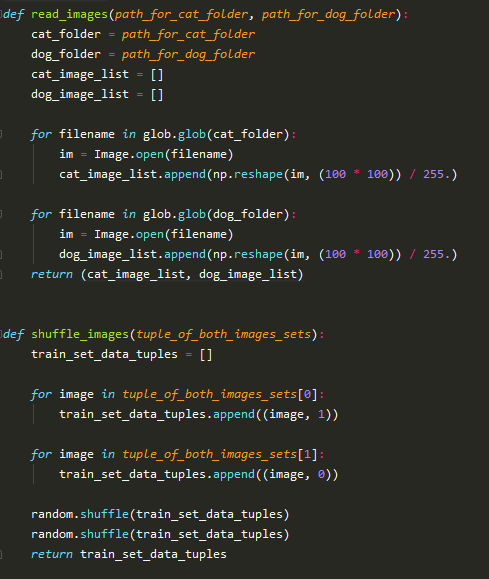
1) preparing the data –

We're using the dataset found here - <https://www.kaggle.com/tongpython/cat-and-dog/data>, these photos are colored dog and cat photos with different resolutions, we used a basic python script to turn them to grayscale photos with resolution 50X50 (we tried 100X100 but it was very SLOW), found in "code" folder under the name "grayscale\_folder", the original dataset contains 8K images as train set and 2K as test set.

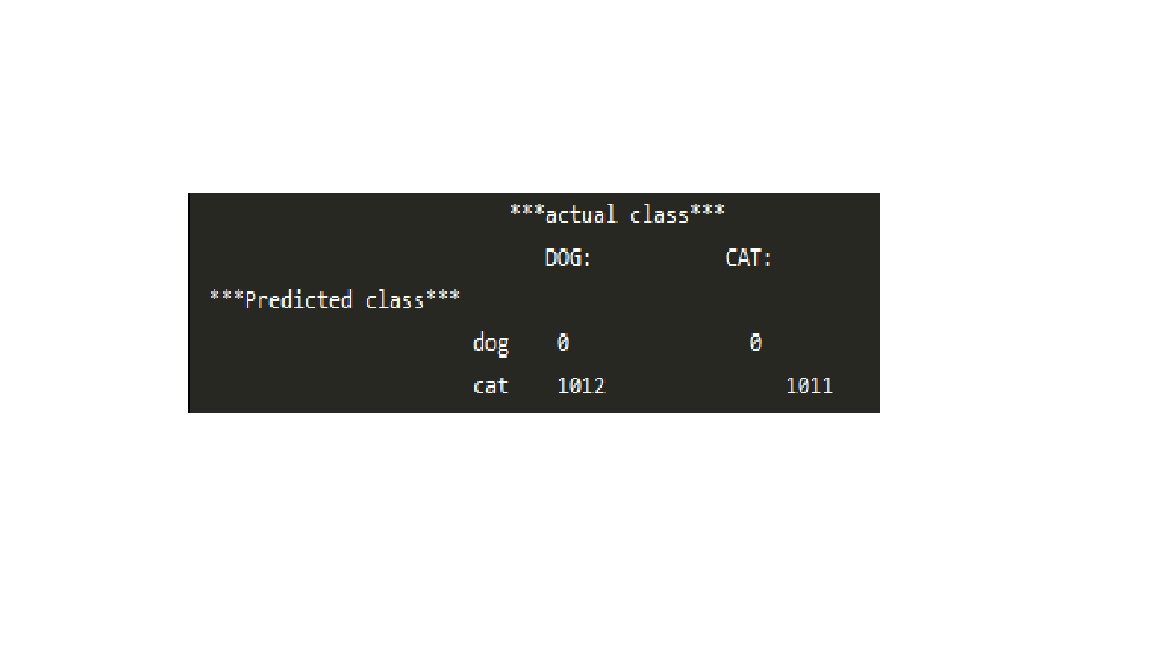
Additionally, we used basic functions to output the data as a set with which we can work:

\*please notice that we divide the pixels by 255 to transform every pixel from [0,255] to [0,1], we discovered that if the pixels are from the first range – the model won't learn anything – more on that later

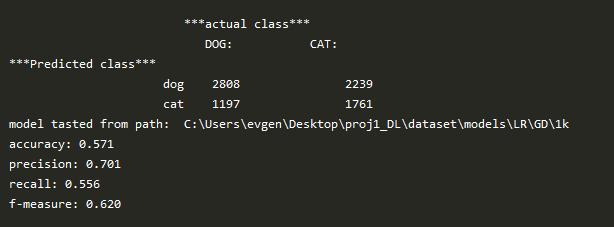


2)Logistic Regression –

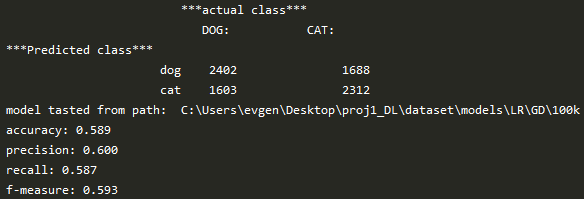
First, we ran the regression on non-normalized pixels (from the range [0,255]) but the learning hasn't work at all – the loss value diverged more and more every epoch:

Of course, we couldn’t predict nothing from test set:

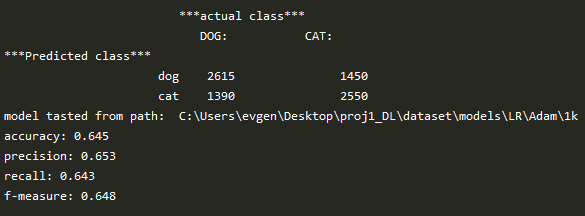
So now we work with only normalized pixels (in range [0,1]), we'll try to run the regression for 1K epochs, with GradientDescentOptimizer with a learning rate of 0.0001 (we tested 0.001 but the loss diverged), we start from ~0.7 loss value and got to 0.689934, this is not that good, it's quite easy to see from train error that the model is still underfitting:



We ran the same model for 100K epochs, got 0.67395353 loss value – which is still high, and still got underfitting:

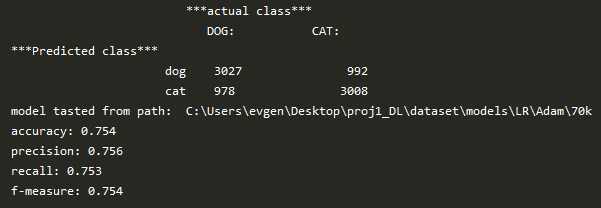


So, we'll try a more modern optimizer – ADAM for 1K epochs, we guess that the model will still underfit, we do this only for comparison - if we'll get a better loss value and better fitting, we'll stick with it, we got 0.63549614 loss value – which is better even than the 100K with DG!, the fitting:

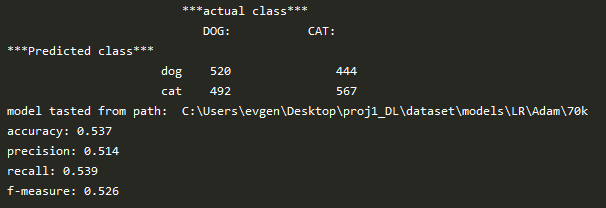


Its still very high for train error, so we'll run 70K epochs to see if we get a better fitting

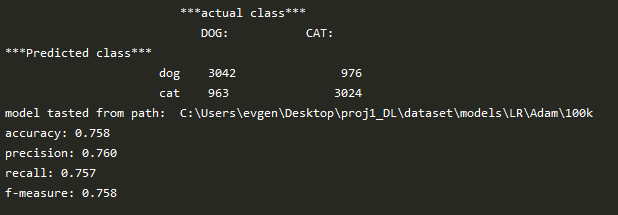
After 70K epochs – we got 0.5027957 loss value (lowest so far), fitting:



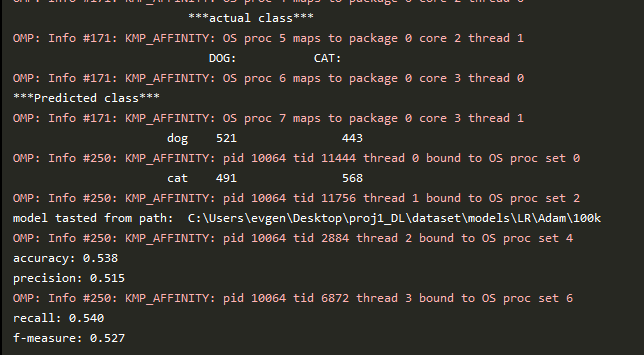
Much better but still low for train, we'll test it on the test set



Not very bad – the algorithm predicted that most of dog images are indeed dog and the same for cat, so it is learning something, but the accuracy is still very low, so we'll try to run the model with 100K epochs, we got loss value of 0.49831918 and the next confusion matrix for train set:



The conclusion here is that the rate in which the model keeps fitting itself to the data converges, as our train accuracy got better only by 0.4 after another 30K epochs, we'll try the test set:



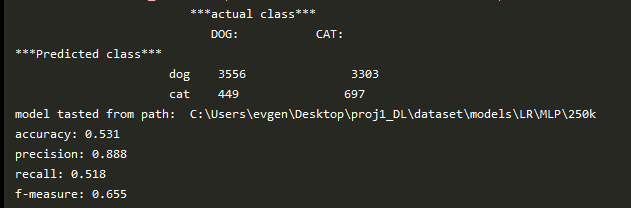
Got better only by 0.1, it seems as this model is too weak for this mission.

3)MLP-

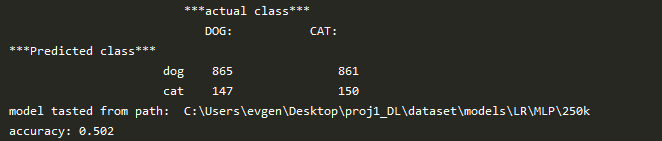
\* architecture of the model –

on the one hand we want a "rich enough" model i.e. enough neurons in our hidden layers to be capable of learning aspects of the image, on the other hand, too many features will require many epochs to "fit", and will slow our learning down (as for this assignment all layers have to be fully connected - too many neurons will result too many weights and biases to compute), so we'll create two hidden layers – the first is 25 neurons (the logic is one neuron for every 100 pixels), the second will be 10 (we want to reduce the number before we enter the logistic layer).

We couldn’t fit our regression after even 100K epochs with Adam, so we'll try 250K epochs for the MLP – we got 0 loss value after about 176K epochs, but the model is still underfitting, confusion matrix for train set:



We can see that the model hasn't really learnt as most of images were classified as dogs, even after 250K epochs, confusion matrix for test set:



NOT good.

So, as a conclusion – both models aren't strong enough for the mission, dog and cats are too similar in our data set –

After researching the web, we found that the only good recognition algorithms for images using CNN, it is quite hard to distinguish between such images of dogs and cats with these basic models, we guess that a very "fat" MLP with a huge amount of neurons would work but this will take VERY long to train, this is not a model we could build on our computer.