



Report

Assignment 3

Supervised Learning algorithms



Author:

Findlay Forsblom ff22ey

Termin: VT20

Ämne: Machine Learning

Kurskod: 2DV516



Table of contents

Table of contents	2
Exercise 1	3
Linear Kernel	3
Rbf Kernel	3
Poly Kernel	5
Exercise 1.1 VG	6
Exercise 2	8
Exercise 3	11
Exercise 4	12



Exercise 1

In this exercise we were asked to test different kernels and see how flexible each was. Then tune the hyper parameters to optimize each kernel.

Linear Kernel

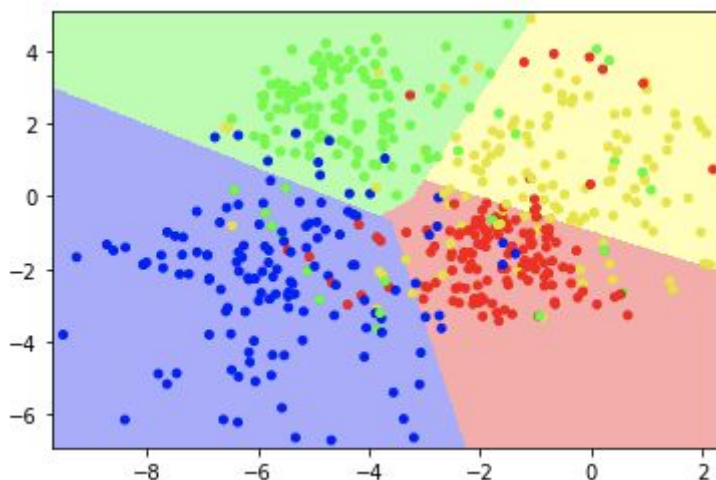
Here the only hyper parameter of interest was the C parameter which was the regularization parameter. To reduce the amount of models that had to be tested and also the training time, few values were started off with, but with large interval in between. Then depending on the results gotten then the interval between the numbers was then reduced and the values more specific. This approach was used when searching for the best hyper parameters for the other kernels as well. In all the test 5- fold cross validation was used. The picture below shows the values that was started with in the first attempt

```
parameters = [{'kernel': 'linear'], 'C': [0.5, 1, 10]}
```

After the first try $c = 1$ was chosen to be the best value with an accuracy of 0.8. So i then added more values that was centered around 1. The picture below shows the values of C that were searched for.

```
parameters = [{'kernel': 'linear'], 'C': [0.8, 0.9, 1, 1.2, 1.3, 0.7, 0.6]}
```

This time around i got that the best C was 0.8 and the score remained the same at also 0.8. The picture below shows the resulting decision boundary.



Rbf Kernel

Here the were 2 parameters of interest. The C parameter and the Gamma parameter. The picture below shows the different values that were started for each parameter

```
parameters = [{'kernel': 'rbf'], 'C': [0.5, 1, 10], 'gamma': ['auto', 0.5, 1, 10]}
```



After the first try i got the following to be the best parameters

Key	Type	Size	Value
C	int	1	10
gamma	str	1	auto

Those results were gotten with a score of 0.82. Since 10 was given to be the best gamma, more values centered around 10 were tested. The picture below shows the values tested and the results gotten

```
parameters = [{'kernel': ['rbf'], 'C': [10, 9, 7, 12, 15]}
```

Key	Type	Size	Value
C	int	1	9

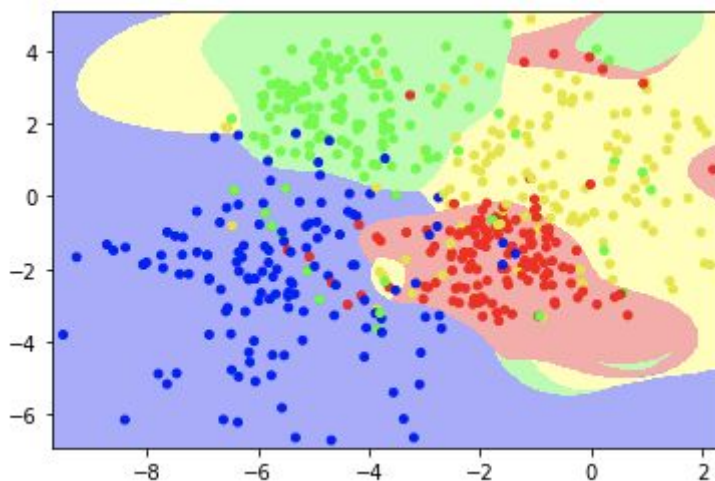
As can be seen from the picture above. 9 was chosen as the best value, but the score remained the same at 0.82.

