Journal of Computer and Communications, 2022, 10, 125-137
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Comparing Machine Learning Algorithms for Improving the Maintenance of LTE Networks Based on Alarms Analysis

Batchakui Bernabe[1], Deussom Djomadji Eric Michel[2,3], Chana Anne Marie[1], Mama Tsimi Serge Fabrice[1]

[1]Department of Computer Engineering, National Advanced School of Engineering, UY1, Yaoundé, Cameroon
[2]Department of Electrical and Telecommunications Engineering, National Advanced School of Engineering, Yaoundé, Cameroon
[3]College of Technology, University of Buea, Buea, Cameroon
Email: eric.deussom@gmail.com

How to cite this paper: Bernabe, B., Michel, D.D.E., Marie, C.A. and Fabrice, M.T.S. (2022) Comparing Machine Learning Algorithms for Improving the Maintenance of LTE Networks Based on Alarms Analysis. *Journal of Computer and Communications*, **10**, 125-137.
https://doi.org/10.4236/jcc.2022.1012010

Received: October 26, 2022
Accepted: December 27, 2022
Published: December 30, 2022

Copyright © 2022 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

Mobile network operators are facing many challenges to satisfy their subscribers in terms of quality of service and quality of experience provided. To achieve this goal, technological progress and scientific advances offer good opportunities for efficiency in the management of faults occurring in a mobile network. Machine learning techniques allow systems to learn from past experiences and can predict, solutions to be applied to correct the root cause of a failure. This paper evaluates machine learning techniques and identifies the decision tree as a learning model that provides the most optimal error rate in predicting outages that may occur in a mobile network. Three machine learning techniques are presented in this study and compared with regard to accuracy. This study demonstrates that the appropriate machine learning technique improves the accuracy of the model. By using the decision tree as a machine learning model, it was possible to predict solutions to network failures, with an error rate less than 2%. In addition, the use of Machine Learning makes it possible to eliminate steps in the network failure processing chain; resulting in reduced service disruption time and improved the network availability which is a key network performance index.

Keywords

4G LTE Mobile Network, Machine Learning, Network Maintenance, Troubleshooting, Decision Tree, Random Forest

1. Introduction

Mobile network operations have grown enormously and rapidly over the past two decades. If satisfaction on the side of subscribers based on the services provided by these mobile networks is appreciable, this is due to a maintenance team of engineers in the back office who are responsible (full-time) for ensuring the maintenance of the network and the usage of new network management techniques that call on artificial intelligence applied to network management and maintenance [1]. Some vendors like Huawei have modified their initial management platform to virtualization based solution which includes the network analysis module based on machine learning [2]. It will be interesting to explore, through this work, the usage of machine learning algorithms that can contribute to the optimization of an LTE FDD mobile network opened to public services.

Many authors have been interested in the usage of artificial intelligence and other techniques applied to maintenance system. For example, in 2013, HOUNKONNOU worked on a semi-automatic modeling approach based on patterns that generically describe the dependencies between the resources used by the services of the IMS (IP Multimedia Subsystem) network [3]. Seghier [4] worked on "Optimization of the radio parameters of the 4G mobile network by fuzzy logic". He dealt with the optimization of radio parameters in the LTE network, by integrating fuzzy logic to help the network to make the decision to perform a Hand Over or not. Deussom *et al.* [5] worked on "Machine learning-based approach for designing and implementing a collaborative fraud detection model through CDR and traffic analysis". In their work, they used machine learning to detect and identify fraud in the mobile network billing system by using the Call detail records and traffic. Sultan *et al.* [6] presented "Call Detail Records Driven Anomaly Detection and Traffic Prediction in Mobile Cellular Networks". Trinh *et al.* [7] worked on "Detecting Mobile Traffic Anomalies through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach".

Our analysis of these research works triggered the following striking question: Which Machine Learning algorithms can follow the learning of the operations carried out by human operators not only to determine the failure that has occurred in the system, but, above all, to propose a solution or the most exact method of solving the failure? This question leads us to hypothesize that: A Machine Learning algorithm that learns from past outage processing operations of LTE mobile networks can predict new outages, anticipate their processing and thus significantly reduce the disruption time and therefore improve the quality of service.

The objective of this work is to evaluate the precision of different machine learning algorithms by using data collected from an alarm management system and propose a model for predicting solutions to failures (incidents) occurring in a mobile network in order to reduce the downtime.

