Development and validation of an application with artificial intelligent feature : Men fertility web service case-study

Zigmann Bastien

bastien@zigmann.org
IM2AG, UGA, Grenoble, France
CS27, Kobe university, Japan

Abstract. Artificial intelligence (AI) is changing software application scope and the way applications are developed. More and more are using machine learning (ML) based AI. With those new techniques come new problems, such as validate, assure security and efficiency of this kind of software. The main challenge is brought by the fact that the main characteristic of ML based software is a kind of *magic box*. We can't predict and control its behavior.
In this context, web services provide an interface for developers in order to use cloud stored ML models. Therefore the validation of this kind of software will be centered on anything but this model.

Keywords: Software · Validation · Machine Learning · Artificial intelligence · Microsoft Azure.

1 Introduction

ML based software development brings new ways to think about logics and operation in development. The data based learning conducts to software with no specifications and unpredictable behaviors. In order to validate this kind of software, we need to rethink the way we were validating classic software.

The collaboration between the LIG (Grenoble, France) and the Graduate School of System Informatics (Kobe, Japan) aims to evaluate the usability of traces and property expression for the validation of ML based software. For this purpose, the project aims at collecting some experience on ML based web service development and validation.

In section 2, we will explain what are the traces and how it could be useful for validating ML based software. Then we will talk about developing intelligent application with web service using Azure ML Studio in section 3. After this, in section 4, we present the application developed. This application will be our base to work with traces later. Finally, we will discuss of the work that will be done with the example application and traces.

2 Traces in validation science : Introducing ParTraP

Traces are a way to keep informations on events and every other things possibly interesting. A trace is a sequence of events that we will verify some properties on the events recorded, such as absence or occurrence of informations, row of events.. Many language exist to treat traces, we use ParTraP.

2.1 ParTraP

ParTraP is a language, developed at LIG [3] designed to easily express properties on traces. ParTraP traces use JSON format. Each JSON have to be composed at least of an ID and a time, after this we can add any kind of informations we need. To study the execution of a program, you need to include logging instructions well chosen, so it is possible to follow the behavior during execution. ParTraP aim to express properties and events with parameters to do some test on traces.

As explain in [2], a toolset is necessary to use ParTraP. This toolset need to be installed on eclipse IDE (details about installation are in [2]).
Once we have a trace, we can write ParTraP properties. With the toolset, properties are executed and output files are created to store the results.

Let us illustrate ParTraP with a chat application we made in the Distributed System UE. A chat is an application that aims at connecting clients amongst themselves in order to send some messages to each others and have some discussions. To use it, a server is launch to host the application, then clients can connect to this server to join the chat. First, the client have to choose a name to be print in chat. Then the server, by default, put the client in the first room of the chat. At launch, server does only have one room. From the moment where the client has join the room, he can speak with all the other clients present in the same room. The client also have a list of available command. The client can create and join rooms. The messages from clients in a room are sent only to the other clients in the same room.

To trace the execution of the chat application, I made a small java class named TraceManager[4] to write JSON corresponding to ParTraP specification. In order to follow chat execution, I created a small trace on sent and received messages. This way it is possible check the following properties:

- the sent message is received by the users in the room (and not users in other rooms)
- the received time is after the sent time
- the sender don't receive his own message

Here is a possible trace for a room with 2 clients in and a single message :

{"id":"Sent", "time":"1553528124008", "sender":"Client1", "senderID":"1", "roomID":"0", "roomclients":"[1, 2]", "content":"hi"}

{"id":"Received", "time":"1553528124013", "sender":"Client1",
"senderID":"1", "roomID":"0", "roomclients":"[1, 2]",
"content":"hi", "receiver":"Client2", "receiverID":"2"}

And here is the ParTraP script used to check it :

```
Reception: after each Sent S,
    forall u in S.roomclients,
    occurrence_of Received R
    where S.content == R.content && S.roomID == R.roomID &&
    u == R.receiverID && R.receiverID != S.senderID &&
    S.time <= R.time;

NotReception: after each Sent S,
    absence_of Received R
    where S.roomID != R.roomID;

ReceptionCheck: Reception and NotReception;
```

To generate traces, we use chat application to send messages between at least two clients, then we evaluate the properties. The evaluation of ReceptionCheck tells that the properties "Reception" and "NotReception" are true. This means that our trace don't present any problems.
I also did tests on more complicated traces with multiples clients and rooms. For each of them, the properties were evaluated to true.

3 Intelligent application using web service

Intelligent applications are using data to learn about how to behave. First the data have to be choosed (or collected) then a ML training is used to create a model. The model is used to make predictions and to allow applications to make some kind of decision. The idea is that the model will behave according to the "experience" brought by the data. This is exactly that moment where developers can't predict the behavior.

3.1 Azure ML Studio and web services

Azure Machine Learning Studio is a tool developed by Microsoft to easily create ML based experiments with your dataset in order to deploy them to real applications. The site allows you to add your own dataset, do some operations on it (for instance select a subset of data, clean or split dataset..etc), build models with ML algorithm and test them. You can also create web services to use the model you created in your applications. The web service is used as an interface between the application and the model prediction (managed by Azure).

