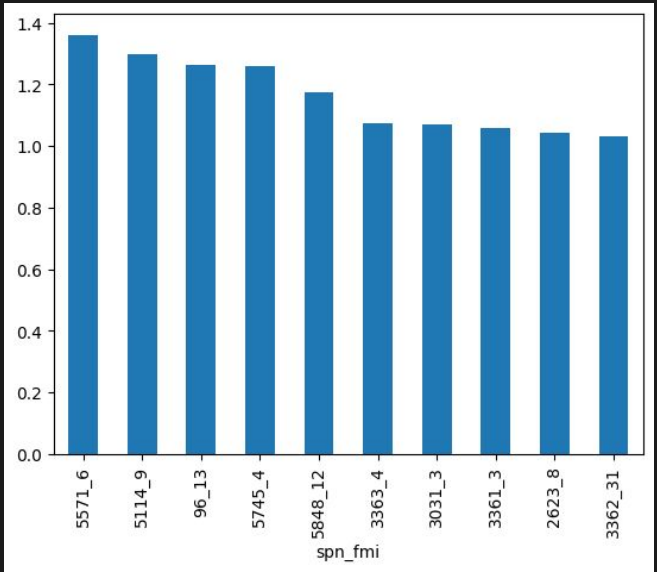


B. the beeswarm plot illustrates how high or low values of each feature impact predictions

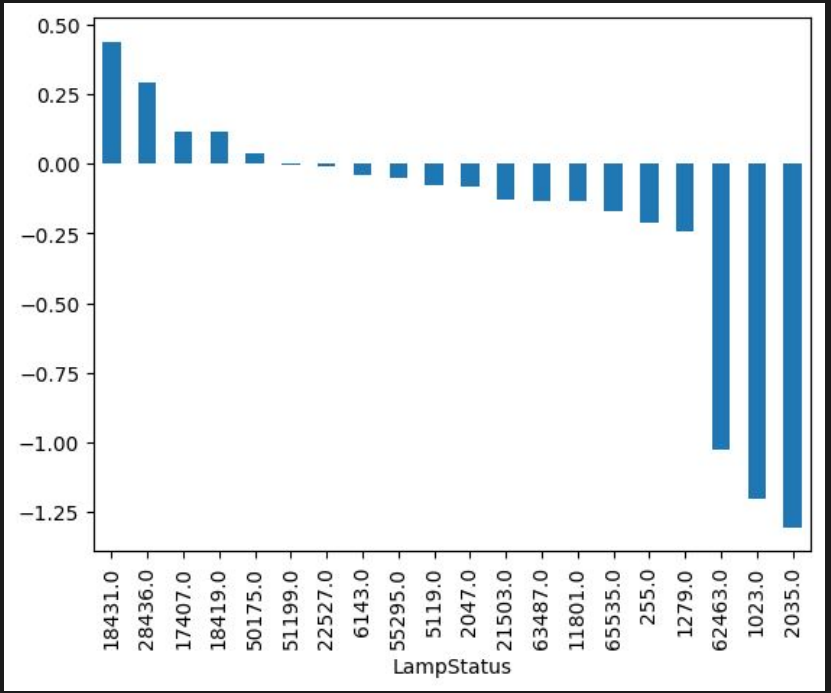
Feature Importance Quantification using XGBoost's Calculation of gain in gradient



