How to put a machine learning model into production?

Tamás Kurics, PhD

Senior data scientist at One Identity

Introduction

Balabit/One Identity – Blindspotter/PAM

More monitoring less control

- **Balabit** is an IT security company specialized in log management and advanced monitoring technologies. Founded in 2000, acquired by One Identity in 2018.

- **Blindspotter** is/was a user behaviour analytics tool that detects deviations from normal behavior and creates a priority list of anomalous, potentially risky sessions. Started as a separate product in 2014, merged with Shell Control Box in 2017 and the unified product is now part of the One Identity Privileged Access Management Solutions package.

Software development who is who

Data science can be used in a variety of ways, from ad-hoc analysis to deployed models used by customers on a daily basis.

When the work of the data science team is incorporated into a software, then cooperation between at least 3 different teams is needed:

- Product team
- Data science team
- Development team

Software development who is who

Data science can be used in a variety of ways, from ad-hoc analysis to deployed models used by customers on a daily basis.

When the work of the data science team is incorporated into a software, then cooperation between at least 3 different teams is needed:

- Product team
- Data science team
- Development team
- Data Engineers (data storage, ETL processes)
- DevOps (deployment, maintaining clusters)
- ...

Life-cycle of a machine learning algorithm

- *Product* (2013?): we need a shiny new product that creates a priority list of sessions based on unusualness, using the session metadata and session content recorded by Shell Control Box

Life-cycle of a machine learning algorithm

- *Product* (2013?): we need a shiny new product that creates a priority list of sessions based on unusualness, using the session metadata and session content recorded by Shell Control Box
- *Development* (Apr. 2014): start to develop a new product called Blindspotter in Python (easy collaboration, quick development cycle)

Life-cycle of a machine learning algorithm

- *Product* (2013?): we need a shiny new product that creates a priority list of sessions based on unusualness, using the session metadata and session content recorded by Shell Control Box
- *Development* (Apr. 2014): start to develop a new product called Blindspotter in Python (easy collaboration, quick development cycle)
- *Development* (July 2015): first release including several anomaly detection algorithms

Life-cycle of a machine learning algorithm

- *Product* (2013?): we need a shiny new product that creates a priority list of sessions based on unusualness, using the session metadata and session content recorded by Shell Control Box
- *Development* (Apr. 2014): start to develop a new product called Blindspotter in Python (easy collaboration, quick development cycle)
- *Development* (July 2015): first release including several anomaly detection algorithms
- *Product* (Oct. 2015): check feasibility of biometric algorithms

Life-cycle of a machine learning algorithm

- *Product* (2013?): we need a shiny new product that creates a priority list of sessions based on unusualness, using the session metadata and session content recorded by Shell Control Box
- *Development* (Apr. 2014): start to develop a new product called Blindspotter in Python (easy collaboration, quick development cycle)
- *Development* (July 2015): first release including several anomaly detection algorithms
- *Product* (Oct. 2015): check feasibility of biometric algorithms
- *Data Science* (Nov. 2015): the answer is affirmative

Life-cycle of a machine learning algorithm

- *Product* (2013?): we need a shiny new product that creates a priority list of sessions based on unusualness, using the session metadata and session content recorded by Shell Control Box
- *Development* (Apr. 2014): start to develop a new product called Blindspotter in Python (easy collaboration, quick development cycle)
- *Development* (July 2015): first release including several anomaly detection algorithms
- *Product* (Oct. 2015): check feasibility of biometric algorithms
- *Data Science* (Nov. 2015): the answer is affirmative
- *Product* (Jan. 2016): create PoC and production code for a keystroke and a mouse movement algorithm