Fig. 1 show example of a learning phase (model creation) for the basic experiment made with Azure tutorial [1]. As it can be seen on the figure, Azure

requires the developer to describe the set of operations to be done as a workflow. Typically in this example, there is a pre treatment phase on data to prepare them for the learning phase, then the dataset is splited taking 75% to train our model and 25% to score it in order to evaluate it. The model is trained to learn a linear regression.

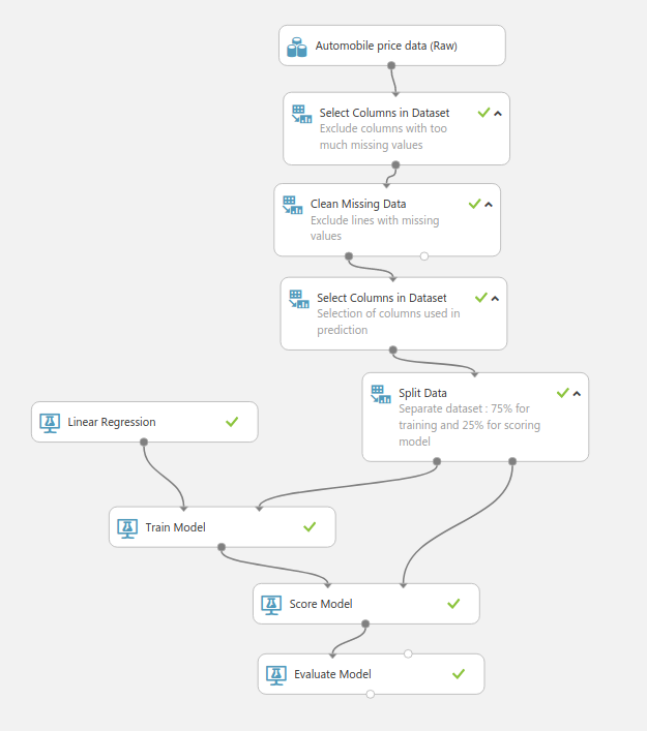

Fig. 1. Model creation for predicting automobile price using Azure ML Studio web interface

When the model is trained it is possible to use it for a prediction. This is where web service is useful. It allows us to enter any kind of data (within external applications) that will be used on the model. Then the web service output (the model prediction) will be send back to the asking application. (see Fig. 3 for details). The web service access is made with an URL and an API key.

4 Men fertility prediction

4.1 Specification and Model creation

In the following, we developed a small application using ML prediction and decision, called "men fertility prediction".

The idea is to ask few questions to 18 to 36 years old men in order to predict if their fertility seems normal or altered. The dataset [5] is quite small and contains only discrete values. It has 9 attributes for 1 single binary decision. The informations required are :

- The season at the moment of the test
- The user's age
- Does he had childish diseases?
- Does he had accident or serious trauma?
- Does he had surgical intervention?
- Does he had high fevers in the last year?
- What's his alcohol consumption?
- Is he smoking (if yes in which regularity)?
- The time he spent sitting in a day

The output of the dataset only tell if the fertility seems Normal or Altered.

To make my ML model, I choosed to use a two class boosted decision tree for its reliability and efficiency [6]. The dataset is not really big so I choosed split the data with 90% to train the model and 10% to score it.
Fig. 2 show the model training experiment.

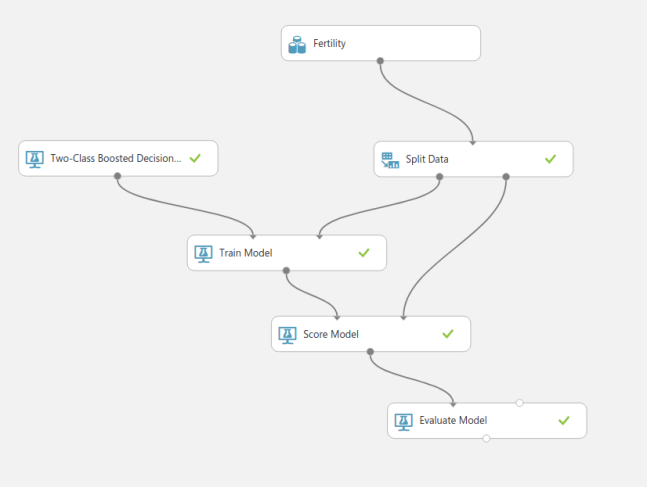

Fig. 2. Model creation for men fertility application

After creating the model and in order to use it in an external application, we need to create a predictive experiment. The predictive experiment is the part web service use to predict result with the model created and the user input coming from the web service.
Fig. 3 show the predictive experiment, we use the model created before (in Fig.

2), and the web service input on this model. Then we select two specific column, Scored Labels and Scored Probabilities. The first one is the answer of the model (Normal or Altered) and the second one is the confidence percentage for this result. This two attributes are sent in the web service output to be used by our script.

