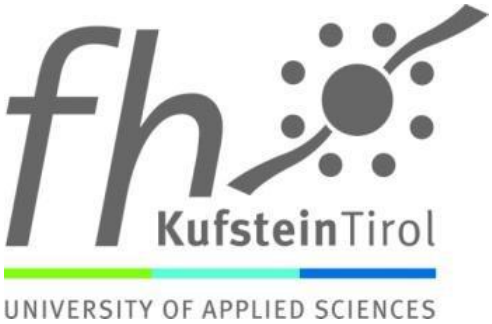

Data Science & Intelligent Analytics

Intelligent Analytics and Artificial Intelligence

Autor:
Jochen Hollich
1810837475

Betreuer:
Dr. Dietmar Millinger

München, 22.08.2020

Table of Content

Problem definition of the present paper

This paper was made in context of the course “Intelligent Analytics and Artificial Intelligence” in the master’s degree programme “Data-Science & Intelligent Analytics” at the university of applied science in Kufstein. The lecture was supervised by Dr. Dietmar Millinger, a research expert in the field of Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) and co-founder of the companies DECOMSYS and AI-Austria.

The basic idea of the paper is that the students get in touch with the elemental concepts of the problem framing of ML and AI related projects. As a frame scaffold for the task the students received following guidelines / formulation of questions:

> We have talked about a concise description of a machine learning project proposal. Please refer to the slide in the part III (page 8: Machine learning problem framing) as a template for your concept work.
>
> 1) Select a real-world problem from your domain of knowledge
>
> 2) Go through the questions of the problem framing template and develop answers with respect to your selected problem
>
> 3) Create a concept document with the found answers
>
> 4) Upload the concept document in the learning platform as PDF

The referenced slide (page 8: Machine learning problem framing) follows:

Machine learning problem framing

Description of **problem to solve** in the context of the user
- ➢ include cost of problem
- ➢ how might you solve your problem **without ML**?

What is the **ideal outcome** of the use of the model?
- ➢ how would the problem be reduced by using your approach

Define success and failure **metrics**
- ➢ measurable quality metrics in context of model AND problem

What **output** would you like the ML model to produce?
- ➢ e.g. classification of images, clustering of sensor data, ...

How will the output be **integrated** in a product
- ➢ interfaces

Identify your **data sources** and **labels**
- ➢ include semantics of data (metadata)

Identify your **data transformations**

https://developers.google.com/machine-learning/problem-framing/framing
https://medium.com/thelaunchpad/a-step-by-step-guide-to-machine-learning-problem-framing-6fc17126b981

Figure 1: "intelligent Analytics and Artificial Intelligence, Part III", slide 8, Dietmar Millinger

[1]

[1]

Machine Learning in the context of encrypted computer-network traffic

I decided to write this paper about the topic of my master's thesis with the title: "Machine Learning and Deep Learning in the context of Computer-Network-Traffic". The basic idea of this research paper is the structured development of AI/ML/DL algorithms being able to classify encrypted TCP/IP network traffic according to its specific individual destinations(Destination = URL-Call e.g a bundles of streams belonging to www.facebook.com, www.youtube.com....) based on their observed metadata. This basic ability to classify computer network traffic is the fundamental building block of further network management tasks like monitoring, network filtering, traffic shaping and pricing of used units.

Such "basic-rule" based classification system were entrenched in different software solutions (e.g. the librariesy nDPI, DPI). The functionality of such systems is the ability to record live network traffic in capture files (usually '*.pcap-files') and the subsequent analytics of the content of the specific capture. Without going too deeply into detail of the computer network traffic operating principle along the ISO-OSI model we can classify the recorded streams with specific data fields (the **Server Name Indication (SNI)** in the Transport Layer Security (TLS) in the package of the Client Hello, following to the TCP-IP three way handshake). This field is during the time of writing this paper still being transmitted in clear text.

In short:
In the most of recorded computer network communication streams is an unambiguous identifier in which context this communication took part. Based on this clear(=not encrypted) information we can build a rule-based software solution assigning this traffic to specific destinations.