Specifically, it is a question of proposing a model capable of:

1) Controlling the failure that has occurred in the system;

2) Predicting, with a relatively lower error rate, the solution to be applied in the event of a failure;

3) Reducing the search time for solutions which is currently of the order of several minutes, or an hour;

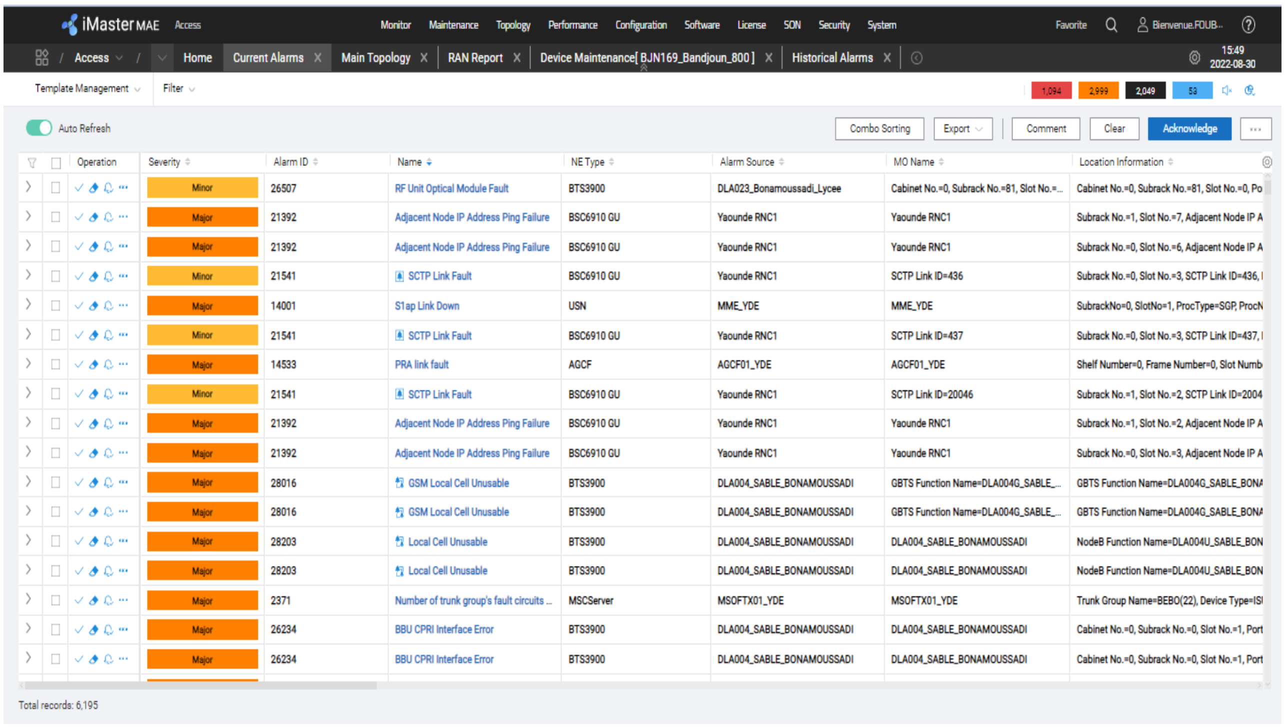
4) Reducing service disruption times, and

5) Improving or even increase customer satisfaction (QoE: Quality of Experience).

The rest of this article presents, in the next section, the materials and methods, and the machine learning algorithms that will be explored. The fourth section deals with the presentation and analysis of result within the experimentation carried out and the results obtained based on data collected on a live 4G network working in 1800 MHz and 2100 MHz bands using Huawei U2020 MBB management platform. This section will also present the discussions of the results obtained. Finally a general conclusion and perspectives are provided.

2. Materials and Methods

In the context of the present work, the research was done by using real data collected on a 4G LTE network in Cameroon using Huawei U2020 MBB. The alarm management feature of the U2020 was selected to monitor the network alarms and process the alarms based on their severity, types, sources and impact on the network (see Figure 1 which presents the appearance of the alarm management platform of the U2020). Hence, the target data to be predicted are discrete data. Algorithms used for the classification of the collected data, as well as for the prediction of our target are presented and reviewed below: K-nearest neighbour (K-NN); decision tree; and random forest algorithms [8].



