A Machine Learning Approach to Diesel Engine Health Prognostics using Engine Controller Data

Steve Nixon, Ryan Weichel, Karl Reichard, James Kozlowski

Applied Research Laboratory, Pennsylvania State University, State College, PA, 16801, USA

sxn5077@arl.psu.edu, rtw141@arl.psu.edu, kmr5@arl.psu.edu, jdk173@arl.psu.edu

ABSTRACT

Many military assets such as surface ships and ground vehicles use diesel engines as their prime movers, and accurately estimating remaining useful life has a high value for enabling predictive maintenance and improving fleet logistics. Most of these diesel engines are already equipped with an array of sensors and digital data busses to support the function of the integrated electronic control module (ECM). There are cost advantages to developing predictive analytics and prognostics using existing embedded sensors. This paper describes a hybrid approach to predictive capabilities that utilizes multiple techniques for the implementation of embedded prognostics using existing sensors. One of the challenges is the fidelity of the data. This paper describes an automated approach to feature and classifier selection for hybrid prognostics. Maintenance records with associated diesel engine sensor data for several different engine classes were acquired, which enabled the training data sets to be organized by failure modes. To help prevent false positives, some filtering of the maintenance logs was required to only include those records likely to be associated with the selected failure mode sensor data sets. The classifier-based, data-driven approach essentially maps multiple channels of the sensor data into subspaces trained to classify multiple distinct failure modes. The intent of this step is to enable fault isolation by quantitatively determining which failure mode class the data best fits statistically. The remaining useful life estimate is provided by tracking the temporal path of the data from the healthy engine classification to one of the known failure mode classes using engine load-hours as the metric for the prognostics.

1. BACKGROUND AND DATA ANALYSIS APPROACH

A recent study of trends in prognostics research showed a significant increase in the use of machine learning techniques, particularly deep learning techniques, starting in about 2007 (Bernardo, 2017). Neural networks (NN) and other techniques which can be classified as machine learning (ML) have been applied in system health monitoring since the 1990s. Japkowicz, Meyers and Gluck (1994) reported on the use of neural network techniques for novelty detection, and the technique was applied to the detection of faults in helicopter gearboxes. While there are many references to the use of neural network techniques in the literature, the connectionist models referenced in the paper are forerunners of today's deep learning techniques. In 2006 Hinton, Osindero, & Teh (2006) introduced deep learning techniques that changed the way neural networks are structured and trained. Further advancements were made in the late 2000s (Deng et al., 2009) with significant demonstrations and applications beginning to appear in 2011. A recent trend has been to focus on the development of prognostic algorithms using low bandwidth sensor data (Grosvenor et al., 2014), such as that available on vehicle control and sensor busses.

The primary objective of this work was to evaluate the feasibility of using existing health monitoring data, originally intended for consumption by physics based models and subject matter experts, for machine learning based prognostics algorithms. The discussed techniques, however, are not specific to diesel engines. Data logs from engine management sensors leading up to specific component failure events are grouped together to form a collection of Unscheduled Maintenance Events (UMEs). These data histories are used to train a ML classifier with the goal of identifying trends in the sensor data that can be correlated with engine component health. The hypothesized ML classifier can then recognize similar trends in new data from operational engines, and provide prognostic estimates on engine component health.

As a means for accomplishing this, a secondary objective was established to implement a software framework that enables the ingestion, preparation, and processing of maintenance records and sensor data through a variety of machine learning techniques. The result is automated generation and validation of classifiers that predict the health of the system relative to the state in which maintenance is required. The subsequent evaluation of these techniques' performance is reported.

1.1. Data Sources

The data used in the following approach originates from two independent databases – one storing engine maintenance records, and one storing control and monitoring system sensor measurements.

The sensor measurements used by the engine management computer are logged by a third party data acquisition system for the purpose of traditional sensor based diagnostics and fault detection. These logs are periodically uploaded from the engine's location to a separate database for record keeping. The sensor data used in this approach is periodically captured, low-bandwidth (sample rate) data, intended to be representative of steady-state engine operation.

1.2. Data Analysis Framework

The framework is a group of scripts developed in Python 3.6 that automates the process of natively interfacing with the database servers, preparing the data, training the classifiers, and scoring their performance. The program accepts configuration files that describe database connection information, a list of UMEs to use for training/test, and a list of preprocessing parameter and ML classifier parameter configurations to be evaluated.

1.3. Limitations and Scope

The training approach described in this paper attempts to correlate multiple engine sensor data trends with each other, according to known points in time where engine component failures occur. It is understood, if not expected, that there may be more than one component demonstrating detectable degradation at any given time. This approach assumes the fault signatures of each failure mode are sufficiently independent from one another such that a ML classifier trained to detect a specific failure mode's trend will not be sensitive to an alternate failure modes' trend. Coupling between trends is the subject of planned future efforts.

