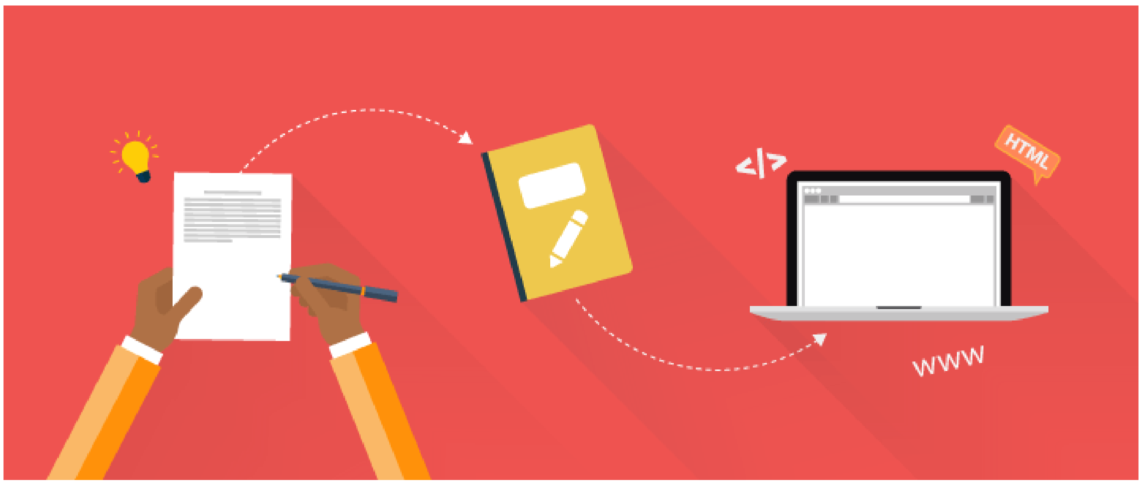
ENSAI 3A SID

Back2Back testing of Machine Learning Algorithms implementation as a service

Specifications of Web Application



Tutors	Students
Mathieu Acher	Laura Dupuis
Olivier Barais	Pierre Laffitte
Johann Bourcier	Flavien Levêque
Olivier Chantrel	Charlène Noé
Carla Schettini	

Introduction

Nowadays, there are many machine learning algorithms in statistics. These algorithms allow scientists to analyze and predict data from a given database. The goal of this project is to implement an easy-to-use tool for comparing different machine learning algorithms through back-to-back testing. Back-to-back testing, which checks the credibility of results obtained from indicators. This makes the comparison between several machine learning algorithms possible. With this comparison, according to the dataset provided, we can provide the best analysis or prediction.

For this project, we have looked at two machine learning algorithms : the classification tree and the random forest algorithms. This project is composed of two parts : the implementation of the performances of these algorithms with Java, and the design of a web application in order to be able to use the performance calculator.

Firstly, ahead of these specifications, we have compared the performance of these algorithms in three Java libraries : Weka, SparkML and Renjin. To this end, we have used three datasets, which are saved in our project. To evaluate the performance and the quality of each algorithm, we have implemented an "accuracy" indicator, calculated with the same dataset. In Concrete terms, the algorithm builds an "internal representation" by itself in order to be able to make a prediction without human interaction. In order to do so, the algorithm takes part of the data set (called learning sample), on which the model trains and improves ie it learns. Then, the other part of the dataset (called test sample), is used to apply the previously trained model and extend the predictions. It is with this test sample that the "accuracy" indicator is calculated. The more precise the prediction on the test sample is, the more reliable the algorithm will be. This is what we have done for the two algorithms used, implemented with the three Java libraries for each of the three datasets. Thus, to compare the performance of each algorithm, we can look at which library is the most efficient using several datasets.

