Applied Data Science: Using Machine Learning for Alarm Verification

by Jan Stampfli and Kurt Stockinger
Zurich University of Applied Sciences, Switzerland

Introduction

False alarms triggered by sensors of alarm systems are a significant cost driver for the blue organizations (i.e. police, firefighters, ambulance) and owners of alarm systems. They occur in a large number and are very costly. These false alarms are often due to either technical failures such as network downtimes or human errors.

To remedy this problem, we develop a novel alarm verification service by leveraging the power of an alarm data warehouse. In addition, we apply various machine learning algorithms to identify false alarms. The goal of our system is to help human responders in their decision about triggering costly intervention forces or not.

Approach

In this project we are working with a security company that is a mayor player in secure alarm transmission. Alarms can be triggered by devices installed at banks, jewelry stores, private homes, etc. As already stated initially, around 90% of alarms are so-called false alarms and are due to technical reasons or unwanted human error. The challenge of our project is to significantly reduce the amount of false alarms.

The problem of alarm prediction is conceptually similar to anomaly detection (Chandola et al. 2009) or prediction of failures (Salfner et al. 2010). Hence, we can borrow some ideas from these fields and take advantage of the latest progress in Deep Learning (LeCun et al. 2015). In our project we have chosen the following four machine learning approaches to predict false alarms:

- Random Forests
- Support Vector Machines
- Logistic Regression
- Deep Neural Networks

We based our experiments on a set of more than 300,000 alarms. For each alarm we had multiple different features such as device ID, device location, type of alarm, alarm trigger time, etc. We used these features as input for our machine learning approaches. An overview of the system architecture is given in Figure 1.

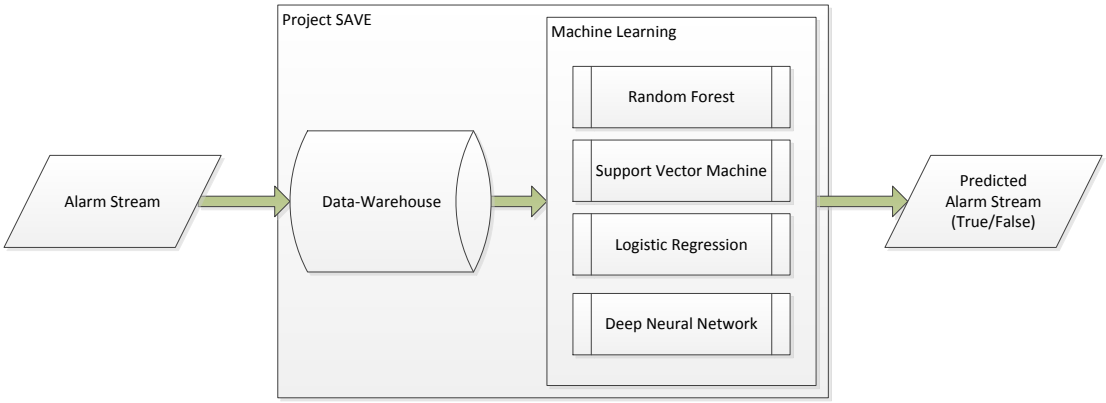

Figure 1. Overview of our alarm prediction system: alarm streams are stored in a data-warehouse and each alarm is evaluated as true or false.

Due to security reasons we do not have direct information about whether an alarm is a false alarm or not. However, we have indirect information that we can use as labels for our machine learning approach. In particular, for each alarm we know when it was triggered and when it was reset again. Our hypothesis is, that if the time difference delta t between triggering and resetting an alarm is small, there is a high chance that the alarm is false.

Consider the case of an alarm system deployed at a private home. Further assume that kids or a pet triggered an alarm. In this case, the parents or pet masters might call the alarm receiving center and inform them about a false alarm. In any case, the alarm receiving center will reset the alarm after verification of the alarm system responsible or the caller identity. In that case, the delta t, i.e. the time between triggering an alarm and resetting it is very small.

In summary, the goal of our machine learning approach is to learn whether the delta t is below or above a certain threshold value.

Details about the Experiments

For our experiments we used Apache Kafka and Apache Spark. Kafka delivers the alarm streams in real time while Spark is used for real-time processing and machine learning.

Our machine learning task was to classify alarms with two possible classes “positive” and “negative” (i.e. negative if the alarm occurred by mistake and positive otherwise). To solve our task we had more than 300,000 alarms containing around 25 different features. Moreover, we engineered additional features based on the given data. From the alarm event time, for example, we retrieved the day of the week and the hour of the day an alarm occurred.

Besides the event time, all our features are categorical and without a natural order (like locations). With these kinds of features the simplest modeling choice is to use Random Forests because no encoding of categorical features is required. In order to apply additional machine learning algorithms, we chose one-hot

encoding (also known as one-of-K encoding). With this encoding a feature results in a binary vector with the length equal to the number of different values with only a single bit set to one. The advantage of this encoding is that it does eliminate any unwanted order, meaning that the values are equitable to each other.

In addition to Random Forests we chose three algorithms that are often used for classification, namely Logistic Regression, Support Vector Machines and Deep Neural Networks.

To train our algorithms we first evaluated the given features. Therefore we used the Pearson correlation to find dependencies between features and labels as well as between features to each other. In addition, we ran a grid search for each algorithm varying the features used to train the models and finally selected the most promising features to learn our classification models.

To train and evaluate our approach, we divided the alarms into two sets for training and testing, containing about 170,000 alarms each. Additionally, we excluded about 10% of the training set to use it as validation set for selecting hyper parameters.

