U.Porto Journal of Engineering, 3:2 (2017) 34-48
ISSN 2183-6493
DOI 10.24840/2183-6493_003.002_0004

Received: 7 April, 2017
Revised: 24 April, 2017
Published: 29 September, 2017

Power Transformer Failure Prediction: Classification in Imbalanced Time Series

Eduardo e Oliveira[1*], Vera L. Miguéis[2], Luís Guimarães[3], José Borges[4]
[1]INESC TEC, Department of Industrial Engineering and Management, Faculty of Engineering University of Porto, Porto, Portugal (gei10037@fe.up.pt), *corresponding author; [2]INESC TEC, Department of Industrial Engineering and Management, Faculty of Engineering University of Porto, Porto, Portugal (vera.migueis@fe.up.pt) ORCID 0000-0001-7831-9140; [3] INESC TEC, Department of Industrial Engineering and Management, Faculty of Engineering University of Porto, Porto, Portugal (guimaraes.luis@fe.up.pt) ORCID 0000-0002-6293-5691; [4]INESC TEC, Department of Industrial Engineering and Management, Faculty of Engineering University of Porto, Porto, Portugal (jlborges@fe.up.pt) ORCID 0000-0001-9946-5614

Abstract
This paper describes a study on applying data mining techniques to power transformer failure prediction. The data set used consisted not only on DGA tests, but also in other tests done to the transformer's insulating oil. This dataset presented several challenges, such as highly imbalanced classes (common in failure prediction problems), and the temporal nature of the observations.
To overcome these challenges, several techniques were applied for prediction and better understand the dataset. Pre-processing and temporality incorporation in the dataset is discussed. For prediction, a 1-class and 2-class SVM, decision trees and random forests, as well as a LSTM neural network were applied to the dataset.
As the prediction performance was low (high false-positive rate), we conducted a test to ascertain if the amount of data collected was sufficient. Results indicate that the frequency of data collection was not adequate, hinting that the degradation period was shorter than the periodicity of data collection.

Author Keywords. Data Mining, Failure Prediction, Time Series, Power Transformer

Type: Research Article

1. Introduction

Asset management plays a central role in ensuring that operational and investment costs are minimized, and ensuring the quality of products or services (i.e., low amount of defects, no energy shortages, low delivery times, etc.). Being able to predict failure in physical assets/equipment can be a powerful tool to aid asset management, as it can help determine the best time for maintenance actions, minimizing the amount of such actions while improving the availability of the equipment. This paradigm of using failure predictions to determine maintenance actions is called predictive (or condition-based) maintenance. In this paper we focus on predicting failure in power transformers, an equipment critical to electricity distribution. Failure in power transformer can lead to power shortages, so predicting when it will happen can help devising maintenance actions that prevent these shortages from happening. We use data that is already collected for safety inspection and non-predictive maintenance. This data was collected in order to evaluate the need for a change of isolating oil due to continuous or sudden degradation. The degradation of the oil may be caused by malfunctioning of the power transformer. These malfunctions may be a symptom of an

upcoming failure, and therefore could be useful to predict upcoming failure, allowing maintenance to be done preemptively, and avoiding failure.

There are three tests that are usually conducted on oil samples in order to determine the oil's degradation: the dissolved gas analysis (DGA), oil quality (OA) analysis (namely physical properties and non-gaseous components), and a furans analysis, which has the objective of determining the condition of the paper isolating the core of the transformer through the presence of furanic elements in the oil (Wang, Vandermaar, and Srivastava 2002).

In order to determine the best model to predict failure in the power transformers of this dataset, we run tests on several machine learning algorithms, namely decision trees, random forests, support vector machines (SVM), and neural networks.

1.1. Data Description

The attributes obtained from each test are listed in Table 1. In the 'Global' column are the attributes common to all instances. An instance corresponds to a test performed on an oil sample collected an oil sample. From the DGA test, in addition to the gases represented by their chemical composition, there are three ratios that are currently used by practitioners to determine the cause of the failure after it has occurred (Miranda and Castro 2005). These ratios are also listed in Table 1. In the 'Oil Quality' column, there are some mechanical and chemical measurements obtained. From the furans test, the concentration of the listed chemical compounds is registered. Aside the global attributes, all the others are numerical.

Global	DGA	Oil Quality	Furans
Serial Number, Date, Label	H2, CH4, C2H4, C2H6, C2H2, CO, CO2, O2, N2, C2H2/C2H4, C2H4/H2, C2H4/C2H6	Color, Density, Viscosity, I.F.T., D.T., A.I, H2O, TG DELTA, PC1	I5HMF, I2FOL, I2FAL, I2ACF, I5MEF

Table 1: Measurements (attributes) obtained from each test

The test database used (referred from here onwards as dataset) is comprised by a total of 15.031 instances (tests) with 30 attributes (variables measured), including the global ones, where the label is either 1 or 0. The instance was labeled as 1 if there was a registered failure between the date of that instance and the date of the next instance with the same serial number, and 0 otherwise. Therefore 1 signifies an upcoming failure, and 0 a normal functioning period.

One of the characteristics of the data set, common to all maintenance problems, is the fact that there are many more instances of normal functioning than of failure. In this data set in particular, the reason of instances labeled 0 for those labeled 1 is 88 (in other words, there is 88 times more instances labeled 0 than labeled 1). Therefore, this is a highly imbalanced data set, and the algorithms and metrics used have to reflect this characteristic, in order to produce relevant results.