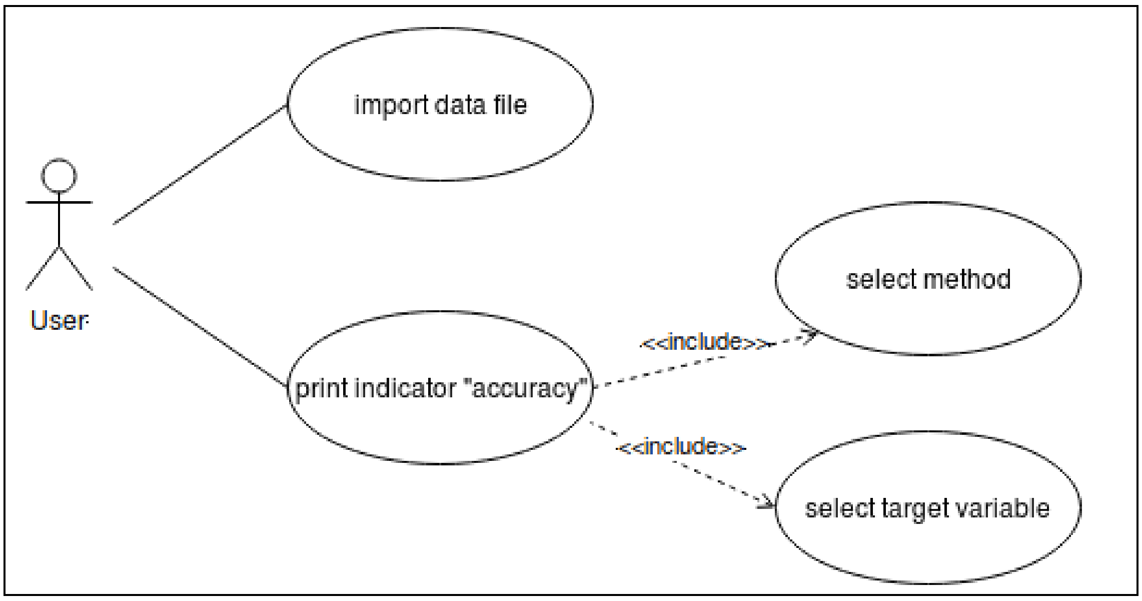
The second part of the project consists of offering this algorithm precision calculation tool to any user. For this to happen, we need to create an easy-to-use web application that compares these different machine learning algorithms. Depending on the dataset and the method (Random Forest or Classification Tree) that the user chooses, he will be informed of the accuracy he can expect under the three Java libraries. Hence, he can choose the best of them to make the predictions he wants. It is this part that we will develop in this specification. We will explain the needs of the user and what the main purposes of the application are. Furthermore, we will describe the implementation of the application and how it internally works.

Design of the application

This section describes the needs of the user and what the main purposes of the application are. The main goal of this product is to create an friendly user application that will employ some Machine Learning methods from different libraries of Java.

Features of the application

This use case diagram shows the two actions that users of the application can do.



Use case Diagram of Application

The first task imports a data file. This data file can be a file already saved in the application or a new file downloadable from a computer.

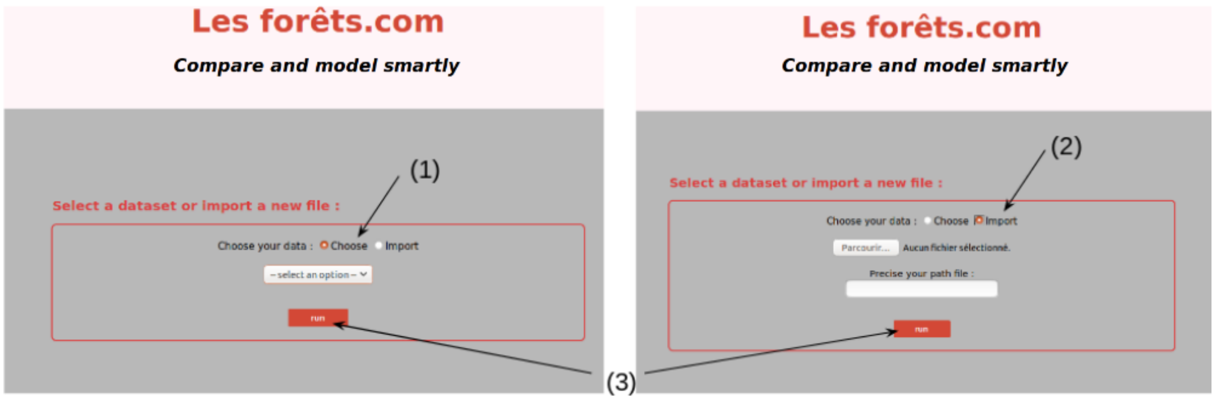
The second action is the display of the "accuracy" indicator. For this, you must first select an algorithm of machine learning (Random Forest or Classification Tree). It is also necessary to choose the variable to predict from the chosen data file.

