Unsupervised Anomaly Detection using *H2O.ai*

Peter Schrott
Berlin Institute of Technology
peter.schrott@campus.tu-berlin.de

Julian Voelkel
Berlin Institute of Technology
voelkel@campus.tu-berlin.de

1. INTRODUCTION

Anomaly detection (commonly referred to as *outlier detection*) is one of a few very common tasks in the field of Machine Learning. The goal of algorithms designed for the purpose of anomaly detection, are concerned with finding data in a dataset that does not conform to a pattern. That means, the goal is to identify data points, that are special in regards to their behavior, compared to the data points in the dataset, that are considered "normal". [2] These data points are called outliers, because they show different behavior than one would expect. Figure 1 shows a basic plot containing data points, with two different populations, along with two outliers, that do not seem to fit either pattern.

The value of identifying outliers in a dataset lies in the action one can take after detecting them. Two common applications of anomaly detection algorithms are health care and fraud detection. In the former, those algorithms can for instance help identifying sick patients, by identifying anomalous vital signs compared in a group of similar patients. In the latter application, those algorithms can account for fast, actionable information in case of credit card fraud, which can be identified by anomalous purchases, given the owners purchase pattern in form of historical purchases.
Anomaly detection can happen in a supervised, semi-supervised, as well as in an unsupervised fashion. The credit card fraud detection would be a semi-supervised learning task, since we can assume, that a new credit card will not be the subject of fraud for at least the first couple of purchases. Hence, we obtain a training set of "normal" purchases (i.e. data point) and can for each newly generated data point decide, whether it conforms to the pattern or does not. Since our project is focused exclusively on the unsupervised case where we do not know which data points are considered normal, but rather have to find a structure or pattern in the data first, in order to then be able to identify data points not conforming to the pattern, the next section will focus on unsupervised anomaly detection.

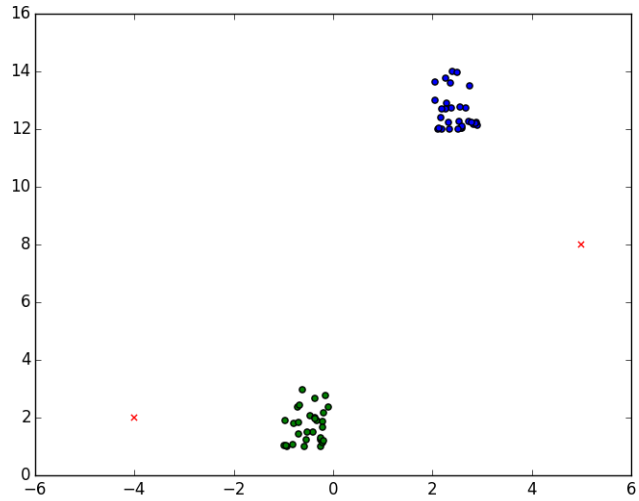

Figure 1: ADD SOME DESCRIPTIVE CAPTION HERE...

1.1 Challenges of Unsupervised Anomaly Detection

One difficulty that arises in unsupervised anomaly detection is, since we do not have any labels for training data, we do not even know what we are looking for. That is, we do not know what a "normal" data point would like, let alone what an anomalous point would look like. To put it in Ted Dunning and Ellen Friedman's words: "Anomaly detection is about finding what you don't know to look for." [1] Since there is no labeled training data in the most widely applicable case of unsupervised anomaly detection, the approach of finding outliers is a different one compared to training a model and then predicting to which class an unseen data point belongs to (much like binary classification). Instead, in case of unsupervised anomaly detection, we generally assume that the number of "normal" data points exceeds the number of anomalous data points by far.[2]. This assumption is fundamental to unsupervised outlier detection, since we would not be able to learn what is normal otherwise. Not being able to determine what "normal" is, means there is no way of finding what is anomalous. Since the goal of anomaly detection is finding what is anomalous, unsupervised anomaly detection usually starts with figuring out

what "normal" is. After achieving this (which, oftentimes, is much harder than it sounds), we can determine the deviation of a data point to what is "normal" using some similarity measure. There are, however, different algorithms dealing with the problem of unsupervised anomaly detection for different fields and problem domains. The main reason why there is no single approach applicable to each problem is, that there are tremendous differences in what is considered normal and what is considered anomalous, depending on the application domain we are looking at. Considering an example for this circumstance given in [2], one can easily imagine why that is the case: "The exact notion of an anomaly is different for different application domains. For example, in the medical domain a small deviation from normal (e.g., fluctuations in body temperature) might be an anomaly, while similar deviation in the stock market domain (e.g., fluctuations in the value of a stock) might be considered as normal. Thus applying a technique developed in one domain to another is not straightforward."

1.2 Deep Learning Auto-Encoder

?

2. PROBLEM STATEMENT

Anomaly detection refers to the task of identifying observations, that do not match the pattern of the data. Oftentimes anomaly detection happens in an unsupervised context. That is, the dataset being operated on is unlabeled, and the goal is to identify exactly those samples, that fit the pattern of the dataset the least. In this project, we shall analyze the performance of a particular algorithm in the field of unsupervised anomaly detection. Specifically, with this project, we aim at providing an answer or at least hints to the answer of the question: **Is a Deep Learning Auto-Encoder[1] well suited for anomaly detection in an unlabeled dataset?**

2.1 Target

As stated above, target of this project is to evaluate the quality of a Deep Learning Auto-Encoder model for the task of identifying anomalies in an unsupervised context.

2.2 Scope

This project consists of various different steps in order to obtain an answer to the problem specified above. These steps can be outlined as follows:

1. Study an existing implementation of the Deep Learning Auto-Encoder model
2. Apply this implementation to a given unlabeled dataset
3. Compare the outcome to already existing outcomes of other algorithms
4. Draw conclusions about the general suitability of the algorithm based on the results of the application of its implementation

3. METHODOLOGY

[1]An artificial neural network used for efficient codings.

4. EXPERIMENTS

5. RESULTS

6. CONCLUSION

7. REFERENCES

8. PROJECT PLAN

In order to being able reach our goal specified in **2. Problem Statement**, we need the following things:

1. An implementation of the Deep Learning Auto-Encoder model
2. A dataset, and
3. Reference results

8.1 Methodology, Objectives and Experiments

In this project, we use *H2O.ai*'s[2] implementation of the Deep Learning Auto-Encoder model through its Scala API on top of Apache spark. The dataset we use in order to being able to compare our results to those of our peers is the *AXA Driver Telematics Analysis* dataset[3], which contains multiple vehicle traces by multiple drivers. The catch with this dataset is that while there is a folder for each driver with a number of his or her respective traces, there is always a varying and unknown number of traces that were being generated by other drivers in that particular folder as well.
Since this dataset does not contain labels for the data, uploading our results to Kaggle, allows us to obtain a performance evaluation and being able to compare our results to other people's results, yielded from other algorithms.

9. REFERENCES

[1] E. F. Ted Dunning. *Practical Machine Learning: A New Look At Anomaly Detection*. O'Reilly, 2014.
[2] V. K. Varun Chandola, Arindam Banerjee. Anomaly detection : A survey. *ACM Computing Surveys*, September 2009.

[2]An open source parallel processing engine for machine learning
[3]This dataset was released to the public as a competition on `https://www.kaggle.com/`. It can be found here: `https://www.kaggle.com/c/axa-driver-telematics-analysis/data?drivers.zip`