Challenge Health Notification\_4

Version 7 (03/12/2019)

Carlos Barranquero ([barranquerodiez@gmail.com](mailto:barranquerodiez@gmail.com))

Contents

[**1.** **Introduction** 1](#_Toc51348355)

[**2.** **Data analysis** 2](#_Toc51348356)

[Regression Model 2](#_Toc51348357)

[Classification Model 4](#_Toc51348358)

[**3.** **Neural Networks** 5](#_Toc51348359)

[3.3. Network architecture approximation 5](#_Toc51348360)

[Network architecture classification 6](#_Toc51348361)

[**1.** **Instructions:** 7](#_Toc51348362)

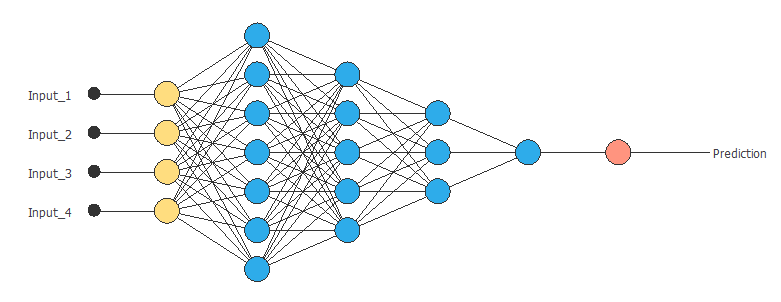
# **Introduction**

The aim of this project is predicted number of submissions in a specific challenge.

Here a method from machine learning, called neural networks, is used. Basically, a neural network is a mathematical expression that models a process from empirical observations of it.

Neural networks have input connections through which they receive input values, and a set of internal parameters (biases and synaptic weights) which interact with the input signals to provide an output value. The neural network parameters are adjusted to provide an accurate response giving a specific input value.

The following picture shows a neural network diagram.

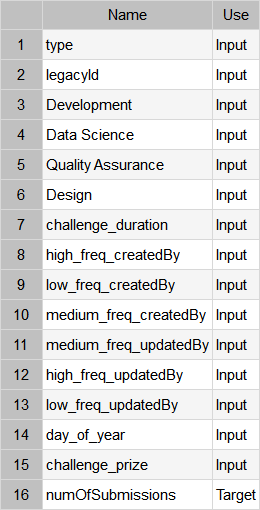
 Neural network diagram.

This project consists of different parts: On the one hand the data acquisition, and the main program is made in python. On the other hand, the neural network training process has been done with OpenNN library written in C ++ (It is not attached but if you need it I will do), once the regression and classification models have been trained in C ++ they have been exported to python where they are used.

# **Data analysis**

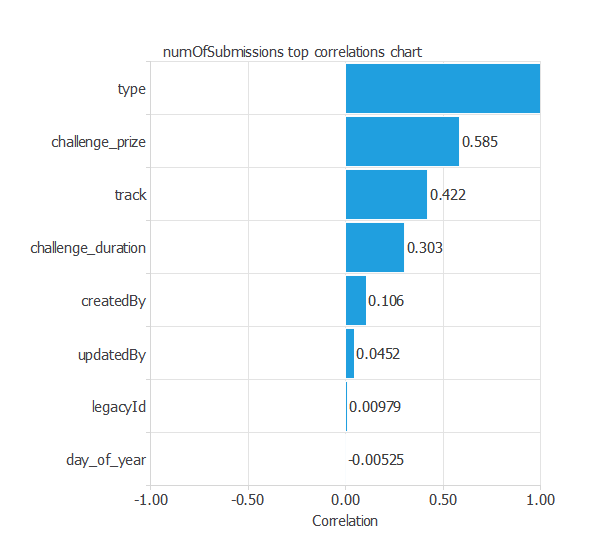
The feature selection process has been done with OpenNN too, I attach some results obtained:

## Regression Model



I have clustered the createdBy and updatedBy variables into 3 groups (high frequency, medium freq, low freq).

