# Modeling Health Status Identification in a Gas Turbine System: Three-Class Classification Approaches

Catherine Cheung[1], Calista Biondic[2], Zouhair Hamaimou[3], and Julio J. Valdés[4]

[1,2,3] *Aerospace Research Centre, National Research Council, Ottawa, Canada*
*catherine.cheung@nrc-cnrc.gc.ca*
*calista.biondic@nrc-cnrc.gc.ca*
*zouhair.hamaimou@nrc-cnrc.gc.ca*

[4] *Digital Technologies Research Centre, National Research Council, Ottawa, Canada*
*julio.valdes@nrc-cnrc.gc.ca*

## Abstract

In this work, the sensor data from a gas turbine system was analyzed with the objective of modeling the health status of the system. A variety of classification models of the system were developed in a three-class problem to differentiate healthy, deteriorated and failed system states to explore the ability of machine learning models to provide early warning of upcoming incidents. However, there are limited examples of failure incidents and the available examples do not cover the scope of possible failures. Therefore the challenge is to create a model that detects failures in new unseen incidents, while also successfully applying that model to different vehicles of the same type. Three approaches to selecting training data were used. The first followed a traditional method of randomly selecting data points from all data according to a target ratio between training and testing data for each data class. The second data selection strategy was to consider data related to failure incidents as a whole and select certain incidents to include in training, and the remaining ones to be unseen in testing. The third approach was a cross-validation-inspired approach, separating data into folds but training and testing models based on failure incidents. In addition to investigating training and data selection strategies, the effect of hyperparameter optimization was explored as well as the effect of varying the time period of the deteriorated class. The classification methods included support vector machines, Gaussian Naïve Bayes, Random Forest, Adaboost, multilayer perceptron, k-nearest neighbor, and extreme gradient boosting. Ensemble models were also created to leverage all the individual classification models that were developed. This paper describes the comprehensive results that were obtained using the various approaches and combinations, highlighting the respective benefits and limitations.

## 1. Introduction

Rapid developments in sensor technology, data processing tools and data storage capability have helped fuel an increased appetite for equipment health monitoring in mechanical systems. As a result, the number of sensors and amount of data collected for health monitoring has grown tremendously. It is hoped that by collecting large quantities of operational data, predictive tools can be developed that will provide operational, maintenance and safety benefits. Fault detection, fault diagnosis, prognostics and health management for mechanical systems is an active research field. Data mining and machine learning techniques are at the core of this analysis and have been key to the ongoing advancements in this area to address the challenge of extracting useful results from the data collected.

In this work, the sensor data from a gas turbine system was analyzed with the objective of modeling the health status of the system. Previous efforts (Cheung, To, & Valdés, 2020) had used a two-class approach for this problem, to distinguish healthy and failed states of the system. Some of the key findings were that the selection of the training data sets is a very important consideration, as the success of the classifiers were significantly diminished for the second technique tested in which whole failure events were withheld from training. Another outcome was that despite simplification of the system into two states, the results showed accurate predictions could still be achieved, and that an ensemble could be used to leverage a variety of classification and anomaly detection models. To build on this work, a third class labeled as deteriorated system data was added prior to each failure event to explore the ability of machine learning models to provide early warning of upcoming incidents.

Three approaches to selecting training data were used. The first followed a traditional method of randomly selecting data points from all data according to desired percentages. The second data selection strategy was to consider data related to failure incidents as a whole and select certain incidents to include in training, and the remaining ones to be unseen in testing. These first two approaches were explored initially in (Cheung et al., 2020). A third approach was added in this work, namely a cross-validation inspired approach. In addition to investigating training and data selection strategies, the effect of hyperparameter optimization was explored as well as the effect of varying the time period of the deteriorated class of data.

In many real-world applications, a severe class imbalance exists between available normal, healthy system data and data from failure incidents. Furthermore, the full scope of possible system failures is not captured by data that can be used for training models. Finally, the variation between vehicles of the same type presents an additional challenge for model development. In the search for system models that are sufficiently robust and flexible to overcome these challenges, investigating different data selection strategies for model training has been a key focus area.

Consequently, the exploration of data selection, or model training schemes as alternatives to conventional random sampling were pursued in order to investigate the ability and flexibility of machine learning models to accurately identify the health status of the gas turbine system in two different vehicles, though not identical but with similar sensor systems.

Through the investigation of these data selection techniques, different classification methods, hyperparameter optimization, and time period of the deteriorated class, a comparison of the different approaches and techniques can be made, providing insight into which techniques are most effective in this challenging problem, as well as highlighting the respective benefits and limitations.

This paper is organized as follows: Section 2 provides details of the gas turbine data and the three data classes, Section 3 gives an overview of the analysis approach, Section 4 describes the methods and tools used for data analysis, Section 5 details the selection of training and testing sets, Section 6 discusses the implementation details and model hyperparameter settings, Section 7 presents and discusses the results of the analysis, and Section 8 summarizes the findings.

## 2. GAS TURBINE DATA

Gas turbine (GT) data from two separate vehicles were obtained for analysis, referred to as Vehicle 1 and Vehicle 2. A total of 76 predictor variables were included for developing models for the GT system, comprising speeds, temperatures, pressures, vibration levels, and control stick positions. Additional details of the sensors are provided in (Cheung et al., 2020).

A total of seven GT-related incidents were recorded by the gas turbine sensor system on the two vehicles, six on Vehicle 1 and one on Vehicle 2. For this analysis, the data associated with the periods of these 'failure' incidents were considered as 'failed' data points and belonging to the 'failed' class of data. Previously, all other data would be considered part of the 'healthy' class of data.

### 2.1. Transition to three-class problem

As a method of early detection, a new data class was added. The deteriorated period is an adjustable length of time prior to known failures that was hoped to show some abnormal behaviour which could help predict the upcoming failure. Time periods of 3 and 6 hours were selected to provide a reasonable distribution of data classes. However, by adding the third class, anomaly detection models could no longer be used as they can only separate the data into two groups, normal and anomalous.

Although it is a reasonable hypothesis that the signals may show indications of system deterioration prior to a failure, there is still a large amount of uncertainty surrounding labeling of data into classes. The sources of this uncertainty are twofold. First, only dates of failures were provided without precise times and therefore errors are introduced in the class labeling with the chosen start and end mark of a failure event. Second, the length of the deteriorated time period is arbitrarily selected, and the introduction of information that could mislead the classifiers in unforeseen ways is a possibility. As correct labeling may be one of the most vital steps of classification, these are issues that may be worth reviewing in the future.

## 3. OVERVIEW OF ANALYSIS APPROACH

The overall analysis approach, including the different data selection strategies that have been tested in this work, is presented in Figure 1. Pre-processing was the initial step, to which raw data was input and from which cleaned and labeled data was output. More details of the pre-processing steps are provided in (Cheung et al., 2020).

The three data selection strategies that were explored (further described in section 5) are shown in Figure 1 in green boxes. Random sampling (section 5.1) and blind failure (section 5.2) were explored using all available data, as well as with representatives and hyperparameter optimization. The outputs from both options were compared within each strategy. In addition to individual classification models, two ensemble classifiers were created using these two data selec-

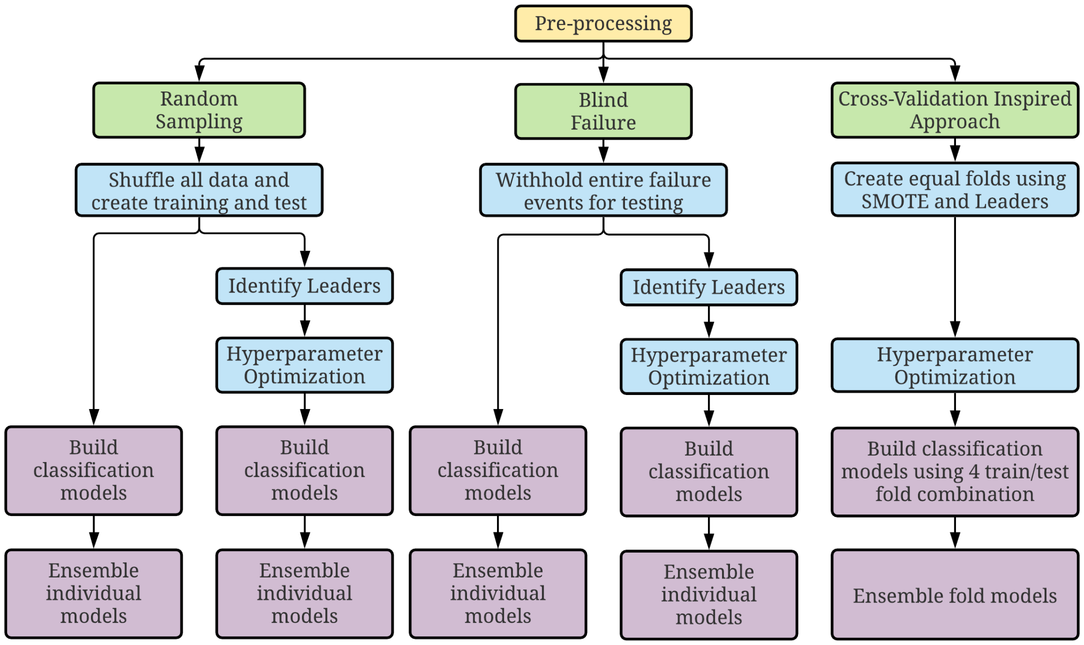
Pre-processing
Random Sampling
Blind Failure
Cross-Validation Inspired Approach
Shuffle all data and create training and test
Withhold entire failure events for testing
Create equal folds using SMOTE and Leaders
Identify Leaders
Identify Leaders
Hyperparameter Optimization
Hyperparameter Optimization
Hyperparameter Optimization
Build classification models
Build classification models
Build classification models
Build classification models
Build classification models using 4 train/test fold combination
Ensemble individual models
Ensemble individual models
Ensemble individual models
Ensemble individual models
Ensemble fold models

Figure 1. Summary of model building approach

tion strategies, differing slightly in their training and testing sets. The right column of the figure summarizes the cross-validation inspired approach (section 5.3), also referred to as CV-inspired approach. Under- and over-sampling techniques (further detailed in section 4.2), such as Synthetic Minority Over-sampling Technique (SMOTE) and the leader clustering algorithm were implemented before applying the hyper-parameter optimization via Sherpa, described further in section 6. For each classification method in the CV-inspired approach, four models were created using different folds and gathered into an ensemble that was tested on a final unseen fold.