Life-cycle of a machine learning algorithm

- *Product* (2013?): we need a shiny new product that creates a priority list of sessions based on unusualness, using the session metadata and session content recorded by Shell Control Box
- *Development* (Apr. 2014): start to develop a new product called Blindspotter in Python (easy collaboration, quick development cycle)
- *Development* (July 2015): first release including several anomaly detection algorithms
- *Product* (Oct. 2015): check feasibility of biometric algorithms
- *Data Science* (Nov. 2015): the answer is affirmative
- *Product* (Jan. 2016): create PoC and production code for a keystroke and a mouse movement algorithm
- *Data Science / Development* (Mar. 2016): Prototypes are ready, keystroke algorithm is in production

Life-cycle of a machine learning algorithm

- *Product* (2013?): we need a shiny new product that creates a priority list of sessions based on unusualness, using the session metadata and session content recorded by Shell Control Box
- *Development* (Apr. 2014): start to develop a new product called Blindspotter in Python (easy collaboration, quick development cycle)
- *Development* (July 2015): first release including several anomaly detection algorithms
- *Product* (Oct. 2015): check feasibility of biometric algorithms
- *Data Science* (Nov. 2015): the answer is affirmative
- *Product* (Jan. 2016): create PoC and production code for a keystroke and a mouse movement algorithm
- *Data Science / Development* (Mar. 2016): Prototypes are ready, keystroke algorithm is in production
- *Data Science / Development* (Jun. 2016): Mouse movement and pointing device detection algorithms are in production

Life-cycle of a machine learning algorithm

In an ideal world this would be the end of the story.

Life-cycle of a machine learning algorithm

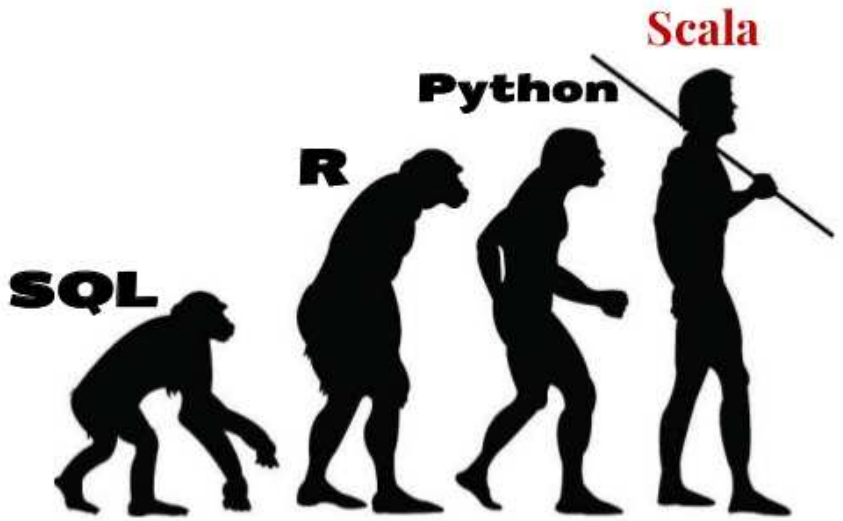
In an ideal world this would be the end of the story.

- *Development* (Aug. 2016): Decided to change the language. Scala and Spark were introduced and Python was retired (we had no idea before what the JVM is...)

Life-cycle of a machine learning algorithm

In an ideal world this would be the end of the story.

- *Development* (Aug. 2016): Decided to change the language. Scala and Spark were introduced and Python was retired (we had no idea before what the JVM is...)
- And that's not all!



Mouse Movement Algorithm

Mouse Movement Algorithm

Raw data format recorded by SCB

record timestamp	client timestamp	button	state	x	y
1434623150.819000	4053813.760000	NoButton	Move	364	123
1434623151.483000	4053814.415000	NoButton	Move	363	125
1434623151.620000	4053814.556000	Left	Pressed	363	125
1434623151.812000	4053814.758000	NoButton	Drag	364	125
1434623151.967000	4053814.852000	NoButton	Drag	374	123
1434623152.148000	4053814.961000	NoButton	Drag	400	123
1434623152.148000	4053815.086000	NoButton	Drag	420	123
1434623152.268000	4053815.211000	NoButton	Drag	425	123
1434623152.459000	4053815.398000	Left	Released	425	123
1434623153.387000	4053816.318000	NoButton	Move	425	124

Challenges:

- The data might be different depending on the OS, the pointing device, the screen resolution, the settings of the mouse sensitivity and so on.
- We have no information on the users' settings. We might give false positives if a user is logged in through a new (unusal) computer.
- The timestamps might show bursting phenomena, depending on protocol and CPU load.