In conclusion for the rbf kernel the best parameters that were chosen were

C = 9

gamma = auto

The picture below shows the resulting decision boundary with the training data for the rbf kernel





Poly Kernel

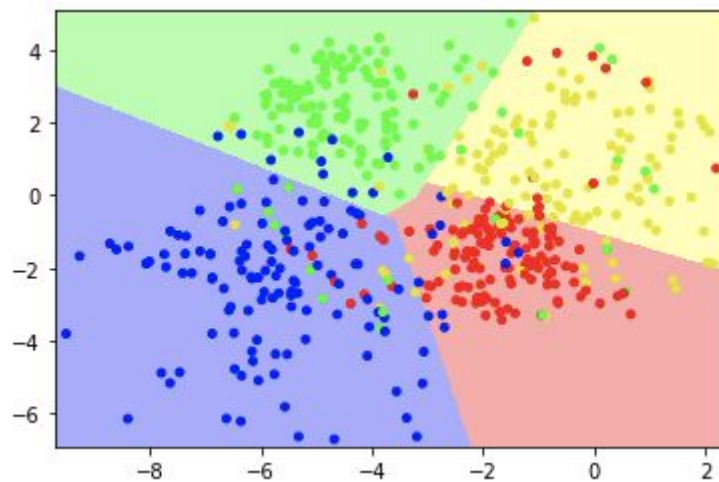
Here there were 3 parameters of interest. The regularization parameter C , gamma and the Degree. Using the same tactics as with the other kernels i came up with the following best parameters

$\gamma = 0.5$

$c = 0.5$

degree = 1

The picture below shows the resulting decision boundary





Exercise 1.1 VG

In this exercise we were asked to implement our own kernel by precomputing it by means of a so-called *Gram-matrix* and then also find the best hyper parameters.

The kernel that was to be implemented was the ANOVA kernel. The ANOVA kernel could be computed as follows

$$k(x, x') := \left(\sum_{j=1}^{\ell} \exp \left(-\sigma (x_j - x'_j)^2 \right) \right)^d ,$$

which in code look something like this

```
def anova(self, xi, xj, sig, d):  
    return np.sum(((xi-xj) ** 2) * (-sig)) ** d  
  
def computeGram(self, X,Y,sig, d):  
    n = X.shape[0]  
    m = Y.shape[0]  
  
    gram = np.zeros([n, m])  
    for i in range(n):  
        for j in range(m):  
            gram[i,j] = self.anova(X[i], Y[j], sig, d)  
    return gram
```

Now the harder part of this was trying to perform gridsearch. The trick here was to wrap the kernel as an sklearn estimator making it a subclass of the *BaseEstimator* and *TransformerMixin*. This gives it methods such as *set_params* and *get_params* making it grid searchable. Then a pipeline was used to automate the steps of transforming and predicting.

The picture below shows the parameters searched for

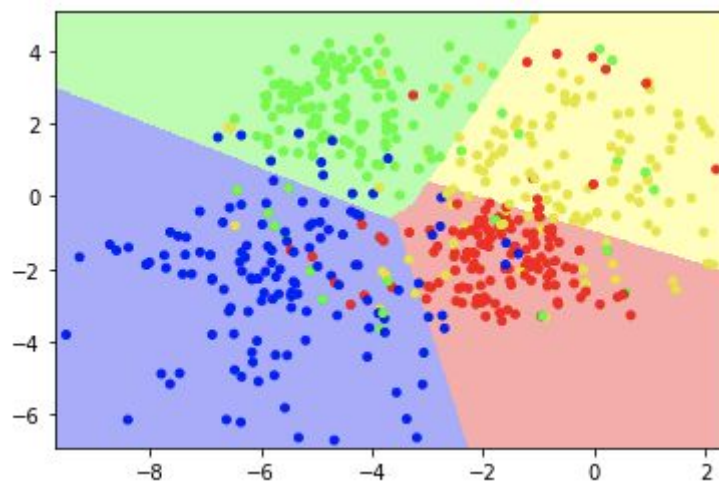
```
cv_params = dict([  
    ('anova__sigma', [0.5,1.5,3]),  
    ('svm__kernel', ['precomputed']),  
    ('anova__d', [2,1,3]),  
    ('svm__C', [0.5, 1,2.5]),  
])
```

and this were the best parameters chosen



best_parameters - Dictionary (4 elements)				
Key	Type	Size	Value	
anova__d	int	1	1	
anova__sigma	float	1	0.5	
svm__C	int	1	1	