Severity	Alarm ID	Name	NE Type	Alarm Source	MO Name	Location Information
Minor	26507	RF Unit Optical Module Fault	BTS3900	DLA023_Bonamoussadi_Lycee	Cabinet No.=0, Subrack No.=81, Slot No.=...	Cabinet No.=0, Subrack No.=81, Slot No.=0, Po
Major	21392	Adjacent Node IP Address Ping Failure	BSC6910 GU	Yaounde RNC1	Yaounde RNC1	Subrack No.=1, Slot No.=7, Adjacent Node IP A
Major	21392	Adjacent Node IP Address Ping Failure	BSC6910 GU	Yaounde RNC1	Yaounde RNC1	Subrack No.=0, Slot No.=6, Adjacent Node IP A
Minor	21541	SCTP Link Fault	BSC6910 GU	Yaounde RNC1	SCTP Link ID=436	Subrack No.=0, Slot No.=3, SCTP Link ID=436, I
Major	14001	S1ap Link Down	USN	MME_YDE	MME_YDE	SubrackNo=0, SlotNo=1, ProcType=SGP, ProcN
Minor	21541	SCTP Link Fault	BSC6910 GU	Yaounde RNC1	SCTP Link ID=437	Subrack No.=0, Slot No.=3, SCTP Link ID=437, I
Major	14533	PRA link fault	AGCF	AGCF01_YDE	AGCF01_YDE	Shelf Number=0, Frame Number=0, Slot Numb
Minor	21541	SCTP Link Fault	BSC6910 GU	Yaounde RNC1	SCTP Link ID=20046	Subrack No.=1, Slot No.=2, SCTP Link ID=2004
Major	21392	Adjacent Node IP Address Ping Failure	BSC6910 GU	Yaounde RNC1	Yaounde RNC1	Subrack No.=1, Slot No.=2, Adjacent Node IP A
Major	21392	Adjacent Node IP Address Ping Failure	BSC6910 GU	Yaounde RNC1	Yaounde RNC1	Subrack No.=0, Slot No.=3, Adjacent Node IP A
Major	28016	GSM Local Cell Unusable	BTS3900	DLA004_SABLE_BONAMOUSSADI	GBTS Function Name=DLA004G_SABLE_...	GBTS Function Name=DLA004G_SABLE_BONA
Major	28016	GSM Local Cell Unusable	BTS3900	DLA004_SABLE_BONAMOUSSADI	GBTS Function Name=DLA004G_SABLE_...	GBTS Function Name=DLA004G_SABLE_BONA
Major	28203	Local Cell Unusable	BTS3900	DLA004_SABLE_BONAMOUSSADI	DLA004_SABLE_BONAMOUSSADI	NodeB Function Name=DLA004U_SABLE_BON
Major	28203	Local Cell Unusable	BTS3900	DLA004_SABLE_BONAMOUSSADI	DLA004_SABLE_BONAMOUSSADI	NodeB Function Name=DLA004U_SABLE_BON
Major	2371	Number of trunk group's fault circuits ...	MSCServer	MSOFTX01_YDE	MSOFTX01_YDE	Trunk Group Name=BEBO(22), Device Type=ISI
Major	26234	BBU CPRI Interface Error	BTS3900	DLA004_SABLE_BONAMOUSSADI	DLA004_SABLE_BONAMOUSSADI	Cabinet No.=0, Subrack No.=0, Slot No.=1, Port
Major	26234	BBU CPRI Interface Error	BTS3900	DLA004_SABLE_BONAMOUSSADI	DLA004_SABLE_BONAMOUSSADI	Cabinet No.=0, Subrack No.=0, Slot No.=1, Port

Total records: 6,195

Figure 1. Structure of U2020 Alarm management system.

2.1. The K-Nearest Neighbour (K-NN)

The K-Nearest Neighbour (K-NN) is a supervised machine learning algorithm that needs only one hyper parameter. According to the method, if the majority of k samples most similar to one sample (nearest neighbours in the eigenspace) belong to a specific category, the sample also belongs to this category [9].

2.2. The Decision Tree (DT)

The decision tree is a model that employs a hierarchical representation of the data structure in the form of sequences of decisions (tests) for the prediction of an outcome or a class [10]. Each individual (or observation), which must be assigned to a class, is described by a set of variables which are tested in the nodes of the tree. Tests are performed in internal nodes and decisions are made in leaf nodes [11].

2.3. Random Forests (RF)

The "Random Forest" algorithm is a classification algorithm that reduces the error of predictions from a single decision tree, thus improving their performance. For this, it combines many decision trees in a bagging type approach [8]. The word "*bagging*" is a contraction of *Bootstrap Aggregation.* It is a technique used to improve classification, especially those from decision trees.

In order to train the system to recognize failures and to propose the most optimal solution, an approach borrowed from "data science" has been followed. Indeed, on a dataset collected from the databases of the operation and maintenance center of a mobile telephone operator for a given period, the three Machine Learning algorithms presented above have been executed, with the aim of selecting the algorithm that will give us the best results with minimal error margin. Figure 2 shows the framework for determining the most accurate model [12]. The process is as follows:

✓ Dataset processing;

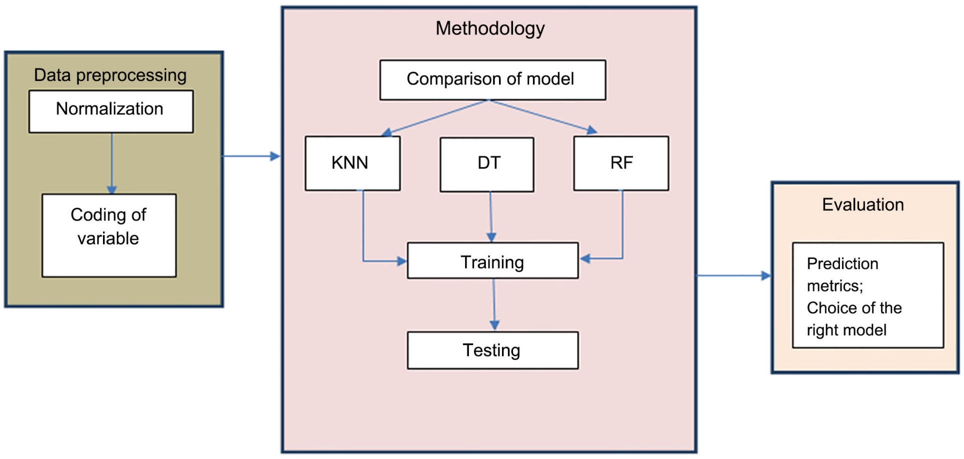


Figure 2. Framework of determination of the right model.

