Predictive Maintenance of Air Pressure System using Boosting Trees: A Machine Learning Approach

Mrugank M Akarte[1], N. Hemachandra[2]

[1] Vishwakarma Institute of Technology, Pune
mrugankakarte13@gmail.com

[2] Department of Industrial Engineering and Operations Research,
Indian Institute of Technology Bombay, Mumbai. nh@iitb.ac.in

Abstract

This paper presents an approach to minimize maintenance cost of Air Pressure System (APS) in Scania trucks using a supervised machine learning approach, gradient boosting trees. The maintenance cost includes the cost of breakdown due to unobserved faulty part and the cost of needless check on satisfactory parts. The dataset consists of highly skewed and unbalanced data with faulty APS cases, referred as positive class and remaining cases as negative class. Skewness is reduced by Log transformation while imbalanced data is addressed by assigning additional class weight to a rare event. Comparison of two models trained using XGBoost, one using equal class weights and one using unequal class weights for positive and negative class is presented. Findings suggest that classifier with unequal class weights with a suitable threshold value achieves better performance compared to classifier using equal class weights with a threshold value of 0.5 in case of unbalanced dataset.

Keywords: Predictive maintenance, Air Pressure System, Cost sensitive learning, Unbalanced dataset, Boosting trees, Classification

1. Introduction

Predictive maintenance plays an important role in cost saving for a company. Unexpected breakdowns of parts cost a lot of money because it can cause unnecessary delays and can become a non-value adding factor for a product. Predictive maintenance using machine learning techniques can help prevent such scenarios by predicting the failures/breakdowns so that some alternatives can be arranged while the machine is under maintenance. The undetected faulty parts lead to breakdowns which have major contribution in the maintenance cost. This paper discusses gradient boosting method to minimize the maintenance cost by predicting failures in Air Pressure Systems (APS) in Scania trucks. The dataset titled 'APS failure at Scania Trucks' is used which was donated by Tony Lindgren and Jonas Biteus on September 2016 to UCI Machine Learning repository[1]. The dataset consists of data collected from heavy Scania trucks in everyday usage. The APS is utilized for various functions in a truck, such as braking and gear changes. The dataset is divided into positive and negative classes. The positive class consists of component failures for a specific component of APS system. The negative class consists of trucks with failures for components not related to APS. For a given problem, the cost of failing to detect the faulty part has 50 times the cost of doing pointless check on acceptable parts. The total maintenance cost was calculated as sum of total cost required for misclassified trucks.

1.1 Contributions

- The problem is to minimize the maintenance cost for misclassified Scania trucks where the cost of missing a faulty truck is 50 times more than cost required to do unnecessary check.
- The dataset is skewed and highly imbalanced as ratio of positive to negative cases in training set is 1:59. To address the problem of skewness, log transformation is done on training and test set independently. Cost sensitive learning is used to overcome the problem of class imbalance.
- XGBoost is used to train the model and hyperparameters are tuned using 5-fold cross-validation on training set. To show the importance of cost sensitive learning and its impact on maintenance cost, cost insensitive classifier is also trained. The results of two classifier are compared on test set.
- Cost sensitive classifier results in much lower cost compared to cost insensitive classifier.

2. Literature Review

Christopher Gondek, Daniel Hafner, and Oliver R. Sampson[2] (2016), approached the same problem using random forest. They used median to replace the missing values and the data was not normalized. New features were calculated as distance to other distributions, mean distribution of positive examples, mean distribution of negative examples, normal distribution with mean = 5 and standard deviation = 1.5, mirrored normal distribution. To overcome the problem of class imbalance, the value for threshold was modified and a value of 95% was found to be most suitable. They were able to achieve an average cost of 0.6, which was calculated as total maintenance cost divided by total number of trucks.

Vıtor Cerqueira, Fabio Pinto, Claudio Sa, and Carlos Soares[3] (2016), solved the problem using boosting trees with meta-features and oversampling technique. The meta-features were generated using box-plots, clustering based outlier ranking and LOF: identifying density based local outliers. SMOTE, oversampling techniques was used to overcome the problem of class imbalance and 10-fold cross validation was used to tune the parameters.

