Machine Learning Documentation

Cycles Length, Menses Length and ovulation Day Model

Finding relationship between data using NN on the FedCycle dataset:

Methodology on cleaning the dataset

When it comes to the machine learning part, the first challenging task was to find a suitable dataset for our project. The task wasn't easy and even to this day, we do not have the perfect dataset that we were thinking of when choosing our topic. However, we found a publicly available dataset which contains the following elements :

Essentially, we have data about different women's (one ID per individual) menstrual cycle.

We could only use the columns that we could ask to the users of our services or calculate/predict.

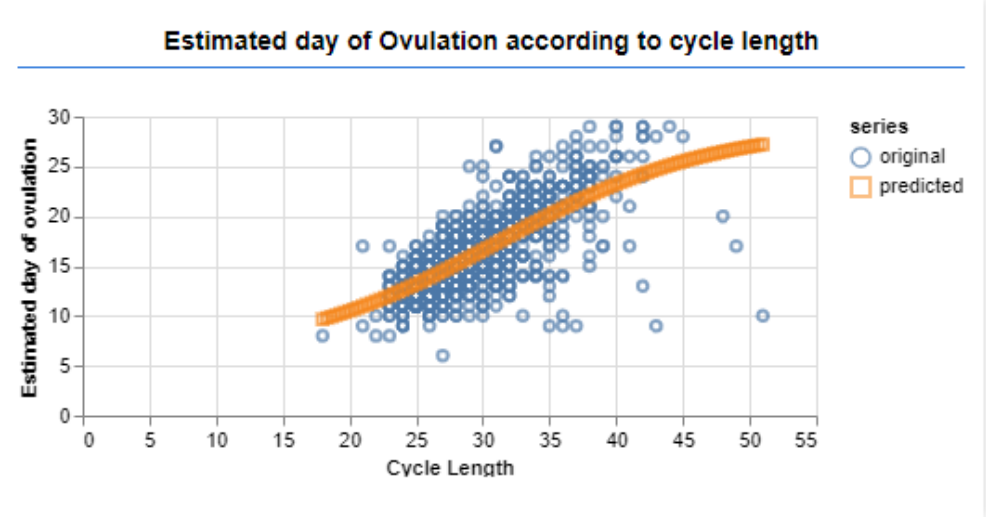
Therefore, we choose to only keep: ClientID, CycleNumber, LengthofCycle, MeanCycleLength, EstimatedDayofOvulation, LengthofMenses, MeanMensesLength, Age, NumberPregnancies and BMI.

To clean and modify the dataset, we used python on a Jupyter Notebook. The dataset contained rows with empty columns (missing values) and needed to be cleaned before we could use it for machine learning purposes.

Methodology on predicting using NN:

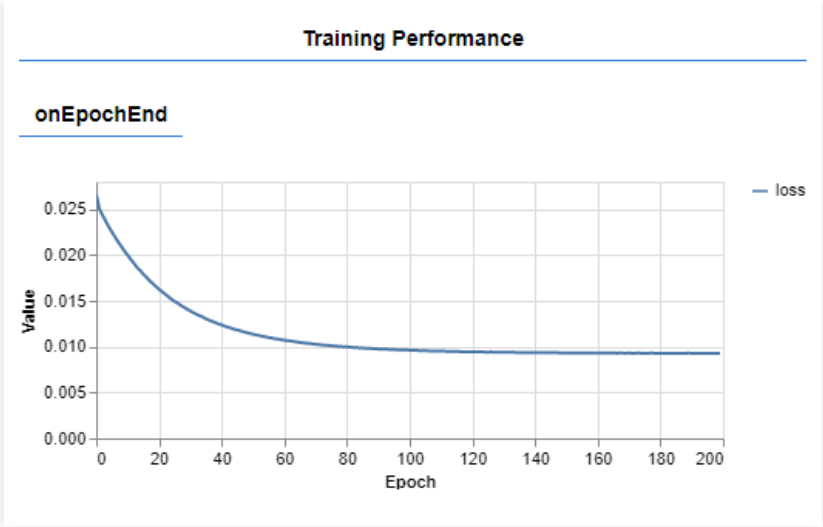
First off, we learned how to use tensorflow.js and tried to realize a simple non linear regression model to predict the estimated day of ovulation of a cycle according to its length as a first practicing example. We started by plotting the relationship between the 2 and the predicted values of the model.