A tool for basic "human-" network analysis is Wireshark. The following image shows the analysis of one stream with the destination to the Gitlab-Server of the FH-Kufstein, and the according SNI:

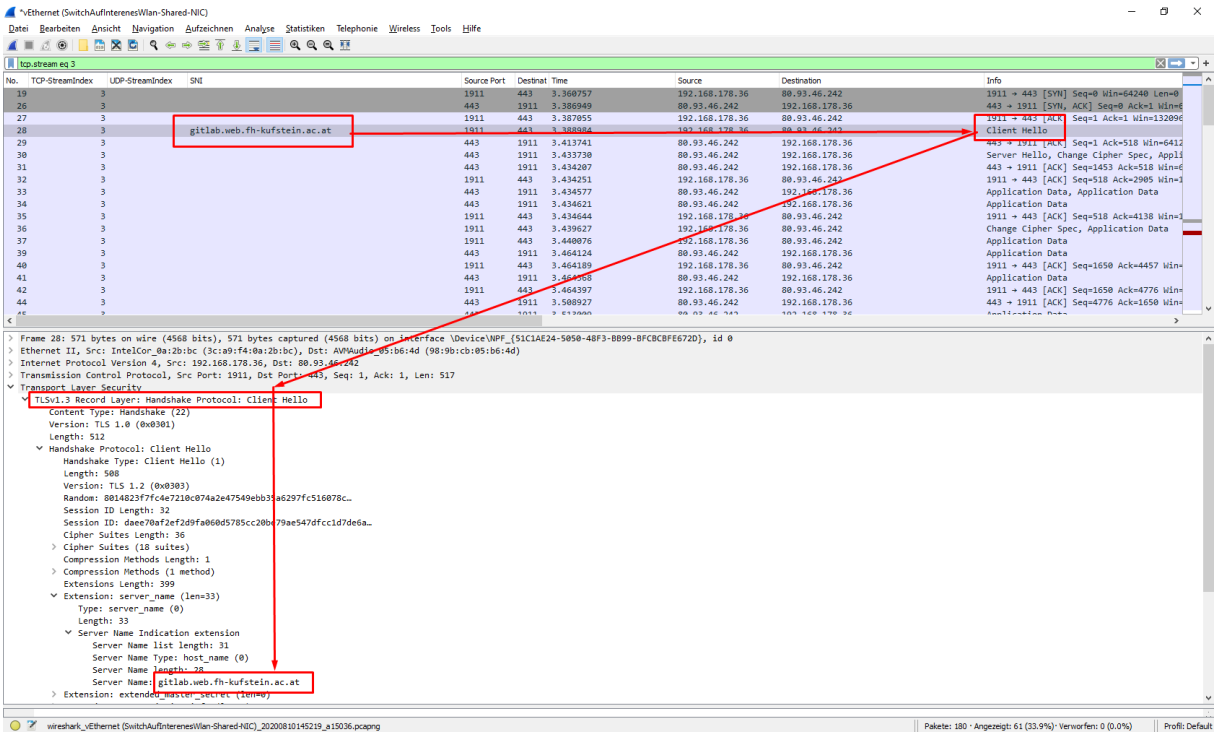

Figure 2 Own Graphics - Demonstration of the Wireshark and SNI

Within this screenshot we can detect the SNI "gitlab.web.fh-kufstein.ac.at" directly after the TCP-Handshake.

The computer-science and IT domain is driven by rapid development of new technologies and the continuous increase of the complexity according to the security measurements. The functionality of computer networking is managed by different companies (IETF, IRTF, IAB, IANA, IEEE) and the confirmed results are published in the Request of Change (RFC) documents. These RFCs are the basic template for producer and manufacturer who build devices and software solutions taking part in the global network (~ each new device which has an IP-Address = almost every IT-device). The basic idea is to provide guidelines for the network traffic operating principle which are implemented in each product in a uniform manner. With this identical implementation a platform independent communication between heterogeneous devices is possible. Consequently, the communication between different hardware vendors (HP, Siemens, Apple, Cisco, Dell, Intel…) and different software solutions (Microsoft, Unix, Debian, Android, MACOS, IOS…) is possible. These basic rules of implementation defined in the RFCs are the backbone of today's global internet technologies. Due to the need for implementation time of the hardware and software vendors according to the current valid RFCs these rules and the timeline of validation are developed in stages. The first stage is the publishment of a draft of RFC which is not valid during the time of publishment. The final RFC is published with a completion date from which the rules in the RFCs are valid.

The Draft "Encrypted Server Name Indication for TLS 1.3 draft-ietf-tls-esni-01" was published on September 18, 2018 and obtains the aim to encrypt the SNI in future network traffic classification.

This SNI-value in a clear (=not encrypted) format within the network traffic captures is the backbone of today's network traffic classification. Without an explicit assignment of captured network traffic, the network management tasks of filtering, traffic shaping and pricing are not possible anymore.