This gave an accuracy of 81% and gave the following decision boundary which was quite similar to the linear kernel

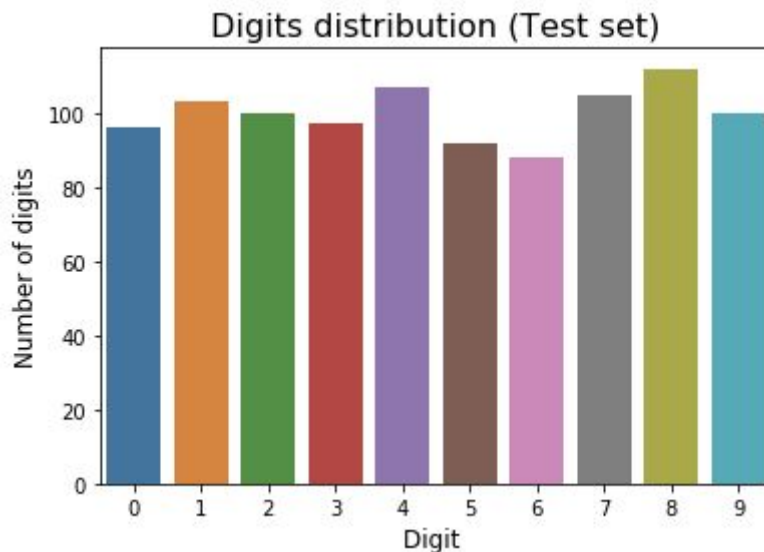
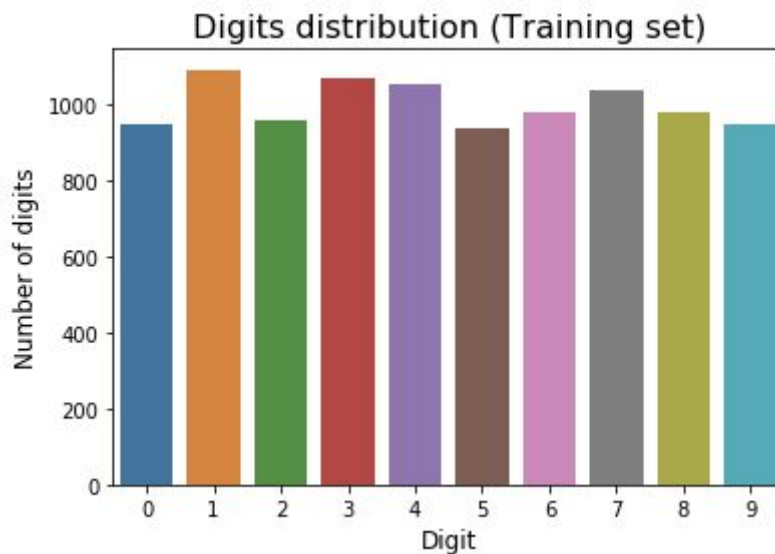




Exercise 2

In the first part of this assignment we were asked to find the best hyper parameters value using the rbf kernel on the real MNIST dataset.

As usual gridsearch was the technique used using 5-fold cross validation when searching for the best values for the hyper parameters. C and γ . Out of the total 60,000 digits for training, only 10,000 were used and out of the total 10,000 digits for testing only 1000 were used. The picture below shows the numbers distribution for both the training set and the test set



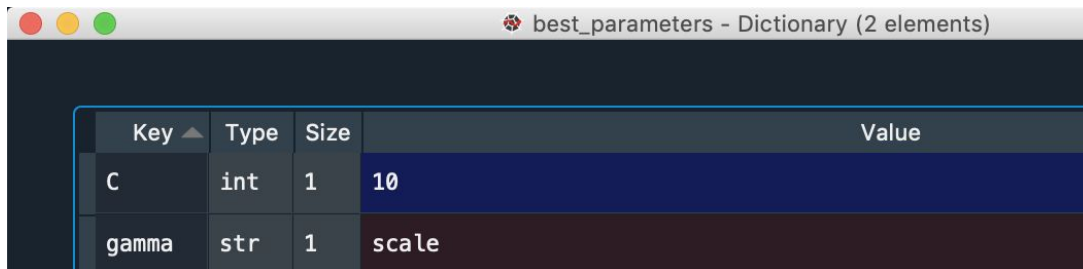
As seen from the pictures above, the numbers are pretty much evenly distributed on both the training set and the test set which is important in classification task if accuracy is to be used as a method of evaluation.