2. DATA PREPARATION

Supervised ML-based classifiers require labeled data for training, where each data observation has an associated class label. In this case, each observation is a vector in the n-dimensional space defined by a combination of n-sensors, and the class label corresponds to how severely degraded the engine is believed to be. To obtain labeled training data for a specific failure mode, first a group of similar UMEs must be identified from the maintenance records. This list of UMEs represents specific dates in which a component of interest failed. Next, sensor data histories ending with the failure date are extracted from the database. The resulting observations are then split into discrete degradation classes according to how close in the data history each observation is to the failure point. Finally, the data undergoes normalization, missing value handling, and additional feature engineering, before it is finally ready for ingestion by the ML training algorithm.

2.1. Maintenance Record Fault Identification

UME list generation for building training data began with analysis of numerous maintenance records collected from the population of engines in consideration, and followed a similar process to that used to develop the system reliability and condition based maintenance strategy for the platform (Banks et al., 2008). First, an appropriate approximation to "failure mode" was created by combining several fields among the maintenance records in a database, allowing records to be grouped by failure mode. After grouping records in this manner, a degrader analysis could be performed. Using Pareto analysis principles, the most damaging failure modes were determined based upon numerous metrics. The resulting list identified the failure modes which, if predicted, would make the most significant impact in increasing engine uptime. These became the focus of the effort. The goal then became finding ways to group UMEs together for the purpose of constructing training data sets comprised of sensor data associated with component failures that occur in a similar fashion.

2.1.1. Maintenance Record Fault Mode And Pareto Analysis

The maintenance records provided for the purposes of creating datasets contained engine identification information, maintenance action details, and other tracking information. The primary challenge was to extract the specific failure mode which prompted the maintenance action recorded, if it existed. In addition to maintenance actions corresponding to equipment breakdown, the population of maintenance records included inspections, repairs due to human error, and minor maintenance actions unrelated to the operation of the engines in question which all needed to be filtered out. All maintenance records also included an opened date, a closed date, and a field describing the criticality of the maintenance action. The criticality field broadly defined whether the engine was operational or not until the maintenance could be completed. Records coded "Critical" or "Major" accounted for 5% of all maintenance actions – only these maintenance actions resulted in engine downtime. After confirming with the customer that this was a valid approach, the opening and closing dates were combined to create a “repair time” field. By further combining this repair time field with the criticality description field, “downtime” could be calculated.

All maintenance records were coded by the maintainer into one or more of 95 possible work breakdown structure codes. Also obtained was a text description of each work breakdown code used, allowing these codes to be human-interpretable. See Figure 1 for a Pareto analysis of the downtime associated with these different work breakdowns, and Figure 2 for a Pareto analysis of the number of occurrences of various work breakdown structures.

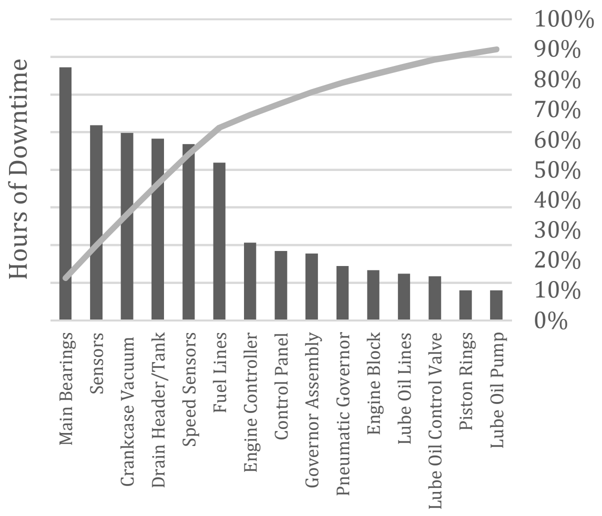


Figure 1: Pareto Chart, Maintenance Event Code vs Downtime

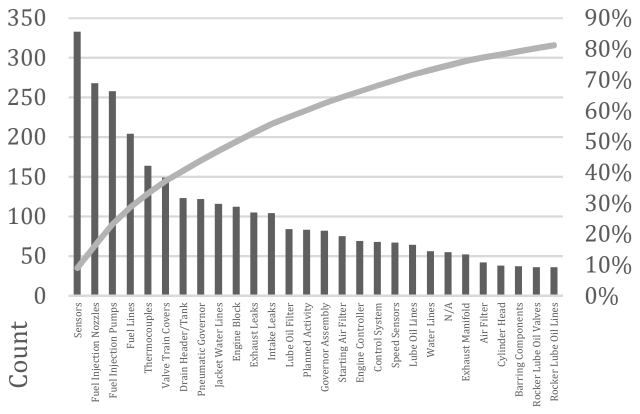


Figure 2: Pareto Chart, Maintenance Event Code vs Counts

Further analysis of records within specific codes revealed that many of the work breakdown structure codes were not specific enough for our purposes if used alone. For example, while UMEs associated with the code for main bearings are sufficiently similar, there were several significantly dissimilar failure modes represented by the fuel injection nozzle code. This field was also prone to misclassifications and false positives.