- ✓ Normalization/coding of variables;
- ✓ Determination of the model and its parameters;
- ✓ Learning;
- ✓ Testing and interpretation of results.

Figure 2 presents the framework for the determination of the right model.

3. Presentation and Analysis of Result

3.1. Environment and Tools Used

To run machine learning algorithms in our computer, the Python version 3.8 programming language have been used. It is the reference language used in the development of applications for artificial intelligence. The Python distribution that is used is the one provided by Anaconda (it contains all the tools and libraries needed to do Machine Learning, namely Numpy, Matplotlib, sklearn, Jupiter, Spider…etc). Package versions are managed by the management system conda packets. The Anaconda distribution is used by more than 6 million users and includes over 250 popular data science packages suitable for Windows, Linux and MacOS [9] [13].

3.2. Data Processing

As seen in the previous section, the dataset refers to the set of examples that the machine must study. In the case of this study, the examples to be studied by the machine consist of data extracted from U2020 MBB involving operation and maintenance platform. These data constitute all the incidents and breakdowns occurring in a mobile telephony network, and the parameters which characterize them. These parameters are, among other things, the severity of the alarm, its identifier, the name of the failures, the type of node (eNodeB) on which the failure occurred, the site name and the location. Table 1 is an extract of the data under consideration.

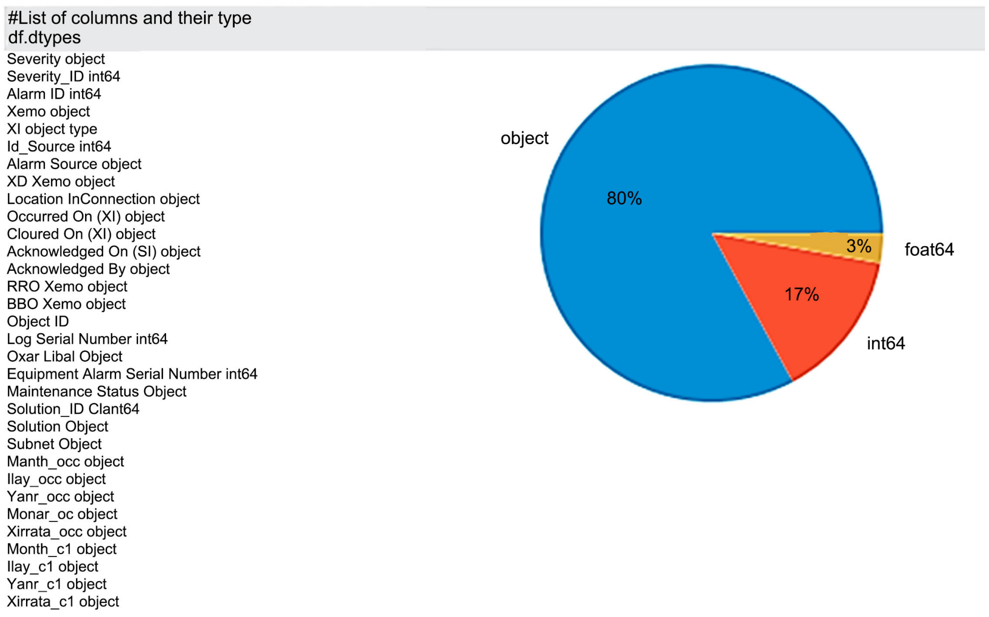
The complete dataset is made up of 35,608 rows and 25 columns of which 19 are of Object type (Object), 5 of Integer type (Int64) and 1 of real type (Float64). The following python code lists the columns of the dataset.

Before implementing the prediction algorithms, some descriptive analysis on the data have been first performed with the aid of some tools offered by the Anaconda environment through its Numpy and Matplotlib libraries. These analyses allowed us to determine, through the variation curve, the level of use of the SOLUTION_ID parameter, and the degree of correlation between the SOLUTION_ID parameter and the other parameters of the database, and the correlation diagram. Figure 3 is a diagram that represents the occupancy size of each data type in this dataset: 80% for Object type data, 17% for Integer type data, and 3% for Real types.

The histogram of the *Solution_ID variable* presented in Figure 4 let us to have the percentage of variations of solutions used in the system.

Table 1. An extract of the dataset considered for this study.