Mouse algorithm – Act I

Used in the Python version of Blindspotter [Nov. 2015 – Feb. 2017]

Before doing anything, read the literature!

- data preprocessing
- model building

Mouse algorithm – Act I

Used in the Python version of Blindspotter [Nov. 2015 – Feb. 2017]

Before doing anything, read the literature!

- data preprocessing
- model building

Data preprocessing steps:

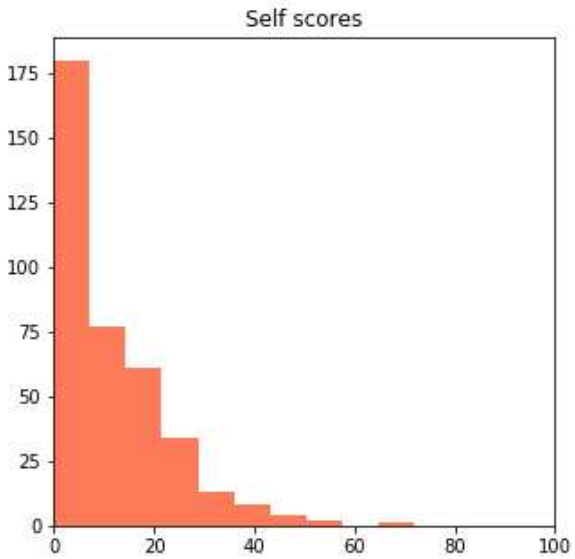
- select timestamp column (when both are available), fix coordinate overflow
- identify gestures: sequence of mouse movement records that ends with a pause or a click
- Each gesture will be represented with a feature vector
- For each gesture we calculate several features like angles, directions, velocities, accelerations and aggregate them by using various statistics (mean, median, min, max, range, percentiles, etc.)
- Handle missing / NA / NaN values

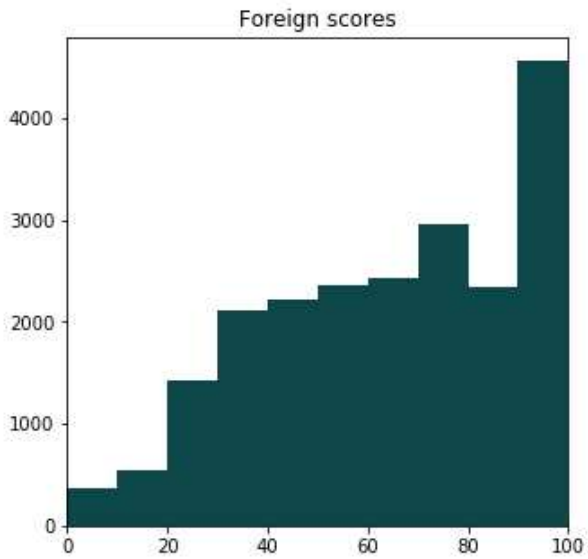
Mouse algorithm – Act I

Modeling steps:

- (Train a logistic regression model for device detection on in-house data since in production we won't have labeled data)
- Create labeled data by mixing own data with samples from other users. Ensure that the foreign data are collected from several different users, the selected sessions should contain enough gestures, but the overall number of foreign and own gestures should be balanced.
- Build a supervised user authentication model for each user on a (pre-selected subset of features), a random forest classifier. (In the PoC phase we tried logistic regression, neural networks, SVM and random forests.)
- At test time score each gesture separately and aggregate them to get a final score

Mouse algorithm – Act I





- gesture-level/user: 0.64 – 0.8
- session-level/user: 0.89 – 1.0
- global session AUC: 0.96

Pros: the model is available in scikit-learn, good performance for small number of users

Cons: slow to build models for everyone, mixing step requires to fetch other users' data

Mouse algorithm – Act II

Used in the Scala / Spark version of Blindspotter [March. 2017 – Nov. 2017]

Data preprocessing is the same as before.