Evaluation and Analysis :



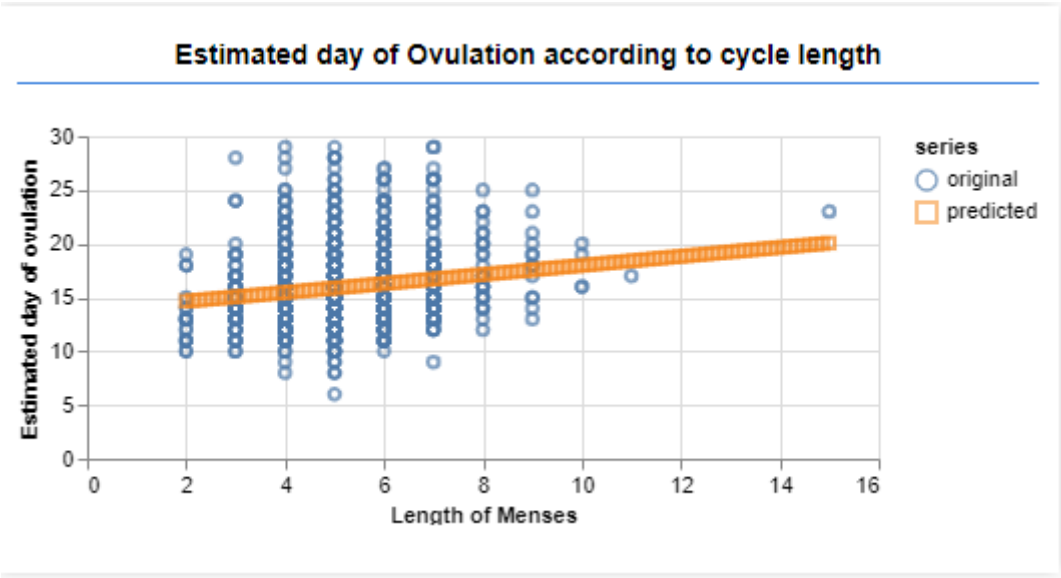
Evaluation and Analysis :

As we can see, it seems that the estimated day of ovulation is linked to the ovulation day. It looks like the ovulation day increases when the cycle length increases. The relationship between those 2 elements seems to be relevant.



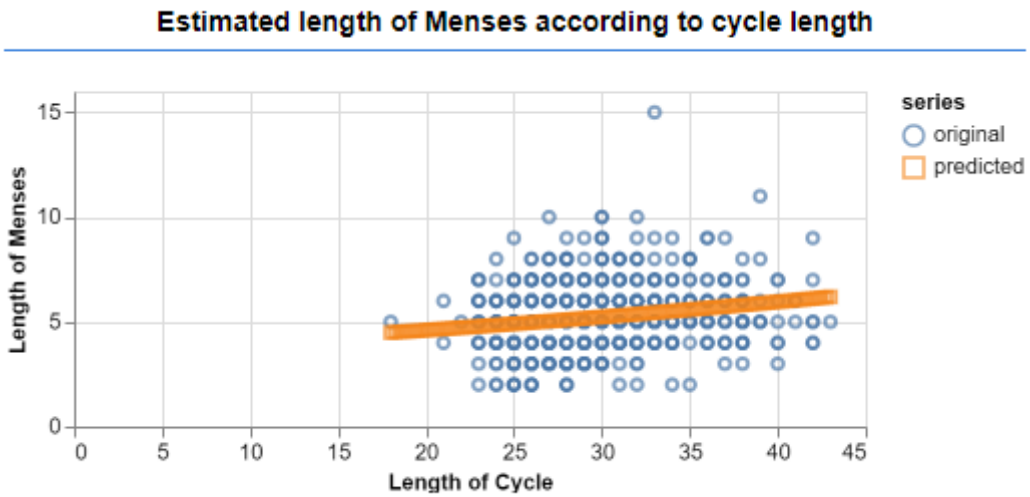
The loss function on this model shows us a significant decrease of the loss throughout the training an a final value of 0.0092 during training, and 0.0096 during testing. For example, with a cycle length of 28 the model predicts the ovulation day to be on the 15th day which matches what we can see on the graph generated previously.

