A machine learning approach for predictive maintenance for mobile phones service providers

Conference Paper · January 2017

DOI: 10.1007/978-3-319-49109-7_69

CITATION
1

READS
1,889

4 authors, including:

Anna Corazza
University of Naples Federico II
75 PUBLICATIONS **678** CITATIONS
SEE PROFILE

Francesco Isgrò
University of Naples Federico II
77 PUBLICATIONS **703** CITATIONS
SEE PROFILE

Roberto Prevete
University of Naples Federico II
67 PUBLICATIONS **545** CITATIONS
SEE PROFILE

A machine learning approach for predictive maintenance for mobile phones service providers

A. Corazza, F. Isgrò, L. Longobardo, R. Prevete

Abstract The problem of predictive maintenance is a very crucial one for every technological company. This is particularly true for mobile phones service providers, as mobile phone networks require continuous monitoring. The ability of previewing malfunctions is crucial to reduce maintenance costs and loss of customers. In this paper we describe a preliminary study in predicting failures in a mobile phones networks based on the analysis of real data. A ridge regression classifier has been adopted as machine learning engine, and interesting and promising conclusion were drawn from the experimental data.

1 Introduction

A large portion of the total operating costs of any industry or service provider is devoted to keep their machinery and instruments up to a good level, aiming to ensure a minimal disruption in the production line. It has been estimated that the costs of maintenance is the range 15-60% of the costs of good produced [14]. Moreover about one third of the maintenance costs is spent in not necessary maintenance; just as an example, for the U.S. industry only this is a $60 billion each year spent in unnecessary work. On the other hand an ineffective maintenance can cause further loss in the production line, when a failure presents itself.

Anna Corazza
DIETI, Università di Napoli Federico II e-mail: anna.corazza@unina.it

Francesco Isgrò
DIETI, Università di Napoli Federico II e-mail: francesco.isgro@unina.it

Luca Longobardo
DIETI, Università di Napoli Federico II e-mail: luc.longobardo@studenti.unina.it

Roberto Prevete
DIETI, Università di Napoli Federico II e-mail: roberto.prevete@unina.it

Predictive maintenance [14, 10] attempts to minimise the costs due to failure via a regular monitoring of the conditions of the machinery and instruments. The observation will return a set of features from which it is possible in some way to infer if the apparatus are likely to fail in the near future. The nature of the feature depend, of course, on the apparatus that is being inspected. The amount of time in the future that the failure will arise also depends on the problem, although we can state, as a general rule, that the sooner a failure can be predicted, the better is in terms of effective maintenance.

In general the prediction is based on some empirical rule [23, 17, 19], but over the last decade there has been some work devoted to apply machine learning [6, 22, 5] techniques to the task predicting the possible failure of the apparatus. For instance, a Bayesian network has been adopted in [9] for a prototype system designed for the predictive maintenance of non-critical apparatus (e.g., elevators). In [12] different kind of analysis for dimensionality reduction and support vector machines [7] have been applied to rail networks. Time series analysis has been adopted in [13] for link quality prediction in wireless networks. In a recent work the use of multiple classifiers for providing different performance estimates has been proposed in [18].

An area where disruption of service can have a huge impact on the company sales and/or the customer satisfaction is the one of mobile phone service providers [8, 4]. The context considered in this work is a the predictive maintenance of national mobile phone network, that is being able to foresee well in advance if a cell of the network is going to fail. This is very important as the failure of a cell can have a huge impact on the users' quality of experience [11], and to prevent them makes less likely that the user decides to change service provider.

In this paper we present a preliminary analysis on the use of a machine learning paradigm for the prediction of a failure on a cell of a mobile phones network. The aim is to predict the failure such in advance that no disruption in the service will occur, lets say, at least a few hours in advance. A failure is reported among a set of features that are measured every quarter of an hour. The task is then to predict the status of the feature reporting the failure within a certain amount of time. As for many other predictive maintenance problems given we are dealing with a very large amount of sensors [15].