The random forest implementation in Spark MLlib provides lower accuracy and its memory management on a single core is not very efficient (at least this was the case around 2016), so we decided to use another ensemble model: Gradient Boosting Machine from H_2O

Problems we have faced with:

- At that time the Scala API was immature and really hard to use with very little documentation.
- How to use Spark and H_2O in production?
- How to keep alive Spark on the box?
- How to achieve the goal that scoring should not use Spark, only baseline-building?
- How to run data science experiments (say, on new data) using the production code?

Mouse algorithm – Act II

In Aug. 2017 the Product team decided to merge Blindspotter (analytics) with Shell Control Box (recording, indexing, search) and hit the PAM market with a unified product.

Some consequences of this decision:

- The data processing, scoring and baseline-building pipeline on the top of SCB should be production-ready within months
- From now on memory and CPU capacity is very limited
- Some algorithms should be ported to PAM as soon as possible, others might wait
- External dependencies should be minimized

As a result, we said good-bye to Spark and H_2O.

Mouse algorithm – Act II

In Aug. 2017 the Product team decided to merge Blindspotter (analytics) with Shell Control Box (recording, indexing, search) and hit the PAM market with a unified product.

Some consequences of this decision:

- The data processing, scoring and baseline-building pipeline on the top of SCB should be production-ready within months
- From now on memory and CPU capacity is very limited
- Some algorithms should be ported to PAM as soon as possible, others might wait
- External dependencies should be minimized

As a result, we said good-bye to Spark and H_2O.

So we still need a mouse algorithm that is

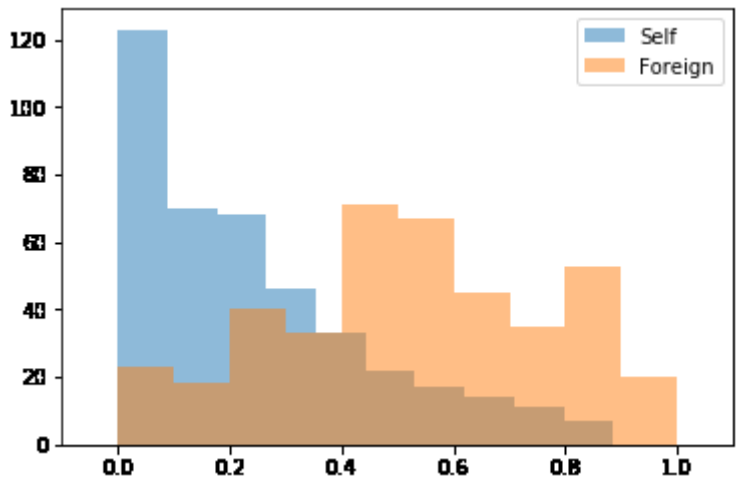
- computationally cheap $\rightarrow$ Act III
- capable of distinguishing users with exceptional AUC and good score distribution $\rightarrow$ Act IV

Mouse algorithm – Act III

not yet in production [Spring 2019?]

A simple idea for a one-class model: compare whether the values of a given feature from the training and test gestures are drawn from the same distribution.

- Data preprocessing and gesture extraction are the same as before
- Feature vector representation of each gesture is the same as before
- For each feaure find statistical evidence against H_0: feature values from the gestures of the session and from gestures in the baseline are drawn from the same distribution
- AUC is good enough, around 0.8
- Score distribution is not that good, unfortunately



Mouse algorithm – Act IV

not in production – waiting for a cloudy day [Maybe sometime?]

Which buzzword have you not seen so far?

Mouse algorithm – Act IV

not in production – waiting for a cloudy day [Maybe sometime?]