Consequently, there is a need for a new assignment logic of computer network communications without the cleartext information of the SNI. There are enough adequate fields of information besides the SNI within network communication which still can be captured in clear and be used for the stream assignment. With the difficulty being, there is no exact identifier like the SNI anymore. The "indirect assignment" being the pattern these values deliver and not one single value as identifier. Consequently, the classification must take part in the context of pattern analysis. For this task the tools of AI, ML and DL are an indispensable.

The resulting model is not directly a product of use and sale. Usually this model is implemented in the logic of a network bridging tool, a router logic, serial port scanner or any device which manages an intelligent internet access for the network participants.

Given Environment

The environment of computer networking infrastructure is volatile. Due to the upcoming of new technologies as well on the hardware as in the software side in the internet infrastructure (=Global Area Network 'GAN', Wide Area Network 'WAN' and Metropolitan Area Network 'MAN') the observed metainformations according on stream differ in irregular intervals. On the other hand, the restructuring of given technologies also results in different observations and captures.

According to the given AI/ML/DL setup this volatile circumstances consequent in regular training the models on "most-recent" labelled data and validate the correct operating mode of the current production model.

Labelled Data - Data-Generation-Lab

Training and validating a classification model (=supervised learning) requires the existence of labelled network traffic. The acquisition of labelled data in the networking context is complex and technical expensive. One way to generate such a labelled dataset is to build a controlled environment (~ Computer network lab) and generate and capture targeted traffic. With this basic capture and the following expensive data preparation, cleaning and filtering it is possible to build a labelled dataset suitable for training and validation classification algorithms. This lab was created on a virtualized environment (based on MS-Hyper-V) and was able to deliver the preclassified network captures.

Just as a rule of thumb:
10 seconds of capturing YouTube -stream traffic generates ~4000 data packages (= row in a table) with ~250 package specific informations (=columns|feature in a table). These packages are organized in ~60 UDP & TCP streams (all belonging to the same YouTube stream for the end consumer) resulting in a *.pcap-capture file of ~5mb.

Within this capture it is secured that the streams which belong to the aimed YouTube-stream call are stored the file. But on the other hand, there are also other streams captured which are not necessarily characteristic for this aimed YouTube-stream. The data preparation, cleaning and filtering process prepares the captures for being used in the training and validation process.

In short, the sole generation of labelled data is technical expensive and time consuming. Due the fact that we need a label dataset as

- most-recent
- to as many url-destinations
- often

as possible it is essential to automate the whole process in a high degree. The scripts for the automated data generation process which are executed in the controlled lab can be found in this Gitlab-Repo.

With storing the captures it is essential to pre-process the data as fast as possible, so that these informations can be used for the training and validation of the models.

Figure 3 Own Graphics: Labeled-Data-Generation-Process

This process is realized within the Python scripts in following GitLab-Repo.

Furthermore, during the above cleaning and data-preparation process occurs different states and metainformation which are needed in different stages during the modelling and the Explorative Data Analysis (EDA). For managing these informations a Maria Database was created to store. This Maria-DB was built with following Data-Definition-Language and has following ER-Structure:

Figure 4 Own Graphics: ER-Diagram

As already mentioned, this process is resource and time consuming. Based on this circumstance a gitlab CI-CD pipeline based on the K8s- infrastructure of the FH-Kufstein was created. An example of the possible implementation of these scripts can be found in this GITLAB-Repo.

Training the models

Based on the labelled dataset of the previous data-generation process it is possible to train different models. These models defer on the focus which specific part of the labelled network traffic (which of the 250 feature/informations/"columns") is used as the input for the current model. The following Links refer to different implementation options:

1) Analysis of the header information from the Ethernet and IP Layer
2) Analysis of the payload
3) Analysis of statistical features
4) Analysis of time series Data

The best models have a test-data accuracy of ~95%.

Description of problem to solve in the context of the user

For this section we will focus on two different possible interactions with the product of an intelligent internet access device with an AI/ML/DL classification concept along an example (the model we trained above).

For example, it is important in the context of a hotel for the guests to have a sufficient bandwidth for their current need. The extent of the individual need (=individual demand for bandwidth) defers depending on the kind of the specific traffic generation. The use of streaming services (Netflix, Amazon Prime Video, Sky, etc.) generates a much higher individual demand of bandwidth than normal websurfing or basic remote working traffic.