G. Weiss[4] (2004) discuss use of cost sensitive learning and boosting as one of the methods to address the problem associated with rarity of class. This paper uses cost sensitive learning and gradient boosting trees to train the model.

3. Problem Statement

The objective is to minimize the maintenance cost of misclassified trucks. The cost matrix for the given problem is shown below.

Table 1: Cost matrix for binary classification

		True class	
		Positive	Negative
Predicted Class	Positive	0	C1
	Negative	C2	0

Where,
C1 = Cost for type1 error (Cost of an unnecessary check)
C2 = Cost for type2 error (Cost of missing a faulty truck)

In context of APS, C1 refers to the cost that an unnecessary check needs to be done by mechanic at the workshop and C2 refers to the cost of missing a faulty truck, which may cause breakdown. The values of C1 and C2 were set to $10 and $500 respectively by the experts at Scania.

The total cost is calculated as:

$$Total\ cost = X1 * C1 + X2 * C2 \quad \text{... (1)}$$

Where,
X1 = Number of False Positive (Type1 error)
X2 = Number of False Negative (Type2 error)

In this paper total normalized cost is used to evaluate the model which is calculated as follows:

$$Total\ normalized\ cost = \frac{X1 * C1 + X2 * C2}{C1 + C2} \quad \text{... (2)}$$

3.1. Methodology

The overall methodology for predicting a faulty APS is shown in Figure 1. This is for a given cost of binary classification problem. Important steps are as follows:

1. Data Preprocessing on training and test set.
2. Training model using XGBoost
3. Hyperparameter tuning using Cross-validation (CV)
4. Predict and evaluate total cost on test set.

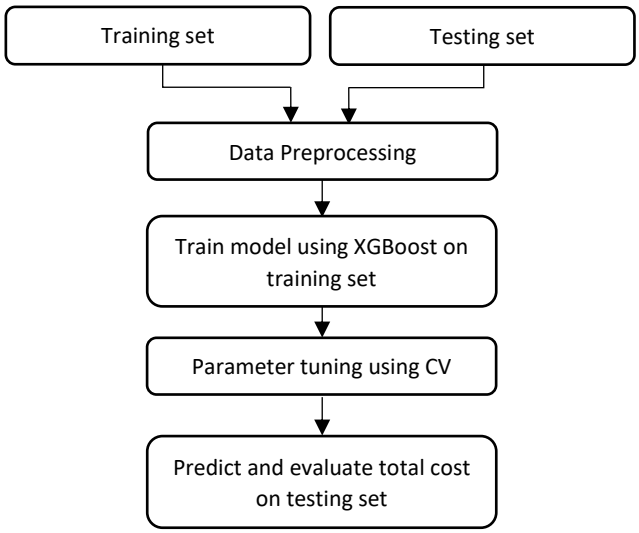

Figure 1: Overall methodology for predicting APS failures

3.2. Review of concepts

Following section gives a brief introduction of the algorithm XGBoost and Cross-validation used to solve the problem.

3.2.1 XGBoost

XGBoost is a scalable end-to-end machine learning system for tree boosting, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges[5]. It is an open source library available in C++, R, Python, and Juila.

XGBoost is faster and more efficient implementation of boosting algorithm. The idea behind the boosting algorithm is to combine weak learners into a strong learner in an iterative manner. Initially a model is fit to data, then another model is fit to the residuals, which corrects the error from previous model. A new model is then created by combining the original model and a model fit to the residuals. This process is repeated until a strong learner is obtained. The classifier designed for this problem is based on this with unequal class weights. The classifier outputs the probability that APS is faulty, and this is then labeled as positive or negative based on threshold value which will be determined based on misclassification cost C1 and C2.