Which buzzword have you not seen so far? (Blockchain, AI, Deep Learning)

- Data preprocessing: replace timestamps with time-difference, scale pixel coordinates, split sessions into parts containing same number of raw mouse records.
- Build a supervised model that predicts the owner of a given mouse record sequence: instead of trying to predict the next element in a sequence (time-series analysis), we assign a label for a given sequence of mouse records, where the order of the sequence is important (sequence learning).

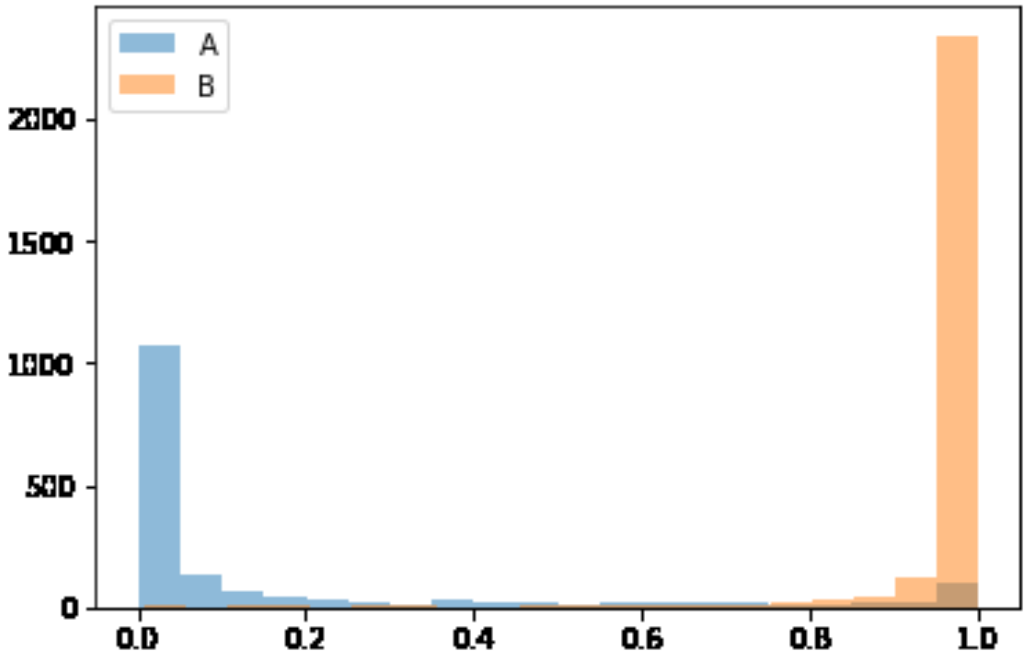
- The model of our choice was LSTM (J. Schmidhuber – S. Hochreiter, 1997, nowadays widely used in speech recognition problems and chatbot applications)

Mouse algorithm – Act IV

- Since this is a supervised model, we need either a large global, or several locally global models to train
- Training is efficient on GPUs
- The PAM box does not have GPU, the resources are limited, and extra dependencies needed (Tensorflow, Keras)
- The problems can be solved by using a cloud infrastructure (AWS, Microsoft Azure), but although it solves some of our problems, introduces many others:
  - maintaining a cloud infrastructure
  - which way to use it: infrastructure as a service?
  - how much does it cost?
  - where is the data?
  - how can we get the baseline
  - how do we score (in the cloud or in the box)

Mouse algorithm – Act IV

Separation of gestures by using a binary model built on 2 users:



- AUC: 0.97

Conclusions

- When a company uses ML/AI/DL in their product, a strong collaboration is required between product team, software engineers and data scientists
- Develop good enough products (accept the resulting method, otherwise nothing will be ready)
- There is no such thing as something is ready or done anyway...
- It would be nice if (apart from the customers) the data scientists could also use the production-grade product developed by the engineers
- Try to come up with domain-driven solutions instead of hype-driven ones
- The real product may contain different solutions than the one the Marketing team advertises/promotes and it's not even a bad thing