The picture shows the values of the hyper parameters that were first tested



```
clf = svm.SVC(kernel = 'rbf')  
parameters = [{'C': [0.05, 0.5, 1, 10],  
               'gamma': [0.05, 0.5, 1, 10, 'scale', 'auto']}]
```

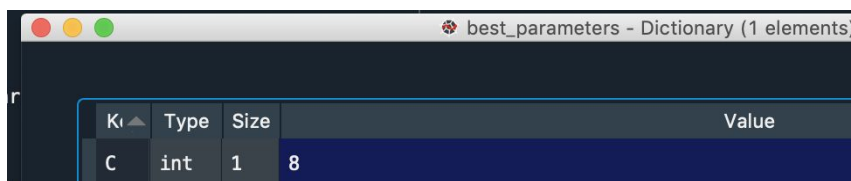
And these were the results gotten



Key	Type	Size	Value
C	int	1	10
gamma	str	1	scale

as can be seen from the picture above $C = 10$ and $\gamma = \text{scale}$ were the best values that were chosen. This gave a validation score of 0.95. On the second try I then tried values around 10 for C to see if scores would improve. The picture below shows the values tested and the results that were gotten.

```
parameters = [{'C': [8, 10, 7, 12]}]
```



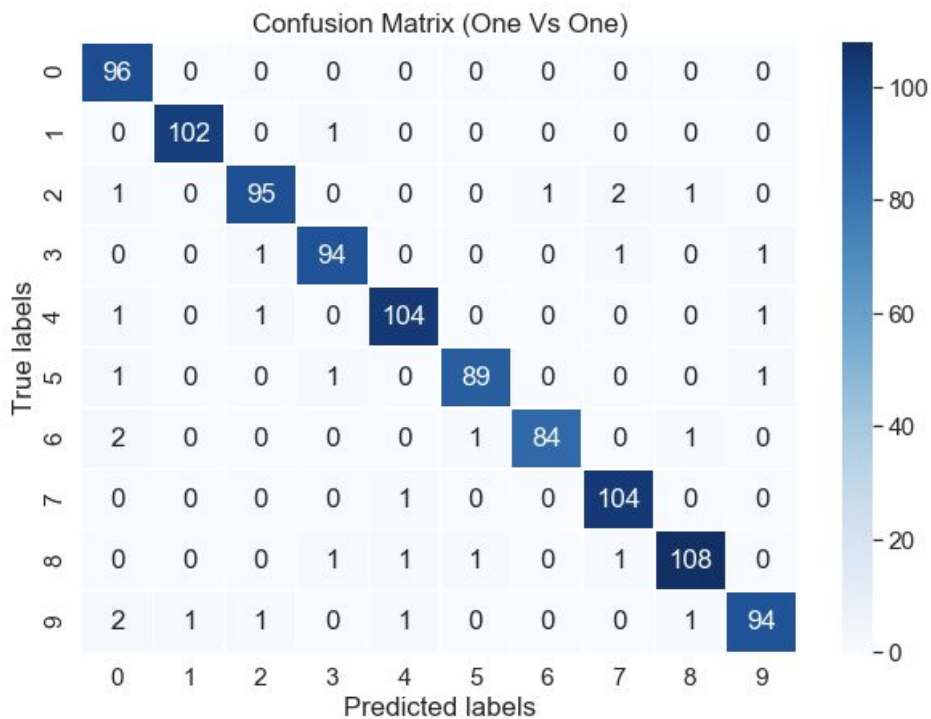
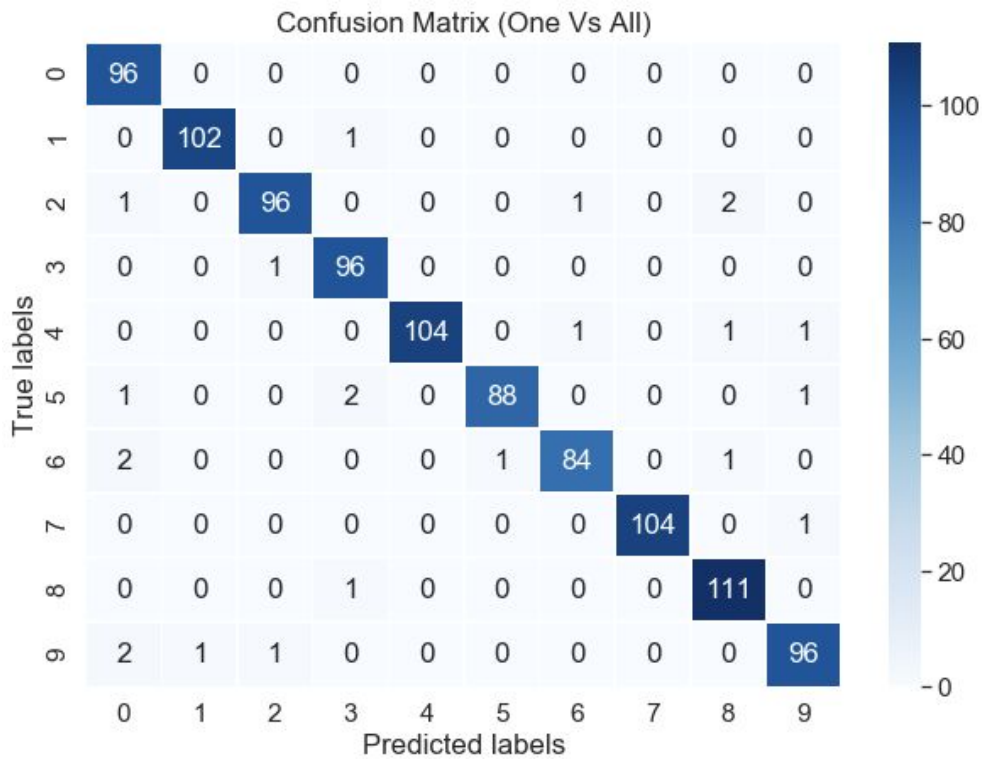
Key	Type	Size	Value
C	int	1	8