The classification methods (more details provided in section 4.1) included Support vector machines, Gaussian Naïve Bayes, Random Forest, Adaboost, multilayer perceptron, k-nearest neighbor, and extreme gradient boosting. Ensemble models were also created, using a decision table classifier, to leverage all the individual classification models that were developed.

Previous results obtained when the data was divided into two classes indicated significant challenges to obtain accurate classification models when all of the data from a specific failure were withheld (i.e. blind failure data selection strategy). The inclusion of anomaly detection models helped offset the poor classification models. However, in a three-class problem, anomaly detection models can no longer be used. In efforts to create diverse classifiers that would perform better on unseen failures, the CV-inspired approach and ensembles were investigated.

## 4. METHODS

This section presents the machine learning and other techniques explored in the analysis of the gas turbine data. In particular, the various classifiers, under-sampling and over-sampling techniques, and ensemble methods are described.

### 4.1. Classifier methods

Several classification methods were explored in this work to develop models for the gas turbine system. The methods included Support vector machines, Gaussian Naïve Bayes, Random Forest, Adaboost, multilayer perceptron, k-nearest neighbor, and XG boost. Ensemble models were also created to leverage all the individual classification models that were developed, implementing a decision table classifier.

Support Vector Machines (SVM) are one of the most robust classifiers as they are the least prone to overfitting (Cortes & Vapnik, 1995), (De Rainville, Fortin, Gardner, Parizeau, & Gagné, 2012). A nonlinear classification function is created by transforming the feature space to a higher dimension, through a kernel, where linear boundaries successfully classify data.

The Gaussian Naïve Bayes (NB) classifier (John & Langley, 1995) makes use of Bayes' theorem to compute the a posteriori probability of belonging to a class given a priori probability and knowledge of likelihoods of each feature belonging to said class. The likelihoods stem from assigning Gaussian distributions for each feature in the feature space. The naïve assumption is invoked by assuming the features to be mutually independent.

The Random Forest (RF) method (Breiman, 2001) outputs the aggregate decision of a large forest of decision trees trained on bootstrapped data, a random subset of rows and columns of the full training data (Hastie, Tibshirani, & Friedman, 2009). This method of bagging or bootstrap aggregation reduces the high variance tied to decision trees by averaging many noisy but approximately unbiased models.

The AdaBoost algorithm (Schapire, 2013) generates a sequence of weak learners and computes the weighted sum of their hypotheses to create a highly accurate classifier. Weak learners are decision trees with performance slightly better than random guessing. Weight is assigned to a weak learner so as to minimize the error made by the sequence of decision trees thus far. The order of the weak learners ensures that they cover for each other's misclassification by adapting the weight of the subsequent weak learner in favor of the misclassified instances.

The data structure of Neural Networks (NN) was inspired from the brain's neural architecture to include multiple nodes with connections analogous to synapses. NNs with multiple hidden-layers, or multi-layer perceptrons (MLP) (Rumelhart, Hinton, & Williams, 1986) linearize the process by activating nodes in a certain layer based on a linear combination of the previous layer's nodes. The initial layer is the set of input parameters, while the final layer consists of K output nodes corresponding to the K classes (De Rainville et al., 2012). The most 'active' node in the final layer determines the class.

The K-nearest neighbor (kNN) classifier (Bentley, 1975), (Omohundro, 1989) considers the data structure and distribution of the labeled training data to make a classification. Unseen data will be labeled according to the majority class among the K closest neighbors by Euclidean distance (Hastie et al., 2009).

Extreme gradient boosting, more commonly referred to as XG Boost, employs a gradient boosting framework where decision trees are built sequentially by minimizing the errors from previous models. It is optimized for performance and computational speed through parallel processing, tree-pruning, and regularization to avoid overfitting (Chen & Guestrin, 2016).

The strength of the decision table rule-based classifier lies in its simplicity as a majority classifier. The 'best-first' search method and cross-validation are employed to evaluate feature subsets (Kohavi, 1995). The generated decision table relates the inputs to the output class in an explicit and simple manner.

## 4.2. Under-sampling and over-sampling techniques

In the CV-inspired approach, described further in section 5.3, three techniques were used to over- and under-sample the existing data in order to achieve the desired data distribution in each fold: Synthetic Minority Over-sampling Technique (SMOTE), random under-sampling, and a leader clustering algorithm. The latter algorithm was also used prior to hyperparameter optimization in the random sampling (section 5.1.2) and blind failure (section 5.2.2) data selection strategies in order to reduce the size of the data set.

SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) restores balance in the data set by synthesizing new examples of the minority class. By generating more failure examples to train the classifier, it is less likely that a biased classifier with low precision is created. SMOTE uses a kNN algorithm to find the nearest neighbors in the feature space and place new samples on the lines connecting existing samples in the feature space.

The leader clustering algorithm (Hartigan, 1975) is used to construct samples from the total data set to facilitate the analysis of data. Given Euclidian distance as a similarity measure, the algorithm reduces the data set to representative objects (leaders) that cluster similar (close in distance) samples into fewer elements. In this way, the constructed samples are inclusive of the entire space covered by the original data set and no outliers are left unaccounted for, as may occur with conventional random sampling.

## 4.3. Ensemble model training

In addition to individual classification models, two ensemble classifiers were developed for two of the data selection strategies, namely random sampling and blind failure. The process followed to generate these ensemble classifiers is summarized in Figure 2. The individual classifiers, trained on the original training set, were used to make an array of predictions. The ensemble models were built by inputting the independent predictions from the individual classifiers mentioned above into a decision table that output a final prediction regarding the health class of the data point. Two approaches were considered to train the ensemble models:

- Training the decision table using predictions from the same training data seen by the individual classifiers, and testing on the same unseen testing data (referred to as 'ensemble' model)
- Training the decision table using predictions from a subset of the individual classifiers testing data, and testing on the remaining unseen testing data (referred to as 'val_ensemble' model)

Although training to testing set ratios were conserved for both ensemble models, the contents of the testing sets differ.

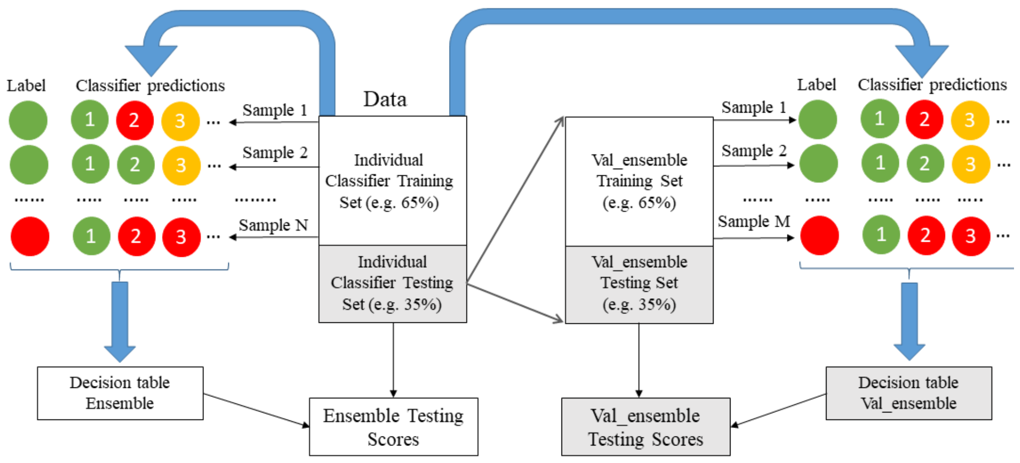
Label
Classifier predictions
Sample 1
Sample 2
Sample N
Data
Individual Classifier Training Set (e.g. 65%)
Individual Classifier Testing Set (e.g. 35%)
Val_ensemble Training Set (e.g. 65%)
Val_ensemble Testing Set (e.g. 35%)
Sample M
Decision table Ensemble
Ensemble Testing Scores
Val_ensemble Testing Scores
Decision table Val_ensemble

Figure 2. Ensemble construction for random sampling and blind failure methods

## 5. DATA SELECTION STRATEGIES

This section provides details of the training and testing sets assembled for the three data selection strategies: random split, blind/withheld failures, and cross- validation inspired approach. An effort was made to have similarly sized training and testing sets between the various data selection strategies.

### 5.1. Random sampling

In the random sampling process, all of the data was shuffled and then separated into training and testing sets. The sets were selected to obtain either 65/35 or 70/30 ratio of available data points for each class in the training vs testing set. In addition, two time periods for the deteriorated class were investigated, one of 3 hours and the other of 6 hours. To study the effects of different deteriorated time periods and data splits, various training and testing data sets were created.