3.2.2 Cross-validation

k-Fold Cross-validation is a technique for measuring how well the machine learning model will perform on an independent test set[6]. The training data set is divided into k sets (also known as folds), and out of 'k' sets 'k-1' sets are used for training the algorithm, and the left-out set is used as a validation set. This process is repeated until every set has the chance to be a validation set. While generating the k-folds, stratified sampling ensures same proportion of observations across each category in each of the set.

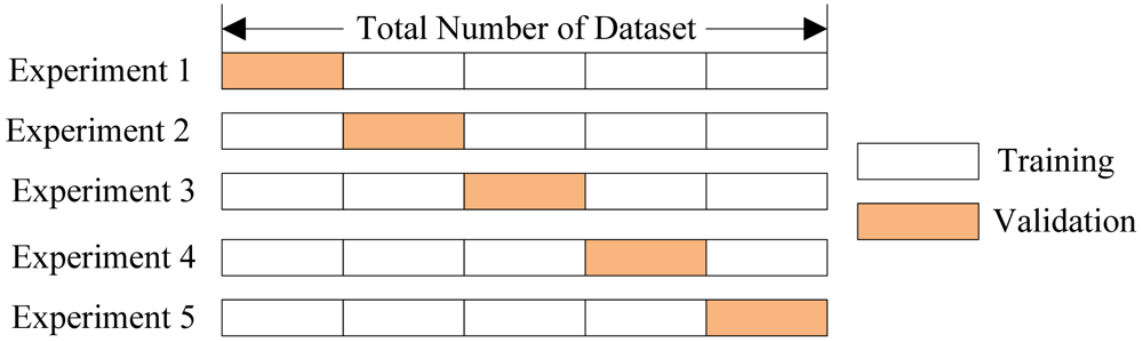

Figure 2: Cross validation (source: https://www.kaggle.com/dansbecker/cross-validation)

The above figure represents a 5-fold cross validation where in experiment 1, first set is used as validation set and rest are used as training set. Once all the experiments are carried out, the cross-validation error is then calculated as mean of errors across all the sets. The advantage of using cross-validation is that it gives accurate measure of the model as all of training data is used as validation set. When we do 70-30 or 80-20 train-validation split of data, it is possible that the validation set selected is easy to evaluate. In such cases the model will perform much better than expected, giving us the false results. The drawback is that it takes more time as it estimates models once for each set. So, depending on size of dataset, a proper method must be selected for evaluation. For this problem the cross-validation approach was used because of highly imbalanced dataset.

4. Data Preprocessing

The training set consists of 60000 samples and 171 anonymized attributes with ratio of 1:59 positive to negative class as shown in Fig.3 and test set consists of 16000 samples. Following graph shows percentage distribution of positive and negative class in training set. Class 0 refers to negative class and Class 1 refers to positive class.

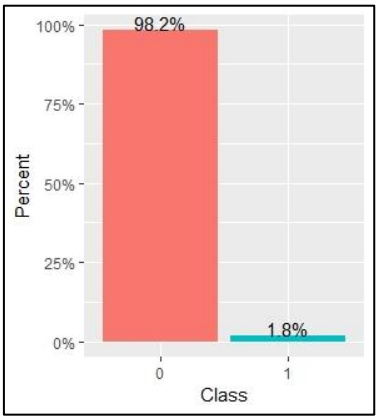

Figure 3: Class distribution for APS dataset
(Red: negative class, Blue: positive class)

The dataset has up to 82% missing values. There are seven attributes with over 70% missing values, which are removed to reduce the noise and size of the dataset. Since most of the attributes were positively skewed, the missing values were replaced with median. The data was log transformed (y = log(x+1)) to have distribution similar to normal distribution. Other transformations like square-root, cube-root, squared were tried but the log transformation showed the distribution more like normal distribution. Following sample histograms (Fig.4 and Fig.5) shows the distribution of attributes *ag_002* and *dp_000* before and after log transformation.

