Tomás Coheur – 93621
Tomás Oficial – 101259

Inteligent Computing for the Internet of Things

Report: Project 1

Intro:

Before starting the project, a good chunk of time was dedicated to looking at the data on excel to understand what we were working with. From the get-go the column “Persons” looked like the output for our neural network. Another thing that caught my attention immediately was that there were some empty cells that needed to be fixed.

After that we started thinking about the columns and which ones would or could be necessary for our neural network. The column “Date” did not seem to be very important because it only had three different days and unless one of the days was a holiday or weekend it should not matter to our neural network. The other column that was interesting was “Time”. This one we felt like it was relative if we needed it or not. Obviously the later we are in the day the less people are going to be in the room which is a good feature for our neural network to use. However, the sensors are also kind of dependent of the time of the day; night is colder or there is less light. So, we could consider the column “Time” like a repeat of the information given by the sensors, or you can use it as an extra factor to regulate the data in case a sensor is broken. The latter was the idea we went with and now, with most of the ideas in place we could finally start coding.

Cleaning of the Data: (code in file dataprocessing.py)

First, we created a new data frame called “df”, deleted the column “Date” and transformed the columned “Time” to integers to be able to process the data when creating the model. Then we started cleaning the data frame starting with filling empty cells with the mean of the column values to avoid errors. There is no special reason to why we used the mean function instead of median, we think it goes back to being relatively the same. After that we had to remove outliers and from a quick glance at the excel there were not that many values that stood out. So, using the quartiles we created a formula because the value being higher or lower than 1.5 times the inter-quartile range was eliminating too many values. At the end of this only four outliers were removed. Finally, we normalized our data set using min-max normalization because, even though it does not handle outliers well, they were already dealt with so using a more complex algorithm like z-score was not necessary.

Creation of the Model: (code in file modelcreator.py)

Having the data cleaned and ready to go it was now time to create or neural network. As we said before the target of our model was without a doubt the “Persons” column so every other column would be part of the inputs. After that we had to create the training, validation and test sets and for this we first divided our data and target into a training set and a temporary set which would then be divided into the validation and testing set. There is no particular reason for the percentage of each set, we used one of the more common divisions.

Now we arrive to the part which needs a little more flair. First, we need to train the model and for that we fit the data and output allocated for training in a new classifier with a random number for the hidden layer size as it is not yet possible to know how many neurons we will need. After that we create a visualizer to be able to score our results using the validation set we created previously. By changing little by little the hidden layer size we try to get the best values possible for our prediction, recall and f1. And by fiddling with these values for a little more than 30 minutes we arrived at the conclusion that the best we could come up with is 4 neurons for the first hidden layer and 2 for the second one. These values were also influenced by an error that would say that we had problems of convergence.

Being almost certain that a size of (4,2) was good enough we decided to finally use the final test and compare the values we got previously with the ones using the test set instead of the validation set. For our relief the values were matching, and the test set values were slightly higher than the validation ones which meant we did not have overfitting. Our model was now good to be used with other data sets.

Unfortunately, due to some time issues it was not possible to respond to the binary question of if there is more than two people in the room. This would be done by creating a binary target with 1 if there were more than 2 people in the room and 0 if there weren’t.

How to run TestMe:

The main.py file creates the model we are going to use on the TestMe file first and only after that calls the function run_test() that asks for the name of the new data set file and then uses the model to test this new data frame. So, to run this little bit of code you have to compile the code and insert the name of the file and it should do the rest.