#### 5.1.1. Random sampling without hyperparameter optimization

For each set of data, four configurations were tested, summarized in Table 1. Results were generated separately from Vehicle 1 and Vehicle 2 data sets as well as a combined data set of data from both vehicles.

Table 1. Random sampling configurations

| Short form Label | Length of Deteriorated Time Period | Ratio of data points per class in training set | Ratio of data points per class in testing set |
|---|---|---|---|
| 3_65_35 | 3 hours | 65% | 35% |
| 3_70_30 | 3 hours | 70% | 30% |
| 6_65_35 | 6 hours | 65% | 35% |
| 6_70_30 | 6 hours | 70% | 30% |

Table 2, Table 3, and Table 4 describe the number of points associated with each class in all the configurations training and testing data sets for Vehicle 1, Vehicle 2, and the combined data set with both vehicles' data, respectively.

Table 2. Class breakdown of data points in different training and testing sets for Vehicle 1 GT data

| Config. | Data set | Failed | Det | Healthy | Total |
|---|---|---|---|---|---|
| 3_65_35 | Training | 23151 | 588 | 302675 | 326414 |
| | Testing | 12466 | 317 | 162980 | 175763 |
| 3_70_30 | Training | 24931 | 633 | 325958 | 351522 |
| | Testing | 10686 | 272 | 139697 | 150655 |
| 6_65_35 | Training | 23151 | 1173 | 302090 | 326414 |
| | Testing | 12466 | 632 | 162665 | 175763 |
| 6_70_30 | Training | 24931 | 1263 | 325328 | 351522 |
| | Testing | 10686 | 542 | 139427 | 150655 |

Table 3. Class breakdown of data points in different training and testing sets for Vehicle 2 GT data

| Config. | Data set | Failed | Det | Healthy | Total |
|---|---|---|---|---|---|
| 3_65_35 | Training | 5616 | 702 | 139002 | 145320 |
| | Testing | 3024 | 379 | 74848 | 78251 |
| 3_70_30 | Training | 6048 | 756 | 149695 | 156499 |
| | Testing | 2592 | 325 | 64155 | 67072 |
| 6_65_35 | Training | 5616 | 1404 | 138300 | 145320 |
| | Testing | 3024 | 757 | 74470 | 78251 |
| 6_70_30 | Training | 6048 | 1512 | 148939 | 156499 |
| | Testing | 2592 | 649 | 63831 | 67072 |

Table 4. Class breakdown of data points in different training and testing sets for the combined data (random sampling)

| Config. | Data set | Failed | Det | Healthy | Total |
|---|---|---|---|---|---|
| 3_65_35 | Training | 28767 | 1290 | 441678 | 471735 |
| | Testing | 15490 | 696 | 237827 | 254013 |
| 3_70_30 | Training | 30979 | 1390 | 475653 | 508022 |
| | Testing | 13278 | 596 | 203852 | 217726 |
| 6_65_35 | Training | 28767 | 2577 | 440391 | 471735 |
| | Testing | 15490 | 1389 | 237134 | 254013 |
| 6_70_30 | Training | 30979 | 2776 | 474267 | 508022 |
| | Testing | 13278 | 1190 | 203258 | 217726 |

### 5.1.2. Random sampling with hyperparameter optimization

In the random sampling training strategy, optimization of the classifier hyperparameters was investigated as described further in section 6.1. Smaller data sets were used in the hyperparameter optimization, as the computational costs were very high for running the optimization on all the available data. These smaller groups of data were found using the leaders clustering algorithm, described in section 4.2. The size of the smaller data sets was chosen to correspond with the size of the folds in the CV-inspired process, described in section 5.3. For each failure incident in the data set, 17,088 points were added to the total points in the training and testing optimization data sets. The total points were then separated into training and testing based on the percentage used in the overall training and testing set.

Table 5 describes the approximate number of leaders used for the optimization training and testing data set for Vehicle 1, the total for each being 102,528 since there were 6 failures in the data set.

Table 6 describes the approximate number of leaders used for the Vehicle 2 optimization training and testing data set, the total for each being 17,088 since there was one failure event in the data set. Since the leader algorithm could not be used to achieve a specific number of points but only a similarity threshold, the final data sets may vary by ± 100 points.

Table 5. Class breakdown of data points in different training and testing sets for the combined data (random sampling)

| Config. | Training | Testing | Total |
|---|---|---|---|
| 3_70_30 | 71770 | 30758 | 102528 |
| 6_70_30 | 71770 | 30758 | 102528 |

Table 6. Number of data points in Vehicle 2 GT optimization data sets for each configuration

| Config. | Training | Testing | Total |
|---|---|---|---|
| 3_70_30 | 11962 | 5126 | 17088 |
| 6_70_30 | 11962 | 5126 | 17088 |

## 5.2. Blind Failure

To test how well the classifiers perform on unseen data, the way the training and testing data sets were created was modified. Entire failure events were withheld for testing, so that groups of failure and deteriorated data were completely unseen by the classifiers in training. Similar to the random sampling data selection approach, the configurations included 3 and 6 hour deteriorated time periods, as well as 2 or 3 withheld failure events for testing. The data for both vehicles were combined for the blind failure training strategy.

### 5.2.1. Blind failure without hyperparameter optimization

Using all the available data, and without hyperparameter optimization, Table 7 describes the number of points associated with each class in the four configurations that were investigated, corresponding to two deteriorated time periods (3 or 6 hours) and number of failure incidents withheld for testing (2 or 3 incidents).

Table 7. Class breakdown of data points in different training and testing sets for the blind failure strategy

| Config. | Data set | Failed | Det | Healthy | Total |
|---|---|---|---|---|---|
| 3_2 | Training | 34561 | 1624 | 494518 | 530703 |
| | Testing | 9696 | 362 | 184987 | 195045 |
| 3_3 | Training | 33121 | 1443 | 467045 | 501609 |
| | Testing | 11136 | 543 | 212460 | 224139 |
| 6_2 | Training | 34561 | 3244 | 493077 | 530082 |
| | Testing | 9696 | 722 | 184448 | 194866 |
| 6_3 | Training | 33121 | 2883 | 465684 | 501688 |
| | Testing | 11136 | 1083 | 211841 | 224060 |

### 5.2.2. Blind Failure with Hyperparameter Optimization

Hyperparameter optimization of the classifiers in the blind failure data selection strategy was also explored. The same breakdown of data points as in Table 7 was used for the blind failure scenario for training and testing.

As with the randomly sampled data, the leaders clustering algorithm was used with the blind failure data sets to reduce the size of the data set. Table 8 describes the approximate number of leaders used for the combined vehicle optimization

Table 8. Number of data points in Combined Blind Failure GT optimization data sets for each configuration

| Config. | Training | Testing | Total |
|---|---|---|---|
| 3_3 | 82535 | 37081 | 119616 |
| 6_3 | 82535 | 37081 | 119616 |

training and testing data set, the total for each being 119616 since there are seven failure incidents in the data set.

### 5.3. Cross-validation inspired approach

In conventional cross-validation, a certain number of folds, with approximately the same number of points and point distribution in each fold, are created. Models are created using all but one of the folds, with the last fold used for testing, allowing for an assessment of classifier performance on unseen data. However, the entirety of the data is utilized to train the final classifier.

The cross-validation inspired approach, also referred to as CV-inspired approach, that was adopted in this work used the idea of separating the data into folds, but also retained the blind failure concept for training and testing the models. Although classifiers were trained using folds, the final classifier was not trained on all the data as is done traditionally in cross-validation, but using an ensemble of the individual classifiers. Therefore, one fold was withheld entirely for testing the ensemble, while training included four folds, where each classifier was trained on three folds and tested on the fourth. Figure 3 illustrates how the data was separated into folds, and how the folds were used for training classifiers and the ensemble. From each of the training data sets on the right of the diagram, two folds of data were withheld, one that the classifier would be tested on and one that the final ensemble would be tested on. In the diagram, classifier 1 corresponds to Fold 1 being withheld for testing, similarly classifier 2 corresponds to Fold 2 withheld for testing. Fold 5 was withheld from all training and used for the ensemble testing as unseen data. The ensemble was trained on the four folds of data that were used for training and testing of classifiers 1-4.