In conclusion $C = 8$ and $\gamma = \text{scale}$ seemed to give the best scores. Using those values on a test set gave an accuracy of 97%

In the later part of the assignment we were asked to compare the built in binarization scheme built for SVC(one-vs-one) against the one-vs-all scheme which was discussed in Lecture 5. To do that we were asked to implement our own version of the one-vs-all scheme.

Using the same dataset and hyper parameters in the first part gave an accuracy of 97.7% which was slightly higher than the one-vs-one scheme used by SVM.

The pictures below show the confusion matrix of both the one vs all and the one vs one



As can be seen from the pictures above. They look pretty much identical. The only difference i noticed was the training and prediction time. It took much longer when using the one vs all approach



Exercise 3

In this exercise we were asked to use a decision tree regressor and random forest regressor to predict the amount of facebook comments a post will generate in h hours.

The pictures below shows the results gotten

Task 1

```
Training mse 0.0036264621846687343
Test mse 4991.93
```

```
Training mse for max_depth = 5: 388.8503894487183
Test mse for max_depth = 5: 10603.583798149606
```

Task 2

```
Training mse 64.48298468947223
Test mse 4749.810993444444
```

```
Training mse for max_depth = 30: 64.68060313858341
Test mse for max_depth = 30: 4626.387838760521
```

Task 3

```
Training mse with DT: 0.0036966966219412014
Test mse with DT: 411.72727272727275
```

```
Training mse for max_depth = 3 with DT: 614.7615415130292
Test mse for max_depth = 3 with DT: 346.95149183232365
```

```
Training mse with RF: 62.72628797169355
Test mse with RF: 262.90589696969704
```

```
Training mse for max_depth = 15 with RF: 65.48261511348453
Test mse for max_depth = 15 with RF: 281.1100959786682
```

The *max_depth* parameters were found using gridsearch on a validation set. Taking a look at the picture above, in task 1, it can be seen that although adding a max depth made the model not to overfit the training data as bad, the test mse went up as well which is undesired

In task 2, 30 was surprisingly chosen to be best *max_depth* parameter. But as can be seen in the results there was barely any difference in the training and test mse scores.

