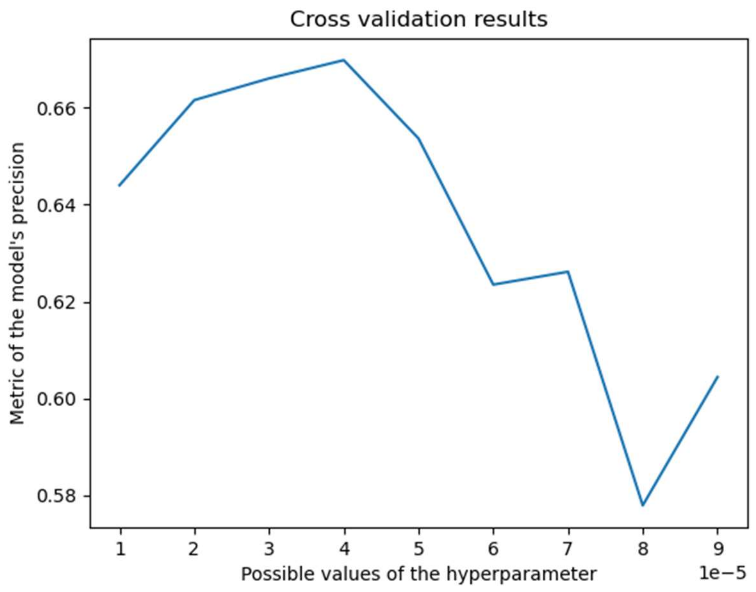
LOGISTIC REGRESSION

We perform gradient descent on the loss function of randomly generated weights to find the weights that generate a minimum of loss. To do this, we have multiple parameters: max_iters which is the maximum number of iterations during the gradient descent, and the learning rate which is the rate of descent. We found that 100 was a good value for max_iters, as a value too big would lead to a sub-optimal runtime and not a notable improvement in results. For the learning rate, we initially tested with a huge range of different values like for example the powers of 10 from -5 to 5 to see which scale of values it should take. We then progressively narrowed the possible values to a little range:

[10^{-5}*i for i in range(1,10)]. Afterwards, we use k-fold cross validation (k=4) on these possible values to find the best value.

For logistic regression, the best value generates the biggest value of the metric which is the macro F1 score. We can see on the following graph that lr=$4 * 10^{-5}$ is the most optimal value.



<u>Score without cross-validation(lr=1e-5,max_iters=100):</u> accuracy=78.80710659898476, macro F1 score=0.6548834675847545

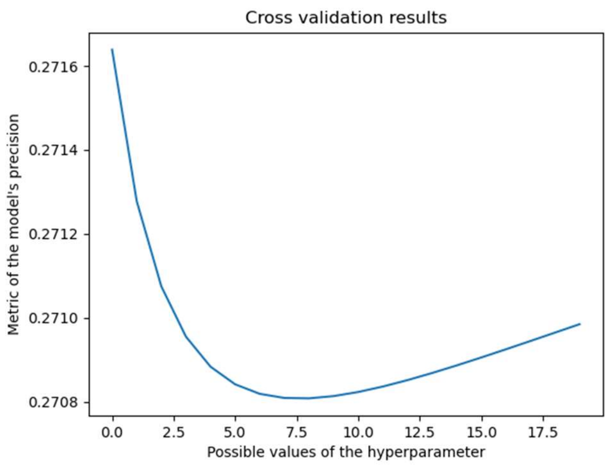
<u>Score with cross validation(lr=4e-5,max_iters=100)</u>:accuracy=79.69543147208121,macro F1 score=0.7168428180741989

<u>Conclusion:</u> Using cross validation, we get a significant improvement in the macro F1 score of about 0.06 which represents around 8% in relative value. Concerning the accuracy we get an improvement of about 0.89 which represents around 1% in relative value. These improvements don’t guarantee a notable difference between the two methods because these results are also due to a certain degree of randomness, but chances are that this follows the general trend and that cross validation is more precise.

LINEAR AND RIDGE REGRESSION

Linear and Ridge regression are very similar methods, where we obtain a closed form for the weights of our linear model, setting the gradient of the loss function to 0 and solving the equation. The only difference is a slight difference in the equation where the lambda parameter of ridge regression intervenes. Linear regression is basically ridge regression with lambda=0. To find the optimal value of lambda, we at first tested with a very large range and narrowed it down to the interval of integers between 1 and 20. Afterwards, we once again use k-fold cross validation (k=4) to obtain the best value from this range.