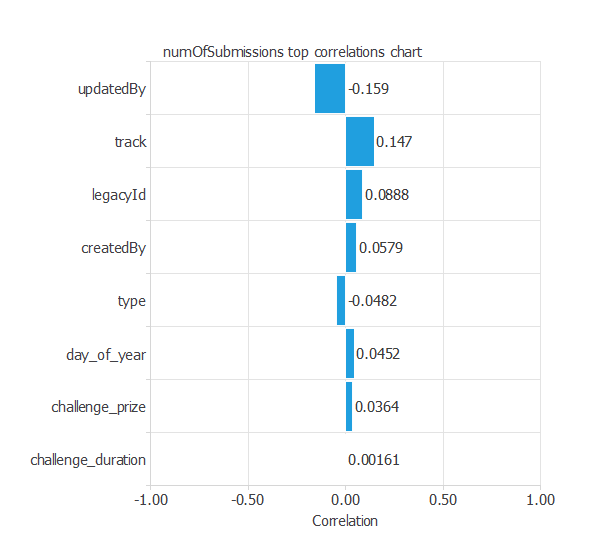
Feature selection has been done with target correlations:



Variables for approximation model:

['type', 'track', 'challenge\_duration', 'createdBy', 'challenge\_prize', 'numOfSubmissions']

## Classification Model



['type', 'track', duration', 'createdBy', legencyId','challenge\_prize', 'numOfSubmissions']

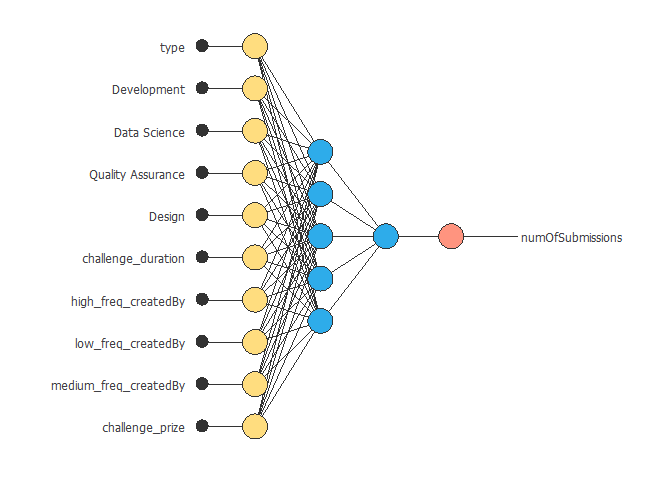
# **Neural Networks**

Neural networks have universal approximation properties, which means that they can approximate any function in any dimension and up to a desired degree of accuracy.

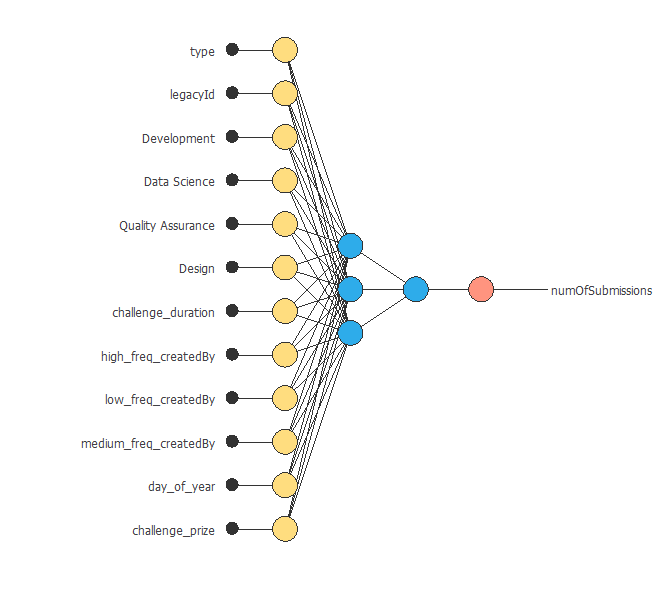
## 3.3. Network architecture approximation

A neural network is built up by organizing layers of neurons in a network architecture.

A graphical representation of the resulted deep architecture is depicted next. It contains a scaling layer, a neural network and an unscaling layer. The yellow circles represent scaling neurons, the blue circles perceptron neurons and the red circles unscaling neurons. The number of inputs is 10, and the number of outputs is 1. The complexity, represented by the numbers of hidden neurons, is 5.



## Network architecture classification



The quasi-Newton method is used here as optimization algorithm. It is based on Newton's method, but does not require calculation of second derivatives. Instead, the quasi-Newton method computes an approximation of the inverse Hessian at each iteration of the algorithm, by only using gradient information.

For Approximation I used Normalize square and for classification I used weighted square error as LossIndex.

# **Instructions:**

Place new json in \data\test\json and execute main.py results will appear in doc folder.