Records also list the specific parts consumed by the maintenance actions, which offers another dimension of comparison in addition to the work breakdown structure codes. Many records listed several parts consumed, while some listed dozens of parts as consumed. This is reasonable to expect, as even simple repairs may require multiple consumables. This does, however, create issues in grouping records, as the same failure mode may require slightly different low value consumables depending on the exact details of the failure, including how far it had progressed. One solution to this is to find a “primary” or “source” part number, ideally the part number corresponding to the part originally responsible for the maintenance action. This kind of information regarding a specific maintenance action most likely is impossible to gather and was not recorded in the maintenance records available. As a substitute, cost data was acquired from another source and merged with each record, allowing the identification of the most expensive part consumed during a maintenance action. Following the assumption that repairs would require one significantly expensive part and zero or several less expensive auxiliary parts, the most expensive part number consumed for each record was recorded as the primary part number. Figure 3 gives one trivial and theoretical example of this analysis with fabricated costs for the different parts.

Part Number	**Part Description**	**Cost**
015623	Main Bearing	$1,100
023654	Seal	$60
010023	1/4 -20 bolt	$1

$\downarrow$

Primary Part Number - 015623

Figure 3: Primary Part Number Determination (part numbers randomly generated to demonstration concept)

If a maintenance record only has a single part number consumed, that part number is obviously recorded as the primary part number. Pareto analysis of the downtime associated with different primary part numbers can be seen in Figure 4. This analysis revealed that only 86 unique primary part numbers were associated with engine downtime in the population of maintenance records, and 16 of those part numbers were associated with 80% of all recorded downtime.

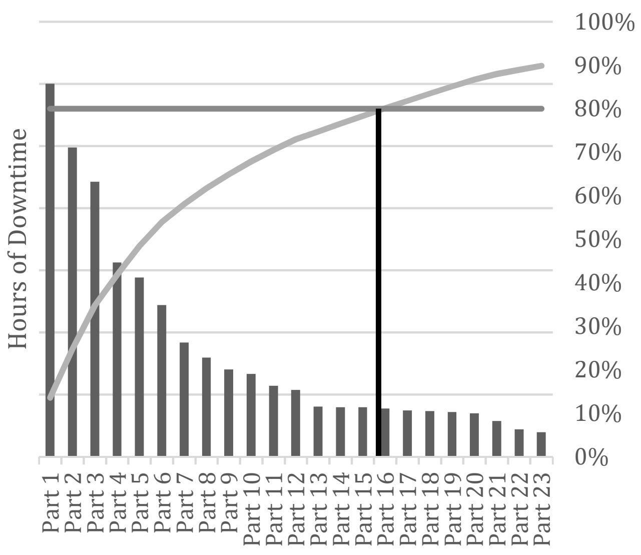


Figure 4: Pareto Chart, Primary Part Number vs Downtime

Ultimately, a combination of the codes used by the maintainers, specific part numbers consumed while preforming the maintenance actions, and several plain text descriptions of the maintenance action were used to classify the maintenance records into different failure modes.

Having classified the maintenance records into different failure modes, and establishing down time as the metric for UME severity, the failure modes found to be responsible for large percentages of total downtime were then analyzed by a diesel engine subject matter expert to determine if that specific failure mode was a good candidate for prediction. Good candidates were failure modes which satisfied the following criteria:

1. Expected to follow a reasonably continuous failure progression
2. Expected to change the operation of the engine in question
3. Not expected to be visible to engine operators until the failure had significantly progressed

The purpose of this filtering was to remove failure modes which would have a low chance of successful modeling, or failure modes where even the best predictive analytics modeling would offer little new information to the operator.

2.1.2. Maintenance Record Final Cleaning and Preparation

Having determined which failure modes significantly impact engine availability, have a potential for successful prediction, and are not immediately obvious to operators, there was some necessary final data cleaning and analysis to be done. Failures which commonly surfaced were main bearing failures, various sensor failures, and fuel system faults.

Having concurrently developed some understanding of the schema of the parametric databases which were to be queried, some fields in the maintenance record database were reformatted for agreement between the two different databases. This relatively simple step significantly simplified downstream work.

Certain automatically generated failure modes were deemed too broad, and required some manual cleaning and further classification. This accounted for a small fraction of the total maintenance record population and was not a surprise.

2.2. Parametric (Sensor) Data Preprocessing

2.2.1. Degradation Class Labeling

We assume that as engine operational hours accumulate, the expected remaining life of its components decreases - the components degrade. It has been established that sensor data observations are assigned a degradation class according to how far away they are in the history from the failure event. As such, a method for establishing this distance is required. A similar approach was explored by Roemer (2017) to examine faults in flow valves.