These two actions can be performed by any user visiting the web application. This use case diagram may evolve if actions are taken in the application.

Graphic interface

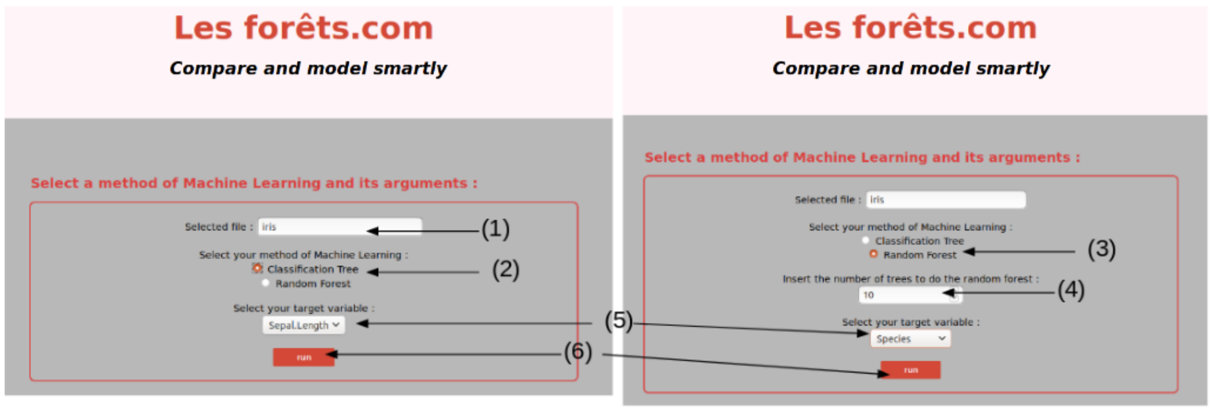
This part presents the graphical interface of the web application. The graphical interface is the representation that the user sees during his visit to the application. This is the part visible to users.

Above is a model of the graphical interface. It illustrates the different features explained previously. Each diagram represents a web page of the application. In total, there are three pages. A homepage to define the data file, a second page to choose the template and a third page to display the performance indicators of the three libraries.


Les forêts.com
Compare and model smartly
Select a dataset or import a new file :
Choose your data : Choose Import
-- select an option --
run
(1)
Les forêts.com
Compare and model smartly
Select a dataset or import a new file :
Choose your data : Choose Import
Parcourir... Aucun fichier sélectionné.
Precise your path file :
run
(2)
(3)