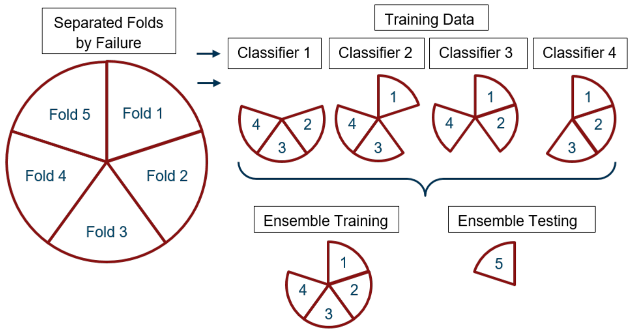

Figure 3. Steps to separate data into cross-validation folds

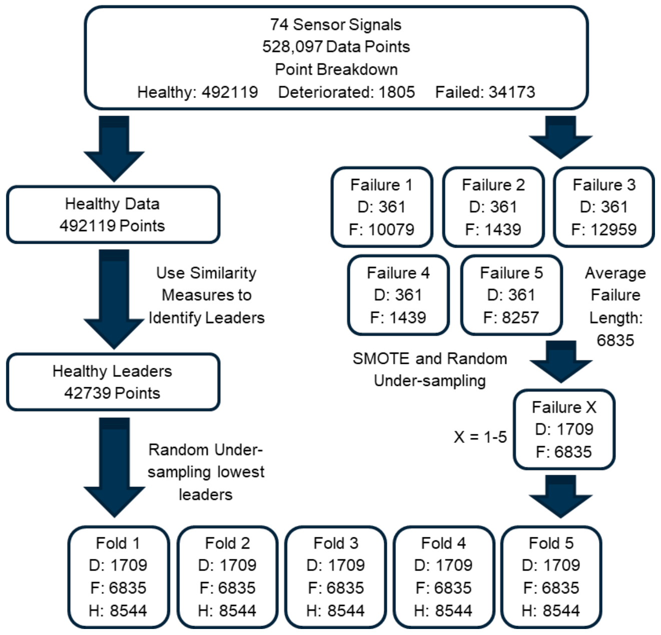

Figure 4. Data sets for training and testing in each step

Figure 4 describes the process performed to create the folds for cross-validation. The goal was to create equal folds with the same class distributions, as it is important in cross-validation for the folds to have similar statistical properties. To try to reduce the class imbalance existing in the original data set, it was decided that each fold should contain approximately 50% healthy points, 40% failed points, and 10% deteriorated points. Three techniques were used to over- and under-sample the data that already existed to obtain the desired distribution in each fold, described in section 4.2: Synthetic Minority Over-sampling Technique (SMOTE), random under-sampling, and leaders.

On the left side of Figure 4 the process to select the healthy data is shown. The identified healthy leaders were randomly shuffled and split into the five equal groups. The right side of Figure 4 displays the process to select the failure and deteriorated data points for each fold. Each failure event was separated into groups containing the failed points and accompanying deteriorated points that occurred prior to the failure. To try to not heavily oversample or under sample a specific failure event, all of the failures were either extracted or reduced to the average number of failed points, 6835. If a failure event had more failed data points than the average, the data set was reduced further using random under-sampling. If a failure event had less points than the average, SMOTE was used to generate more points. SMOTE was also used to oversample each group of deteriorated points to create the 10% representation desired in each fold.

The folds were created by combining the finalized failure groups and randomly split leaders of the healthy data. Each fold contained 17088 points in total: with 8544 healthy points, 6835 failure points, and 1709 deteriorated points.

The groups of training data built one of each type of model, with the hyperparameters of each group of data and classifiers being optimized. Therefore seven classifiers were trained using each of the 4 training groups; creating 28 classifiers in total. The predictions of the classifiers were then combined in a decision table to create the ensemble trained on 4 of the 5 folds. The final step was to test the ensemble model on the last withheld fold of data.

## 6. IMPLEMENTATION DETAILS

This section discusses the implementation details of the techniques that were used, including the range of settings explored for optimization of the hyperparameters in each of the classifiers built.

The model development and analysis was carried out on the Python platform. In particular, the Scikit-learn package (Pedregosa et al., 2011) was used to implement the classifiers described in section 4.1.

The Synthetic Minority Over-sampling Technique (SMOTE) was implemented through the imblearn package (Lemaître, Nogueira, & Aridas, 2017), as was the random under-sampling technique described in section 4.2.

Optimization of hyperparameters of the machine learning models was carried out using Sherpa, which is a Python library for hyperparameter tuning of machine learning models (Hertel, Collado, Sadowski, Ott, & Baldi, 2020). It provides a variety of optimization algorithms that can be used to explore any number of hyperparameters. The GPyOpt Bayesian Optimization was selected as it was efficient when using multiple parameters and when the number of trials is large.

### 6.1. Hyperparameter optimization function

When using the Sherpa optimization of hyperparameters, this process was driven by the selected objective function. Typical scoring functions of classification include precision, recall, F1 score, and many others that reflect true/false negative and true/false positive rates.

For the GT data, it was important that the classifiers correctly identify the failed and deteriorated data points since they are both the minority classes and correspond to when the system is not functioning as expected. Finally it is more costly to have false negatives as it is very important to identify when there is a malfunction, and would be more harmful to be misclassified as healthy. These considerations led to the use of a combination of the failed and deteriorated F2 score as the optimization function.

Equation (1) is the Fbeta equation which is a combination of precision and recall that is controllable via the beta coefficient; for the F2 score, beta is equal to 2. Each class has separate F2, precision, and recall scores.

$$Fbeta = \frac{(1 + beta^2) * (Precision) * (Recall)}{(beta^2 * Precision) + Recall} \tag{1}$$

$$Precision = \frac{True\,Positives}{True\,Positives + False\,Positives} \tag{2}$$

$$Recall = \frac{True\,Positives}{True\,Positives + False\,Negatives} \tag{3}$$

The final optimization function uses the classification results from the test group in equation (4).

$$Optimization\,Function = \frac{F2_{failure} + F2_{deteriorated}}{2} \tag{4}$$

### 6.2. Classifier hyperparameters

For the configurations where the hyperparameter values for the classifiers were not optimized (sections 5.1.1 and 5.2.1), the set of values used are included in Table 9 of the Appendix. Where the hyperparameter value is not specified, the default value was used.

For the configurations where Sherpa was used to optimize classifier hyperparameters (sections 5.1.2, 5.2.2, 5.3), Table 9 in the Appendix lists all the hyperparameters and the values explored in the classification results that appear in section 7.1.2, 7.2.2, and 7.3. Table 10 in the Appendix details the outcome of the hyperparameter optimization for the various configurations.

## 7. RESULTS

Results were generated for a number of configurations and combinations of training strategy, deterioration period, and hyperparameter optimization. The results are presented in the following order:

- Random sampling
- Random sampling with hyperparameter optimization
- Blind failure
- Blind failure with hyperparameter optimization
- Cross-validation inspired approach with hyperparameter optimization and SMOTE

In this section, the results from the many classification methods are compared using two metrics, namely the macro F1 score (Equation (5)) and the modified macro F2 score used for optimization (Equation (4)). Equation (1) can be used to calculate the F1 and F2 score of each class, where beta is equal to 1 and 2 respectively. The macro F1 score is an average of the three F1 scores from each class, as described in Equation (5). The modified macro F2 can be calculated using Equation (5), which is an average of the failure and deteriorated F2 scores. The macro F1 provides insight into how well the classifier performs on all the classes, and the modified macro F2 shows how it behaves during anomalous behavior.

$$F1_{macro} = \frac{F1_{healthy} + F1_{failure} + F1_{deteriorated}}{3} \quad (5)$$

In the plots of macro F2 score vs macro F1 score, the various shapes represent the different configurations and the colors correspond to the different classifier types.

In addition to individual classifier models, two ensemble classifiers were also used in the batch learning. By using the predictions of the individual classifiers, decision tables were built using two techniques as described in section 4.3.

### 7.1. Random sampling results

Details of the training and testing sets for the random sampling data selection strategy were provided in section 5.1, including the two deteriorated time periods that were considered and individual vehicle and combined vehicle data sets.

#### 7.1.1. Random sampling results without hyperparameter optimization

Figure 5 and Figure 6 plot the F2 (modified macro) metric vs the F1 (macro) metric from training and testing, respectively, of each classifier using Vehicle 1 GT data. Figure 7 and Figure 8 show these plots for the Vehicle 2 GT data. Figure 9 and Figure 10 plot the metrics from the training and testing, respectively, of each classifier for the combined data sets of Vehicle 1 and Vehicle 2 GT data.

From the Vehicle 1 random sampling results, the strongest classifiers are Random Forest and XGBoost, while AdaBoost and GaussianNB display the weakest performance. Slightly higher scores were obtained when using 6 hour deteriorated time period as compared to the 3 hour deteriorated time period.