1.2. Outline of the paper

The remaining of the paper is organized as follows: in section 2 we present related literature; in section 3 we will describe the pre-processing; section 4 presents the evaluation procedures and a brief description of the algorithms used; in section 5 we present the results of the tests and discusses them; section 6 finalizes with the presentation of the main conclusions of this paper.

2. Related Literature

In this section some literature on the topic of using data mining (DM) techniques in power transformer data and fault prediction is exposed.

Huang, Huang, and Sun (2012) reviewed different DM approaches to oil-immersed power transformer maintenance. It focused more on diagnosis, identifying the cause of failure after the failure has occurred, with brief mentions to prognosis of faults. It only considered literature focusing on data from DGA tests.

Wang (2004) developed a 2-part model for fault forecasting in oil-immersed power transformers. The first part consisted in using a modified gray model to forecast the gas levels in the DGA, while the second part performed diagnosis for the forecasted values, effectively predicting faults. The use of a gray model was due to the lack of points (4 points per transformer), which may represent an obstacle to the use of traditional forecasting methods, such as regression. Although the results are positive, some questions remain as if this method is the best when more information is available.

Fei et al. (2009) developed a support vector machine (SVM) to forecast dissolved gas levels, optimizing its parameters using particle swarm optimization (PSO). The justification of using a PSO to optimize SVM parameters is due to being time efficient while being easier to operate than genetic algorithms (GA). The selected Kernel was RBF, and an SVM regression model was developed for each gas. Data set contained daily measurements. It compared its performance with a gray model and an artificial neural network (ANN), achieving better results than those.

Trappey et al. (2015) developed a system that focuses on real-time monitoring of key parameters and uses DM to detect transformers' potential failure under various operating conditions. The system consisted in a PCA for dimensionality reduction followed by a feed-forward ANN trained with back-propagation. It only uses data from DGA, and tries to predict the status of a power transformer in "Normal", "Waiting Acknowledgment" and "Abnormal". The data set had 500 points, which in this case represent the last measurements of a given transformer (in other words, each point corresponds to a different transformer). Therefore, it does not consider the data as a time series. Also, the data set is balanced between the 3 possible status. It performs an 80-20 split evaluation.

Wang (2004) and Fei et al. (2009) focus on predicting the continuous gas levels, and then making a discrete prediction about failure. Trappey et al. (2015) discretizes the condition of the power transformer. The last paper does not consider the temporality of the data. All the reviewed papers focus only on the DGA analysis. Also, none of them makes reference to the imbalanced nature of the data, as they balance the data set beforehand.

This paper adds to the literature as it uses more tests than the DGA analysis to predict failure in the power transformers. Also, it deals explicitly with the imbalanced nature of the data set, by using weights in the observations during the training of the algorithms. It also considers the temporality of the data by considering values of the attributes from previous observations.

3. Pre-Processing

One important characteristic of this problem is that the tests conducted on oil samples were not done all at the same time. Some instances have only the values of DGA test, while others just the oil quality test. This led to a large amount of missing values, that made the utilization of the raw data impossible in some algorithms. Therefore, a pre-processing algorithm was considered in order to eliminate all missing values (see Figure 1). In the data set used, all transformers were identified with a serial number. In order to ease the pre-processing of the algorithms, the data set was first sorted in order by the serial number first, and then sorted

by date. Then the algorithm in Figure 1 was implemented. A brief description of this algorithm is that when it detects a missing value, it checks if there is a value from a previous observation of the same transformer, and if there is the missing value is replaced by that value. The intuition behind this algorithm stems from the assumption that, given the lag between tests, in the absence of information of the value at that time, the last known value is the best estimation. In other words, it assumes that the amount of tests of all types is similar. Although this was true for the DGA and oil quality (OA) tests, the furans tests were rarer than the other two (DGA had 13.335 tests, OA 12.963 and furans 8.074). This caused a very large amount of instances to be eliminated, possibly compromising the performance of algorithms that used this data. After validation from expert practitioners, it was decided to eliminate the attributes related with the furans test, as they compromised results.

ALGORITHM 1: Pre-Processing Algorithm

```
Data: Raw data
Result: Pre-Processed Data (No Missing Values)
for each attribute in data set do
    for each instance in data set do
        if value of dimension in current instance is a missing value and serial number of previous instance is the same then
            the value of this instance's dimension is equal to the previous instance's value
        else
            do nothing
        end
    end
end
Eliminate all instances with a missing value
End
```

Figure 1: Pre-processing algorithm used

From this process, the data set was reduced from 15.031 instances with 30 attributes, to 9627 instances with 25 attributes. There is a large reduction due to some power transformers only having one or two registered measurements, and/or only from one test.

3.1. Standardization

In addition to the process already mentioned, standardization of the data set was conducted in order to improve the speed and effectiveness of the data mining algorithms. The standardization formula used was the Z-score, $x_{new} = \frac{x-\mu}{\sigma}$, in order to prevent outliers from turning normal data into very small values. Mean and standard deviation are obtained from the data set.

3.2. Adding Temporality

In order to capture the temporal nature of the data, we opted to add attributes to the data corresponding to previous values of the collected values. In other words, if we select a time lag of 1 (i.e., we consider that the previous observation's value and the current observation's value are relevant), we add an attribute (e.g. H2_1) for each original attribute (e.g. H2), which corresponds to the value of the original attribute in the previous observation. Several time lags were considered. However, adding temporality this way leads to a reduction in the number of usable instances, as for the first *n* instances of each transformer (*n* equal to time lag) there are not *n* previous observations. As such, not very high time lags were considered (maximum of 6).