Then we tried to do the same with other elements of the dataset to find a relationship between them. We first tried to predict the ovulation day according to the length of menses.

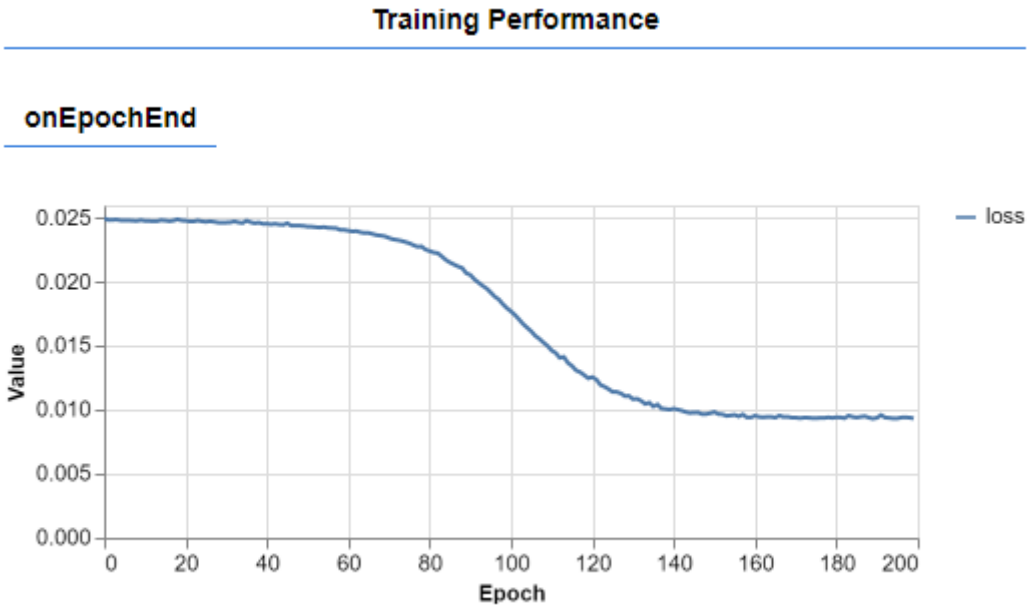


As we can see, we cannot observe a clear relationship between the 2. It looks like the ovulation day seems to be sooner with the lower values of the length of Menses. However, the range of value for the ovulation day is very broad for every data points of the Length of Menses variable.

We also tried to predict the length of Menses of a cycle based on its length but couldn't observe a clear relationship between the 2.



We obtained similar results when trying to use BMI, Age and number of pregnancies one at a time to predict the ovulation day Moreover, when using LengthofCycle, LengthOfMenses, BMI, Age and NumberOfPregnancies to predict the day of ovulation, we obtained results similar to when we only used the Length of Cycle.



As we can see on the graph, at the end of training we had a loss of 0.0093 and also a loss of 0.0093 on testing. This doesn't look like an improvement compared to the loss of 0.0092 obtained before. We can see that the loss value on the testing set was slightly lower and closed to the loss obtained on the training set when using those 5 features, but the difference too small to not be significant enough and draw conclusions.