The paper is organised as follows. Next section describes the data we used and reports some interesting properties of the data that have been helpful in designing the machine learning engine. The failure prediction model proposed is discussed in Section 3, together with some experimental results. Section 4 is left to some final remarks.

2 Data analysis

To predict failure, considered data is obtained by monitoring the state of base transceiver stations (also known as cells) in a telecommunication network, during a 1 month (31 days) time-span. A cell represents the unit for a telecommunication net-

work in the tackled case study. Cells are grouped into antennas, so that one antenna can contain several cells. The goal for the problem is to predict a malfunctioning (pointed out by an alarm signal originated from cells) in a cell.

Furthermore, information about the geographical location of the cell can be relevant. When the Italian peninsula is considered, the total number of cells amounts to nearly $52,000$. For instance, when considering the total number of measurements, we get more than 150 millions of tuples.

Several kinds of statistical analysis were implemented to explore the data, and some interesting key-points and critical issues emerged from this analysis.

First of all, more than the 60% of the cells did not show any alarm signal. This is a quite usual behavior, as the system works smoothly for most of the time. Even when such cases are excluded, the average number of alarms per cell is only 3 in a month. In order to obtain a data set which is meaningful enough for a statistical analysis, only cells with at least 6 alarms have been kept: in this way the number of cells is further reduced to less than $2,000$. Moreover, among the remaining cells the proportion between alarm tuples and non-alarm tuples still remains high, as the former represent barely 1% of the total. However, we considered it acceptable, as malfunctioning must be considered unlikely to happen. In the end, this unbalance strongly influence the pool size of useful input data, and must faced with an adequate strategy.

Another critical issue regards the presence of several non-numeric values spread among the tuples. There are four different undefined values, among whose, `INF` values are the most frequent. Indeed, `INF` is the second most frequent value among all fields. All in all, discarding tuples containing undefined values in some of the fields would cut out 80% of data, leaving us with a too small dataset. We therefore had to find a different strategy to face the problem.

As already stated, another problematic issue regards the temporal dimension of data. Time-span is only one month, which on a time series related problem is not very much, to begin with the very basic problem of properly splitting data into training and test sets.

Another key-point regarding the data was found by looking at scatter plot diagrams between pairs of features. These diagrams highlighted two aspects: the first one is that alarm occurrences seem related to the values of some specific features. The second one is that there are two identical features. Since we don't have information about the meaning of the various features, we can't tell if this is supposed to be an error on the data supplied.

In addition to these, other statistical analysis were performed, focusing mainly on the values assumed by the features. Average values are summarized, along with standard deviations in Figure 1.

Inspecting Figure 1 we can see that FEATURE_7 and FEATURE_9 show a significant difference in term of average value between alarm and non-alarm events and thus, can trace a good starting point for a machine learning approach. Moreover, we can split features in three different groups.

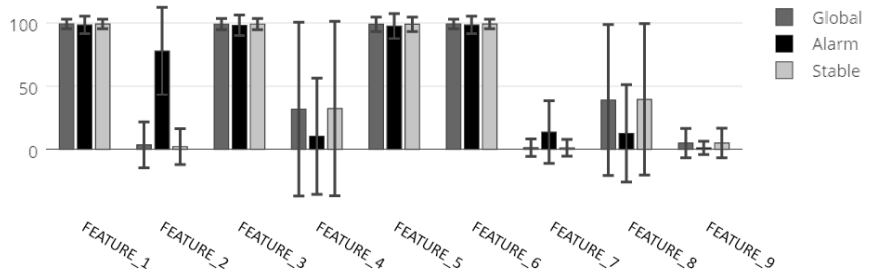

Fig. 1 Average values for the features in a stable or alarm situation. The line on every bar represent the standard deviation