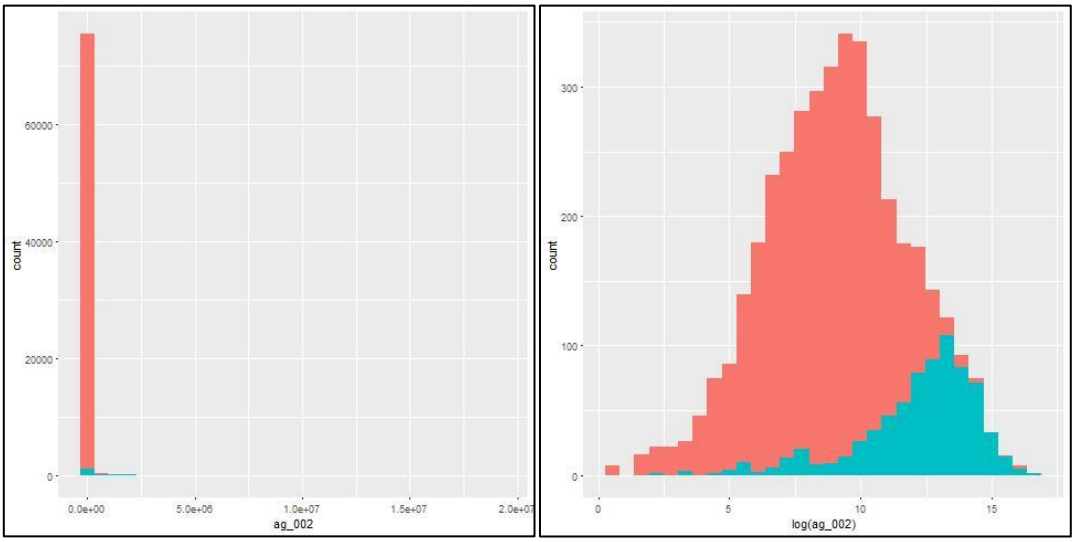

Figure 4: Histogram of attribute ag_002 before and after log transformation (Red: negative class, Blue: positive class)

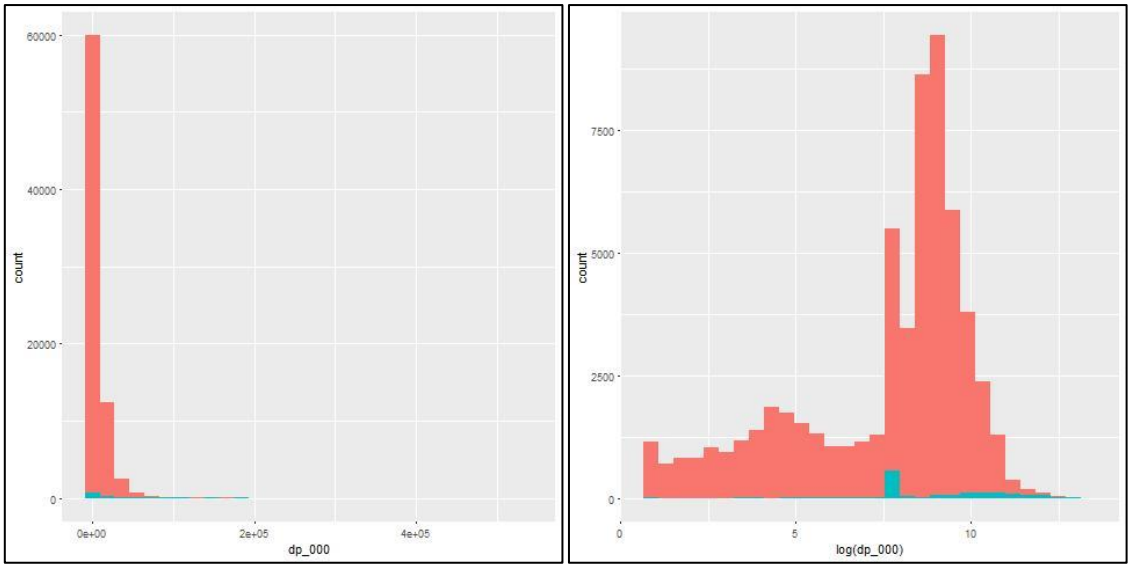

Figure 5: Histogram of attribute dp_000 before and after log transformation (Red: negative class, Blue: positive class)

5. Design of a Cost Sensitive Classifier for APS

XGBoost is used to train the model by minimizing the total cost as formulated in problem statement. The hyperparameters *eta*, *subsample*, *colsample_bytree*, *lambda* and *nrounds* were tuned using 5-fold stratified cross validation. The values for above parameters used in cross-validation are tabulated in Table 2.