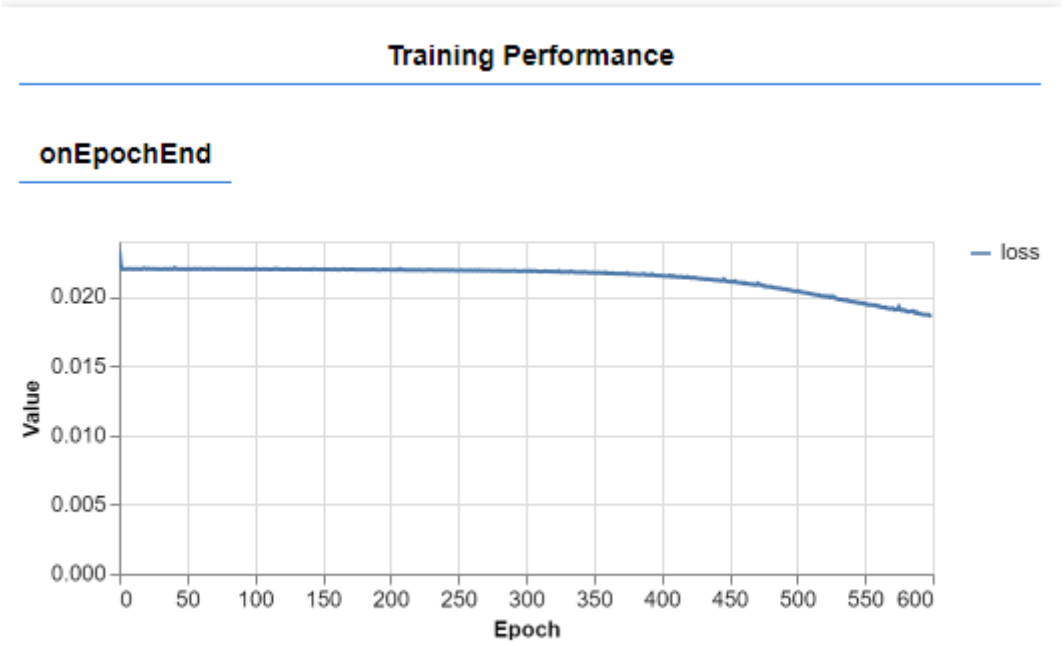
At the end of this "training" and features research using a non linear regression model in order to learn how to clean a dataset, import it, create an AI model and train it on the dataset, we had to move to other AI models and use the dataset differently for multiple reasons.

First off, our goal isn't to wait for the end of a cycle to predict when was the ovulation day that already happened, that would be pointless as the user needs to know a prediction for the next cycle, and not a cycle that already ended.

Our AI models need to predict the length of the next cycle, length of Menses and the estimated day of ovulation based on the data obtained about previous cycles.

Previous research papers that we read used the body basal temperature at their main feature to predict the day of ovulation, as it gives a clear and precise indication as when it happened. However, this is not something we can obtain with our service. The BBT (body basal temperature) needs to be measured daily, when waking up, with a special kind of thermometer that most people do not own. But what we could learn from this research paper is how they used CNN (Convolutional Neural Network) models and LSTM (Long Short Term Memory model, a type of Recurrent Neural Network or RNN) models to predict the day of ovulation and how effective those model were, especially the LSTM model.

But before moving on to an LSTM or CNN model, we tried to predict the ovulation day using the mean cycle length and the length of menses of the current cycle. As the ovulation day happens a few days after the menses, we could use the mean value of previous cycles and the length of menses of the current cycle to try to predict the ovulation day. Sadly this method did not provide really good results. This could be explained by the fact that only the length of Menses varies from cycle to cycle of an individual, same as the ovulation day. However we've seen that the ovulation day does not show a clear relationship with the length of Menses but more with the length of Cycle, which here does not change between cycles of a same individual because we are using the same mean value every time.



We only managed to obtain a loss of 0.187 on the training set and 0.2 on the testing set, but after a huge number of epochs compared to what we had for previous model. Which could mean that we only overfitted the data and do not necessarily have a working model.

Time series prediction on Cycle length, Menses length and Ovulation day :

Methodology :

We began working on an LSTM model, but this attempt was unsuccessful as we encountered errors which we couldn't solve. Tensorflow.js community is not as big as TensorFlow on python, and less tutorials or examples are available. Which is why we couldn't find enough documentation about the encountered errors in order to fix them. The error was about the dimension of the input, saying that it shouldn't be of dimension 3 instead of 2. However, as we printed our input tensor in the console it was indeed a 3d tensor. We couldn't fix this specific error or understand why it happened.

For that reason, we decided to work on another model that gave good really on different projects about time series: 1D CNN. In a 1D CNN, the kernel of the convolutional layer moves in 1 direction. And even if we have to transform our input tensor into a 3d tensor for the model to work, it is basically a 1-dimensional array of features. When working with time series prediction, the goal being to predict the next value using previous values as input, the label used is the next value following the features value in the dataset. For example if we have to predict the length of the next cycle of a women using n=44 values of previous cycles, we would choose a number of previous cycles as the features (lets take 4 as an example) and the next value after each group of features as a label. Thus, transforming the dataset using 2 for loop in order to have a number of samples = n-features = 40, composed of 4 features and a label for each.