The amount of the resource bandwidth of an complete institution (e.g. the hotel) depends on the geographical location, the individual contract with the Internet-Service-Provided (ISP), the used technology for the network communication (fiber(light), copper (electrical), radio (wave) transmission) and the level of development of the individual technology. This common resource an institution receives is shared by the individual participants in this institution.

POV Hotel Management

The responsible persons of a hotel want to provide useful and suitable infrastructure for their guests. Consequently, they must be able to automatically manage different load situations. In the scenario of a high demand of bandwidth the existence of an automated traffic shaping is indispensable. In case a guest needs more resources than the automatic assignment, this person must be able to buy further bandwidth resources. Meaning a guest can buy a priority account, with which a higher amount of networking resources is assigned to this specific privileged account in the network management logic.

POV Hotel Guest

Nowadays every hotel guest expects the existence of working internet infrastructure for the basic internet interactions. In case of a higher demand for bandwidth the guest should receive the information about his “unusual” high demand, the explanation why this individual demand can’t be handled at the time of request and the reference to the ability to buy a privileged account.

Ideal outcome of the use of the model

The basic idea of the ideal outcome is of the model working properly. This can be measured by a technical/statistical and on an economical POV.

Statistical approach

According to a rational statistical approach the fundamental approach of the problem is a classification model (=estimation of category/url-destinations from captured input values). Classification is an AI /ML /DL strategy of the supervised algorithms. Consequently, the following performance measures relate on the requirement of the existence of a labelled dataset.

The performance of a “fresh-trained” classification model is measured by following KPIs:

- **Accuracy**
  *“Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model. For our model, we have got 0.803 which means our model is approx. 80% accurate.” (*https://blog.exsilio.com/ *)*

- **Precision**

  *"Precision is the ratio of correctly predicted positive observations to the total predicted positive observations."(*https://blog.exsilio.com/)

  In terms of the hotel example:

  How many of the predicted streams by our model where actually correct. High precision relates to the low false positive rate.

  Recall

  *"Recall is the ratio of correctly predicted positive observations to the all observations in actual class" (*https://blog.exsilio.com/)

  In terms of the hotel example:

  How many of all known labelled streams of our known test-dataset did the algorithm detect.

  F1-Score

  *"F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall." (*https://blog.exsilio.com/*)*

  In terms of the hotel example:

- **Receiver Operating characteristic (ROC-Curve) & Area under the curve (AUC)**

  *"AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes." (*Link*)*

  In terms of the hotel example:

  The higher the AUC is, the better the model can distinguish between the different classes.

Economic approach

With the developed product there should be value in the offered service. The product with the integrated AI/ML/DL solution will generate value in the user experience as well on the side of the managing persons as on the side of the consuming persons.

Consequently, the customer feedback will directly mirror the quality of service of the whole product and indirectly the quality of the AI/ML/DL implementation.

Desired output of the Lifecycle and tech-stack

As already descried in the chapter "Given Environment" the circumstances for the model are volatile. This volatile surrounding must be taken into consideration during the development of the model. Rephrasing the previous sentence:

There must be a logic which generates controlled and labelled traffic by which the developer is able to build the models, as well as to validate the already trained models and confirm the correct operating principle. These two steps should be done in an effective, transparent, and comprehensible way. With the existence of an automated "labelled-data-generation tool" it is possible to build a pipeline to train and validate the models in an automated way.

For the creation of robust, flexible and powerful pipeline it is possible to use technologies for containerization-tool like Docker and the container orchestration tool Kubernetes for the basic development of the models. With the use of this tech-stack we can build efficient, complete automatable, powerful, and self-scaling pipelines which adapt to the current environment and the

given needs. These pipelines start by picking up the labelled data and end with the automated deployment of the most recent trained model to the production systems.

With the tool “data-version-control” (DVC) we can realize the concept of transparent and comprehensible development to the architecture. This tools tracks which dataset results in which deployed model.

In short:

The developer or data scientist aims to build a logic by which the task of building an effective classification model has a degree of automation which is as high as possible. Additionally, the process should be transparent, comprehensible, and efficient.

Integration of the model in a product

Most of the “intelligent” network traffic management devices are based around the Linux-kernel. Consequently, the final model is used in the programming language C. It is possible to load and use (predict) python trained models in a C-Environment. With this the most recent trained model is deployed to a shared folder between the device having trained the model centralized on a computational high potent machine. The final model is deployed to a specific shared instance (Git-Repo/FTP-Server/Directly to production system). The production system picks the most recent model (e.g. by a webhook or an automated script) and deploys it in the production system.