A simple metric of elapsed time does not work well in practice due to the fact that the engines of the population do not see identical usage patterns. Instead, fuel consumption was used as the measurement for expended life. Fuel consumption effectively represents “load-hours” and accounts for periods of shutdown as well as normalizing light vs. heavy engine load. An example fuel history vs. time and how degradation classes are defined is shown in Figure 5. It is seen here that various engines burn through fuel at differing rates when all engine’s fuel histories are aligned at the point of the Unscheduled Maintenance Event.

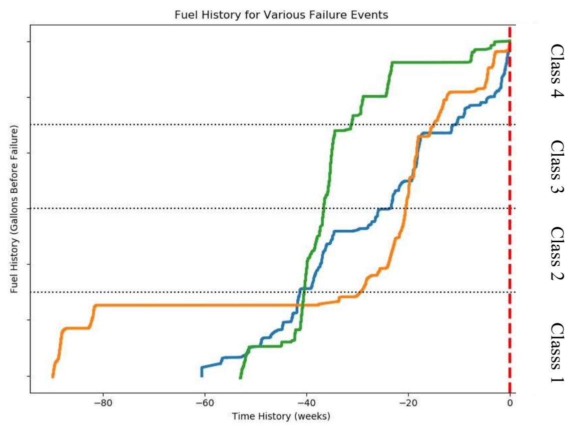


Figure 5: Example fuel history vs time showing fault classification levels.

2.2.2. Missing Value Handling, Additional Processing

Many ML algorithms are unable to handle missing data. Linear Discriminant Analysis, in particular, requires a fully dense observation matrix for determining the subspace transformation mapping. As such, a method was developed for paring down the sensor data to remove incomplete observations.

Procedure for handling missing data:

1) Resample data by averaging each N-minute span into a single data observation. This averages neighboring data observations into a single vector that may have more non-missing values than each of the individual source vectors.
2) Calculate the percentage of missing values in each row, and determine the maximum value
3) Calculate the percentage of missing values in each column, and determine the maximum value
4) If the row maximum percentage is greater than the column maximum percentage, delete all rows having that percentage score. Else, remove the first column having that percentage score
5) Loop back to step 2 until no missing values remain

The framework includes flexibility to add further subject matter expert defined preprocessing/feature engineering steps as well. An example of this would be starting with a list of engine bearing temperature measurements, and using them to derive a new feature representing the maximum differential bearing temperature.

2.3. Machine Learning

Upon completion of the previously described process, we arrive at a set of engine operating point observations labeled with degradation classes corresponding to fuel history distance from failure. This is the form required for training a ML classifier. As the data processing framework was developed in Python, the scikit-learn toolbox was used for ML functions. To make the trained classifiers and preprocess results portable for classifying new data, a high-level, custom classifier object is defined. This object encapsulates a list of the specific sensor channels used for training after conducting the missing values handling, sensor data normalization coefficients, and the scikit-learn ML classifier objects themselves (i.e. Linear Discriminant Analysis (LDA) transform coefficients, Bayesian weights, decision trees, etc.) (Pedregosa, et al., 2011). The framework is intended to be flexible, enabling any classifier that accepts labeled data observations to be evaluated for use with little additional integration effort.

2.3.1. Linear Discriminant Analysis-Naïve Bayes

Many standard classifiers are available for use in the scikit-learn toolbox, which is one of the strengths of Python for data science. One of the modeling architectures proposed throughout this effort is a combination classification model which uses LDA for subspace creation and dimensionality reduction in combination with a Naïve Bayes classifier trained on the transformed data. At a conceptual level, LDA calculates the subspace transformation of labeled data that maximizes Fisher's discrimiant ratio, which can be thought of as the scatter between classes divided by the scatter within classes. While simple, hard decision boundaries can be formed based on these transformations, a Naïve Bayes model was fit to the subspace features to further improve the model and to provide more information about how likely each class is for a given observation. The Naïve Bayes classifier estimates the most likely normal distribution for each input dimension for each classification level. New data is classified by calculating the likelihood that each different observation belongs to each different classification level, variable by variable – the class with the highest combined likelihood is then the predicted class.

Used in combination, LDA maximizes the separation between classes in an optimized subspace, and Naïve Bayes then learns the details of where each different degredation class is most likely to be found within this subspace. New observations are first transformed into the subspace learned by the LDA model, and then the likelihood of membership to each class is calculated based upon the distributions estimated by the Naïve Bayes model.

3. EXAMPLE: INJECTOR PUMP FAULTS

To test the approach and associated framework, a UME list was generated focusing on Fuel Injector Pump faults. A list of 7 UMEs was used for this testing. Several ML classifiers were evaluated. Details of the UMEs used in this example can be seen in Table 1.

Table 1: Maintenance Events Used for Section 3 Example

UME	Total Observations	Class 1 Obs	Class 2 Obs	Class 3 Obs	Class 4 Obs
1	178	0	0	63	115
3	802	267	224	192	119
4	650	100	264	130	156
5	806	141	240	186	239
6	580	18	23	152	387
7	956	213	334	201	208
8	927	96	290	265	276

3.1. Grid-search of learning algorithms

Following identification of the UMEs to be used, configuration scripts defining a grid search of ML techniques were developed and executed by the Framework. Two holdout methods were used for testing, one where a percentage of each UME's data was held out to form a testing set, and one where entire UMEs were held out to form a testing set. The combined LDA-naïve bayes classifier was compared to both a Random Forest Classifier and a Support Vector Machine (SVM) classifier. After extensive experimentation, a third order polynomial kernel function was found to provide the best results during use of the SVM. The results are shown in Table 2 and Table 3.

