01/06/2020

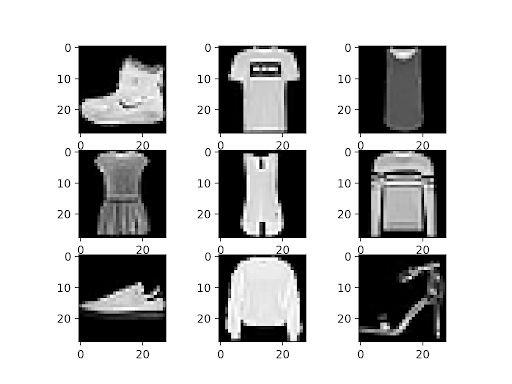
HW Completion – Anya Katz

The task was to use “fashion\_mnist” dataset, to run some machine learning algorithms and to analyze the results.

The “fashion\_mnist” dataset is a known dataset of images of clothes.

The images are labeled.

The data allows us use supervised learning.

  
Figure 1: fashion\_mnist image examples

Attachments:

- utils\_fastion\_mnist.py

- decision\_tree\_fashion\_mnist.py

- naive\_bayes\_fashion\_mnist.py

- svm\_fashion\_mnist.py

- dl\_fashion\_mnist.py

- output – Folder

- \*.png - ‘Confusion Matrix’ files

- Results.txt – Accuracy plots

- Conclusions.docx – This file

Analyzing the data

The data is set of 70,000 images of clothes.

Every image is 28 on 28 pixels converted to gray scale [0 – 255].

The images are labeled to 10 categories:

["T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"]

60,000 images were used as a train and 10,000 used as a test.

Usually, the dataset is divided into two parts: train dataset (70% of all data) and test dataset (30% of all data). Yet, in this case, we have enough data to change the balance in order to get more precision.

The data is post Exploratory Data Analysis (EDA), therefore it does not need further manipulations to get the sufficient accuracy.

Data processing implementation

I have used the following algorithms: Decision Tree, Naive Bayes, Support Vector Machine and Neural Network from the ‘keras’ and ‘sklearn’ libraries.

The code is in five python files.

- General Utils file: utils\_fastion\_mnist.py.

This file loads the “fashion\_mnist” dataset, prepares the data to use, generate the reports of the algorithm results and plots the confusion matrix.

decision\_tree\_fashion\_mnist.py, naive\_bayes\_fashion\_mnist.py, svm\_fashion\_mnist.py and dl\_fashion\_mnist.py are the algorithm files. They all using the General Utils file (utils\_fastion\_mnist.py).

Results overview

Naive Bayes

The Naive Bayes is an algorithm based on probabilities. This algorithm uses the probability of feature occurrence in the dataset.

For example: The algorithm will help to identify spam emails by the presence of specific words most common to spam such as ‘Buy’, ‘Limited’, ‘Today’.

In order to calculate the probabilities, the features needs to be very similar, yet in our case, the clothes features are not alike. The sleeves of a t-shirt will not be the same in all of the images.

As expected, the Naive Bayes accuracy is very low:

---------------- Naive Bayes Report ---------------

F1 score: 0.5561601931134046

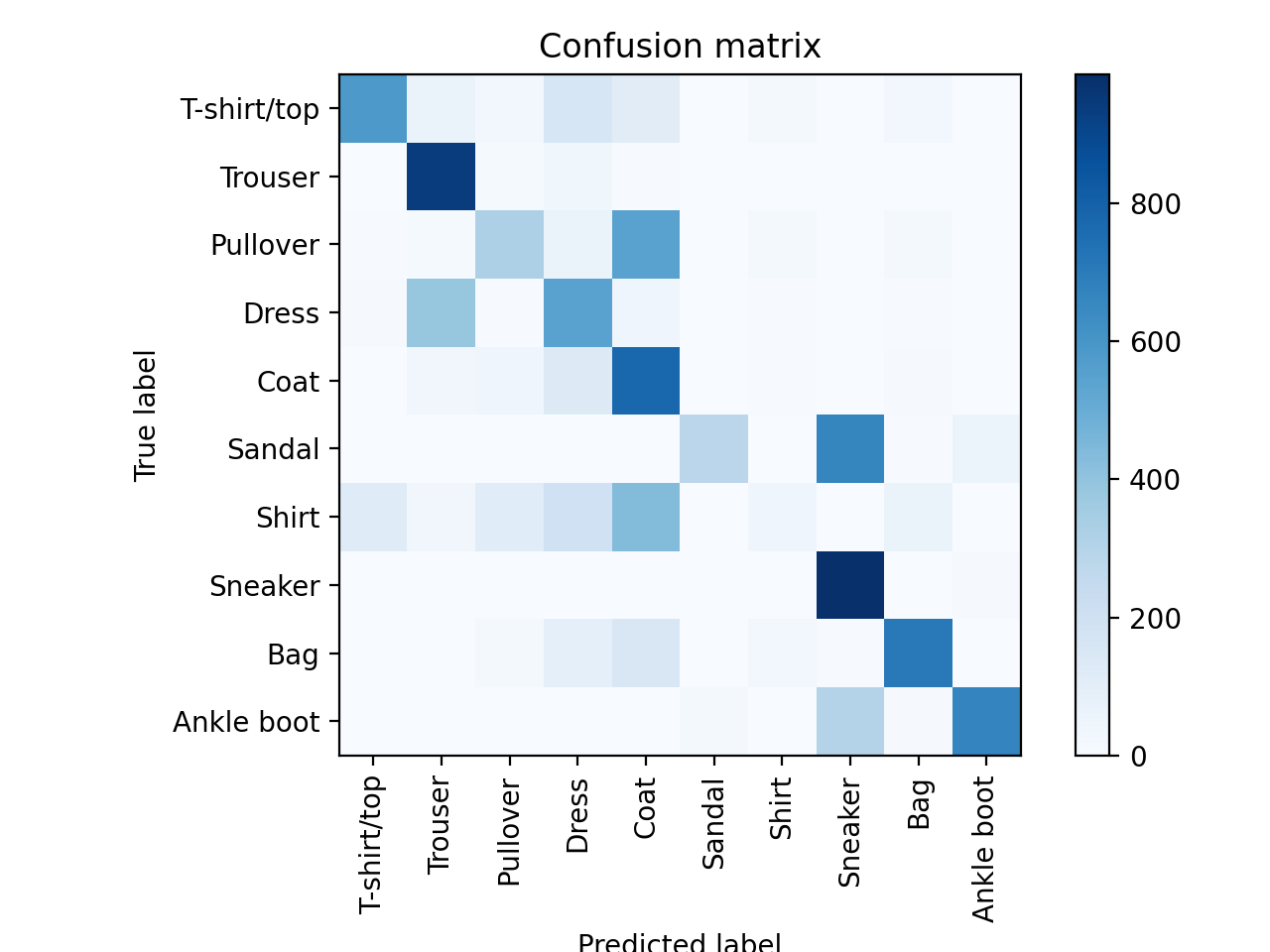
Accuracy score: 0.5856

precision recall f1-score support

accuracy 0.59 10000

macro avg 0.64 0.59 0.56 10000

weighted avg 0.64 0.59 0.56 10000



Decision Tree

This algorithm fits tasks such as picking a house based on the house features. It is a very easily traceable algorithm – you can see the full route of the decision.

This algorithm prefers well defined features and uses binary decisions such as ‘is the house has 4 rooms’. When dealing with real images the feature edges are not sharp. If the images where artificial we would expect some better accuracy yet not enough:

---------------- Decision Tree Report ---------------

F1 score: 0.752632719511957

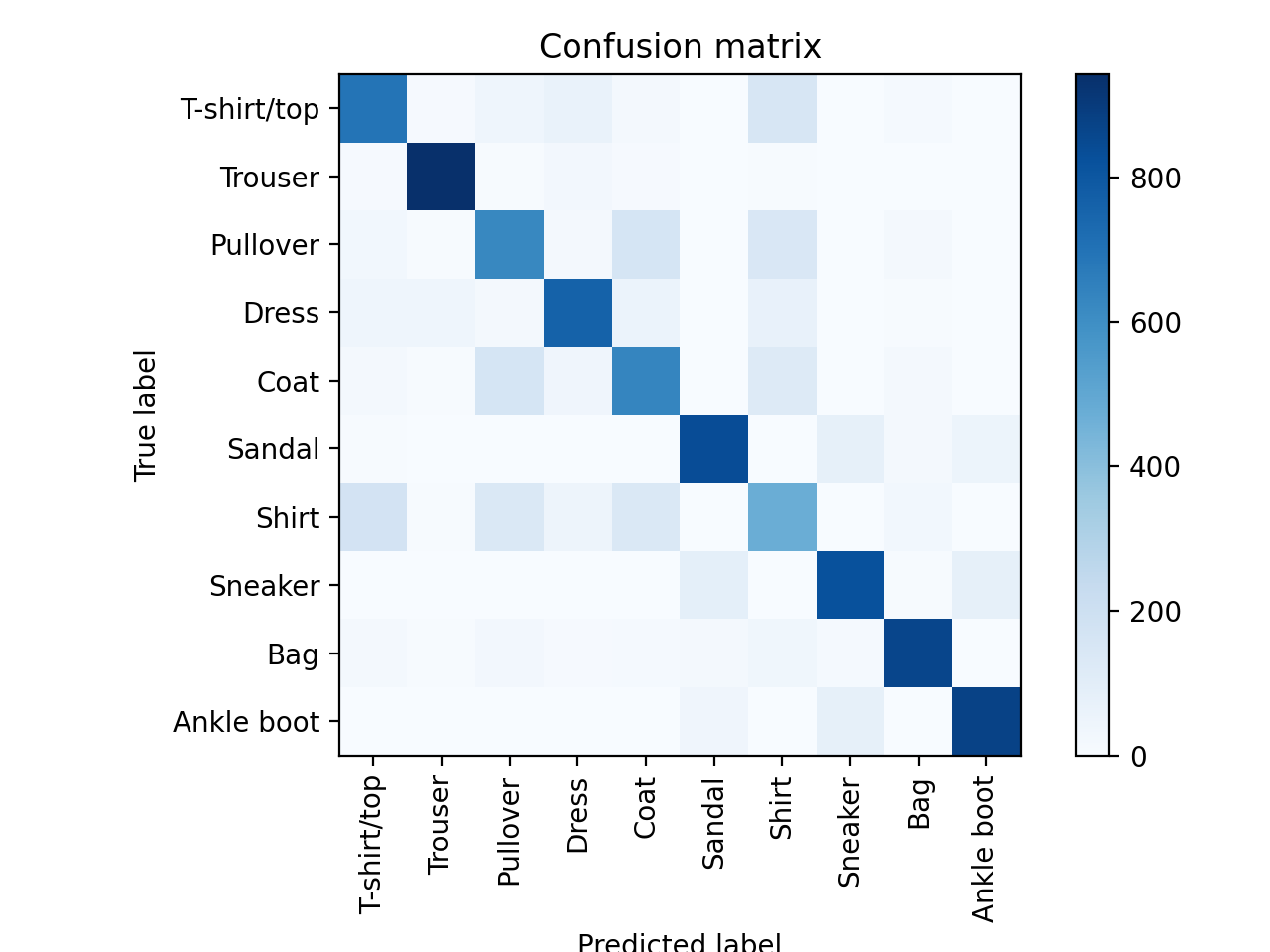
Accuracy score: 0.7523

precision recall f1-score support

accuracy 0.75 10000

macro avg 0.75 0.75 0.75 10000

weighted avg 0.75 0.75 0.75 10000



Support Vector Machine

The SVM is a less feature based algorithm, this is an arithmetic based algorithm. The SVM searches for the difference between the images rather then the similarities. This allows to separate sleeves and heels more easily. The SVM is a powerful tool for images and other signals such as audio, medical signals and video.

As we expected, the SVM performed much better then the Naive Bayes and the Decision Tree:

---------------- SVM Report ---------------

F1 score: 0.845599053028593

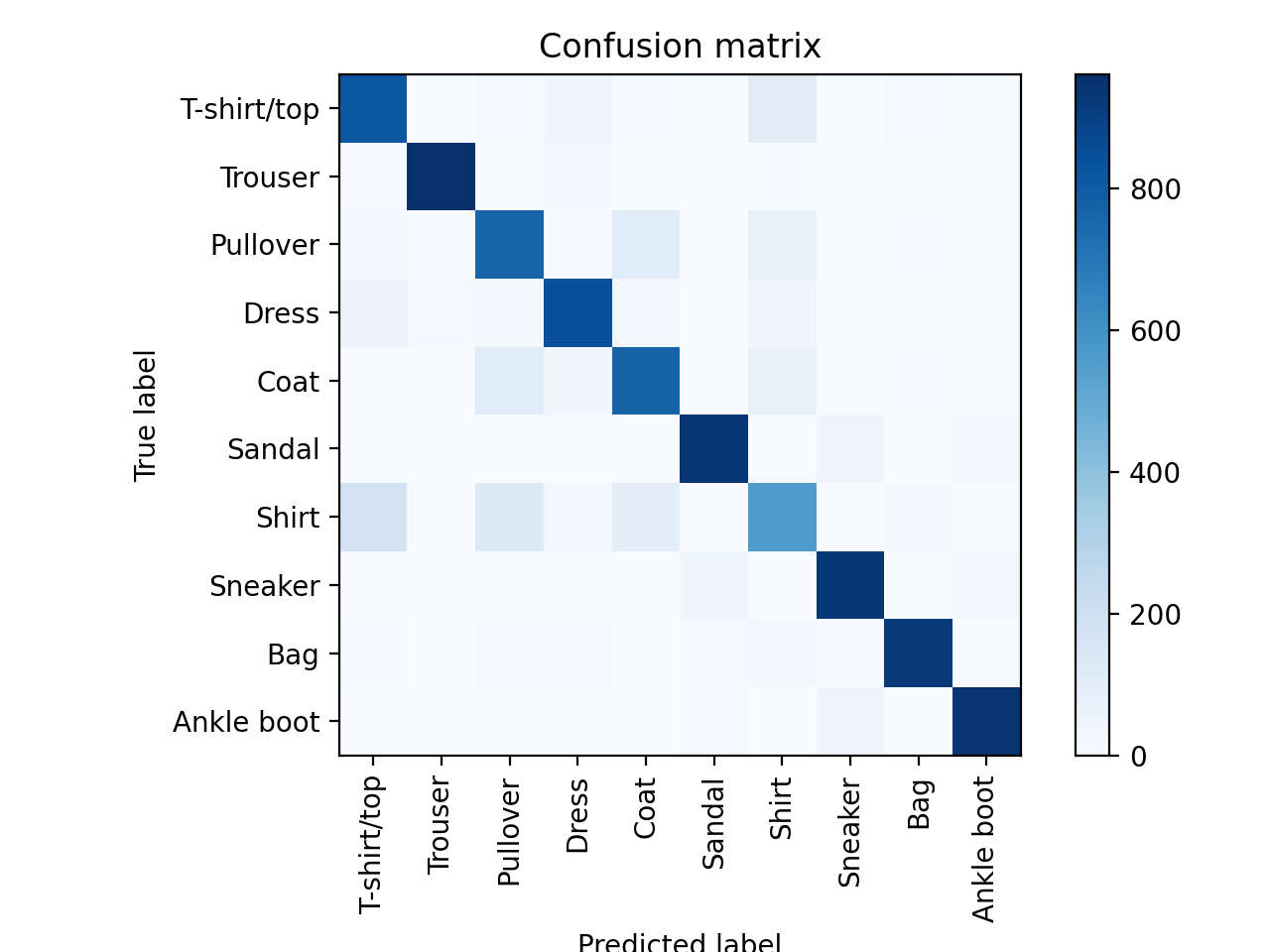
Accuracy score: 0.8463

precision recall f1-score support

accuracy 0.85 10000

macro avg 0.85 0.85 0.85 10000

weighted avg 0.85 0.85 0.85 10000



Deep Learning – Neural Network

The DL is a very powerful tool, arithmetic based algorithm. The DL is more configurable, yet, much harder to adjust and to train. This algorithm is not traceable – there is no way to understand why the decision was made. When configuring the algorithm correctly, the outcome on arithmetic problems may be better then SVM. In our case the DL performed a bit better then the SVM:

---------------- DL Report ---------------

F1 score: 0.8737270024114278

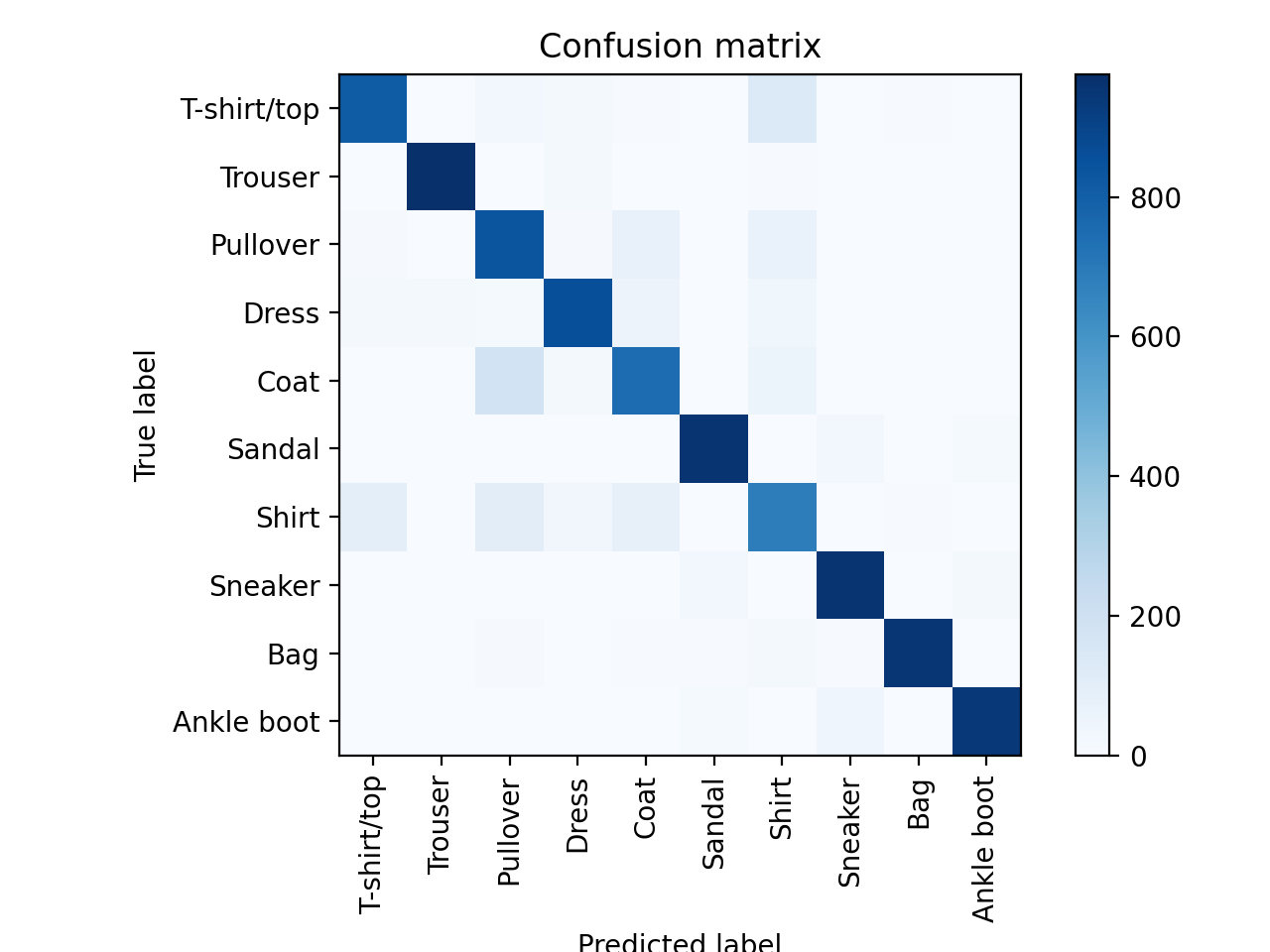
Accuracy score: 0.8731

precision recall f1-score support

accuracy 0.87 10000

macro avg 0.88 0.87 0.87 10000

weighted avg 0.88 0.87 0.87 10000



Conclusion

We have tested 4 machine learning algorithms: Naive Bayes, Decision Tree, Support Vector Machine and Neural Network on “fashion\_mnist” dataset. All the algorithms performed as expected and the accuracies where under 90%. I believe it is possible to increase the accuracy of the Support Vector Machine and the Neural Network using more data EDA, configurations and combining some algorithms together.