4. Algorithms Used and Performance Metrics

In this section it is presented a short description of the algorithms used in the exploratory analysis. The results of each of the algorithms described are in section 5.

4.1. Evaluation Metrics and Methods

Given the imbalance between the classes in the data set, it was necessary to use measures able to capture what is desired from the algorithms, leading to meaningful results.

Using accuracy (Equation 1), in a context where the data in very imbalanced will simply lead the used algorithms to assign all instances to the majority class, as that ensures that the fewest instances are misclassified, which are all the ones from the minority class, precisely what needs to be predicted.

It was desired to predict as many failures as we could, while not allowing the number of false positives to be very high. For this reason, the selected metrics were *recall* and *false-positive (FP) rate* (see Equations 2 and 3, respectively).

$$accuracy = \frac{number\ of\ instances\ classified\ correctly}{number\ of\ instances} \quad (1)$$

$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (2)$$

$$FP\ rate = \frac{False\ Positives}{False\ Positives + True\ Negatives} \quad (3)$$

Recall reflects how many failures the algorithm was able to detect from all the failures that occurred. FP rate reflects the amount of instances classified as leading to failure that were normal, relatively to all normal instances. The objective is to maximize recall while minimizing FP rate. Another metric, related to the two above is the F1 metric (Equation 4), which establishes a compromise between the two, and is used as an auxiliary metric when optimizing some algorithms.

$$F1 = \frac{2 * True\ Positives}{2 * True\ Positives + False\ Negatives + False\ Positives} \quad (4)$$

In addition to the use of this metrics, the evaluation process will use an adaptation of 10-fold cross validation to assess the performance of the algorithms. This modification is motivated by the fact that the instances can be grouped by serial number to create a time series, or sequence, and a standard, out-of-the-box, cross validation could "break" these sequences, compromising the integrity of the data. Therefore, instead of selecting 10% of the instances, as in standard 10-fold cross validation, it will be considered 10% of the serial numbers, ensuring that no sequence is "broken".

4.2. Decision Trees

Decision trees are learning methods that do not require an assumption of the population distribution, and are as such considered to be non-parametric. They can be used for both classification and regression. Their basic functioning consists in a divide-and-conquer approach, where a more complex problem is divided in simpler problems, and this strategy is applied recursively (Gama et al. 2012), creating a tree-like structure of hierarchical decisions (hence the name).

Structurally, a decision tree is composed by nodes and leaves, where in each node there is a decision based on one of the attributes and a threshold, and in the leaves the instances are assigned to a class or value.

The decision tree creation algorithm can be briefly described as, at each iteration, selecting the best attribute and the best threshold to divide the data, creating a node that represents that decision, followed by the creation of two new leaves that stem from the new node. This process is repeated until a stopping criterion is reached, which can be, for example, the purity of leaf nodes (how many instances in that leaf node were classified wrong), or the depth of

the tree (the length of the longest path from the first node to a leaf). Several criteria can be used to decide which attribute to make the split, but the two most common ones are information gain/ entropy, and the Gini index. The first one focuses on making the more general decisions first, as this gains more information (reduces entropy), while the second one is a measure of purity. The two most popular algorithms for decision tree induction are the C 4.5 (Quinlan 1993) and the CART (Breiman et al. 1984). In this study, the algorithm used will be CART due to being able to better deal with outliers and noise (Questier et al. 2005).

This algorithm has some hyper-parameters that should be optimized in order to improve the performance. To do that, a grid search was conducted, as according to Table 2. For optimizing these parameters, the first 1000 instances were selected as a tuning subset (while ensuring serial number integrity). The tuning was performed using a 10-fold cross validation.

Criteria	Maximum Number of Leaves	Maximum Depth
Gini	None	None
	2	5
Entropy	5	10
	10	20

Table 2: Decision tree grid search space

One of the main issues that come up after looking at the data is that there are many more observations where the model is not supposed to predict a failure (label equal to 0 – considered normal behavior by the transformer) than observations immediately preceding a failure (label equal to 1). There were around 88 times more observations of normal behavior than those with “faulty” behavior. As such, class weights were introduced in order to balance the classes and achieve useful results. The formula for the weights of erring a prediction in a certain class is inversely proportional to the class frequencies, and is given by Equation 5. The number of samples and number of classes is fixed.

$$weight(Y) = \frac{Nº\ of\ samples}{Nº\ of\ classes * Nº\ of\ samples\ of\ class\ Y} \quad (5)$$

The hyper-parameters that were deemed more relevant to tune for this problem were the criteria, the maximum number of leaves, and the maximum depth of the tree. The use of different criteria can change the approach to the problem and affect the results obtained, and as such should be considered in the grid search. The maximum depth of the tree enables us to select how complex we wish our decisions to be, as more depth means that more decision nodes can be "stacked" on top of each other. The maximum number of leaves represents in how many "groups" the data will be divided, with each "group" belonging to a class (note that there can be several "groups" with the same class). This parameter allows us to refine how precise the results will be, as more leaves will lead to higher purity in each one of them, and less leaves will make the leaves more prone to misclassification, but also less prone to noise and more capable of generalizing. In both the maximum number of leaves and the maximum depth, "None" means that no limit is imposed. The number of possible combinations are $2 * 4 * 4 = 32$.

The results of the grid search will be discussed in section 5.1.

4.3. Support Vector Machines (SVM)

Support Vector Machine is an optimization based machine learning algorithm that focuses on defining the best linear frontier that separates two classes is a data set. It does so by considering support vectors, which are the points from both classes closer to the other class, and defining a frontier that maximizes the distance to the support vectors from both classes.