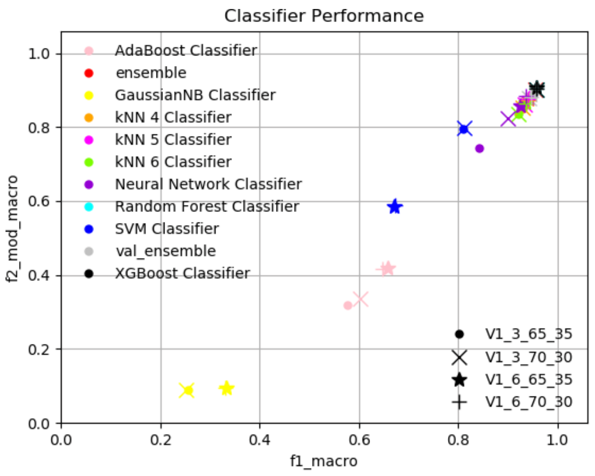

Figure 5. Vehicle 1 random sampling - training results

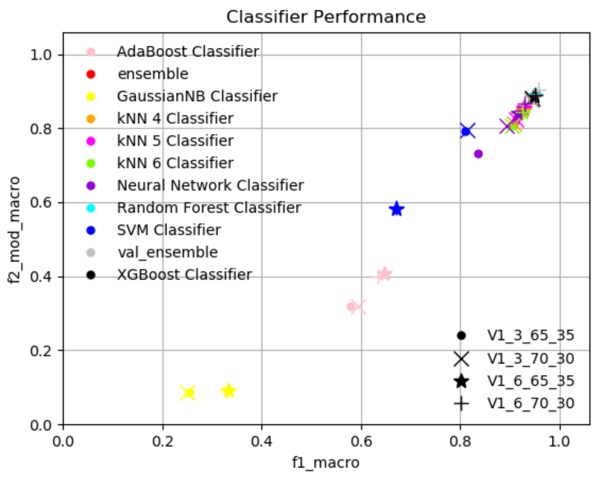

Figure 6. Vehicle 1 random sampling - testing results

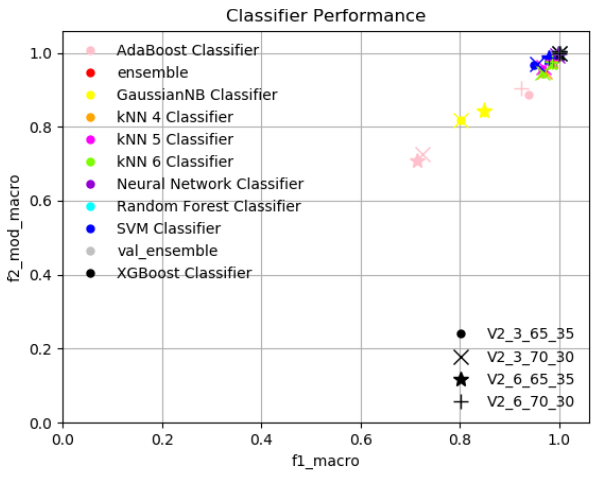

Figure 7. Vehicle 2 random sampling - training results

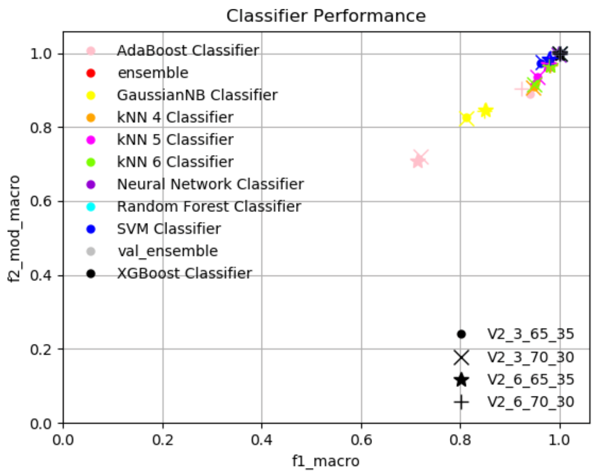

Figure 8. Vehicle 2 random sampling - testing results

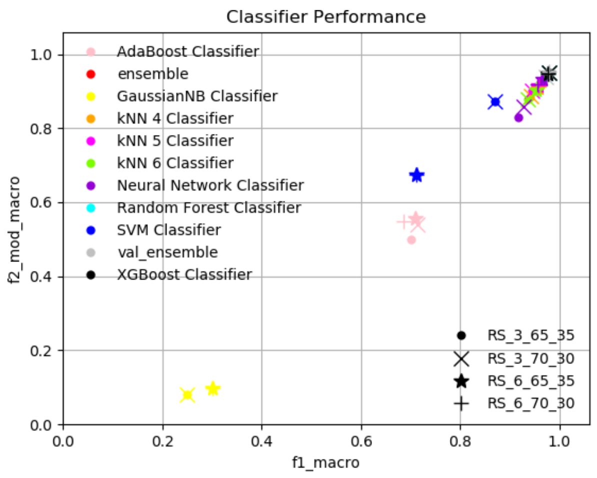

Figure 9. Combined Vehicles 1 and 2 random sampling - training results

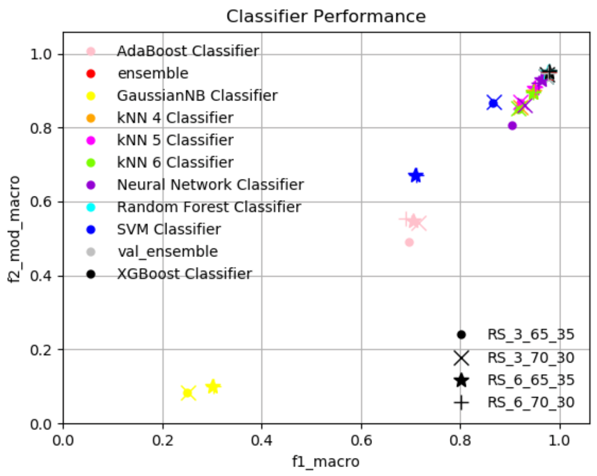

Figure 10. Combined Vehicles 1 and 2 random sampling - testing results

There were no obvious differences noticed in the training/testing split of 65/35 or 70/30. In the 6_70_30 configuration, corresponding to a 6 hour deteriorated time period, and 70/30 training and testing split, the val_ensemble obtains the best performance overall, perhaps because the test set is smaller.

From the Vehicle 2 random sampling results, again the strongest classifiers were Random Forest and XGBoost. Although the weakest classifiers were AdaBoost and GaussianNB, generally good performance was obtained from all classifiers, since F1 and F2 values were greater than 0.7.

Both ensembles, ensemble and val_ensemble, achieved similar success to the best individual classifiers, with val_ensemble having slightly better performance perhaps because the test set is smaller. F1 and F2 scores are very similar in value for each classifier. Slightly higher scores were obtained when using 6 hour deteriorated time period as compared to the 3 hour deteriorated time period. Similar to Vehicle 1, there were no obvious differences noticed in the training/testing split of 65/35 or 70/30.

From the combined data results, consistent improvements in performance were evident across the various configurations, likely due to the larger test data used. The strongest classifiers are still Random Forest, XGBoost, and the two ensembles. Slightly higher scores were noticed again when using 6 hour deteriorated time period.

#### 7.1.2. Random sampling with hyperparameter optimization and leaders

The data sets used for the training and testing of the finalized classifiers in this section are the same as those described in section 5.1.2. Since there was not much variation in the results obtained previously between the 70/30 and 65/35 configurations, only the 70/30 data sets were tested in this section. Figure 11 and Figure 12 plot the two main metrics for Vehicle 1 GT data sets following hyperparameter optimization. Figures 13 and 14 plot the two main metrics for Vehicle 2 GT data sets following the hyperparameter optimization.

From the Vehicle 1 results, it is noted that AdaBoost, XGBoost, kNN experienced slight improvements from optimization. However, RF and SVM performance significantly reduced from optimization, which could be because the hyperparameter values that were explored were not well suited to the data set. From the Vehicle 2 results, many classifiers performed very well, showing improvements on the approach with no optimization. Most models achieved scores very close to the (1,1) ideal result except for AdaBoost and GaussianNB, which is consistent with the results obtained previously. The Vehicle 1 results were not as strong as those for Vehicle 2.

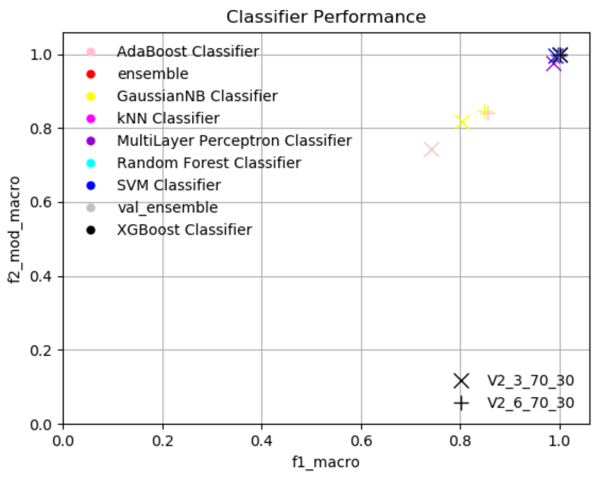

Figure 11. Vehicle 1 random sampling with hyperparameter optimization - training results

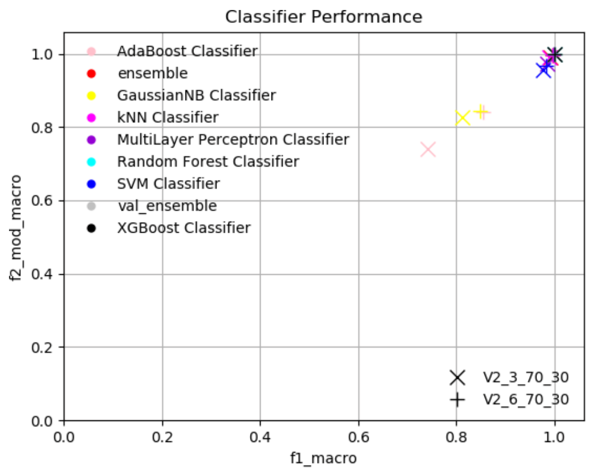

Figure 12. Vehicle 1 random sampling with hyperparameter optimization - testing results

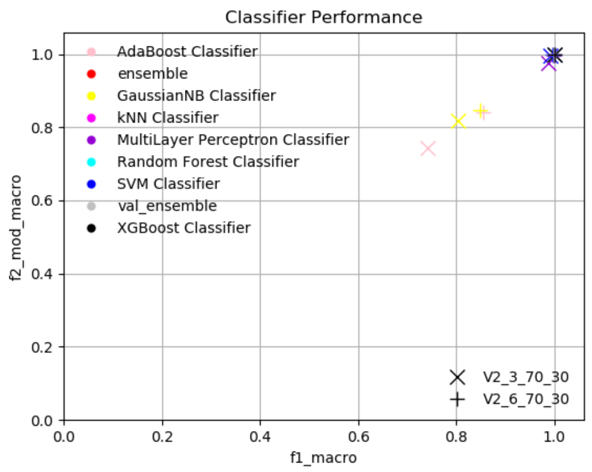

Figure 13. Vehicle 2 random sampling with hyperparameter optimization - training results