Severity	Alarm ID	Name	NE Type	Location Information	Additional Information	Occurred On (NT)
Critical	65034	AC Power Failure	BTS3900	Cabinet No. = 0, Subrack No. = 0, Slot No. = 19, Port No. = 0, Board Type = UPEU	RAT_INFO = UL, AFFECTED_RAT = UL, DID = NULL	08/17/2019 17:24:46
Major	65041	Battery Fuse Break	BTS3900	Cabinet No. = 0, Subrack No. = 0, Slot No. = 19, Port No. = 7, Board Type = UPEU	RAT_INFO = UL, AFFECTED_RAT = UL, DID = NULL	08/17/2019 17:25:54
Major	26232	BBU Optical Module Transmit/Receive Fault	BTS3900	Cabinet No. = 0, Subrack No. = 0, Slot No. = 1, Port No. = 0, Board Type = UBBP, Specific Problem = Receive Power Too Low	RAT_INFO = UL, AFFECTED_RAT = U, DID = NULL, Cumulative Duration(s) = 90	08/17/2019 17:26:25
Major	25888	SCTP Link Fault	BTS3900	Link No. = 6, Description = NULL, Peer IP Address = 10.32.248.251, Service Type = NCP	RAT_INFO = UL, AFFECTED_RAT = U, DID = NULL	08/17/2019 17:27:14
Major	25888	SCTP Link Fault	BTS3900	Link No. = 7, Description = NULL, Peer IP Address = 10.32.248.250, Service Type = CCP	RAT_INFO = UL, AFFECTED_RAT = U, DID = NULL	08/17/2019 17:27:14
Major	25888	SCTP Link Fault	BTS3900	Link No. = 0, Description = YDE MME, Peer IP Address = 10.31.102.1, Service Type = S1-AP	RAT_INFO = UL, AFFECTED_RAT = L, DID = NULL	08/17/2019 17:27:14
Major	25888	SCTP Link Fault	BTS3900	Link No. = 70002, Description = DLA MME, Peer IP Address = 10.32.202.1, Service Type = S1-AP	RAT_INFO = UL, AFFECTED_RAT = L, DID = NULL	08/17/2019 17:27:14
Minor	26263	IP Clock Link Failure	BTS3900	Link No. = 0, Server IP Address = 10.32.101.1	RAT_INFO = UL, AFFECTED_RAT = UL, DID = NULL, Cumulative Duration(s) = 30	08/17/2019 17:27:40
Warning	26819	Data Configuration Exceeding Licensed Limit	BTS3900	-	RAT_INFO = UL, AFFECTED_RAT = L, DID = NULL	08/17/2019 17:28:18
Major	29240	Cell Unavailable	BTS3900	eNodeB Function Name = YDE017_Messassi, Local Cell ID = 2, Cell FDD TDD indication = FDD, Cell Name = YDE_11017_2, eNodeB ID = 11017, Cell ID = 2, Specific Problem = Insufficient license	RAT_INFO = UL, AFFECTED_RAT = L, DID = NULL, Cumulative Duration(s) = 90	08/17/2019 17:28:18
Major	25880	Ethernet Link Fault	BTS3900	Cabinet No. = 0, Subrack No. = 0, Slot No. = 7, Port No. = 0, Specific Problem = Ethernet Link Fault	RAT_INFO = UL, AFFECTED_RAT = UL, DID = NULL	08/17/2019 17:24:36
Major	25880	Ethernet Link Fault	BTS3900	Cabinet No. = 0, Subrack No. = 0, Slot No. = 7, Port No. = 0, Specific Problem = Ethernet Link Fault	RAT_INFO = UL, AFFECTED_RAT = UL, DID = NULL	08/17/2019 17:31:26
Major	29240	Cell Unavailable	BTS3900	eNodeB Function Name = YDE056_Bastos_ARMP, Local Cell ID = 10, Cell FDD TDD indication = FDD, Cell Name = YDE_11056W_0, eNodeB ID = 11056, Cell ID = 10, Specific Problem = External Link Fault	RAT_INFO = UL, AFFECTED_RAT = L, DID = NULL	08/17/2019 17:25:12



```
#List of columns and their type
df.dtypes
Severity object
Severity_ID int64
Alarm ID int64
Xemo object
XI object type
Id_Source int64
Alarm Source object
XD Xemo object
Location InConnection object
Occurred On (XI) object
Cloured On (XI) object
Acknowledged On (SI) object
Acknowledged By object
RRO Xemo object
BBO Xemo object
Object ID
Log Serial Number int64
Oxar Libal Object
Equipment Alarm Serial Number int64
Maintenance Status Object
Solution_ID Clant64
Solution Object
Subnet Object
Manth_occ object
Ilay_occ object
Yanr_occ object
Monar_oc object
Xirrata_occ object
Month_c1 object
Ilay_c1 object
Yanr_c1 object
Xirrata_c1 object
```

Figure 3. Diagram of the variable types listed in our dataset.