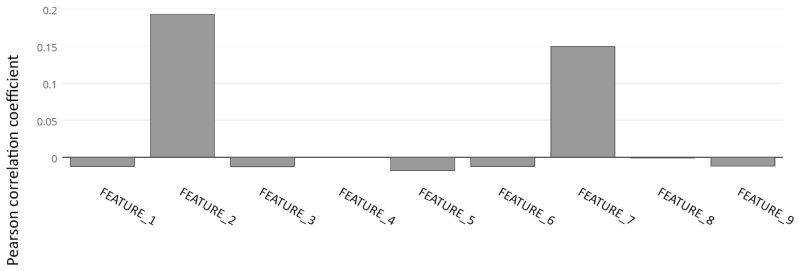

Fig. 2 Pearson correlation coefficient between features and alarm indicator

1. The first group is composed of: FEATURE_4, FEATURE_8, and FEATURE_9. These features have constant values in all the three conditions but also a relatively high standard deviation.
2. The second group is composed of: FEATURE_6, FEATURE_3, FEATURE_5, and FEATURE_1. Also in this case, the features tend to have constant values in all three situations, but with a relatively lower standard deviation.
3. The third group is composed of: FEATURE_7 and FEATURE_2. These features show a large difference in terms of both average value and standard deviation between alarm and non-alarm situations.

To better analyse the differences between alarm and non-alarm situations, Pearson correlation coefficients have been calculated between each feature and the alarm indicator. Results are shown in Figure 2, and confirm that FEATURE_2 and FEATURE_7 appear to be more related with an alarm occurrence, although the correlation value is always lower than 0.2.

Last but not least, the alarm propagation effect has been analyzed, to check if an alarm occurring in a cell is correlated to alarms in nearby cells. The results in Figure 3 show that this is the case only for cells belonging to the same antenna. In general, when the distance increases and cells of different antennas are considered, the probability of cooccurrent alarms drops close to 0. We can therefore conclude that, according to our data, there is no propagation effect.

3 Failure prediction

Alarm prediction is approached as a binary classification in a vector space model [6]. In simple words, the input to the classifier is given by a vector of features measured at times $t, t+1, \ldots, t+d$ in a given cell, while the output is positive or negative, depending whether an alarm in the same cell is foreseen or not for the time $t+\Delta$, with $\Delta > 0$. Such an approach is very general, and can be ported to similar problems in different domains.

More specifically, we chose a classification model based on Tikhonov regularization [21] (also known as Ridge regression). This is a direct classification model, based on the assignment of binary label to a tuple of features in accordance with the presence or the absence of an alarm signal after a chosen amount of time.

The representation of the features we use for classification is important for the system performance. In fact, we have a time series and how the temporal information is represented is a crucial point. Our approach consists of the following two steps:

1. Feature expansion;
2. Feature selection.

Feature expansion is obtained by calculating new features derived from the existing ones. An example of this is the application of some aggregation measures like means, standard deviations and/or variances. Another used technique for increasing the dimensionality is related to the use of convolution filters on the variables. More specifically, we used a set of filters belonging to the family of wavelets. One single filter was used multiple times with different parameters. The output for this step are tuples with a much larger dimensionality. We switched, in some extreme cases, from a 9 dimensions problem to a 1200 dimensions problem. Such number of variables

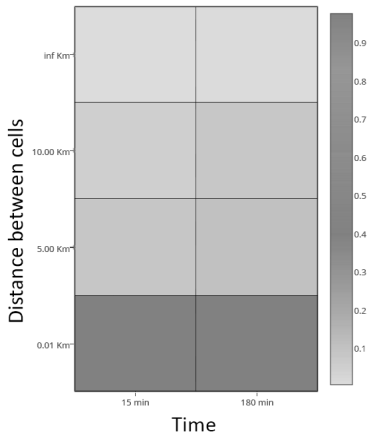

Fig. 3 Illustration of the probabilities of getting an alarm for the cells close to a cell signaling an alarm, after 15 and 180 minutes

is vastly overabundant. For this reason a further step for feature selection becomes necessary.