Table 2: Parameter values for cross-validation

Hyperparameters	Range of values used in cross-validation
eta	0.3, 0.1, 0.05, 0.03, 0.01
Subsample	1, 0.8, 0.6
Colsample_bytree	1, 0.8, 0.6
lambda	0, 1

The optimal values of the hyperparameters for Model 1 using above parameters were found to be; *eta* = 0.05, *subsample* = 0.6, *colsample_bytree* = 1, *lambda* = 1 and *nrounds* = 398. Other parameters were set to their default values; *booster*: gbtree, *max_depth* = 6, *min_child_weight* =1 and *objective* = binary:logistic.

XGBoost allows to take care of unbalanced class by setting up a parameter called '*scale_pos_weight*', which takes care of rare event by increasing the cost it fails to correctly predict them. The value of the parameter was set as ratio of number of negative instances to number of positive instances. Another parameter '*early_stopping_rounds*', was set to 100, to prevent overfitting to training set. This parameter stops the training if the total cost on validation set does not decrease after certain iterations, in this case after 100 iterations.

J.Friedman (1996), formulated an equation, as shown below, to minimize misclassification error by Bayes rule[7].

$$y(x) = 1\left(f(x) \geq \frac{l0}{l0 + l1}\right) \quad \ldots (3)$$

Where,

l0 = cost incurred for misclassifying class 0
l1 = cost incurred for misclassifying class 1

Using the same formula (Eq. 3), the threshold value for classification was set to *0.0196*; above this value the class was labeled as positive else negative class. The mean cross-validation normalized cost across five folds for tuned parameters was found to be 14.77.

5.1 Design of a Cost insensitive classifier for comparison

To show the importance of using cost sensitive loss function in case of unbalanced classes, a new model (Model 2) is trained and tuned for same parameters shown in Table 2. The values for C1 and C2 were set to 1 in the cost function. The optimal values for the hyperparameters of this classifier were found to be; *eta* = 0.03, *subsample* = 0.8, *colsample_bytree* = 0.8, *lambda* = 1, *nrounds* = 498 and other parameters were set to their default values as mentioned above. The parameter '*scale_pos_weight*' was not used for this model and the value for '*early_stopping_round*' was set to 100. The threshold value for this cost insensitive classification was set to 0.5. The mean cross-validation normalized cost across five folds for tuned parameters was found to be 31.2.

The following table (Table 3) shows comparison of various metrics for the two tuned models on 5-fold cross-validation; one with unequal class weights (Model 1) and one with equal class weights (Model 2). Since the objective is to minimize the total maintenance cost and the cost for false negatives is 50 times more than cost for false positives, it is important to maximize recall. Model 1 dominates the Model 2 in terms of recall and hence has lower cost.

Table 3: Metric comparison for equal and unequal class weights classifier on cross-validation set

	Total normalized Cost	F1 score	Recall	Precision
Unequal class weights (Model 1)	**14.77**	0.5736	**0.9518**	0.4105
Equal class weights (Model 2)	31.2	0.8298	0.7607	0.9129

5.2 Computation Platform

All the computations were done on Win10, 8GB RAM and Intel 5th Gen i5. The models were trained and tuned using cross-validation in R v3.5.1 using XGBoost library v0.71.2. The histograms were plotted using ggplot2 library v3.0.0 in R v3.5.1.