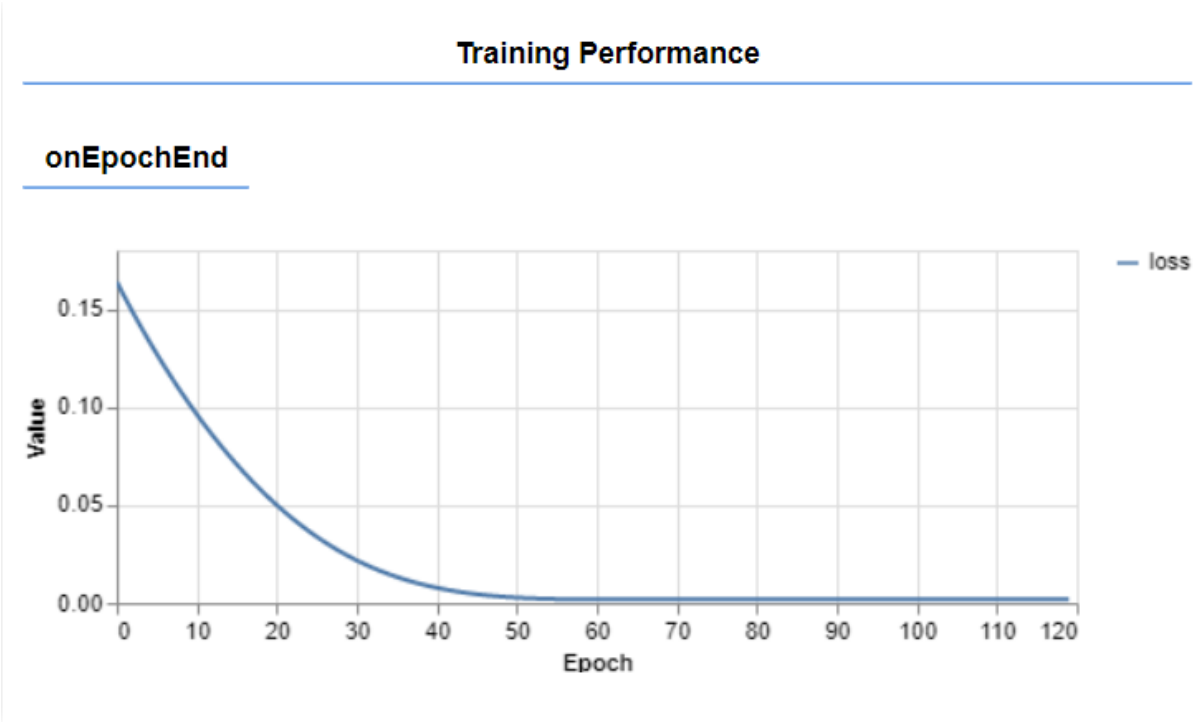
27
40
27
40
28
40
27
39

Cycle1	Cycle2	Cycle3	Cycle4	NextCycle
27	40	27	40	28
40	27	40	28	40
27	40	28	40	27
40	28	40	27	39

How we would modify a dataset of 8 cycles to use it with our model, Cycle 1 to 4 are features, and NextCycle is the label

Lacking a dataset made of irregular cycles, we made an artificial dataset to make it work with an LSTM model or a CNN. Then, training the model using the data of a user of our own made dataset, we achieve the following loss value (our model using mean squared error as a loss function).