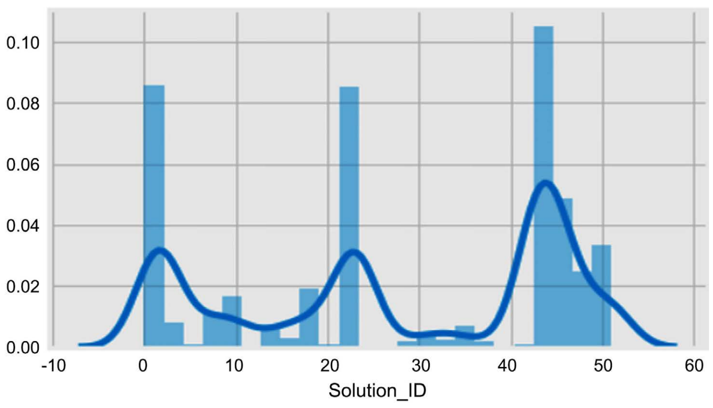


Figure 4. Histogram of the Solution_ID variable.

The analysis of correlations through a heatmap makes it possible to represent correlations (or relationships) between variables graphically as seen in **Figure 5** [14]. The closer the value is to 1 (dark red color), the more positive and strong the correlation is. Conversely, the closer the correlation is to −1 (dark blue), the more the correlation is negative and strong [13].

One can observe from the results that the variable Solution_ID is negatively correlated to Severity_ID, Alarm_ID; ID_Source, Log Serial Number and Equipment Alarm Serial Number.

3.3. Creation of Training and Test Sample

After the descriptive analysis of our data, it is a question of defining the sample of data on which our learning should be performed, as well as the test sample data. As part of our work, 70% of the data has been used for learning and 30% for testing. The following python codes were used to generate these sub datasets:

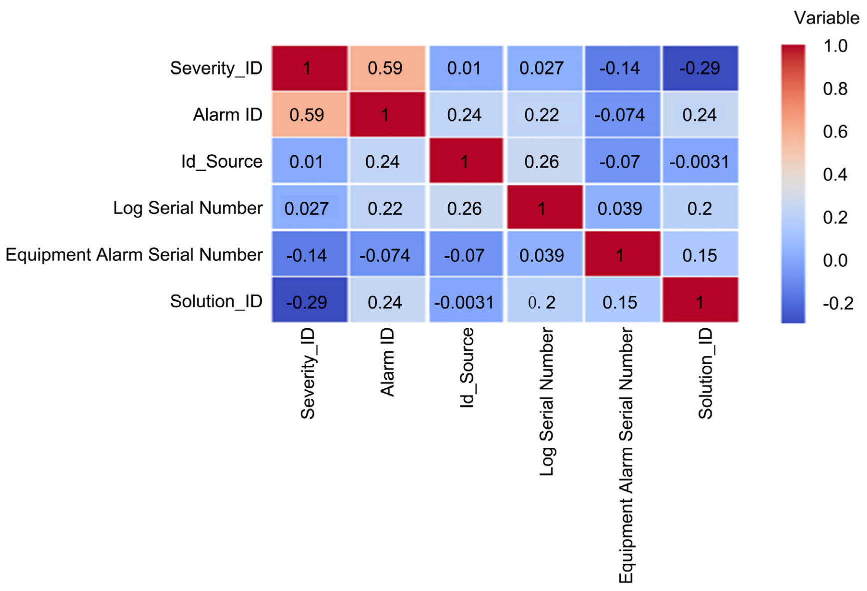


Figure 5. Heatmap of correlations between variables.

#creation of 4 datasets:

# - x_train contains 70% of x

# - y_train contains the Solution_ID associated with x_train

# => x_train and y_train will train the algorithm

#

# - x_test contains 30% of x

# - y_test contains the Solution_ID associated with x_test

# => x_test and y_test will evaluate the performance of the algorithm once trained on the train

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(dff,Y,test_size=0.30, random_state=2020).

3.4. Learning and Building Prediction Models

After the creation of the data sets for training and for testing, it was then a question of applying a learning model on them. As it is said previously, the data to be predicted being discrete data, that is phased with a classification problem (unsupervised learning), and not a regression one (supervised learning). It is reminded that the best classification model will be selected amongst the KNN, the Decision Tree and the Random Forest algorithms. The next section will present the results obtained after the implementation of each method and the data set. [15].

3.5. Results and Discussions

Here are presented the results obtained after the implementation of the three machine learning methods above on the dataset obtained from the Huawei U2020 MBB platform.

3.5.1. Prediction with the K-NN Algorithm

The algorithm has been imported from the sklearn library using the following code:

```
from sklearn.neighbors import KNeighborsClassifier
```

Then the KNN model is created, having randomly 10 neighbours. The following line shows the appropriate python code:

```
model_KNN = KNeighborsClassifier(n_neighbors=10)
```

And the learning on of the dataset started. The python code to invoke for this is the following:

```
model_KNN.fit(x_train, y_train)
```

For different values of the neighbour number K, different prediction error values got out as seen in **Table 2**.

By optimizing the score on the test data in graphical representation, the best prediction is obtained for K = 1. **Figure 6** represents the evolution of KNN test scores as a function of the number of neighbours K. The abscissa axis contains the number of neighbours K to be considered, while the ordinate axis returns the value of the corresponding error rate.

