The Quality Prediction :

Concept :

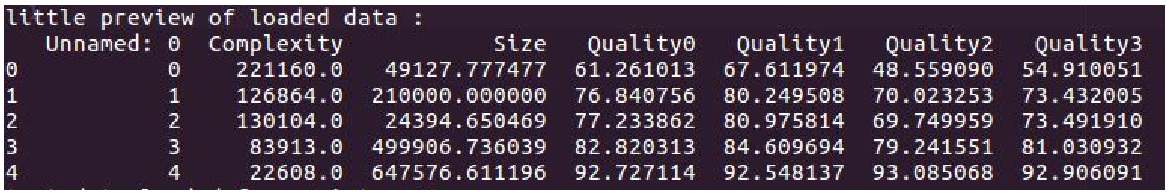
The whole concept of quality prediction (regression in our case) stemmed from a simple yet annoying problem : once an object is printed by one of the printers of the factory, it has already been printed with a certain quality percentage that will be later determined by the inspection stations; we wish however to be able to somehow predict that quality on each of the available printers in order to optimize a cost function (through the use of the GA algorithm) that decreases with the quality and increases with the scheduling makespan.

Since each printer has its own ontology (parameters that play a role in the quality of printing, see the quality function in the “Inspection station” section) and each object has its own properties (Complexity and Size), the quality that is determined by the aforementioned inspection stations must be a function of the complexity and size of the printed object as well as the ontology of the printer that printed the object itself.

This is where Deep Neural Networks (DNNs) come in very handy. They are considered to be universal function approximators, and even if they are sometimes considered a little overkill for prediction purposes (other simpler architectures like SVMs or logistic regression can be used) they have been shown to outperform most of the regression methods for most regression tasks.

The quality prediction is achieved through the use of a Feed Forward Deep Neural Network (DNN) consisting of 2 layers of 15 units (neurons) each.

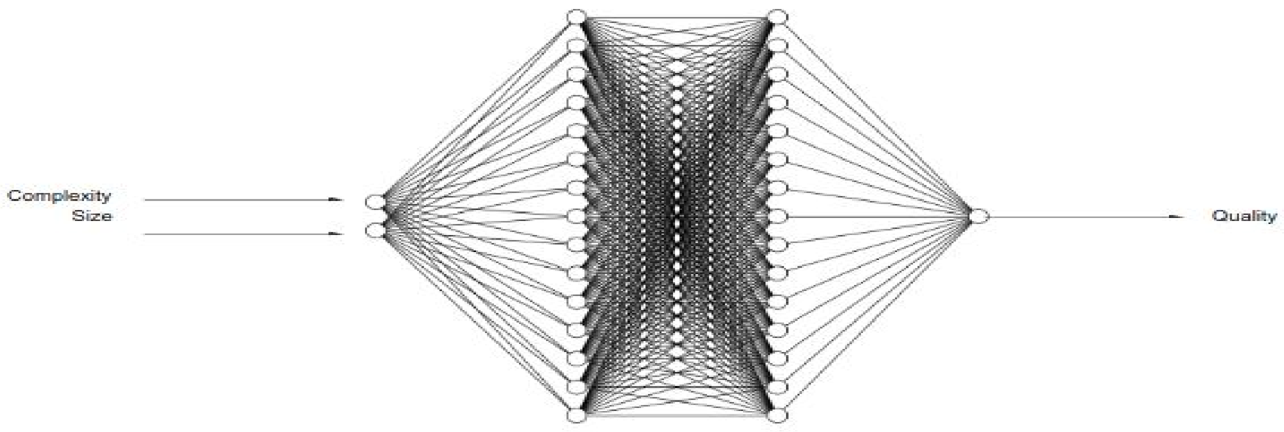
An overview of the Data that our DNN will train and be tested on is useful to understand how and why the architecture of the Quality Prediction is the way that it is :



```
little preview of loaded data :
   Unnamed: 0  Complexity           Size   Quality0   Quality1   Quality2   Quality3
0           0    221160.0   49127.777477  61.261013  67.611974  48.559090  54.910051
1           1    126864.0  210000.000000  76.840756  80.249508  70.023253  73.432005
2           2    130104.0   24394.650469  77.233862  80.975814  69.749959  73.491910
3           3     83913.0  499906.736039  82.820313  84.609694  79.241551  81.030932
4           4     22608.0  647576.611196  92.727114  92.548137  93.085068  92.906091
```

The DNN’s architecture :

Since each printer has its own ontology, each printer will have a different quality prediction even though the same object (Complexity and Size) is considered. In other words, the quality function that the DNN learns to approximate will be different from printer to printer. Since 4 printers are considered, this required us to build and train 4 different DNNs each with the following architecture :



Training/Testing procedure :

Results :

Technology :

Since writing from scratch the code for a deep neural network would have been to tedious, the API KERAS with the framework THEANO running as a backend was used in order to build and customize our network in a very modular way.
In order to split the data and pre-process it, PANDAS and SCIKITLEARN have been used.
Last but not least, NUMPY for vector and matrix representations and MATPLOTLIB for graphical representations (The DNN’s architecture above was generated using matplotlib).