Homepage of the application

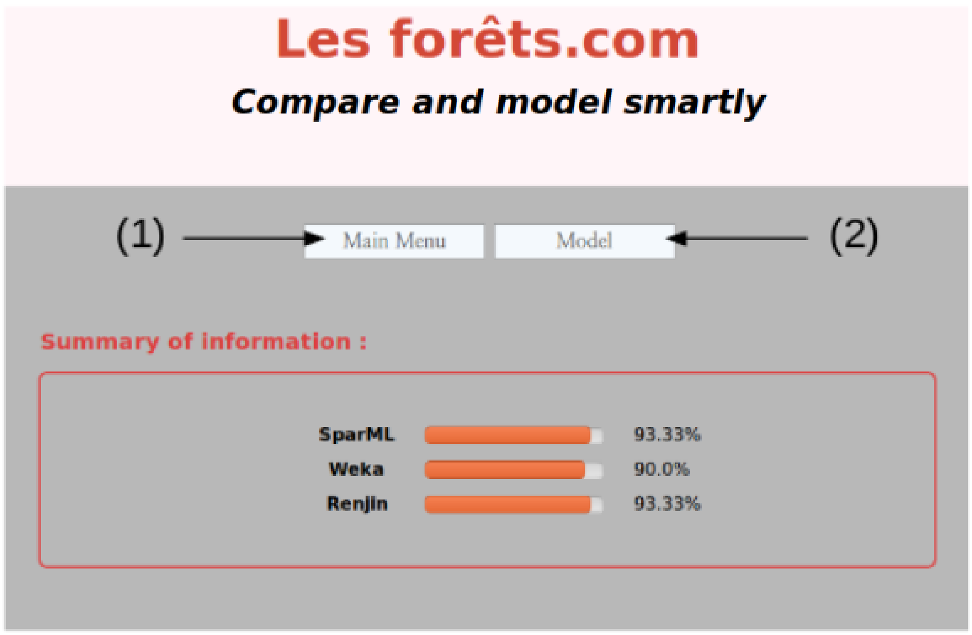
Above is the homepage of the application. This is a first page that the user sees. From the interface the performances are selected. As previously explained in the use case, the user can choose to select an already existing file (1), then click on "Choose" and select from the drop-down list the desired file. Otherwise, it is possible to choose a data file present to user's computer (2) by clicking on "Import". For that, it is necessary to specify the way to the file on the computer in the entry (the white rectangle). Finally, the user must click on "run" (3) to go to the next page.


Les forêts.com
Compare and model smartly
Select a method of Machine Learning and its arguments :
Selected file : iris
Select your method of Machine Learning :
Classification Tree
Random Forest
Select your target variable :
Sepal.Length
run
(1)
(2)
Les forêts.com
Compare and model smartly
Select a method of Machine Learning and its arguments :
Selected file : iris
Select your method of Machine Learning :
Classification Tree
Random Forest
Insert the number of trees to do the random forest :
10
Select your target variable :
Species
run
(3)
(4)
(5)
(6)

The second page of the application

Once the file is selected (e.g here the iris file), the second page appears. Then, a method of machine learning and its arguments are selected. The file name is specified in text (1) to be sure to work on the correct one. The user can then click on Classification Tree (2) or Random Forest (3). In this second case

(3), the user must specify the number of trees wanted in the algorithm (4). The user then must choose the target variable to predict, using the drop-down list. In this list, all variables of the selected file (e.g iris), are present. Finally, when all these parameters have been chosen, the user can click on "run" (6) to access the results page.



SparML		93.33%
Weka		90.0%
Renjin		93.33%

The third page of the application : summary of information

Then the performance of the three libraries (SparkML, Weka and Renjin) can be vizualized according to the selected algorithm and data set. On this page, the value of the accuracy indicator and progress bar for the three libraries are shown. This makes the comparison between several performances possible and helps the user to choose a library. The main menu button (1) is used to return to the homepage. The model button (2) returns to the previous page (corresponding to the second picture presented).