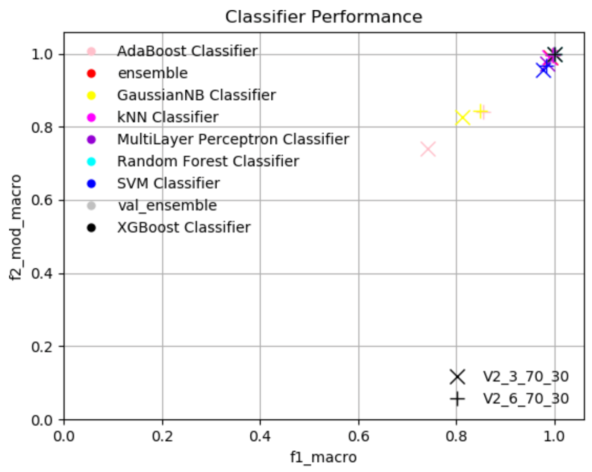

Figure 14. Vehicle 2 random sampling with hyperparameter optimization - testing results

## 7.2. Blind failure results

The following section contains information regarding the runs of the combined (Vehicles 1 and 2) GT data. To test how well the classifiers will perform on unseen data, entire failure events were withheld for testing, meaning groups of failure and deteriorated data were completely unseen by the classifiers in the training. The configurations include 3 and 6 hour deteriorated time periods, as well as 2 or 3 withheld failure events for testing.

### 7.2.1. Blind failure results without hyperparameter optimization

Table 7 in section 5.2.1 describes the number of points associated with each class in all the configurations training and testing data set. Figure 15 displays the scores from the training, while Figure 16 shows the results of testing of each classifier.

Promising training results were obtained from some classifiers, but then poor performance ensued in testing. This behavior was also seen previously in the two class problem (Cheung et al., 2020). Extremely poor performance from all classifiers resulted on the withheld failures in testing. The F1 score is consistently higher than the F2 score in training, indicating that the failure and deteriorated classes were not predicted well. Near zero values for F2 scores in testing means that the classifiers were classifying all points as healthy. There did not seem to be any clear and consistent difference between the classifiers trained with 2 vs 3 failure incidents, especially since the classifiers' performance on the unseen failures in testing was very poor all around.

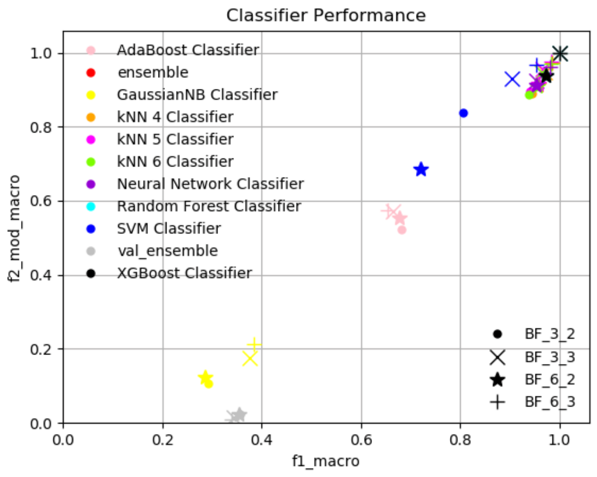

Figure 15. Combined Vehicle 1 and 2 data – Blind failure training results

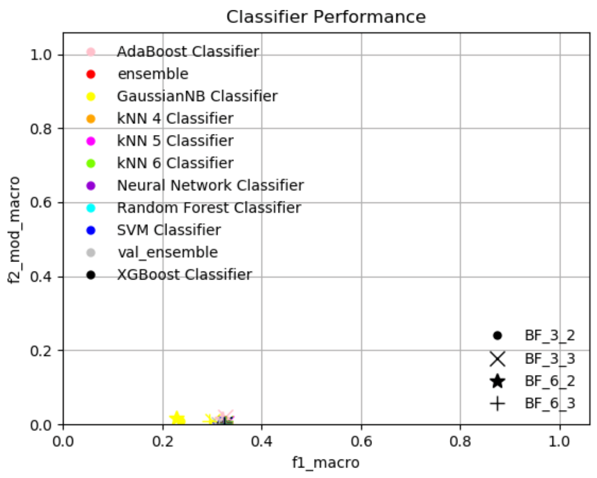

Figure 16. Combined Vehicle 1 and 2 data – Blind failure testing results

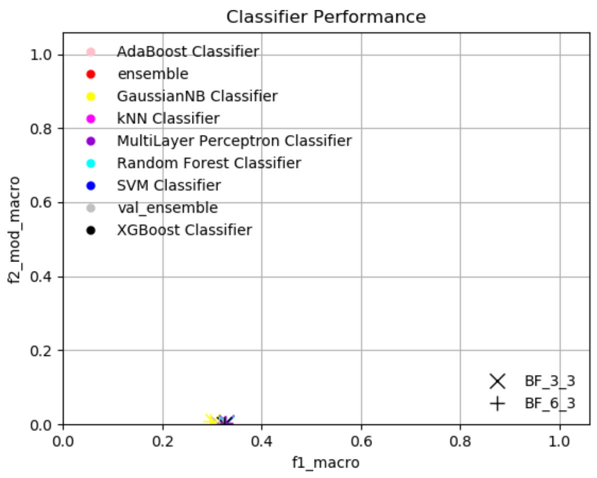

Figure 18. Combined Vehicle 1 and 2 data – Blind failure with hyperparameter optimization – testing results

#### 7.2.2. Blind failure results with hyperparameter optimization and leaders

The addition of hyperparameter optimization in the blind failure data selection strategy was also explored, but only for 3 withheld failure events for testing. Figure 17 and Figure 18 display the scores from the training and testing of each classifier.

The hyperparameter optimization had virtually no effect on the classifier performance. Similar to the results without optimization, there were some good classifiers that performed well in training, but the testing results were poor for all classifiers with F1 scores consistently at 0.3 and near zero F2 scores.

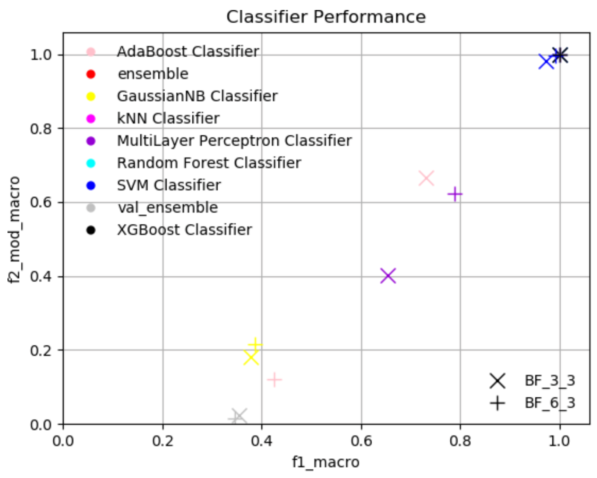

Figure 17. Combined Vehicle 1 and 2 data – Blind failure with hyperparameter optimization – training results

### 7.3. Results of cross-validation inspired approach

Cross-validation is typically used as a technique to evaluate how a classifier will perform on unseen data while still using the entirety of the data to train the final classifier. In this work, the intent was to modify the method to build multiple classifiers that may be more diverse and combine their predictions in an ensemble. This section presents the results from the CV-inspired approach which included hyperparameter optimization, SMOTE to oversample for deteriorated and failure classes, and leaders for the healthy class.

To begin, some of the intermediate results from the individual models built before the ensemble were inspected. The classifiers that had been successful during training in the past, like the Random Forest and XGBoost models continued to classify very well on the training folds but were still unsuccessful at classifying the withheld fold. However, the classifiers that did not typically excel in training, sometimes performed more reasonably on the testing fold. The confusion matrices from three examples are shown in Figure 19.

An ideal classifier would have non-zero values on the main diagonal and zero values in all the other cells. The AdaBoost and Neural Networks tend to over classify the data as healthy, and the GaussianNB Classifier very successfully classifies the deteriorated data. These examples are quite different from the majority of the classifiers which classify almost all the test set data as healthy.