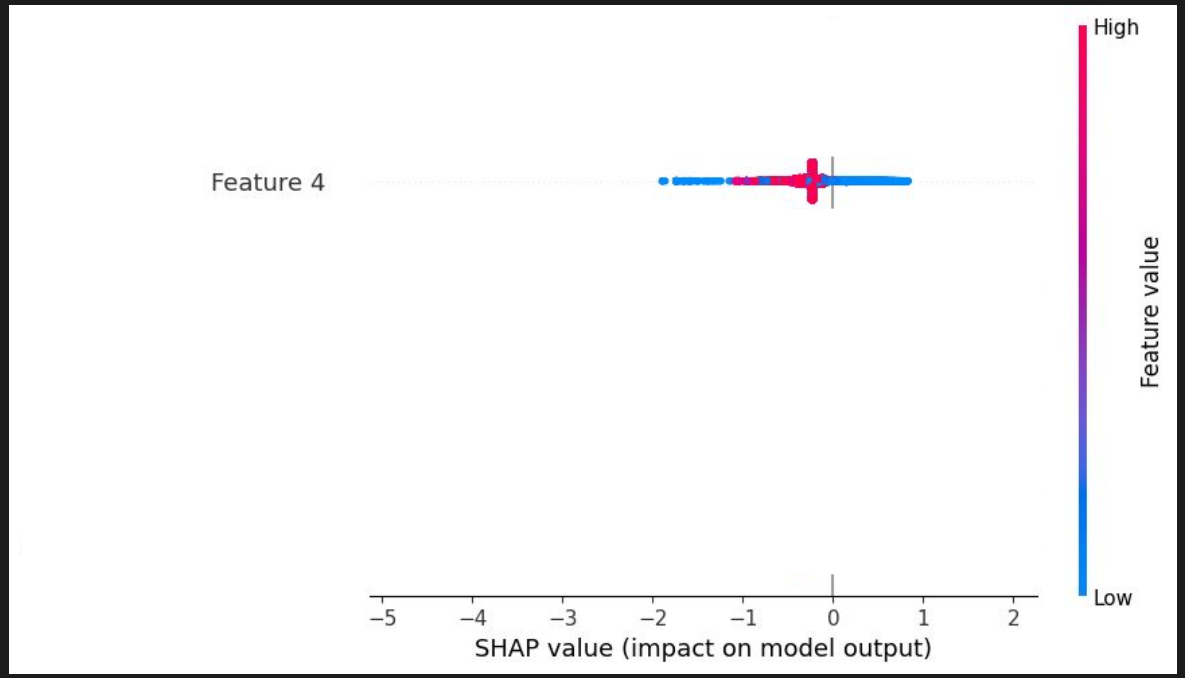
Most Impactful Feature: SPN_FMI



2nd Most Impactful Feature:
LampStatus

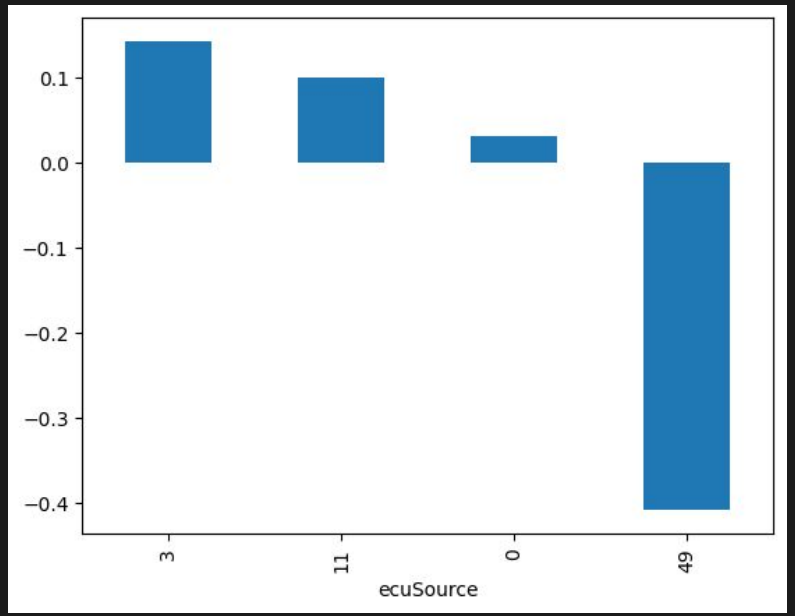


3rd Most Impactful Feature:
Fuel Level



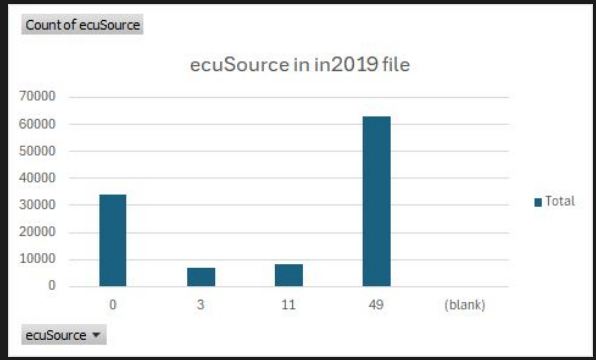
4th Most Impactful Feature: ecuSource Codes