A process of automatic *feature selection* was chosen to increase portability and maintain a data-oriented approach. In particular we used an algorithm for L1L2 regularization implemented in the "l1l2py"[1] Python package. This algorithm combines the classic shrinkage methods given by Ridge Regression and Lasso Regression [20].

We consider a regression problem where the output y is reconstructed from features $x_i, i \in [1,p]$ by combining them with coefficients $\beta = \{\beta_i\}$. Ridge Regression uses a L2 norm in order to force a constraint on the regression coefficients size by reducing their absolute value:

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \left\{ \left\| \mathbf{y} - \sum_{j=1}^{p} x_j \beta_j \right\|^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{1}$$

On the other hand, Lasso Regression instead uses a L1 norm forcing sparsity in data and the annulment of some of the coefficients:

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \left\{ \left\| \mathbf{y} - \sum_{j=1}^{p} x_j \beta_j \right\|^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{2}$$

The final step is the effective experimental assessment of the classifier. First of all, we have to decide how to solve the critical issues emerged from data analysis and pointed out in the preceding section: how to mandage undefined values and how to split data into training and test set while reducing unbalancing of positive and negative examples.

With relation to the issue of undefined values we decided to operate a fixed substitution of the most frequent of such values (`INF`) based on the average value assumed by the considered feature, according to the scheme in Table 1. The number

Table 1 Substitution of undefined values in the features which assume such value.

Feature	Substitution value
FEATURE_6	120
FEATURE_3	120
FEATURE_5	120
FEATURE_1	120
FEATURE_7	-10

of occurrences of the other undefined values is relatively negligible and the tuples containing those values were simply dropped.

[1] http://slipguru.disi.unige.it/Software/L1L2Py/

In order to fix the balance between positive and negative samples we kept all the available positive samples, which were the ones with a minor number of occurrences N_p, and randomly choose $N_n = 4N_p$ negative examples.

The splitting of data into training and test set has been solved by a temporal based partitioning: we selected the first $2/3$ of the month for the training, and the remainder of the data was used as test set. We could act like that because positive examples have a nearly uniform distribution in data. Therefore, even if we applied the split without considering the frequencies of positive and negative examples, we obtained an acceptable balance for both sets. Furthermore, we want to underline how it is fundamental to operate an accurate sampling of data composing the training set, because including tuples related in some way with the occurrence of an alarm results in a hike of performance.

In the experiments different time shifts Δ have been considered. An analysis of the results showed some few interesting points.

First of all, we tested both generic and location-based models. The former does not consider geographical information, while the latter is a location-based model. Classification results have shown how the geographical information is crucial to the classification, while a single generic model for the whole area fails to catch the different variety of underlying key factors specific to each geographic subarea. One example is illustrated in Figure 4, where we compare the results, in terms of ROC curve, from a sample of such two models. Training strongly geo-localized models resulted, in some of the best performance, with AUC values (the area under ROC curve) of $0.7-0.8$.

Another point regards the inverse proportionality between classification performance and the time shift: performance decreases while the time shift between observations and alarm increases. In fact, a regular loss in performance can be observed when the time shift raises from a quarter of an hour up to $6-7$ hours; after that performance fundamentally go close to a random guess.

Last, we run some tests to analyse how performance changes in relation to the introduction of automatic feature selection. Models built directly using all the features produced by the feature expansion phase and models where we an *l1l2py* step of feature selection was applied have been compared. The performance of the system with feature selection shows a constant (although relatively low) improvement with respect to the one without it. One example of this is showed in Figure 5.

We noticed that the set of features chosen by the feature selection step changes depending on the location of the considered cell. However, the final features are always correlated with the two which showed the largest coefficient of linear correlation with the output, that is: FEATURE_7 and FEATURE_2. Such analysis can also help the service provider to analyse which are the most likely causes of malfunctioning.