Table 2: Prediction Accuracy of testing data by various algorithms when randomly selecting 25% of data as testing data, 20 replications.

	Prediction Accuracy Mean	Prediction Accuracy Standard Deviation
LDA-Naïve Bayes	0.687498	0.006862
Random Forest	0.95582	0.002835
SVM	0.913857	0.002695

Since the deployment of the model will require accurate prediction of degredation classes on new maintenance events rather than assigning the correct class when filling in the gaps of a maintenance event, a different strategy for creating training and testing datasets was required.

Table 3: Prediciton accuracy of various algorithms when randomly assigning 2 of 7 maintenance events as testing data, 20 replications.

	Prediction Accuracy Mean	Prediction Accuracy Standard Deviation
LDA-Naïve Bayes	0.251868	0.118754
Random Forest	0.311276	0.086861
SVM	0.165678	0.068346

The significant dropoff in scores for all three modeling algorithms suggests that overfitting to specific failure events is a significant concern for all modeling architectures used.

The decision was made to move forward with the LDA-Naïve Bayes model despite it's lower mean predictive accuracy after significant, manual examination of predictions made by all three models. For certain combinations of maintenance events, LDA-Naïve Bayes models demonstrate significantly more predictive power than similarly trained SVMs and random forests, while remaining robust to overfitting and easily interpretable.

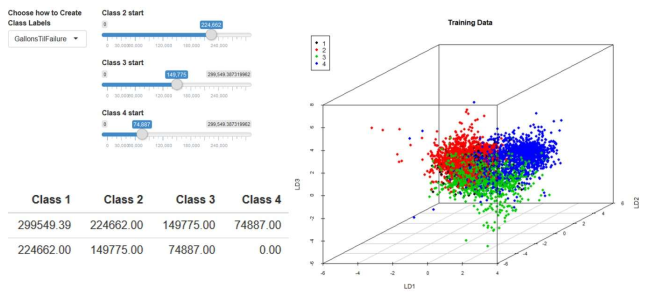
3.2. Analysis GUI tool in R/Shiny

The way in which fuel history is split into degradation classes is a crucial part of the algorithm tuning process. To improve the efficiency of this process, a Shiny web application was developed using the R statistical computing language, which provides users with an easy, graphical interface to upload different datasets, experiment with different class labeling schemes, and receive immediate feedback (Chang et al., 2018). Available tuning parameters within the app are:

- Number of Classes (integer, 2-10)
- Class Division Units (gallons until failure and days until failure)
- Individual Class starting point (individual slider inputs for classes 2 and up)
- Maintenance events to include in training data (checkboxes which set inclusion in training set)
- Maintenance events to include in the testing data (Checkboxes which set inclusion in testing set)

The application responds to user input by altering the class label vector based upon the users chosen number of different classes, class label division units, and specific class label parameters. Having generated a class label vector, the application partitions the data into training and testing data per the user's choices, and performs linear discriminant analysis on the selected training data. Using the transformation matrix generated from the training data, the training data and the testing data are plotted in the feature space (up to the first three features), with color determined by the actual class label of the data point. A confusion matrix is also calculated and displayed, as well as the percentages of correctly and incorrectly predicted values in the testing dataset. Several figures demonstrating this application (with discussion) as applied to the degraded fuel pump example, will follow.

As each of the different algorithms from section 3.1 was tested using 4 evenly divided classes, this is the starting point for this tuning exercise.



Class 1	Class 2	Class 3	Class 4
299549.39	224662.00	149775.00	74887.00
224662.00	149775.00	74887.00	0.00

Figure 6: Screen shot from the algorithm tuning app showing Injector Pump Data, plotted in feature space.

Figure 6 demonstrates the bunching of all the various classes of data using the default settings. There is very little visible of the first class of degradation, as it appears to be very similar with class two. By changing the slider value of "Class 2 start", the user can choose to combine these two groups by covering roughly the first 150,000 gallons before failure together and further dividing the final 150,000 gallons of failure. Another option is to reduce the total number of fault classes from 4 to 3, or to select one or more maintenance records to remove from the training dataset. After some experimentation, the following settings were considered – maintenance events 3, 5, and 6 were used for training data, while the classes were set to the following start and stop points:

Table 4: Intermediate step and experimentation, class definitions, Fuel Injector Pump Data example

Class	1	2	3	4
Start (gallons until failure)	299549	195000	90000	30000
End (gallons until failure)	195000	90000	30000	0

The class definitions in Table 4 produced the training data plot in Figure 7, which displays 4 fairly distinct clusters. This leaves maintenance events 1, 4, 7, and 8 as optional testing dataset components, and maintaining these settings shows poor results when predicting the failure progression of these maintenance events.