Results

In order to evaluate the effectiveness of our machine learning approach, we experimented with various values for delta t ranging between 1 and 10 minutes. The goals of our evaluation were as follows:

- Evaluate the accuracy of four different machine learning algorithms
- Study the impact of various deltas t on the prediction accuracy

The selection of hyper parameters for each of the learning algorithms (e.g. architecture of Neural Networks) was essential for the prediction accuracy. Even though, once we found suitable parameters we did not have to adjust them for the different deltas t (e.g. neither under- nor overfitting), showing that the complexity of the models is appropriate for our task.

Finding the hyper parameters was done with grid search. The following tables show the best suitable parameters found for our classification task.

Random Forest

Maximum depth of a tree	30
Number of trees to train	50

Support Vector Machine

Maximum number of iterations	2,000
Step size	1.0
Mini batch fraction	0.2
Regularization parameter	1e-2
Kernel	Linear
Update Function	Squared L2

Logistic Regression

Maximum number of iterations	500
Convergence tolerance of iterations	1e-6

Deep Neural Network

Maximum number of epochs	10,000
Mini batch size	200
Loss function	Cross Entropy
Update function	Nesterov Momentum
Learning rate	0.1
Momentum	0.9

Architecture (5 Layers)

Layer	**# Nodes**	**Type**	**Activation Function**
Input	803 Nodes		
Hidden 1	803 Nodes	Fully connected	ReLU
Hidden 2	16 Nodes	Fully connected	ReLU
Hidden 3	2 Nodes	Fully connected	ReLU
Output	2 Nodes	Fully connected	Softmax

Next we evaluated the performance of the machine learning algorithms and studied the impact of the alarm reset time delta t. Apart from Support Vector Machines, we observe that the performance of the algorithms is not affected by the deltas t (see Figure 2). Random Forest and Deep Neural Networks show the best performance with a prediction accuracy of up to 92%. These results show that our system is reliable even if we replace our hypothetic labels when we get access to the real ground truth in the future.

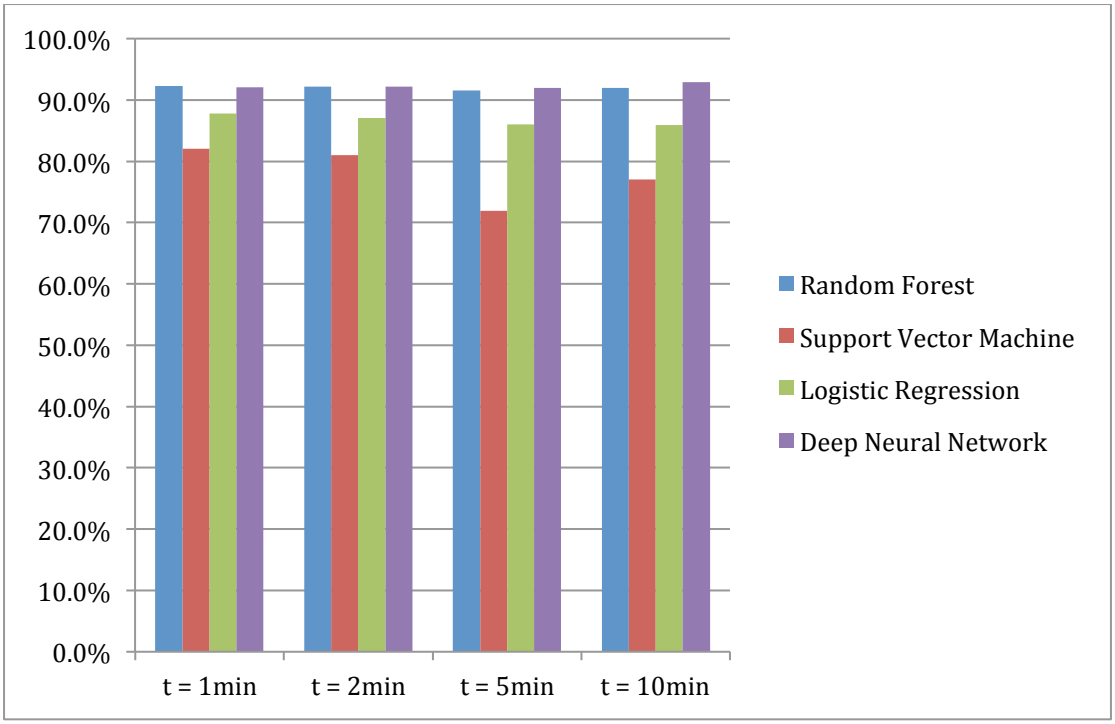

Figure 2. Evaluation of the machine learning algorithms used to predict alarm validity: prediction accuracy vs. delta t.

Conclusions

The results demonstrate that our machine learning approaches are very effective for predicting false alarms with an accuracy of up to 92%. These results can be directly used by typical alarm receiving centers for prioritizing alarms and thus have a large potential to significantly reduce the costs of dispatching intervention forces.

As part of future work we will integrate this machine learning approach in our alarm data warehouse to enable stream and batch processing. The idea is to apply the machine learning algorithms in real time on alarm streams and correlate the results with the alarm response to potentially further increase the accuracy of false alarm prediction.

Contact

Kurt Stockinger, Zurich University of Applied Sciences, Switzerland
Email: Kurt.Stockinger@zhaw.ch

References

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3, Article 15

LeCun, Y., Bengio, Y., & Hinton, G. 2015. Deep learning. *Nature*, *521*(7553), 436-444.

Felix Salfner, Maren Lenk, and Miroslaw Malek. 2010. A survey of online failure prediction methods. *ACM Comput. Surv.* 42, 3, Article 10