In order to solve non-linear problems (i.e., that are not linearly separable), a kernel trick is used, where the dot product of instances in the original solution space is transformed using a kernel function to a space of higher dimension. Common kernels are the linear kernel ($k(x,y) = x^T y + c$), the polynomial kernel ($k(x,y) = (x^T y + c)^d$), and the radial basis function (RBF) kernel ($k(x, y = \exp(y\|x - y\|^2)$). For a tutorial of SVM, refer to Cristianini and Schölkopf (2002).

As in the decision tree case, a grid search was done in order to optimize the hyper-parameters of SVM. The search space can be seen in Table 3. Please note that the degree is only applicable when using the polynomial kernel, and the gamma only in the RBF kernel, so we have $5 * 1 + 5 * 3 + 5 * 3 = 5 * (1 + 3 + 3) = 35$ possible combinations.

C	Kernel	Degree	Gamma (γ)
100	Linear	2	0.001
1000	RBF	3	0.0001
2500	Polynomial	4	0.00001
5000			

Table 3: 2-Class SVM grid search space

The kernel functions used were already explained. The degree represents the degree of the polynomial kernel (*d*). Gamma can be defined intuitively as how far the influence of a single training example reaches, as it can be seen as the inverse of the radius of influence, with lower values meaning that an instance can influence many others. The C parameter represents the trade-off between misclassifying instances against the simplicity of the decision frontier. Smaller values of C mean a smoother frontier, and higher values lead to a more "over-fitting" frontier. In addition to this parameters, class weights were balanced the same way as in the decision tree case.

4.4. One-Class SVM - Novelty Detection

Chawla, Japkowicz, and Kotcz (2004), state that when negative examples vastly outweigh the positive ones, one-class learners trained on the positive class alone may lead to improved results. SVM has a one-class version for novelty detection, which was tested in this problem, training it on the positive samples. This version, instead of finding the best frontier between classes, tries to separate all points from the origin of the feature space, and maximizes the distance of this frontier to the origin (for more details see Scholkopf et al. (1999)).

The grid search was done using the same search space, but without the C hyper-parameter, for a total of 7 combinations. As it is a one-class algorithm, it does not have class weights.

4.5. Random Forest (RF)

Random forests (Breiman 2001) are an ensemble method, that focuses on constructing several small decision trees, fitting them to sub-samples of the data set and combining them in order to improve performance. Ensemble methods combine several learning algorithms in order to improve predictive performance. The results of the several algorithms are combined usually with a voting scheme, where the most voted class by the different algorithms is the output of the final model. There are several types of ensembles, and random forests are an example of bagging/ bootstrapping, where each model is trained on a random subset, and each model has an equal weight when voting.

The hyper-parameters used for random forests are the same as decision trees, as these are the base unit of this method. The only additions are the number of trees to use, and the maximum number of features to consider when looking for the best split. The number of

features to consider was set to the square root of the total number of features. Table 4 shows the space of the grid search. This led to $2 * 4 * 4 * 4 = 128$ combinations.

Criteria	Number of Trees	Maximum Number of Leaves	Maximum Depth
Gini	5	None	None
	10	2	5
Entropy	20	5	10
	50	10	20

Table 4: Random forest grid search space

4.6. Neural Networks – Long Short-Term Memory (LSTM)

Artificial Neural Networks (ANN) are a machine learning/ artificial intelligence approach to model complex functions, for both classification and regression. They consist on using several smaller units, called neurons, that comprise a linear combination of inputs, followed by a non-linear activation function. These units are organized in layers and interconnected, forming a network with the capability of modeling highly complex functions. In this paper, we use a version of neural networks that is capable of handling recurrence, i.e., capable of handling temporal/ sequential data, called Long Short-Term Memory (LSTM) networks. It is inherently able to handle high time gaps efficiently, by using a memory block comprised a central unit called cell, and other units to control the flow of the error called gates. The original formulation was in Hochreiter and Schmidhuber (1997). For further information on the more modern additions to LSTM, consider reading Greff et al. (2015).

The testing performed with the neural networks was a more complex procedure, as it not only involved optimizing hyper-parameters, but also the structure of the neural network (in this case the number of layers the network should have). In addition, a test on the sensitivity of the model to time lag was performed to see if the model could benefit of increased time lag when compared to the other models. These steps were done sequentially, i.e., first an optimization of the hyper parameters was performed using a grid search, then an optimization of the number of layers (using the best hyper parameters), and then a study on the influence of time lag (using the best hyper parameters and number of layers).

A LSTM neural network has as hyper parameters the batch size, which determines after how many instances the weights of the connections will be updated, the learning rate, which determines how much the weight will change at the end of each batch, and the number of epochs, i.e., the number of times the network is trained over the data set. The optimization algorithm used was the Adam algorithm (Kingma and Ba 2015).

As such, a grid search was conducted with the following parameter list in Table 5.

Batch Size	Learning Rate	Neurons	Epochs
20	10^{-1}	1	50
50	10^{-2}	10	100
100	10^{-3}	20	200
200	10^{-4}	50	300
	10^{-5}		

Table 5: LSTM grid search parameter list

The LSTM algorithm was implemented in Python using the Keras framework (Chollet 2015). A 5-fold cross validation performed. A time lag of 2 was chosen, based on previous experiments. The loss function chosen was binary cross-entropy, and is defined in Equation 6, where y_i represents the target/true values of the label (either 0 or 1), and $\hat{y}_i$ represents the predicted values.

$$\sum_{i}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \quad (6)$$

Then, in order to see if increasing the complexity of the machine learning algorithm would improve performance, tests using the parameters that performed the best in the first grid search were used with an increasing number of layers, from 2 to 10.