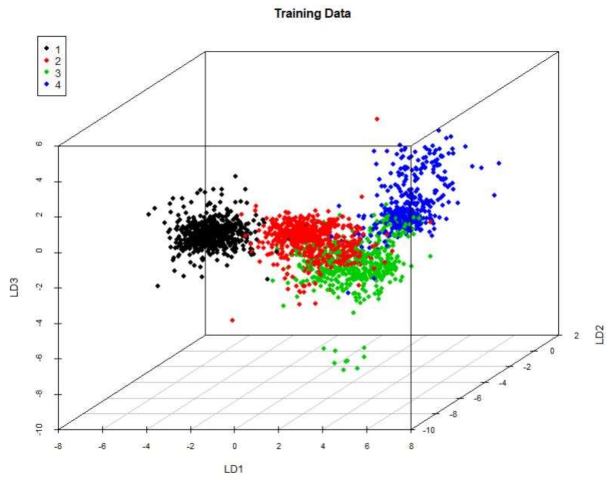
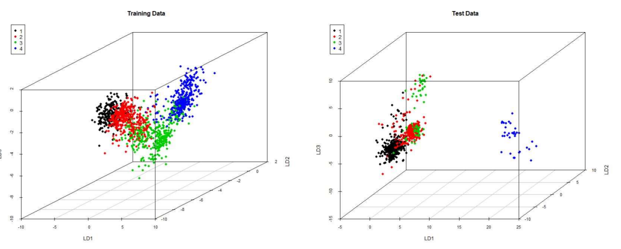
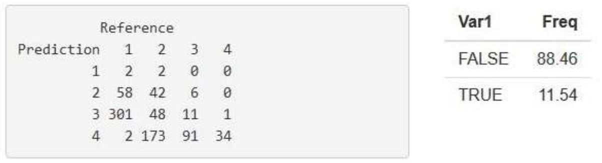


Figure 7: Training data associated with intermediate experimentation step

More interesting is how the fault mapping changes when removing one of the three events which currently make up the training data, and testing on this.



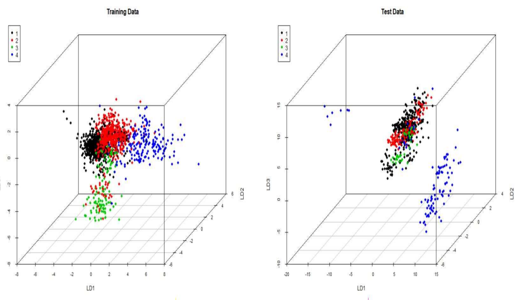


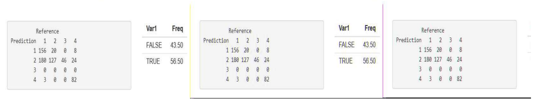
```
        Reference
Prediction   1   2   3   4
         1   2   2   0   0
         2  58  42   6   0
         3 301  48  11   1
         4   2 173  91  34
```

Var1	Freq
FALSE	88.46
TRUE	11.54

Figure 8: Training (Events 5 and 6) and Testing (Event 3) mapping and results

When event 3 is removed from the training set and added to the testing set, a new model is trained, and the data from UME 3 is transformed and plotted according to the new LDA transformation (Figure 8). The predictive accuracy is very low (11% correct), but the failure progression follows a similar but shifted pattern through subspace. Investigating the confusion matrix shows that a large portion of the data which is actually class 1 has been misclassified as class 3, while a very large portion of data which is class 2 has been misclassified into class 4. Similar analysis can be done by separating events 5 and 6, drawing different conclusions regarding the similarity between different maintenance events. After some more tuning and experimentation, a training dataset is created from events 3 and 7, and a prediction is made on event 4 producing the results in Figure 9**Error! Reference source not found.**. The model in Figure 9 has a much higher overall prediction accuracy, mostly due to strong performance predicting class 2 (86% correct) and 4 (72% correct). Most of the data for class 4 is separated by high values of feature 1 (LD1 in the graph).





```
        Reference
Prediction   1   2   3   4
         1 156  20   0   8
         2 180 127  46  24
         3   0   0   0   0
         4   3   0   0  82
```

Var1	Freq
FALSE	43.50
TRUE	56.50

```
        Reference
Prediction   1   2   3   4
         1 156  20   0   8
         2 180 127  46  24
         3   0   0   0   0
         4   3   0   0  82
```

Var1	Freq
FALSE	43.50
TRUE	56.50

```
        Reference
Prediction   1   2   3   4
         1 156  20   0   8
         2 180 127  46  24
         3   0   0   0   0
         4   3   0   0  82
```

Figure 9: Training (Events 3 and 7) map and Testing (Event 4) map and results

The user can decide to further tweak the number of classes, try to more effectively split classes 1-3 apart, or move onto a completely different selection of maintenance events if not a whole different failure mode. The application provides tremendous flexibility to experiment with the tuning parameters available to this machine learning technique.