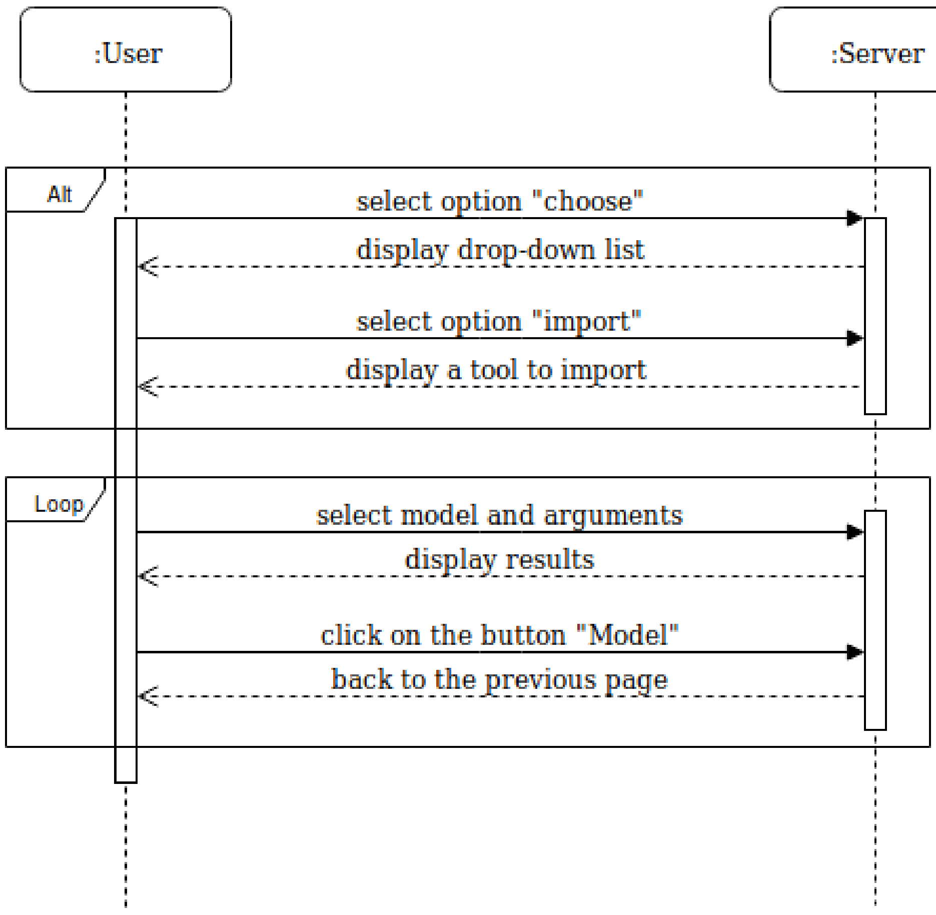
Implementation

This section provides a detail description of the application and how it internally works.

Communication between User and Server

This part consists of explaining the interaction between the user and the server.

To better understand the link between the user and the server, a sequence diagram is shown below. It shows the different interactions in the scenario of the use case diagram presented in the first part. The user is the main actor on the left and the system (the server) is represented on the right. This diagram illustrates how the user's actions take place and how they are reflected on the server over time.



Sequence Diagram : Interaction between the user and the server

At the beginning, the user selects an option to add to the dataset. As explained previously, the choice is possible between option "choose" or option "import". This is then sent to the server and the chosen option is displayed. The server is active only when called. After each request, the server becomes active. At the end of request, this server is down. This is the case when the model and arguments are selected. The server is then active and displays the results of the indicators. Then the user clicks on the button

"Model", so the server comes back to the previous page. And these steps are repeated until a new comparison is restarted.

Therefore, this diagram illustrates the interaction between the actor and the system. The next part details the structure of the application and the communication between the files.

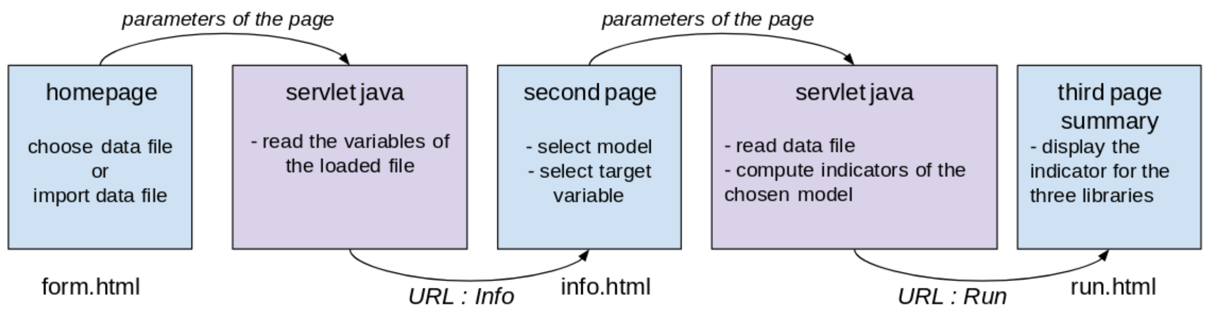
Structure of the application

To insert the first step of the project, the application must be implemented under JAVA.