After that, in order to confirm if this more complex algorithm could make use of an increased time lag, tests on different time lags, using the network with the best performance, were done. The time-lag varied between 3 and 6.

Results of LSTM tests are presented in section 5.2.

5. Results

5.1. Tree-based and SVM-based algorithms

The tests consisted in using the adapted 10-fold cross validation (see section 4.1.) in each of the selected algorithms with the optimal hyper-parameters.

In addition to testing the different algorithms, they were also tested on several time lags, more precisely between 0 and 3, which represent either no time lag (the original data set), or the addition of the previous 3 values for each attribute. Note that increasing the time lag leads to a decrease in the number of available instances, as there is a need to eliminate missing values which will originate in the first instances of each transformer (same serial number).

Another test that was performed was to differ the scoring function when doing the grid searches, between the F1 metric, and the recall metric. F1 should be a more balanced metric, while recall should force more misclassification errors to increase the number of positive instances correctly classified. Results of the parameter optimization can be seen in Table 6.

The tests results can be seen in Table 7.

		Decision Trees			***SVM 2-Class***			***SVM 1-Class***		***Random Forests***			
	Time Lag\ Parameters	*Criteria*	*Max. Leaves*	*Max. Depth*	*C*	*Kernel*	*Degree/ Gamma*	*Kernel*	*Degree/ Gamma*	*Nº of Trees*	*Criteria*	*Max. Leaves*	*Max. Depth*
F1	*0*	Entropy	None	None	5000	Polynomial	3	Linear	-	10	Entropy	5	100
	1	Gini	None	None	100	Polynomial	3	Linear	-	10	Gini	5	50
	2	Gini	5	None	100	Polynomial	4	Linear	-	5	Entropy	5	100
	3	Gini	2	None	100	RBF	0,001	Linear	-	5	Entropy	5	None
Recall	*0*	Entropy	2	None	100	RBF	0,00001	Linear	-	10	Gini	2	20
	1	Entropy	2	None	100	RBF	0,00001	Linear	-	5	Gini	2	100
	2	Entropy	2	None	100	RBF	0,00001	Linear	-	5	Entropy	2	None
	3	Entropy	2	None	100	RBF	0,00001	Linear	-	5	Entropy	2	20

Table 6: Grid search results by scoring metric, algorithm used and time lag. Best parameters for each type of model, time lag and optimization criteria are presented

		Time Lag											
	Parameter Selection/Criteria	0			1			2			3		
		Error	Recall	FP-rate	Error	Recall	FP-rate	Error	Recall	FP-rate	Error	Recall	FP-rate
Decision Tree	F1	0,02	0,03	0,01	0,01	0,00	0,00	0,37	0,70	0,37	0,60	0,73	0,60
	Recall	0,52	0,71	0,52	0,65	0,86	0,66	0,48	0,86	0,48	0,63	0,70	0,64
SVM 1	F1	0,53	0,66	0,53	0,27	0,24	0,26	0,66	0,80	0,66	0,49	0,51	0,49
	Recall	0,53	0,66	0,53	0,27	0,24	0,26	0,66	0,80	0,66	0,49	0,51	0,49
SVM 2	F1	0,23	0,27	0,22	0,10	0,14	0,09	0,03	0,07	0,02	0,11	0,15	0,10
	Recall	0,44	0,62	0,44	0,33	0,68	0,33	0,35	0,77	0,35	0,29	0,66	0,29
Random Forests	F1	0,20	0,34	0,19	0,16	0,37	0,16	0,21	0,41	0,20	0,17	0,30	0,17
	Recall	0,23	0,42	0,23	0,34	0,56	0,34	0,35	0,63	0,35	0,38	0,45	0,38

Table 7: Exploratory analysis results. Evaluation metrics used are misclassification error, recall and false positive rate. The scoring criteria is related with the parameters in Table 6

5.1.1. Discussion

While evaluating these results, due to practical concerns, an algorithm is considered "successful" when its recall is above 0.60, and its FP-rate is below 0.4. Taking this into consideration, and consulting Table 6, we can see that the "successful" cases are decision trees when optimized to F1 and with time-lag 2, random forest when optimized to recall and time lag 2, and SVM 2-class, for time lags 1, 2 and 3, when optimized to recall. The greatest difference between FP-rate and recall is achieved by SVM 2-class with time lag 2. It can be argued that time lag 2 achieves the best results. The fact that result worsen from time lag 2 to 3 may be caused by the reduction in the number of observations that occurs when time lag is increased (see section 3.2).

When it comes to decision trees or random forest, it is unclear which is the best criteria (Gini or entropy), but it seems evident that limiting the number of leaves has a positive effect on performance, especially in recall, while maximum depth should not be controlled. This indicates that there is a need for complex decisions (high depth) and resistance to noise (limited number of leaves).

When analyzing the SVM algorithms, the 1-Class SVM was outperformed by all the other algorithms. It was, however, the most stable of all algorithms, with the same results when optimizing either F1 or recall. The 2-Class SVM was the most successful algorithms, which may come from its ability to model complex decision boundaries between two groups. In terms of hyper-parameters, the RBF kernel was the most successful, with a C value of 100 and low gamma (10^{-5}).

5.2. LSTM

5.2.1. Parameter Grid Search

For the sake of brevity, results will be presented as graphics representing the trends of the difference between recall and FP-rate given the different values of the parameters. In the following Figures (2, 3, 4, 5), each parameter is represented separately, with the difference over the other parameters averaged.