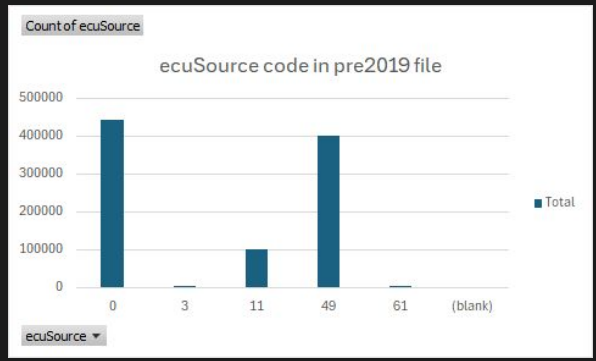
ecuSource code's impact on SHAP Value (log odds)



Source Address	Address Description
0	Primary Engine Controller (CPC, ECM)
3	Primary Transmission Controller (TCM)
11	Brakes - System Controller (ABS)
49	Cab Controller - Primary (MSF, SHM, ECC)
61	Exhaust Emission Controller (ACM)

Though highly pertinent, Code 61 (Exhaust Emission Controller) did not show up in **Testing** (intra-/post-2019) Dataset





https://static.nhtsa.gov/odi/tsbs/2018/MC-10141869-9999.pdf

Results

Tabulated on a day-to-day basis, per derate per truck:

- 35 true positive predictions
- < 42 false positives predictions
  - Max penalty count of 1 per day, per EquipmentID_Index
  - Summed over the course for *n* days
  - Hence we have tabulated ***more*** counts of False Positives than Actual # of False Positives
- Net savings : >$119,000
- Prediction alert benchmarked against the period of 2~24 Hours ahead of a full derate: not too early / late

```
Summary of Results:
Total Count of EquipmentID_Index by Full Derates or Lack Thereof: 787
Total Savings: $140,000.00
Total Losses: <$21,000.00
Net Savings or Losses: >$119,000.00

Sample equipment results:
   EquipID_Index  Has_GroundTruth_TimeIntervalClass_1  \
0    105338729_0                                False
1    105344451_0                                False
2    105349576_0                                False
3    105370255_0                                False
4    105411909_0                                False
5    105427203_1                                False
6    105437340_0                                False
7         1688_0                                False
8         1698_0                                 True
9         1698_1                                False
10        1698_2                                False

    Has_Predicted_TimeIntervalClass_1_Where_GroundTruth_TimeIntervalClass_1  \
...
8      4000       0  4000
9         0       0     0
10        0       0     0
```

Future Work

Future Dev 1	Try **LSTM** (Long-/Short-Term Memory) and **TCN** (Temporal Convolutional Neural Network): to learn then predict on sequence of FaultID triggering events
Future Dev 2	**Distance** Calc based on Latitude & Longitude: Between the locale where each Fault Code popped up and the subsequent Full Derate. Alternatively, would be nice to have **odometer** data.
Future Dev 3	Optimize model to run quickly and efficiently (i.e. less demanding of computational resources), train on/predict using fewer features. E.g. LightGBM, maybe fewer *for* loops
Future Dev 4	Try other tree-based models, e.g. YDF, MICE Forest (Light GBM combined with Random Forest) for Imputation + Hyperparameter Tuning. Hopefully it won't overfit, as XGBoost Imputer did.

Acknowledgements

Traversing on shoulders of giants:

Instructors	**Michael** & Alexa: for their excellent teaching & elucidation of ML concepts through this project
Billy's Team	For sharing Haversine code that removed rows at/near Service Center locales
Each Team Member	For synergistic contributions & collaboration

Questions?