6. Results and Conclusions

If all the samples are predicted as positive class, then the total normalized cost would be 306.37 and the total cost would be $156250 and if all the samples are predicted as negative class then the total normalized cost would be 367.64 and the total cost would be $187500. The confusion matrix for classifier using unequal class weights in shown in Table 4 and confusion matrix for classifier using equal class weights is shown in Table 5.

Table 4: Confusion matrix for classifier using unequal class weights

		True class	
		Positive	Negative
Predicted Class	Positive	363	414
	Negative	12	15211

Table 5: Confusion matrix for classifier using equal class weights

		True class	
		Positive	Negative
Predicted Class	Positive	285	13
	Negative	90	15612

Using a classifier with unequal weights (Model 1), the total normalized cost and total cost for test set were found to be 19.88 and $10140 respectively. The total normalized cost and total cost for classifier using equal weights (Model 2) was calculated to be 51.5 and $45130 respectively. The total normalized cost for classifier using equal weights (Model 2) is 2.59 times more than cost for classifier using unequal weights (Model 1).

Comparison of suitable metrics like Recall, Precision and F1 score for both the classifier is tabulated in Table 6.

Table 6: Metric comparison for equal and unequal class weights classifier on test set

	Total normalized Cost	F1 score	Recall	Precision
Unequal class weights (Model 1)	**19.88**	0.6302	**0.968**	0.4671
Equal class weights (Model 2)	51.5	0.8469	0.76	0.9563

Since the classifier using unequal class weights is good at minimizing false negative, this result in high false positives. Thus, this model has high recall but low precision. On the other hand, the classifier with equal weights gets biased to majority class (negative class), hence this classifier is good at minimizing false positives which results in high false negatives. Thus, this classifier has high precision but low recall value. F1-score weighs precision and recall value equally, but in our case, as mentioned earlier, the cost for false negatives is 50 times more than the cost for false positives, recall is more important than precision. As a result, classifier with unequal class weights has low F1 score compared to classifier using equal weights.

The importance of using cost sensitive loss function in case of unbalanced dataset was shown by modifying the weights of minority class and by using suitable threshold value for classification.

Using machine learning can help company save lot of money by predicting faults which can lead to breakdowns. Tuning of parameters and using a cost sensitive loss function helps to reduce the problem of bias in unbalanced class and gives much better results.

Further study to reduce the cost can be done by using different algorithms like random forests, neural networks or ensemble models. Sampling techniques like oversampling, undersampling and SMOTE and creating new features based on existing features can be studied to see if sampling and feature engineering can help in reducing the cost.

7. References

1.Tony Lindgren and Jonas Biteus (2016). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks]. Irvine, CA: University of California, School of Information and Computer Science.

2. Gondek C., Hafner D., Sampson O.R. (2016) Prediction of Failures in the Air Pressure System of Scania Trucks Using a Random Forest and Feature Engineering. In: Boström H., Knobbe A., Soares C., Papapetrou P. (eds) Advances in Intelligent Data Analysis XV. IDA 2016. Lecture Notes in Computer Science, vol 9897. Springer, Cham

3. Cerqueira V., Pinto F., Sá C., Soares C. (2016) Combining Boosted Trees with Metafeature Engineering for Predictive Maintenance. In: Boström H., Knobbe A., Soares C., Papapetrou P. (eds) Advances in Intelligent Data Analysis XV. IDA 2016. Lecture Notes in Computer Science, vol 9897. Springer, Cham

4. G. Weiss. Mining with rarity: A unifying framework. SIGKDD Explorations, 6(1):7-19, 2004.

5. Tianqi Chen, Carlos Guestrin, XGBoost: A Scalable Tree Boosting System, June 2016, available at: https://arxiv.org/abs/1603.02754 and accessed on August 18, 2018.

6. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, 'Resampling Methods' *Introduction to Statistical learning*, Springer, 2013, pp. 176-186

7. Friedman, J.H. Data Mining and Knowledge Discovery (1997) 1:55. https://doi.org/10.1023/A:1009778005914