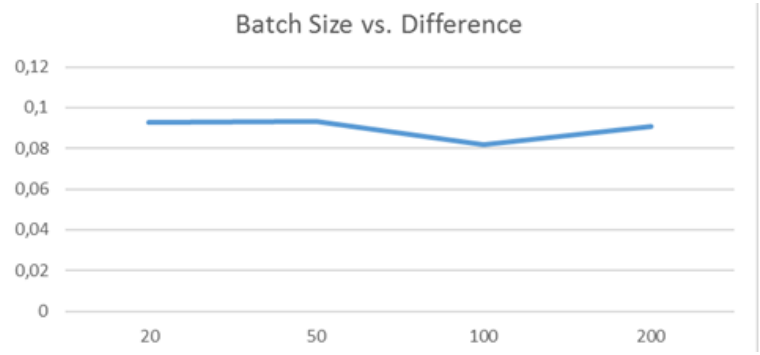

Figure 2: Difference between recall and FP-rate in vertical axis and batch size in horizontal axis

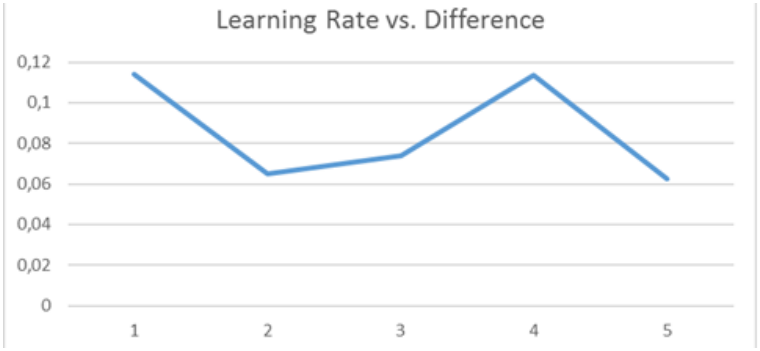

Figure 3: Difference between recall and FP-rate in vertical axis $log_{10}(leraning\ rate)$ in horizontal axis

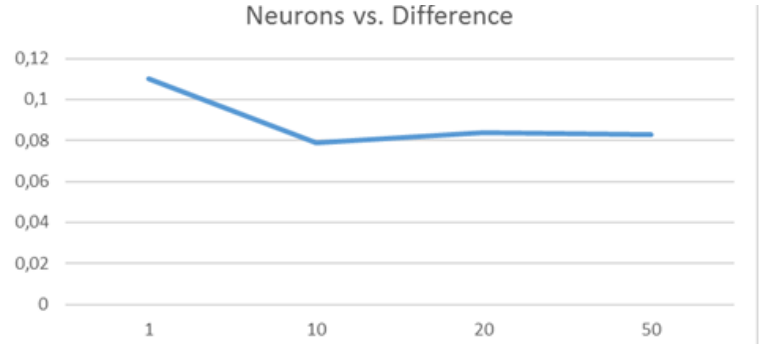

Figure 4: Difference between recall and FP-rate in vertical axis and number of neurons in horizontal axis

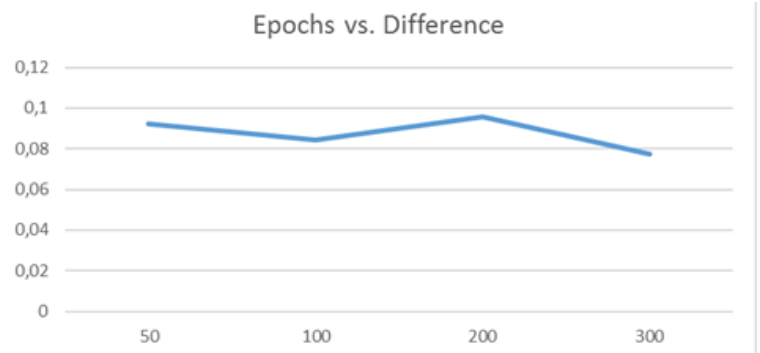

Figure 5: Difference between recall and FP-rate in vertical axis and number of epochs in horizontal axis

It can be seen that batch size and number of epochs had little impact in the performance of the model, with batch size 50 and epochs equal to 200 having very slightly better results. In terms of the number of neurons, 1 neuron had a greater difference in average than more neurons. Learning rate was more unstable, having the greatest average difference for 10^{-4}. These trends may not be very informative due to the lack of points, so conclusions should be

drawn carefully (computing more points would have been very time consuming). Also, combination effects between parameters have to be studied more carefully (an exploratory analysis of the combination has been conducted, but no visible effects were noted).

Despite all the trends, the best result from a combination of parameters came from batch size of 50, a learning rate of 10^{-5}, 50 neurons in the layer, and 200 epochs, with the maximum difference of 0.28, from a recall of 0.54 and a FP-rate of 0.26. As such, these were the parameters selected for the tests involving increasing the number of layers.

5.2.2. Layers Tests

Results from increasing the number of layers are presented in Table 8.