3.3. Sensitivity correlations of UMEs

The analysis from section 3.2 should provide some sense of how complex the interactions between different maintenance events can be within the linear discriminant analysis framework, even in very similar fault groupings. One way to further explore these subtleties is to test the predictive power of a model trained on a single maintenance event on the dataset constructed for a different maintenance event within a specific failure mode, and to repeat this methodology for all given maintenance events. This procedure allows quick analysis of how similar the progression through feature space is between two or more different maintenance events. Results of this strategy are presented in
Table 5.

The bottom row is the average of all scores that are not the model scoring on itself, or the cross validated (CV) score for that model. The diagonal, where the model is scored only on the training data, is shaded grey as this is not a valid score for this methodology. Valid scores over 0.5 are highlighted green to make patterns visually clear. Looking at the averaged scores, models trained on maintenance events 1, 5, and 7 are marginally more accurate than others while models trained on event 8 are fairly weak.

Table 5: UME sensitivity matrix, fuel injection pumps

	Model 1.0	Model 3.0	Model 4.0	Model 5.0	Model 6.0	Model 7.0	Model 8.0
data 1.0	0.98	0.05	0.40	0.56	0.13	0.43	0.12
data 3.0	0.25	0.84	0.39	0.15	0.23	0.12	0.34
data 4.0	0.24	0.41	0.94	0.13	0.22	0.53	0.11
data 5.0	0.33	0.35	0.20	0.96	0.19	0.25	0.28
data 6.0	0.66	0.05	0.67	0.66	0.98	0.66	0.03
data 7.0	0.25	0.22	0.11	0.26	0.32	0.87	0.12
data 8.0	0.43	0.28	0.26	0.38	0.19	0.21	0.93
CV Score	0.36	0.23	0.34	0.36	0.21	0.37	0.17

Most interestingly, event 6 is predicted either very strongly (nearly 66% correct) or very weakly by the other events grouped as degraded fuel injector pumps. Other combinations of model-data which scored well include model 5 on data 1 and model 7 on data 4, indicating that each of these pairs of maintenance events has a similar path to failure and are worth further investigation.

4. DISCUSSION

4.1. Data Sparsity and Performance

The datasets which formed the basis for the training data were extremely sparse throughout this project – for example, only 12.9% of the dataset originally constructed for the example used throughout section 3 were real values - the remaining 87.1% were missing when originally flattened for this analysis. After the final resampling and cleaning the final dataset contained 0.476% of the original entries, and 38% of the original sensor columns. Given this level of data density, a large part of the preprocessing is primarily concerned with creating a full dataset for the algorithm to work with.

Much of this has to do with the fact that the database holding the parametric data was never intended to be flattened in this way or used for this purpose. Adapting this data to algorithms that are sensitive to missing values makes the dataset cleaning, preparation, and selection even more critical than originally anticipated. Much of the strength of dimensionality reduction machine learning algorithms rests on the ability to combine different data channels, and this power is significantly inhibited when the data is this sparse. One possible option is to create an ensemble model, built on models reliant only on single data streams – this approach would disregard the data interactions, however, and would lose much of the power discussed above. Significant feature engineering is one other possible avenue to combat this issue.

4.2. Classifying Multiple failure modes

Up until this point, all discussion has been focused on training a classifier to detect a specific failure mode. In practice, it is useful to be able to discern multiple failure modes from each other, as well as a normal state from an abnormal state. A proposed approach is shown in Figure 10, where a hierarchical classification structure is presented. Raw sensor data is first classified into either normal or abnormal status. If it is abnormal, another classifier determines which fault is most probable to cause the abnormaility, at which point the degradation classifier can be used to assess level of degradation and subsequently remaining useable life (prognostics).

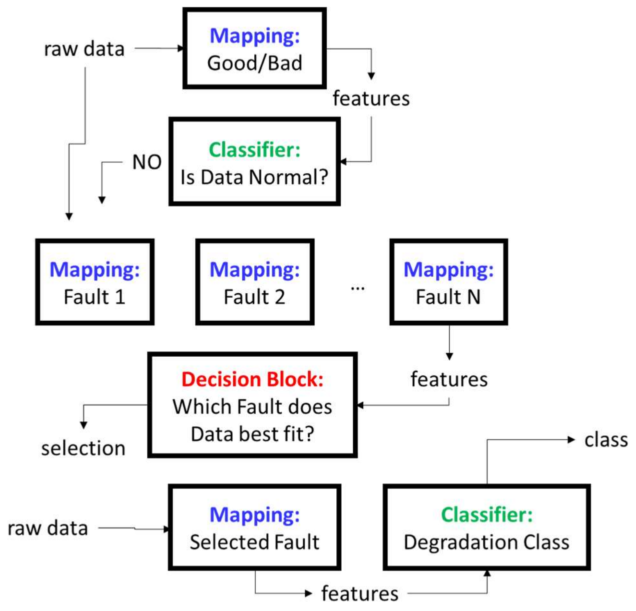


Figure 10: Processing flowchart for use with multiple failure modes

5. CONCLUSION

The results presented in this paper demonstrate the possibility for certain datasets and models to map similarly in feature space, and therefore perform well for both classification/diagnostics and prediction. The techniques applied here to diesel engines are also applicable to other complex engineering systems, such as drive trains and transmissions, for which operational and sensor data are collected by control and monitoring systems.