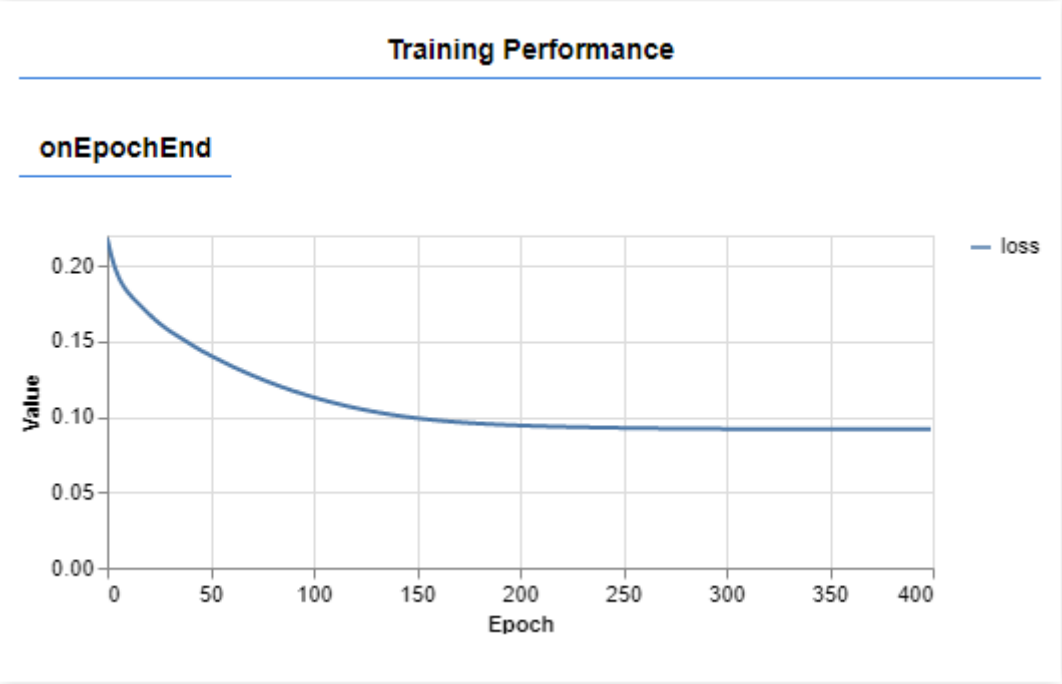
Evaluation and Analysis :



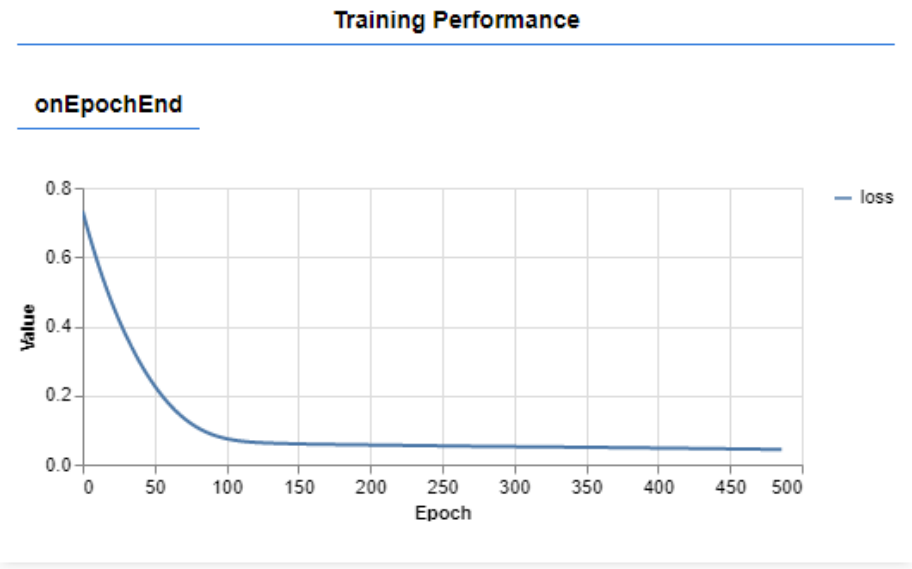
After training, the loss value is equal to 0,0018 on average. Giving us excellent results when predicting a value.

For example, using [28,40,27,40] as an input when predicting, the output is equal to 27, which is equal or very similar to the value found in our dataset. Same when using [40,27,40,27], the output is equal to 40, following the pattern found for this particular user.

Using the same approach and the same type of model to predict the length of Menses and ovulation day we achieve different results. For the length of Menses, the loss value obtaining is not as promessing.



But when it comes to the ovulation day, we achieve a good loss value. This can be explained by the link between the ovulation day and the length of cycles that we illustrated before when working on a regular Neural Network. As the ovulation day value changes according to the length of cycles, we could expect that if the model performed on length of cycles, it could perform on ovulation day too.