After creating an ensemble from 28 classifiers using a decision table, the ensemble included the predictions of the following four classifiers to predict the status of the GT system:

- GaussianNB classifer 2 (testing on Fold 2)
- Neural Network classifier 3 (testing on Fold 3)
- XGBoost classifier 1 (testing on Fold 1)
- XGBoost classifier 2 (testing on Fold 2)

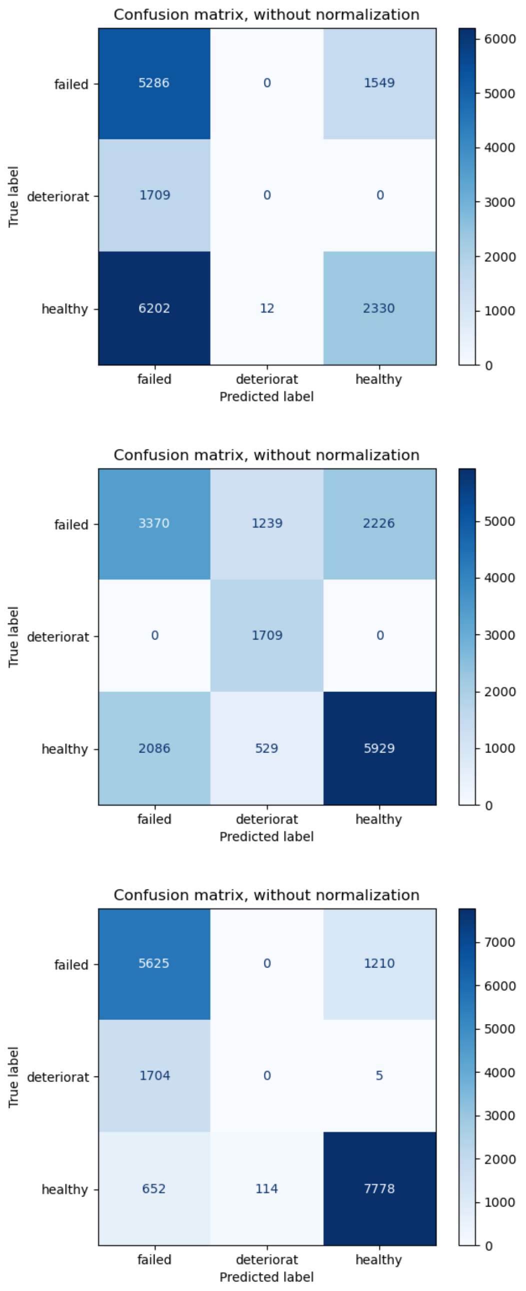

Figure 19. AdaBoost classifier 3, GaussianNB classifier 4, and Neural Network classifier 1 Testing

It is interesting to note that none of the classifiers from the 4th group of folds (Fold 4 used for testing) were selected, and that the GaussianNB classifier was selected even though it does not typically predict as well as some of the other classifiers.

In Figure 20, the results from the training and testing of the final decision table ensemble model are shown. The results from training were very successful with only one point being misclassified. However, the testing results seem to continue

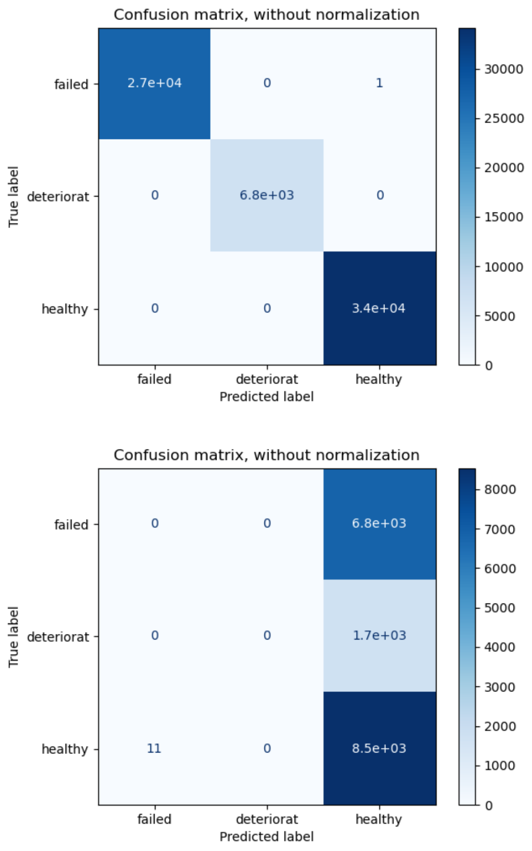

Figure 20. Ensemble results for training (top) on Folds 1-4 and testing (bottom) on Fold 5

to suffer, with almost all the data points classified as healthy, despite the attempts to diversify the classifiers, optimize the hyperparameters and reduce the imbalance of data.

Figure 21 plots the F1 and F2 scores for the ensemble from

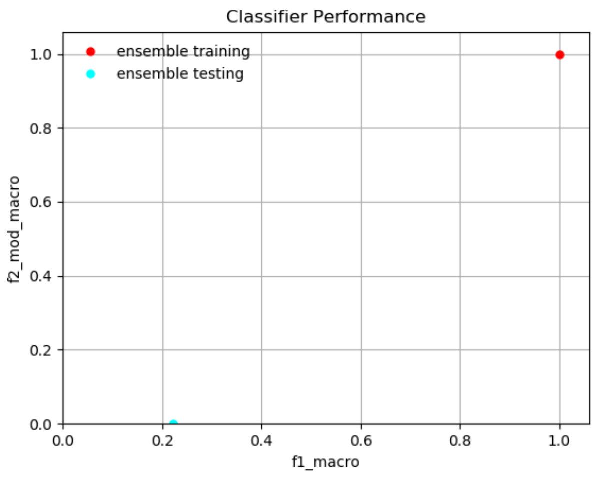

Figure 21. Training and testing scores for ensemble in cross-validation inspired approach

this approach after training and testing, again reflecting the almost perfect performance during training but poor performance in testing.

### 7.4. Discussion

Random sampling for constructing the training set, where examples of all failure incidents were included in training, led to high performing models, as noted by the high macro F1 and macro F2 scores. XG Boost and Random Forest models perform extremely well, and the ensembles also perform similarly. While improvement upon weaker individual classifiers is expected in the ensembles due to the reduction of the variance of the estimates, the contribution of the different methods depends on many factors related to the individual error distributions. For this reason, it is useful to examine the relationship between the ensemble and the contributing methods.

Optimization of hyperparameters seemed to provide some benefit based on the results obtained for random sampling. These results are very promising as they indicate that providing samples of all available data in training can lead to models that can provide some early warning of known failure types.

The training approach of keeping data from an entire failure incident excluded from the training data, what is referred to here as ‘blind failure’, led to very poor performing models in the three-class problem. Similar results were seen using this approach with classifiers in a two-class problem (Cheung et al., 2020), so these results are not all that surprising. In the two class problem, anomaly detection models could be leveraged in the blind failure approach to generate reasonably accurate ensemble models, however those anomaly detection models are not applicable in a three-class problem.

The CV-inspired approach merged aspects of the random sampling and blind failure approaches. There were some promising results obtained in training with an ensemble model but unfortunately those results did not translate into good results in testing.

## 8. Concluding Remarks

In many real-world applications, a severe class imbalance exists between available healthy system data and failed system data. To add to the complexity, the full scope of possible system failures is not covered by training data and vehicles of the same type may not be identical. In the efforts to achieve system models that are sufficiently robust and flexible to overcome these challenges, investigating different data selection strategies for model training has been a key focus area.

Using gas turbine system data from two vehicles which included 7 failure incidents and 76 predictor variables, a variety of classification models of the GT system were developed in a three-class problem. Three approaches to selecting training data were used. The first followed a traditional method of randomly selecting data points from all data according to a target ratio between training and testing data for each data class. The second data selection strategy was to consider data related to failure incidents as a whole and select certain incidents to include in training, and the remaining ones to be unseen in testing. The third approach was a cross-validation inspired approach, separating data into folds but training and testing models based on failure incidents. In addition to investigating training and data selection strategies, the effect of hyperparameter optimization was explored as well as the effect of varying the time period of the deteriorated class.

Similar to the results obtained when the problem was approached as a two-class problem (Cheung et al., 2020), high performing models were achieved with the first approach of randomly selecting data from all data for training, in particular the Random Forest, XG Boost classifiers, as well as ensemble models. Very poor results were obtained using the blind failure approach for all classifiers. The CV-inspired approach merged aspects of the random sampling and blind failure approaches. There were some promising results obtained in training with an ensemble model but unfortunately those results did not translate into good results in testing.

This paper describes the comprehensive results that were obtained using the various approaches and combinations, highlighting the respective benefits and limitations. Promising results were obtained using the random sampling approach, indicating that providing samples of all available data in training can lead to models that can provide some early warning of known failure types.

It should be noted that comprehensive studies like what was done in this work can be useful in other failure modeling or fault detection applications. Understanding the impact of the statistical properties of the data set, as well as understanding the variations in the classifiers are both important. They provide further knowledge of the problem, and can be used to study the consistency and reliability of the models built. Exploring different classifier types, data sampling techniques, and hyperparameter optimization are all ways to complete a thorough study of any modeling problem.

As has been noted in previous work, it is important to point out that one of the weaknesses in the GT data and the analysis that has been carried out in this work is how the initial labeling of the data into the three classes is carried out, as the assigned labels have a significant impact in model training.

### 8.1. Recommended Future Work

Overall, there seems to be opportunity in each method to improve the consistency of the classifiers and resilience to new unseen failure data. In combination with trying to access more failure data which would help the classifiers prepare for more failure types, some of the future work that may be explored include:

- Consider the use of other metrics in place of F measures for evaluating the classifiers, such as the Mathews correlation coefficient.
- Explore hyperparameters more thoroughly and how to select the appropriate ranges of values to explore in optimization.
- Further research in intelligent oversampling and under sampling.
- Implement alternative algorithms to the leader algorithm for prototype selection.
- Investigate methods to improve the labeling of the data into classes.
- Given that there is data available from two vehicles, investigate how successful the models trained on one vehicle's data can be applied to another vehicle.

## ACKNOWLEDGMENT

This work was supported by Canada's Department of National Defence and Defence Research and Development Canada.

## REFERENCES

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, *16*(1), 321–357.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (p. 785–794). New York, NY, USA: Association for Computing Machinery.

Cheung, C., To, D., & Valdés, J. (2020). Failure modeling in a gas turbine system: Combining classification with anomaly detection models for two data selection strategies. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 842–849).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273-297.

De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). Deap: A python framework for evolutionary algorithms. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation* (p. 85–92). New York, NY, USA: Association for Computing Machinery.

Hartigan, J. A. (1975). *Clustering Algorithms* (99th ed.). USA: John Wiley & Sons, Inc.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). New York, NY: Springer.