The classification results described in this paper show the effect of data sparsity in both the engine sensor data and the maintenance record data. The algorithm's requirement of a fully dense data matrix leads to the removal of a substantial amount of sensor data. For the data used in this study, the data collection system was not designed with the intention of providing fully dense data tables, and many other prime mover systems with data collection systems designed before the advent of big data will present similar issues. This is common in many industrial plant and vehicle health monitoring and control systems.

Future work will focus on investigating the possibility that more complete data, either through improved data collection processes or better data cleaning, will yield explainable, practical, and useful machine learning classifiers for failure prediction on large diesel engines.

None of the progress or findings from this effort would be possible without subject matter expertise in the engines which are the topic of classification. Understanding of the engine's operation and associated maintenance practices is crucial to the construction of useful datasets when using maintenance records to label training data. This is in accordance with accepted best practice for machine learning team development, where subject matter experts in the topic of interest play a critical role in goal setting, feature extraction, dataset creation, and model validation.

ACKNOWLEDGEMENT

This work was supported under NAVSEA Contract Number N00024-12-D-6404. The content of the information does not necessarily reflect the position or policy of NAVSEA, and no official endorsement should be inferred. This publication has been approved for unlimited public release

REFERENCES

Banks, J., Reichard, K., & Drake, M. (2008), System Reliability and Condition Based Maintenance, *IEEE Reliability and Maintainability Symposium* (pages 423-427), January 28-30, Las Vegas, NV. doi: 10.1109/RAMS.2008.4925833

Bernardo, J. T., & Reichard, K. M. (2017). Trends in Research Techniques of Prognostics for Gas Turbines and Diesel Engines. *Annual Conference of the Prognostics and Health Management Society 2017*, October 2-5, St. Petersburg, FL.

Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J., (2018). shiny:Web Application Framework for R. R package version 1.1.0. https://CRAN.R-project.org/package=shiny

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 248–255). IEEE.

Grosvenor, R. I., Prickett, P. W., Frost, C., & Allmark, M. J. (2014). Performance and condition monitoring of tidal stream turbines.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Japkowicz, N., Myers, C., & Gluck, M. (1995). A novelty detection approach to classification. *International Joint Conferences on Artificial Intelligence* (Vol. 1, pp. 518-523).

Pedregosa, F., et al., (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. pp. 2825-2830.

Roemer, M. R., (2017). *Fuzzy Logic, Neural Networks and Statistical Classifiers for the Detection and Classification of Control Valve Blockages*. Master of Science Thesis. The Pennsylvania State University, University Park, PA.

BIOGRAPHY

James D. Kozlowski received his B.S. and M.S. in Engineering in 1993 and 1996 from Temple University, Philadelphia, Pennsylvania and received his Ph.D. in Electrical Engineering in 2002 from the Pennsylvania State University, State College, Pennsylvania. His primary area of research is machine condition monitoring, diagnostics and prognostics. Specifically, data collection, sensor optimization, data fusion and component failure prediction and precursor detection. The types of systems that has been his focus in the last ten years include electrochemical systems (batteries, fuel cells, etc.) and drivetrain components (gears, bearings, etc.). He is currently employed by the Applied Research Laboratory at the Pennsylvania State University, supporting programs sponsored U.S. Department of Defense and NASA.

Steven R. Nixon received his B.S. Mechanical Engineering from the Pennsylvania State University in 2015. He spent two years working in industry on wireless vibration analysis and joined the Pennsylvania State University Applied Research Laboratory in 2017 where he is a research engineer. His areas of research include failure prediction of mechanical systems, statistical analysis, condition based maintenance

Ryan T. Weichel received his B.S. and M.S. in Electrical Engineering in 2008 and 2010 from the Pennsylvania State University, State College, Pennsylvania. His areas of research include power electronics and control systems, machine learning applications and techniques, robotics, and embedded sensing and control. He is currently employed by the Applied Research Laboratory at the Pennsylvania State University.

Karl M. Reichard received the Ph.D., M.S. and B.S. degrees in Electrical Engineering from the Virginia Polytechnic Institute and State University (Virginia Tech). Dr. Reichard is an Associate Research Professor with the Pennsylvania State University Applied Research Laboratory, and the Penn State Graduate Program in Acoustics. His research experience includes the development of embedded and distributed sensing and control systems for prognostic health management, robotics, noise cancelation, and acoustic monitoring and classification.