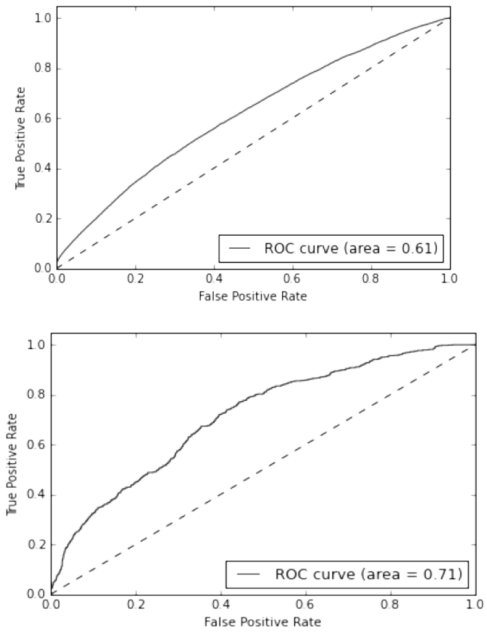

Fig. 4 Comparison between ROC curves generated on test set by a model trained and tested on the whole Italian area (top) and another specific to a single location (bottom). Time shift for prediction is 3 hours.

4 Conclusions and future work

In this paper we described a possible strategy to tackle a problem of alarm prediction in a domain where time series of features are available. From a first analysis of the dataset some issues have been raised, including the problem of undefined values. The alarm prediction has been defined as a classification problem which was solved by ridge regression. From the experimental results, we can conclude that geographical localization is important for the performance and that it is only possible to preview alarms occurring in a few hours.

Future directions of this analysis should explore models able to exploit the mutual position among cells, or, in general, to better exploit data. Among these, one of the most promising is the *cascade based model*. This approach is not focused on a single defined model to use, but on the chaining of various models for data classification which are applied sequentially. It aims to reduce, through several steps, the range of variables and dataset size. In a way which reminds of the decision tree approach, every stage is built to reduce the pool of data that is given as input to the next stage or the dimensionality of the problem.

The idea had origin from the implementation of AdaBoost [16] algorithm, which consists of a sequence of steps aiming to reduce the number of elements to classify.

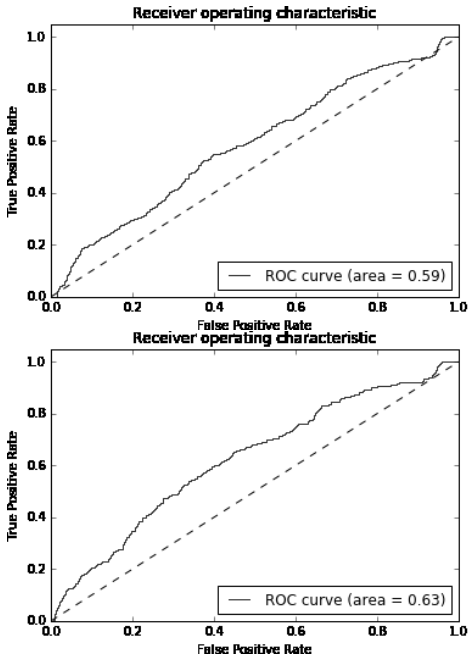

Fig. 5 Comparison between ROC curves obtained from models in a case without using feature selection (top) and in a case with using it (bottom)

The main key-point is to use a sequence of models increasing in complexity in order to cut out as many samples as possible in each stage. The early stages are typically the ones where the biggest cut is operated, while the latest stages are reserved to more refined models which have to take the hardest decision on the most difficult data.

In our case, this approach could not be applied because of the relativly small size of the sample dataset which remained after removing all non-relevant cells and balancing data. In fact, such multi-stage system is effective when it can exploit a really large dataset. On the other hand, in actual cases, a nearly unbounded quantity of data can be collected and therefore this approach could express its potentialities.