For linear/ridge regression, the best value achieves the smallest value of the metric: the mean squared error. We can see on the following graph that lambda=8 is the best value.



Score without cross validation(lambda=1): We obtain a final loss of 0.45681751676468

Score with cross validation(lambda=8): We obtain a final loss of 0.4542817400007059 (with an obtained metric of 0.2708087214529691 during the validation step for lambda=8).

Score without ridge regression(linear regression): We obtain a final loss of 0.45761932660137167.

Conclusion: We can observe that default ridge regression improves the loss result by approximately 0.1% (0.0008 absolute difference) whereas cross-validated improves the loss result by approximately 0.7% (0.003 absolute difference) compared to regular linear regression.

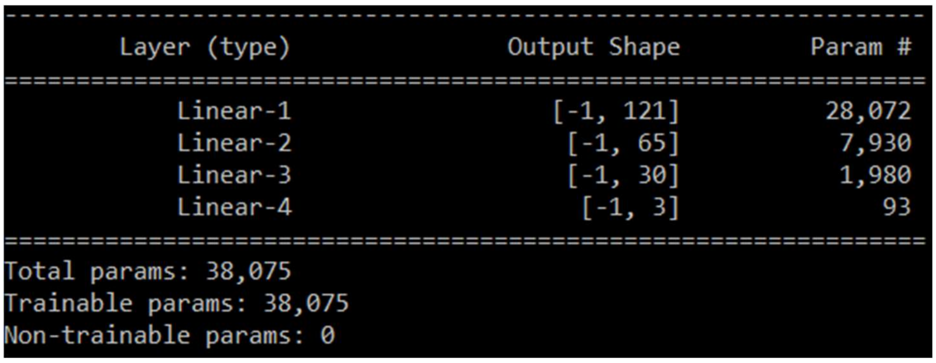
FINAL RESULTS

	Linear Regression	Ridge Regression	Logistic Regression
CROSS VALID.	/	≈0.4542 MSE	≈ 79.7% acc, ≈0.72 F1 score
NO CROSS VALID.	≈0.4576 MSE	≈0.4568 MSE	≈78.8% acc ≈0.65 F1 score

NEURAL NETWORK

Architecture of the model

For the implementation of the neural network, we chose to not use some convolutional layers, who will not improve much the results (low number of features + possible relation between N°1 feature and N°230 feature that could not be taking in account using convolution). Hence, we constructed a classical MLP with ReLU activation function.
We went for 4 hidden layers (the first three activated with ReLU and the last one not activated). For choosing the number of neurons for each layer, we simply took the mean between the input number of features and the numbers of classes.
Using this simple model, we got some convincing scores (c.f *results*). We hence chose to keep this architecture.
The final architecture:



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                 [-1, 121]          28,072
            Linear-2                  [-1, 65]           7,930
            Linear-3                  [-1, 30]           1,980
            Linear-4                   [-1, 3]              93
================================================================
Total params: 38,075
Trainable params: 38,075
Non-trainable params: 0
```

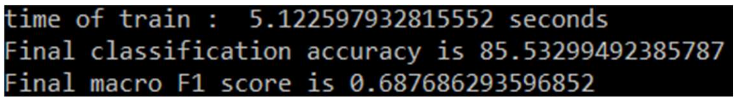
Batch size, learning rate, Epochs

To speed up the training process, we chose to take a batch size of *128 samples*, we know that increasing the batch size reduces the accuracy, so to compensate this loss we designed this strategy:

Start with a high learning rate, then each two epochs reduce this learning rate.

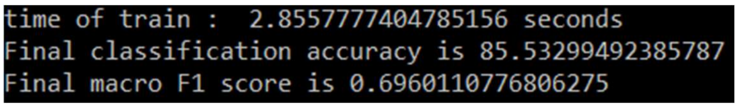
Indeed, we found that increasing the learning rate increase the accuracy when using big batch size. The objective of this strategy is then to increase the learning rate by keeping the advantage of a small learning rate (higher chances of converging to a good solution).
Using this process, we got those results:

Batch size = 32 ; learning rate = 0.01 ; epochs = 10 ; without strategy