Hertel, L., Collado, J., Sadowski, P., Ott, J., & Baldi, P. (2020). Sherpa: Robust hyperparameter optimization for machine learning. *SoftwareX*, *12*, 100591.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (p. 338-345). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kohavi, R. (1995). The power of decision tables. In *European Conference on Machine Learning* (pp. 174–189).

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, *18*(1), 559–563.

Omohundro, S. M. (1989). *Five balltree construction algorithms* (Tech. Rep. No. TR-89-063).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536.

Schapire, R. E. (2013). Explaining AdaBoost. In *Empirical Inference* (pp. 37–52). Springer.

## BIOGRAPHIES

**Catherine Cheung** holds a MASc in Aerospace Studies (2002) and BASc in Engineering Science (1999) from the University of Toronto. She is a Senior Research Officer in the Aerospace Structural Integrity group at the National Research Council Canada in Ottawa. Her recent science and technology contributions have focused on usage and load monitoring research for helicopters and other structures using machine learning techniques. She is a member of the Vertical Flight Society HUMS technical committee.

**Calista Biondic** is completing a BASc in Engineering Science in the Aerospace option at the University of Toronto (2022). She completed a 16-month work term at NRC Aerospace in Structural integrity as a research assistant contributing to research in machine learning models for fault detection, helicopter health management, and various components of full-scale testing. Her research interests are in supervised classification for system state estimates, data selection and manipulation strategies, and structural damage predictions via loads estimation.

**Zouhair Adam Hamaimou** is completing a BASc in Engineering Science in the Aerospace option at the University of Toronto in Toronto, Ontario, Canada (2023). He is currently involved in numerous research projects at NRC Aerospace related to various Health and Usage Monitoring Systems technologies. His research interests lie in the field of Machine

Learning where he wishes to apply his knowledge to optimize prediction algorithms for aerospace among other applications.

**Julio J. Valdés** has a PhD in Mathematics and Physics. He has worked at research institutes and universities in Canada, Germany, Czechoslovakia, Spain and Cuba. Since 2001 he is with the National Research Council Canada and is Adjunct Professor at the Universities of Ottawa and Carleton. His scientific activities cover artificial intelligence, computer science, mathematics, earth, environmental and space sciences. His topics of interest are data analytics, machine learning, computational intelligence (neural networks, evolutionary computation, fuzzy logic, probabilistic reasoning, rough sets), pattern recognition, digital image and signal processing and data visualization. He also works on computational intelligence and data analytics of earth, environmental and space sciences. He is a Senior Member of the IEEE and has conducted multidisciplinary research in a broad variety of domains covering several branches of engineering (aerospace, civil and geological), medicine and astrophysics. His record includes 300 publications in books, journals, conference papers and technical reports.

## APPENDIX

Table 9. List of hyperparameters of classifiers

| Classification method | Hyperparameter | Type | Original value | Optimization range |
|---|---|---|---|---|
| SVM | C | continuous | 1.0 (default) | [0.4,1.2] |
| | gamma | continuous | ‘scale’ (default) | [0.5,4.0] |
| | shrinking | choice | True (default) | [True, False] |
| | decision_function_shape | choice | ‘ovr’ (default) | [’ovo’, ’ovr’] |
| | class_weight | choice | {0.0:1, 1.0:20, 2.0:150} | [’balanced’, {0.0:1, 1.0:30, 2.0:200}, {0.0:1, 1.0:20, 2.0:240}] |
| GaussianNB | var_smoothing | continuous | 1e-9 (default) | [1e-8, 1e-10] |
| Random forest | n_estimators | discrete | 100 (default) | [100, 150] |
| | criterion | choice | ’gini’ (default) | [’gini’, ’entropy’] |
| | max_features | choice | auto’ (default) | [’auto’, ’log2’, None] |
| | bootstrap | choice | True (default) | [True, False] |
| | class_weight | choice | None (default) | [None, ’balanced’, ’balanced_subsample’, {0.0:1, 1.0:30, 2.0:200}, {0.0:1, 1.0:20, 2.0:240}] |
| AdaBoost | n_estimators | discrete | 50 (default) | [30, 80] |
| | learning_rate | continuous | 1 (default) | [0.5,4.0] |
| NN MLP | hidden_layer_sizes | choice | (15, 10, 5) | [(100,), (100,10,), (15,10,5), (100,10,10)] |
| | activation | choice | relu’ (default) | [’identity’, ’logistic’, ’tanh’, ’relu’] |
| | solver | choice | adam’ (default) | [’lbfgs’, ’sgd’, ’adam’] |
| | alpha | continuous | 1.00E-05 | [1e-4, 1e-6] |
| | learning_rate_init | continuous | 0.001 (default) | [1e-3, 1e-4] |
| | max_iter | discrete | 500 | [250, 800] |
| kNN | n_neighbors | discrete | [4,6] | [4, 9] |
| | weights | choice | uniform’ (default) | [’uniform’, ’distance’] |
| | algorithm | choice | auto’ (default) | [’ball_tree’, ’kd_tree’, ’auto’] |
| | p | choice | 2 (default) | [1, 2, 3, 4] |
| XGBoost | booster | choice | gbtree’ (default) | [’gbtree’, ’dart’] |
| | eta | continuous | 0.3 (default) | [0.1, 1] |
| | max_depth | discrete | 9 | [6, 10] |
| | max_delta_step | discrete | 6 | [0, 10] |
| | tree_method | choice | auto’ (default) | [’auto’, ’approx’, ’hist’] |

Table 10. List of optimized hyperparameters for several configurations

| Classification method | Hyperparameter | Random sampling V1 3_70_30 | Random sampling V1 6_70_30 | Random sampling V2 3_70_30 | Random sampling V2 6_70_30 | Blind failure 3_3 | Blind failure 6_3 |
|---|---|---|---|---|---|---|---|
| SVM | C | 0.789 | 0.793 | 0.706 | 0.726 | 0.947 | 0.622 |
| | gamma | 0.613 | 0.500 | 0.525 | 0.549 | 2.586 | 1.573 |
| | shrinking | FALSE | FALSE | TRUE | TRUE | FALSE | TRUE |
| | decision_ function_ shape | ovo’ | ’ovr’ | ’ovr’ | ovo’ | ’ovr’ | ovo’ |
| | class_weight | {0.0: 1, 1.0: 20, 2.0: 240} | {0.0: 1, 1.0: 30, 2.0: 200} | {0.0: 1, 1.0: 30, 2.0: 200} | {0.0: 1, 1.0: 30, 2.0: 200} | ’balanced’ | {0.0: 1, 1.0: 30, 2.0: 200} |
| GaussianNB | var_smoothing | 3.86E-10 | 2.52E-09 | 3.12E-09 | 2.18E-09 | 6.31E-09 | 2.88E-09 |
| Random forest | n_estimators | 113 | 108 | 121 | 129 | 109 | 111 |
| | criterion | ’entropy’ | ’entropy’ | ’entropy’ | ’entropy’ | gini’ | ’entropy’ |
| | max_features | None | None | None | ’log2’ | None | None |
| | bootstrap | TRUE | FALSE | TRUE | FALSE | FALSE | FALSE |
| | class_weight | {0.0: 1, 1.0: 20, 2.0: 240} | {0.0: 1, 1.0: 30, 2.0: 200} | {0.0: 1, 1.0: 30, 2.0: 200} | {0.0: 1, 1.0: 20, 2.0: 240} | {0.0: 1, 1.0: 20, 2.0: 240} | ’balanced’ |
| AdaBoost | n_estimators | 60 | 66 | 47 | 45 | 56 | 61 |
| | learning_rate | 1.341 | 1.301 | 1.340 | 2.764 | 1.912 | 3.948 |
| NN MLP | hidden_layer _sizes | (100) | (100, 10,10) | (100, 10) | (15, 10, 5) | (100, 10) | (100,10, 10) |
| | activation | ’tanh’ | ’relu’ | ’tanh’ | ’tanh’ | ’identity’ | ’identity’ |
| | solver | ’adam’ | ’adam’ | ’adam’ | ’adam’ | sgd’ | ’adam’ |
| | alpha | 4.42E-05 | 3.76E-05 | 3.33E-05 | 2.78E-05 | 2.84E-05 | 1.30E-05 |
| | learning_ rate_ init | 4.50E-04 | 7.89E-04 | 8.82E-04 | 4.71E-04 | 8.02E-04 | 8.46E-04 |
| | max_iter | 633 | 386 | 679 | 344 | 444 | 263 |
| kNN | n_neighbors | 4 | 4 | 6 | 8 | 7 | 5 |
| | weights | ’distance’ | ’distance’ | ’distance’ | ’distance’ | ’distance’ | ’distance’ |
| | algorithm | ’ball_tree’ | ’ball_tree’ | ’ball_tree’ | ’kd_tree’ | ’ball_tree’ | ’kd_tree’ |
| | p | 1 | 1 | 1 | 1 | 1 | 1 |
| XGBoost | booster | ’gbtree’ | ’dart’ | ’gbtree’ | ’gbtree’ | ’dart’ | ’gbtree’ |
| | eta | 0.451 | 0.588 | 0.239 | 0.975 | 0.301 | 0.291 |
| | max_depth | 7 | 7 | 9 | 9 | 6 | 9 |
| | max_ delta_ step | 1 | 1 | 8 | 3 | 7 | 8 |
| | tree_method | ’approx’ | ’approx’ | ’approx’ | ’hist’ | ’hist’ | ’hist’ |