Acknowledgements

The research presented in this paper was partially supported by the national projects CHIS - Cultural Heritage Information System (PON), and BIG4H - Big Data Analytics for E-Health Applications (POR).

References

1. Amato, F., De Pietro, G., Esposito, M., Mazzocca, N.: An integrated framework for securing semi-structured health records. Knowledge-Based Systems **79**, 99–117 (2015)
2. Amato, F., Moscato, F.: A model driven approach to data privacy verification in e-health systems. Transactions on Data Privacy **8**(3), 273–296 (2015)
3. Amato, F., Moscato, F.: Exploiting cloud and workflow patterns for the analysis of composite cloud services. Future Generation Computer Systems (2016)
4. Asghar, M.Z., Fehlmann, R., Ristaniemi, T.: Correlation-Based Cell Degradation Detection for Operational Fault Detection in Cellular Wireless Base-Stations, pp. 83–93. Springer International Publishing, Cham (2013)
5. Barber, D.: Bayesian Reasoning and Machine Learning. Cambridge University Press (2012)
6. Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)
7. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
8. Damasio, C., Frölich, P., Nejdl, W., Pereira, L., Schroeder, M.: Using extended logic programming for alarm-correlation in cellular phone networks. Applied Intelligence **17**(2), 187–202 (2002)
9. Gilabert, E., Arnaiz, A.: Intelligent automation systems for predictive maintenance: A case study. Robotics and Computer-Integrated Manufacturing **22**(5), 543–549 (2006)
10. Grall, A., Dieulle, L., Berenguer, C., Roussignol, M.: Continuous-time predictive-maintenance scheduling for a deteriorating system. IEEE Transactions on Reliability **51**(2), 141–150 (2002)
11. Jain, R.: Quality of experience. IEEE MultiMedia **11**(1), 96–95 (2004)
12. Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., Hampapur, A.: Improving rail network velocity: A machine learning approach to predictive maintenance. Transportation Research Part C: Emerging Technologies **45**, 17 – 26 (2014). Advances in Computing and Communications and their Impact on Transportation Science and Technologies
13. Millan, P., Molina, C., Medina, E., Vega, D., Meseguer, R., Braem, B., Blondia, C.: Tracking and predicting link quality in wireless community networks. In: 2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 239–244 (2014)
14. Mobley, R.K.: An introduction to predictive maintenance, 2nd edn. Butterworth-Heinemann (2002)
15. Patwardhan, A., Verma, A.K., Kumar, U.: A Survey on Predictive Maintenance Through Big Data, pp. 437–445. Springer International Publishing, Cham (2016)
16. Schapire, R.E.: Explaining AdaBoost, pp. 37–52. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
17. Scheffer, C., Girdhar, P.: Practical machinery vibration analysis and predictive maintenance. Elsevier (2004)
18. Susto, G.A., Schirru, A., Pampuri, S., McLoone, S., Beghi, A.: Machine learning for predictive maintenance: A multiple classifier approach. IEEE Transactions on Industrial Informatics **11**(3), 812–820 (2015)
19. Swanson, D.C.: A general prognostic tracking algorithm for predictive maintenance. In: Aerospace Conference, 2001, IEEE Proceedings., vol. 6, pp. 2971–2977. IEEE (2001)
20. Tibshiriani, R.: Regression shrinkage and selection via the Lasso. Journal of the Royal Statistical Society. Series B **58**(1), 267–288 (1996)
21. Tychonoff, A., Arsenin, V.: Solution of ill-posed problems. Winston & Sons, Washington (1977)
22. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
23. Zhou, X., Xi, L., Lee, J.: Reliability-centered predictive maintenance scheduling for a continuously monitored system subject to degradation. Reliability Engineering & System Safety **92**(4), 530–534 (2007)