```
time of train :  5.122597932815552 seconds
Final classification accuracy is 85.53299492385787
Final macro F1 score is 0.687686293596852
```

Batch size = 128 ; learning rate = 0.1 ; epochs = 10 ; with strategy

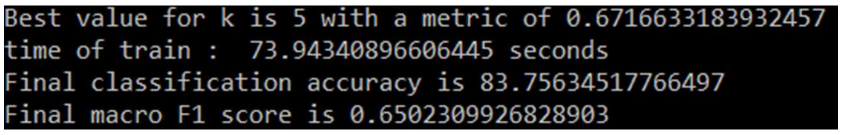


```
time of train :  2.8557777404785156 seconds
Final classification accuracy is 85.53299492385787
Final macro F1 score is 0.6960110776806275
```

Finally, we managed to highly decrease the training time by not impacting the scores

KNN

To perform the knn model, we chose to perform cross validation on the set [0, 10] for the value k, we then found that the best value for k is 5. Hence, we got the following results:



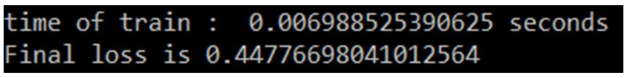
```
Best value for k is 5 with a metric of 0.6716633183932457
time of train :  73.94340896606445 seconds
Final classification accuracy is 83.75634517766497
Final macro F1 score is 0.6502309926828903
```

The final score is not bad but the running time is extremely long compared to neural network or even logistic regression (the computer on which the model was tested is the same for every model)

PCA DIMENSIONALITY REDUCTION

For the dimensionality reduction, we first searched the lowest dimension *d* such that the explained variance is bigger than a predefined threshold *max_exp_avr.* We also managed to keep a relatively high *max_exp_var* to not decrease accuracy. Hence, with a *max_exp_var* of 0.8, we found that the best dimension reduction is d = 58, with an explained variance of 0.8040915. Hence, we got those results:

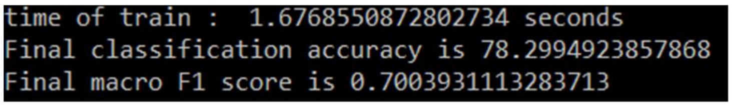
Linear regression :



```
time of train :  0.006988525390625 seconds
Final loss is 0.44776698041012564
```

Training time improvement ≈1260% ; loss increasement ≈2%

Logistic regression :

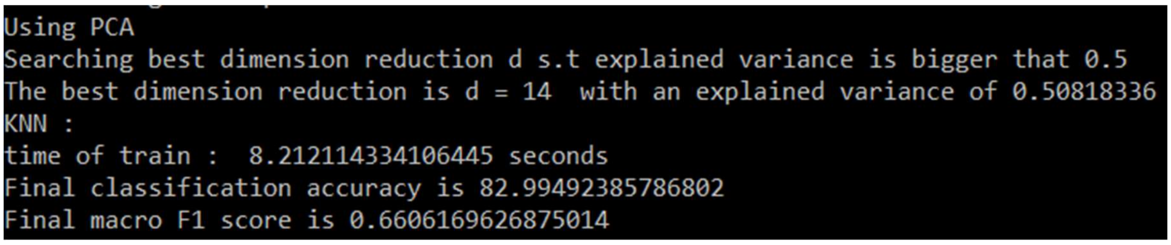


```
time of train :  1.6768550872802734 seconds
Final classification accuracy is 78.2994923857868
Final macro F1 score is 0.7003931113283713
```

Training time improvement ≈200% ; ≈no accuracy decrease

KNN :

Because the running time for knn is extremely long, we chose to decrease the threshold for the explained variance to try to get proper running time. We then setup PCA with *max_exp_var = 0.5* and got :



```
Using PCA
Searching best dimension reduction d s.t explained variance is bigger that 0.5
The best dimension reduction is d = 14  with an explained variance of 0.50818336
KNN :
time of train :  8.212114334106445 seconds
Final classification accuracy is 82.99492385786802
Final macro F1 score is 0.6606169626875014
```

Training time improvement ≈800% ; accuracy decrease ≈0.9%

CONCLUSION

Regression task :

By his simplicity, we think that the best model to perform regression on those data is a classical linear regression (ridge regression did not improve much the scores). Moreover, we can perform PCA on the data to significantly improve the time performance.

Classification task :

The best scores achieved by one model for the classification is from the neural network. Moreover, we do not need a complicated cross-validation process as with logistic regression or KNN and the running time is convincing. The neural network with the described architecture is then, for us, the best model for classification on those data, beyond all the tested ones.