The loss value is not as low as when predicting the cycles, but we still have a loss value equal to 0.04

Machine Learning

Pill Prediction Model

Methodology:

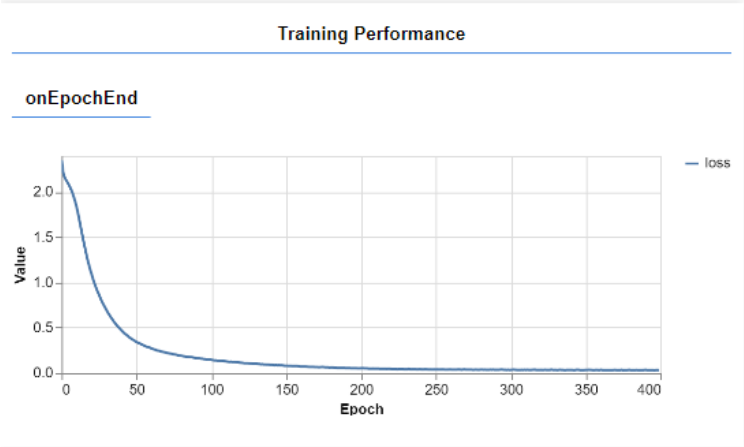
There is 12 different type of pills to predict using 14 features, therefore we can use a multi class classification algorithm. This type of AI model is used when your output is a class, and not a number. Compared to binary classification we won't be using a single binary output but 12 outputs. The number of layers, the number of neurons per layer and their activation functions were determined by trial and error, trying different settings in order to achieve the best loss value and the quickest. The last layer must be using softmax for the activation as it output a probability value between 0 and 1 for each output here being the type of pill. For now, the best results we had were using : a first dense layer of 28 neurons, with bias and a sigmoid activation function, 2nd dense layer of 56 neurons, sigmoid activation and bias and then a last dense layer of 12 output neurons, bias and softmax activation. For a multiclass classification problem, we are using a categorical Cross Entropy loss function and adam as an optimizer.

One thing to mention is One-hot Encoding, which is mapping your categorical variables names or value with n values, from 0 to n-1. Indeed the output layer is composed of n neurons and the output of the model is an array of n features, with values between 0 and 1 and index from 0 to n-1. Therefore we have to remap our label array for training from values between 1 and 12 (the type of pills) to values from 0 to 11. Then when predicting a value we have to achieve the opposite and map the output value of the model to the possible value of the categorical labels. To determine the final output (type of pill to take, from 1 to 12) we observe which value of the output array of the model is the highest, but this label will be between 0 and 11, not 1 and 12. Therefore we have to remap our output to values matching the categorical label (here being values between 1 and 12, the type of pills).

We coded the model using Tensorflow.js on JavaScript, opening the csv dataset from either a link or a local file, then separating the dataset in a 70/30 ratio between training and testing. We do not need to normalize the value of the features and labels as the labels are categorical and the values of the features are all between 0 and 1. Then we can train the model using the training set, then testing it and displaying the loss value of both when testing it. For testing, we executed our javascript script in an html to be able to generate and display real time graph during training in our browser. We used the tfjs-vis library to display the graphs found below.

Evaluation and Analysis:

With the model mentioned before, we achieved a loss value around 0,03 on both training and testing using 400 epochs and a batch size of 32.



Work Remaining on both prediction models :

The remaining task to perform for both pill prediction and cycles, menses and ovulation prediction is to access user's data on our database. For the pill prediction we don't train the model user by user as the output won't change according to users, but their symptoms. Which is why we can train the pill prediction model only once using the pill dataset found online. We then can save the model and simply load it when needing to predict which pill to user according to any user's symptoms saved on our SQL database.

For the cycles, menses and ovulation prediction we need to access user's data on our SQL database to train the model every time new data is entered for the 3 variables, in order to retrain a new model and predict values for the next month.

Related work:

To learn how to create such a model but also how to use TensorFlow.js in general, we followed an online course on Udemy called *Machine Learning in JavaScript with TensorFlow.js.*

https://www.udemy.com/course/machine-learning-in-javascript-with-tensorflow-js/

Non linear regression model using 1 feature :

http://www.mediafire.com/file/g4xsgjklt8a0ovn/regression1feature.html/file

Non linear regression model using multiples features (here using 5 features):

http://www.mediafire.com/file/3vka6tbdgzb852d/tensorflow5features.html/file

Non linear regression model using mean length of Cycle, length of menses to predict ovulation:

http://www.mediafire.com/file/ayafw3b1mymrdfx/regressionusingmeancyclelength.html/file

Original Dataset :

http://www.mediafire.com/file/a1nrr7yluwqtgcg/FedCycleData.csv/file

Jupyter Notebook File for the Dataset cleaning :

http://www.mediafire.com/file/e4xdf3af0k0ct6e/AI_PROJECT.ipynb/file