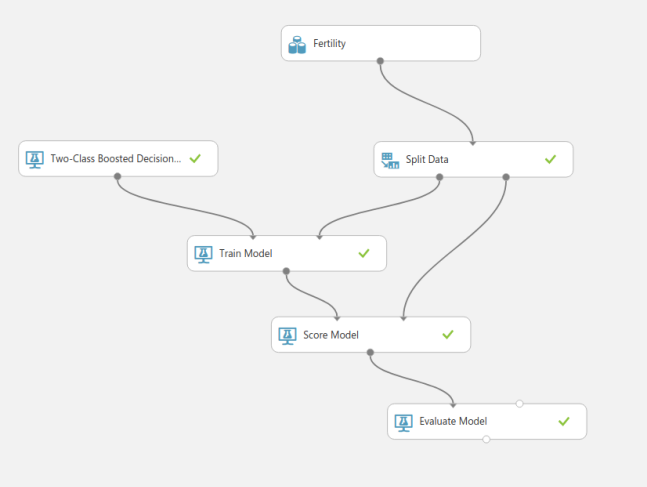

Fig. 3. Predictive experiment creation for men fertility application

4.2 Web service framework and Validation

The web service was based on the outline provided by Azure ML Studio. A small command line interaction asks the user to enter the needed informations. Then the computation is made by Azure and its web service. Fig 4. show the script execution.
The application gives some personal advices according to user entries (in the case of altered fertility). The Fig. 5 show all the possible advices that can be return by the script.

```
-------------------------------------------------------------
This test is for men fertility between 18 and 36 years old.
-------------------------------------------------------------
Please enter the actual season
1:winter
2:spring
3:summer
4:fall
2
-------------------------------------------------------------
Please enter your age (must be between 18 and 36) :
22
-------------------------------------------------------------
Did you had childish diseases (for instance chicken pox, measles, mumps, polio ..)?
0:yes
1:no
1
-------------------------------------------------------------
Did you had any accident or serious trauma ?
0:yes
1:no
1
-------------------------------------------------------------
Did you had any surgical intervention ?
0:yes
1:no
1
-------------------------------------------------------------
Did you had high fever during last year?
-1:less than three months ago
 0:more than tree months ago
 1:no
1
-------------------------------------------------------------
What's your alcohol consumption frequency ?
1:several times a day
2:every day
3:several times a week
4:once a week
5:hardly ever or never
3
-------------------------------------------------------------
Are you smoking?
-1:never
 0:occasionnally
 1:daily
1
-------------------------------------------------------------
How much time do you spend sit per day?
10
-------------------------------------------------------------
It's the good time to have a baby !
```

Fig. 4. Normal execution with good news for the user

```
Your fertility seems altered.
The best season for fertility is winter
You should go for a check
You should wait for few month
Maybe you should try to change your habbits (less drinking or smoking for example)
```

Fig. 5. All the possible advices for altered fertility

In order to validate this application, we need to find some interesting points during execution to create a trace to analyze. The next step is to create a trace of entries and output return by the application. This is possible because we are working with full set of discrete values. So when a set of values is send, we can trace these values and the following answer. With all those traces stored we will

easily detect if application behavior is suspicious. For instance if a tier attacked the software and modified the result, we will see the difference with the stored result.

5 Conclusion and perspectives

In order to get familiar with AI based software we used Azure and its web service to develop a decision application working with a machine learning model. Using Azure is a good way to focus on software side resting on Microsoft implementation for ML algorithms. We also worked with ParTraP to learn about trace validation and discover its functionalities.

As we seen in 4.2, possible traces and ParTraP analyzes can be done to assure veracity of results. We now need to think about deeper validation and other properties to trace. The goal of the next step is to evaluate the usability of ParTraP to ensure validity and efficiency (even security) of our ML based application.

6 Acknowledgment

I would like to thanks Nakamura Masahide, all his team and Kobe University for welcoming us in there laboratory. Also thanks to Lydie Du Bousquet and Philippe Lalanda for accompanying us during our internship, the preparation of our travel and also while we are in Japan. Finally thanks to Auvergne-Rhne-Alpes for financing our travel to Japan.

References

[1] *Azure tutorial for making first experiment on automobile price prediction.* URL: https://docs.microsoft.com/fr-fr/azure/machine-learning/studio/create-experiment.

[2] Ansem Ben Cheikh et al. "An Environment for the ParTraP Trace Property Language (Tool Demonstration)". In: *Runtime Verification - 18th International Conference, RV 2018, Limassol, Cyprus, November 10-13, 2018, Proceedings*. Ed. by Christian Colombo and Martin Leucker. Vol. 11237. Lecture Notes in Computer Science. Springer, 2018, pp. 437–446. DOI: 10.1007/978-3-030-03769-7_26. URL: https://doi.org/10.1007/978-3-030-03769-7%5C_26.

[3] *LIG, Grenoble informatic laboratory.* URL: http://www.liglab.fr/?lang=fr.

[4] *My trace manager java class.* URL: https://github.com/Sauww/Cahier-de-labo/blob/master/Sources/TraceManager.java.

[5] *Source and documentation of the fertility dataset I used.* URL: https://archive.ics.uci.edu/ml/datasets/Fertility.

[6] *Two-class Boosted Decision Tree Azure explanation.* URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-boosted-decision-tree.