Thus for K = 1, the K-nearest neighbour algorithm predicts the solution to be applied to the system with the least error of 10.85%.

3.5.2. Prediction with the Decision Tree

To do this, the import of the algorithm concerned is done from the sklearn library via the following python code:

```
from sklearn.tree import DecisionTreeClassifier
```

Then, the creation of a decision tree is done with a depth of 1 via the following code:

```
model_DT = DecisionTreeClassifier(random_state=40, max_depth=1)
```

And the learning on of the training dataset started. The following code is executed:

```
model_DT.fit(x_train, y_train)
```

After the classification test, the accuracy error is 0.602873. In order to optimize this error, the evolution curve of the error is plotted as a function of the depth of the tree. The result obtained indicates the optimal error for a maximum depth of 13 as illustrated in **Figure 7**. The graph has as abscissa, the values of the depth of the tree, and as ordinate, the value of the error of prediction observed by the algorithm.

Thus, after running the decision tree model with a maximum depth of 13, a prediction of the solutions to be applied is obtained with an optimal error rate of 1.4365%.

3.5.3. Prediction with Random Forest

To perform prediction with random forest, algorithm from the sklearn library is imported via the following python code:

```
from sklearn.ensemble import RandomForestClassifier
```

Then, the creation of a random forest of 300 trees is done using the following code:

model_RF=RandomForestClassifier(n_estimators=300, random_state=2020)

Learning on our training dataset was then performed with the following python code:

model_RF.fit(x_train, y_train)

By running this model, with the number of trees = 300, a predicted solution is obtained with an error rate of 1.8437%. This rate remained optimal because after optimization, the number of trees to be used in this model was 303, still with an error rate of 1.8437%.

At the end of the experimentation that was carried out in the previous paragraphs, the machine learning model chosen is the decision tree. Indeed, this model is retained because it predicts the optimal solution of a failure occurring in the network with an accuracy of 1.4% as presented in Table 3.

Table 2. Value of precision error for different values of the neighbour number K.

K	Error
1	0.108585
2	0.120801
5	0.148286
10	0.170682

Table 3. Test results and performance metrics.

Model used	Meta-parameter value	Precision error
K-Nearest Neighbor (K-NN)	Number of neighbor K = 1	10.8585%
Decision tree	Shaft depth p = 13	1.4365%
Random forest	Number of trees n = 303	1.8437%

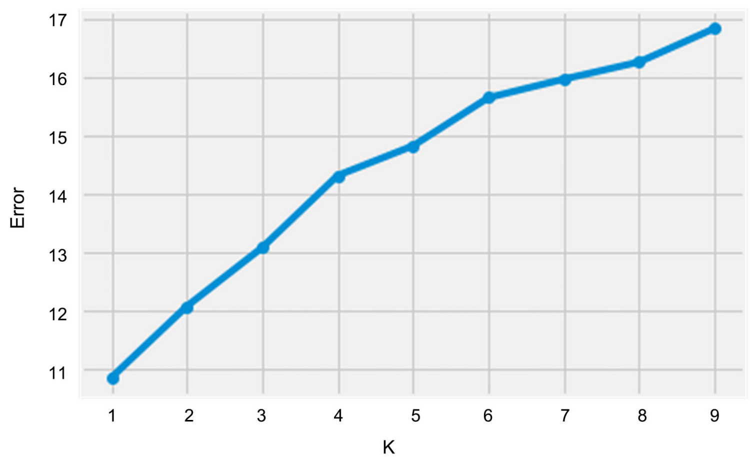


Figure 6. Evolution of KNN test scores as a function of the number of neighbour K.

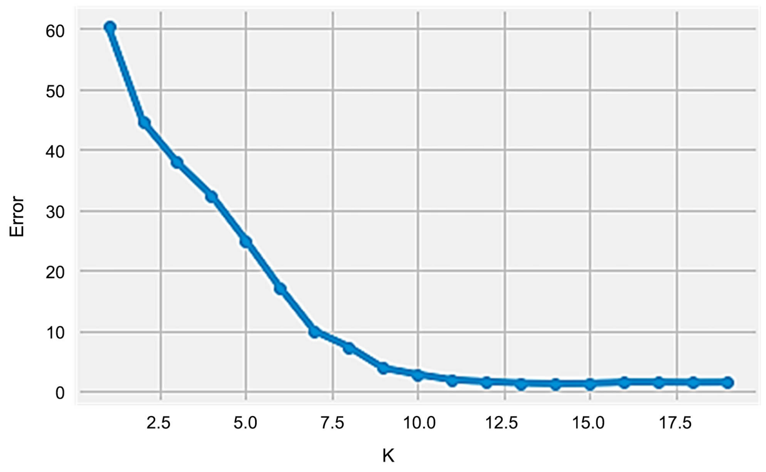


Figure 7. Evolution of DT test scores as a function of tree depth.

