Final project – Data visualization

ID: 209904606

Date: 01/09/2021

In order to test how powerful the XGBoost classifier is, I decided to use it exclusively in all the notebooks and focused on the quality of my classifier instead of the amount of different types of classifiers. Using the xgboost learning api instead of sklearn's XGBClassifier, I was able to tune my model much faster and more accurately for two main reasons:

Reason number 1: The xgboost module has custom data structure called DMatrix that was optimized to work with the xgboost learning api

Reason number 2: The xgboost learning api supports early stopping rounds, meaning that each model we make finds its own amount optimal iterations, compared to sklearn's classifier where we would have to either search manually for the best number of iterations or use the same amount for all models, both of which are not optimal and time consuming.

Classification notebook:

In the last semester, I managed to reach an accuracy of 71.7% in identifying the winner of a League of Legends game based on each team's statistic at the 10^{th} minute mark. I managed to improve this model by using PCA to reduce the amount of features I used and by using a tuned XGBoost model. I managed to improve my accuracy by 0.8%. Although it is not much, I also managed to do so using only 1 feature instead of 8.

Fashion MNIST notebook:

In this notebook I managed to reach an accuracy of 88.72% using 121 out of 784 features with our tuned XGBoost model, an improvement of 3.79% compared to the default XGBoost model. However, the model preformed poorly in identifying shirts reaching an F1 score of only 0.58.

Cats vs dogs notebook:

In order to load images faster, I decided to use the multiprocessing module to load all the images in parallel fast and without using keras. Since I had some experience using OpenCV and processing images with it, I felt more comfortable using it.

Once we have the images we loaded using OpenCV we could use the transformations provided by the module easily, since we could not use neural networks, I decided to try to make a model that uses a very low amount of features, yet retains as much accuracy as possible, to do that, I used erode, dilate and grey scale image transformations to slightly distort the images, this led to less dimensions when using a PCA with 0.95 feature variance.

It might seem strange that I picked the dilate and erode parameters that I chose, however, they have been tested beforehand and had consistently reached the highest accuracy on the validation set, although they seem to be bad parameters to choose.

Overall, our tuned XGBoost model had an accuracy of 65.26% with 303 features, an improvement of 2.14% from our baseline XGBoost model.

Hands notebook:

I started off by removing the time column from all the data sets since each person seemed to have a unique range of time in their dataset and just like Frame ID, it's a parameter that holds little to no importance. If, for example, we had an extremely long recording of someone, he would have really high values in the time column and it might lead to false prediction since our model didn't train on data that had such high time values.

In addition, since every valid recording had a right hand and a left hand, I decided to add 3 new features, elbow distance, position distance and wrist distance.

At the end, I managed to reach an accuracy of 87.92% with 19 features using a tuned XGBoost model, an improvement of 4.86% from our baseline XGBoost model. This kind of metric however is quite misleading since our model could easily classify the “Alone” state since it always had the same right hands, but the “Spontan” and “Sync” states had a low f1 score of 0.82

To not overfit by learning person specific traits, I decided to split the validation sets by person instead of picking random samples since it is easy to overfit this kind of dataset.