Layers	Recall	FP-Rate	Error	Times (seconds)	Difference
2	0,490	0,251	0,254	690,352	0,239
3	0,455	0,245	0,248	1054,709	0,210
4	0,363	0,218	0,222	1427,059	0,145
5	0,373	0,227	0,231	1777,480	0,146
6	0,390	0,232	0,236	1880,026	0,157
7	0,346	0,241	0,245	2294,013	0,105
8	0,348	0,238	0,243	2651,101	0,110
9	0,298	0,231	0,236	2757,684	0,067
10	0,330	0,211	0,216	3229,153	0,119

Table 8: Results from the layer addition tests. Batch size = 50, learning rate = 10-5, 50 neurons per layer and 200 epochs

It can be seen that adding more layers to the model, making it more complex will result in worsening the performance, while also increasing the running time of the training. As such, time lag tests will be done with a one-layered network, with the same parameters.

5.2.3. Layers Tests

Results from the time lag tests are presented in Table 9.

Time Lag	Recall	FP-Rate	Error	Times (seconds)	Difference
3	0,479	0,186	0,189	347	0,293
4	0,452	0,075	0,078	332	0,377
5	0,300	0,017	0,016	297	0,283
6	0,000	0,004	0,005	258	-0,004

Table 9: Results from time lag tests. Batch size = 50, learning rate = 10-5, 50 neurons, one layer and 200 epochs

From these results we can see that the LSTM network makes better use of more past information, since increasing the time lag from 2 to 4 actually improves performance in the LSTM case, while for the other classifiers, using a time lag greater than 2 led to worse performance. However, this improvement was not enough to surpass the other best classifier (the 2-class SVM - difference of 0.42).

5.2.4. Discussion

From these results, it was possible to conclude that trying to solve the problem using a more complex model, did not lead to an improved performance. The best performance obtained remained the one from the 2-class SVM, with a recall of 0.77, and a FP-rate of 0.35, which corresponds to a difference of 0.42. This led to the conclusion that the main issued was effectively with the data and not the method used. From this conclusion, two possibilities remained: either the data was not adequate, i.e., the attributes had nothing to do with the problem (in other words, the isolating oil degradation is not related with upcoming failures in

the machine, and could only be used for diagnosis), or the data was not collected at a right periodicity (i.e., tests have to be more frequent in order to obtain meaningful results). In order to verify which of these possibilities were true, a validation procedure on the periodicity of the data collection was devised, as will be explained in the next section.

5.3. Data Collection Periodicity Validation

The time between tests was around one year on average. However, the degradation that indicates an upcoming failure may only start to appear in a shorter period of time (e.g. a month). In order to test if the data collection periodicity was frequent enough, a validation procedure was devised.

Please remember that observations were labeled as 0 or 1 whether there was a failure between that observation and the next one. This does not reflect how much time between the current observation (the one labeled) and the actual failure occurred. We only know that is at most one year. Also, a failure can be detected between observations, so we have a specific date for the occurrence of the failure. Therefore, it should be possible to verify if this time lapse between the labeled observation and the actual failure influences the performance of the predictor.

The validation procedure consisted in using the best predictor possible to make predictions, and compare the time lapse between the prediction (made on each observation), and the failure. The predictor selected was a 2-class SVM with and RBF kernel and a gamma of 10^{-5} and a *C* of 100, trained on the whole data set. This will lead to an optimistic evaluation of the model, but what we are trying to do is to have the best possible predictions a model could theoretically make in the data, and see the effect of the time lapse on these predictions.

From this procedure, the average number of days between the prediction and the failure was higher for failures the predictor failed to detect (as it can be seen in Table 10).

True Value	Predicted	Average Time Lapse (days)	Std. Deviation
1	0	236,455	160,734
1	1	143,583	121,041

Table 10: Results from the data collection periodicity validation procedure

The performance of the optimistic model resulted in a recall of 0.85 and a FP-rate of 0.36. Given the fact that the average time lapse is higher for situations where the model fails to predict, and that the high level of recall and FP-rate, it can be concluded that the poor performance of the model comes from the periodicity of the data. However, the impact on performance does not manifest itself so much on the failure to predict, but in the amount of false positives the model led to. It appears that the long time lapses between the time the model had to make a prediction and the time the failures occurred, led the model to "strain" itself trying to predict too far into the future.

6. Conclusions and Future Work

In this paper, we tried to predict failure in power transformers. Doing so would provide a useful tool to improve maintenance policies. We tested several machine learning algorithms to a data set of insulating oil tests in order to find the best one for predicting failure. The data set, common to many maintenance problems, was highly imbalanced between situations with normal and faulty behavior. The fact that the data set was highly imbalance in conjunction with the fact that the observations had a temporal nature was a challenge, as common procedures to handle imbalanced data (e.g. under sampling and oversampling) do not take into consideration related observations, and as such we used weights in common data mining algorithms to deal with the imbalanced nature of the data.

The predictive model with the best performance was a SVM for two classes, which achieved a performance of recall equal to 77% and a false-positive rate of 35%. These results were not satisfactory since, given the high imbalance, as it corresponds to having around 77 true predictions for around 3000 false positives. This hinted that the degradation period that precedes the failure was shorter than the periodicity of data collection. In order to verify this, a test was performed, which indicates that the hypotheses should be true.

On a more technical level, adding more time lag after the 2 period threshold worsened performance for most classifiers, and more than a time lag of 4 also worsened performance for the LSTM case (which can make better use of past data). Trying more complex models, such as networks with more layers, also did not improve performance. Most likely, the LSTM network did not have a better performance because of the low frequency the data was collected, which not only affected the prediction of the failures, it also eliminated the need to use information from long time lags, decreasing the usefulness of the LSTM. Please note that time lags for the model are measured in number of instances and not real time. Finally, it was concluded that tests have to be made more frequently in order to enable the model to have a clearer picture of when and how the failure mode starts to show signals of occurring.