Any visible part on the web application must be integrated in HTML and CSS. HTML is used to define the logical structure of the application. It allows forms to be created on pages. CSS is used to format the HTML page (e.g font size, color, etc). These two languages make it possible to manage the "static" part of the application.

The "dynamic" part is managed by the JAVA code using "servlet". A "servlet" is a Java class that allows the process of requests and the customization of responses. A "servlet" can receive a request sent by a browser and send back a response in return.

The application must be implemented as follows :



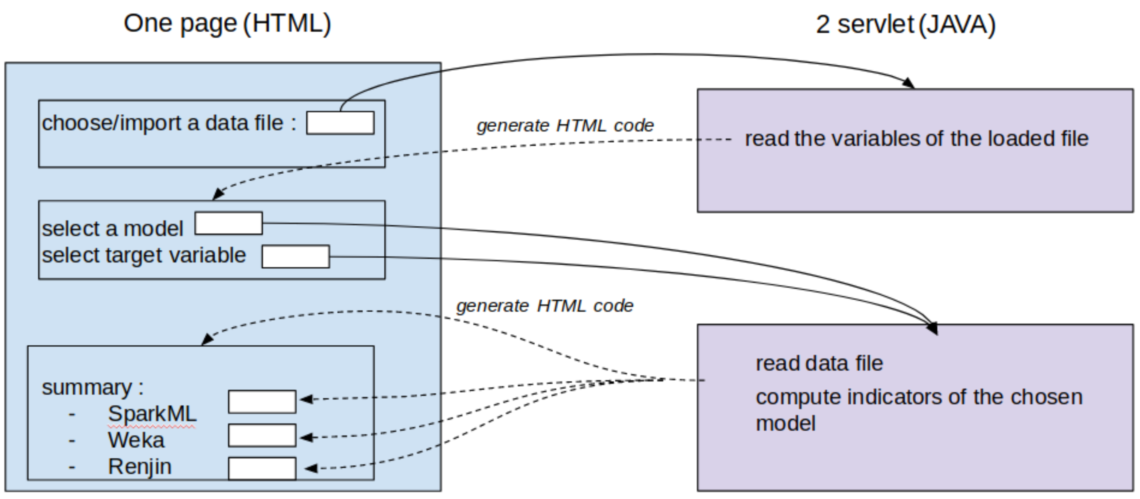
Links between files

The homepage is used to choose the data file on which the performance comparison has to be performed. A Java servlet retrieves this file and reads only the file header, which corresponds to the name of the variables. Then the servlet returns to a new URL : "Info". This one displays a second html page for selecting the model used and the target variable. The second Java servlet retrieves the given parameters and uses them to read the entire data file, selects the target variable and computes the three performance indicators according to the three libraries. The servlet returns a new "Run" URL that displays the html page displaying the three indicators. Therefore, the servlet returns HTML code in a precise URL.

Improvements

This specification covers the features that must be present in the application. However, they can be improved or expanded. Here are some ideas for improvements that can be exploited, if the working time allows for it.

First of all, it would be possible to implement the web application on the same page, known as "One-Page". This is a very popular phenomenon at the moment on the Internet. But this is more complicated, since it would manage the dynamic parts on the same page. For that, a single HTML file and a single Java servlet would be useful. Here is how these two code files would be implemented :



Links between the single HTML and a single Servlet

This diagram uses the same principles as the implementation part. When a parameter on the HTML page is filled, they are sent to the Java servlet. The servlet compiles and returns HTML code in the appropriate frame in "OnePage". Thus, it will no longer be necessary to manage the different pages and buttons. But this brings new technologies in Java code : Ajax and JQuery. It allows to manage the "dynamic" part of the site on the same page.

Another possible improvement would be to add a new action. The application could support new machine learning algorithms. To compare the performances of the libraries using to a new algorithm, the user will be able to implement the algorithm in Java and add some code to the existent one. It would therefore be possible to create a new action on the application, allowing the addition of an algorithm not yet implemented.

And finally, one last possible improvement would be to use a database linked to the application. This would make it possible to store the history of the comparisons made, at each calculation of the application. When a user configures the settings and displays the results, these one are added to the database. Thus, it would be possible to retrieve information about the performances already saved.