The results obtained from our experimentations allow us to predict the faults and incidents and the solutions to be applied with a fairly low error rate of <2%. The system not subject to machine learning offered more or less exact fault solutions, depending on the level of expertise of the engineer in charge of maintenance. So, for maintenance engineer with no experience, it is more difficult to maintain the network, and the down time can be big, thereby reducing the network availability and customers' satisfaction. The work carried out and explored in the method section here shows us that the exploitation of machine learning in the maintenance of mobile networks has brought great efficiency. However, these works present only the detection of faults, and not their resolution. With the advent of expert systems, the problem began to be addressed, but was stymied by the fact that these systems very quickly become obsolete in the face of a dynamic and extended environment. The use of the decision tree as a supervised learning model thus enables the system to learn the solutions of past failures in order to predict those to come. The methodical learning offered by Machine Learning models therefore brings a step forward towards the intelligent processing of mobile network maintenance work. Also, once the failure has been diagnosed, the solution to be applied will be directly proposed in order to be implemented.

4. Conclusions

Customers consuming mobile telephony services are increasingly demanding, not only in terms of their quality, but also in terms of their availability. Operators are therefore called upon to multiply their efforts and performances in order to guarantee not only the quality of service, but also the quality of experience by reducing the downtime of services. The use of Artificial Intelligence techniques, in particular Machine Learning, considerably reduces the service disruption time. Three machine learning techniques are presented in this study and compared with regard to accuracy. Only one hyper parameter has been varied for each technique so that the design process is simplified. This study demonstrates that the appropriate machine learning technique improves the accuracy of the model. By using the decision tree as a machine learning model, it was possible to predict solutions to network failures, with an error rate less than 2%. In addition, the use of Machine Learning makes it possible to eliminate steps in the network failure processing chain; resulting in reduced service disruption time and improved the network availability which is a key network performance index.

Out of the panoply of machine learning solutions or algorithms available, it was a question for us to determine the model that will provide the expected result with the lowest possible error rate. Also, it is deduced that with the decision tree model, failure solutions could be predicted with an accuracy error of less than 2%. This reduces the time taken by the engineers who, after consulting the alarm management system, must analyze the fault and find the appropriate solution. The use of machine learning solution, therefore, constitutes an effective decision support tool in the sense that not only the failure analysis time is reduced, or even eliminated, but the proposed solution is the right one at least at 98%.

However, improvements could always be made to this model with the aim of further reducing the accuracy error rate and getting closer to 0%. Moreover, research work could be undertaken with the aim of producing an end-to-end Self-Healing solution for mobile networks.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Paquette, G. (2002) Modélisation des connaissances et des compérences. Presse de l'Université du Québec, Québec.

[2] Jean-Paul Pallois, M.D. (2018) IA: Rendre les réseaux mobiles intelligents. IA Technologies d'apprentissage, n° %14/2018. https://www.semanticscholar.org/author/Jean-Paul-Pallois/134200585

[3] HOUNKONNOU (2013) Semi-Automatic Modeling Approach Based on Patterns or "Patterns".

[4] Chahineze, S. (2016) Optimization of the Radio Parameters of the 4G Mobile Network by Fuzzy Logic.

[5] Deussom, E., Matemtsap, M.B., Tchagna, K.A., *et al.* (2022) Machine Learning-Based Approach for Designing and Implementing a Collaborative Fraud Detection Model through CDR and Traffic Analysis. *Transactions on Machine Learning and Artificial Intelligence*, **10**, 46-58. https://doi.org/10.14738/tmlai.104.12854

[6] Sultan, K., *et al.* (2018) Call Detail Records Driven Anomaly Detection and Traffic Prediction in Mobile Cellular Networks. *IEEE Access*, **6**, 41728-41737. https://doi.org/10.1109/ACCESS.2018.2859756

[7] Trinh, D.H., *et al.* (2019) Detecting Mobile Traffic Anomalies through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach. *IEEE Access*, **7**, 152187-152201. https://doi.org/10.1109/ACCESS.2019.2947742

[8] Breiman, L. (2001) Random Forests. *Machine Learning*, **45**, 5-32.

https://doi.org/10.1023/A:1010933404324

[9] Mint, M. (2018) Machine Learning Made Easy. https://thenewstack.io/machine-learning-made-easy/

[10] Bénard, A. https://blog.ysance.com/algorithme-n1-comprendre-ce-quest-un-arbre-de-decision-en-5-min

[11] O'Neil, C. (2019) Weapons of Math Destruction. https://en.wikipedia.org/wiki/Weapons_of_Math_Destruction

[12] Saint-Cirgue, G. (2019) Apprentissage supervisé. https://machinelearnia.com/apprentissage-supervise-4-etapes

[13] Vieille, M.-J. https://www.lovelyanalytics.com/2020/06/08/random-forest-tutoriel-python

[14] Jakobson, G. (1993) Alarm Correlation. *IEEE Network*, **6**, 52-59. https://doi.org/10.1109/65.244794

[15] Amini, M.-R. (2015) Apprentissage Machine de la théorie à la pratique, Eyrolles. https://hal.archives-ouvertes.fr/hal-01211214