Compared to the literature reviewed, the data used in this paper, although it has more observations than in the other papers (approximately 9000 compared to 500), the data was collected less frequently. In other words, the time horizon of data collection was larger and more transformers were under analysis, which lead to increased number of observations, “hiding” the frequency that the observations were done (1 year). We can conclude that more than having a large number of observations, it is more relevant to ensure that they are collected in an adequate frequency when compared to the degradation period.

In terms of future work around this topic, an obvious one would be to repeat these tests but with a data set with more frequent tests. Another approach that has not been tested would be to consider alternative ways of incorporating temporality in the data, for example using time series data mining techniques to represent the data from each transformer as a sequence. This, however, may still be affected by the frequency of data collection.

References

Breiman, L., Jerome Friedman, Charles J. Stone and R. A. Olshen. 1984. *Classification and regression trees*. Monterey, CA: Wadsworth and Brooks.

Breiman, Leo. 2001. "Random forests". *Machine Learning* 45 (1):5–32. Accessed April 7, 2017. DOI: 10.1023/A:1010933404324.

Chawla, Nitesh V., Nathalie Japkowicz and Aleksander Kotcz. 2004. "Editorial: Special issue on learning from imbalanced data sets". *ACM SIGKDD Explorations Newsletter* 6 (1):1–6. Accessed April 7, 2017. DOI: 10.1145/1007730.1007733.

Chollet, François. 2015. “Keras.” GitHub. Accessed January 31, 2017. https://github.com/fchollet/keras.

Cristianini, Nello and Bernhard Schölkopf. 2002. "Support vector machines and kernel methods: The new generation of learning machines". *Artificial Intelligence Magazine* 23 (3):31–42. Accessed April 7, 2017. DOI: 10.1609/AIMAG.V23I3.1655.

Fei, Sheng-wei, Ming Jun Wang, Yu-bin Miao, Jun Tu and Cheng-liang Liu. 2009. "Particle swarm optimization-based support vector machine for forecasting dissolved gases content in power transformer oil". *Energy Conversion and Management* 50 (6):1604-1609. Accessed April 7, 2017. DOI: 10.1016/j.enconman.2009.02.004.

Gama, João, André Ponce de Leon Carvalho, Katti Faceli, Ana Carolina Lorena and Márcia Oliveira. 2012. *Extração de Conhecimentos de Dados: Data Mining*. Edited by Manuel Robalo. 1st ed. Lisboa: Edições Sílabo, Lda.

Greff, Klaus, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink and Jurgen Schmidhuber. 2016. "LSTM: A search space odyssey". *IEEE Transactions on Neural Networks and Learning Systems* (99):1-11. Accessed April 7, 2017. DOI: 10.1109/TNNLS.2016.2582924.

Hochreiter, Sepp and Jurgen Jürgen Schmidhuber. 1997. "Long short-term memory". *Neural Computation* 9 (8):1735–1780. Accessed April 7, 2017. DOI: 10.1162/neco.1997.9.8.1735.

Huang, Yann-Chang, Chao-Ming Huang and Huo-Ching Sun. 2012. "Data mining for oil-insulated power transformers: An advanced literature survey". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2 (2):138–148. Accessed April 7, 2017. DOI: 10.1002/widm.1043.

Kingma, Diederik P. and Jimmy Lei Ba. 2015. "Adam: A method for stochastic optimization". International Conference on Learning Representations 2015: ICLR 2015, San Diego, USA, May 7-9, 2015. Accessed April 7, 2017. https://arxiv.org/abs/1412.6980.

Miranda, Vladimiro and Adriana Rosa Garcez Castro. 2005. "Improving the IEC table for transformer failure diagnosis with knowledge extraction from neural networks". *IEEE Transactions on Power Delivery* 20 (4):2509–2516. Accessed April 7, 2017. DOI: 10.1109/TPWRD.2005.855423.

Questier, F., R. Put, D. Coomans, B. Walczak and Y. Vander Heyden. 2005. "The use of CART and multivariate regression trees for supervised and unsupervised feature selection". *Chemometrics and Intelligent Laboratory Systems* 76 (1):45–54. Accessed April 7, 2017. DOI: 10.1016/j.chemolab.2004.09.003.

Quinlan, J. Ross. 1993. *C4.5: Programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Scholkopf, Bernhard, Robert Williamson, Alex Smola, John Shawe-Taylor and John Platt. 1999. "Support vector method for novelty detection". *Proceeding NIPS'99: Proceedings of the 12th International Conference on Neural Information Processing Systems,* 582–588. Accessed April 7, 2017. MIT Press. http://dl.acm.org/citation.cfm?id=3009740.

Trappey, Amy J. C., Charles V. Trappey, Lin Ma and Jimmy C. M. Chang. 2015. "Intelligent engineering asset management system for power transformer maintenance decision supports under various operating conditions". *Computers and Industrial Engineering* 84:3–11. Accessed April 7, 2017. DOI: 10.1016/j.cie.2014.12.033.

Wang, M. H. 2004. "Grey-extension method for incipient fault forecasting of oil-immersed power transformer". *Electric Power Components and Systems* 32 (10):959–975. Accessed April 7, 2017. DOI: 10.1080/15325000490257999.

Wang, M., A. John Vandermaar and K. D. Srivastava. 2002. "Review of condition assessment of power transformers in service". *IEEE Electrical Insulation Magazine* 18 (6):12-25. Accessed April 7, 2017. DOI: 10.1109/MEI.2002.1161455.

Acknowledgements

This paper was done under Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020", which is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).