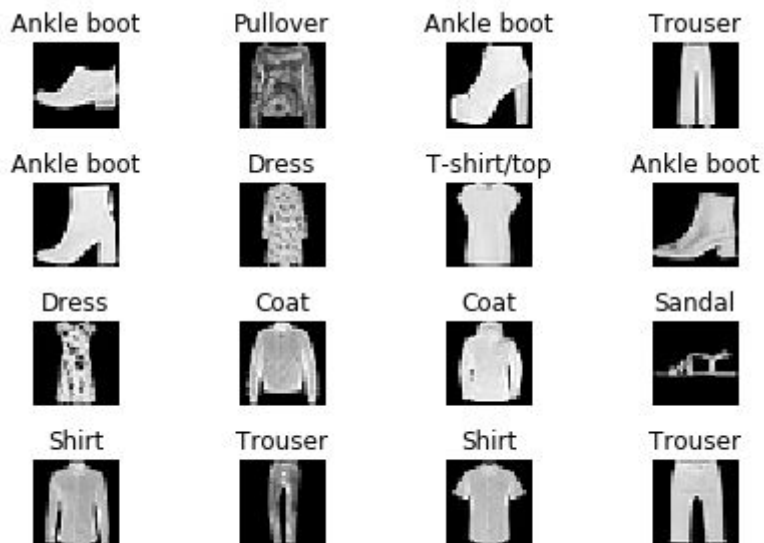
In task 3, when using decision tree with a max depth of tree and a dataset where the 39th feature was equal to 24, the model returned some kind of an unknown result with a test error that was much lower than the training error. Doing same test using random forest and the same dataset gave a much better result, but setting the *max_depth* to 15 which was found using gridsearch did not seem to improve the results.



Exercise 4

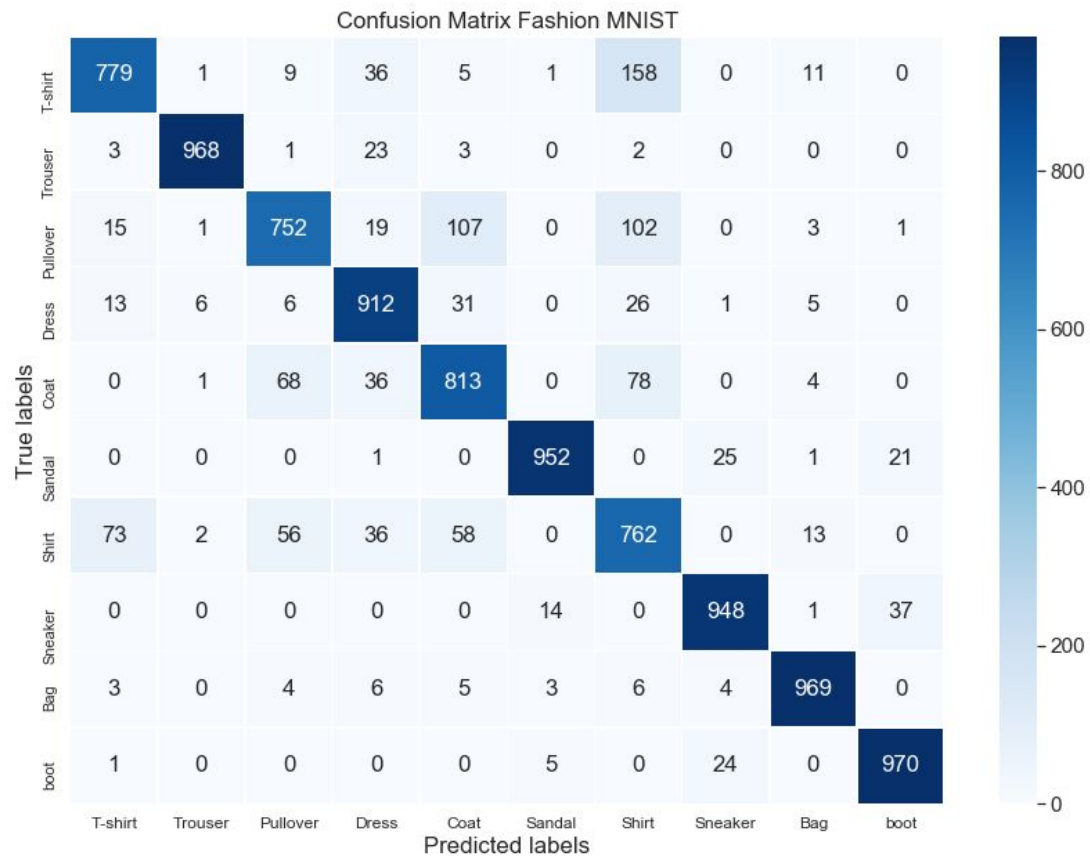
In this task we were asked to train a multilayer perceptron (MLP) to be able to classify the Fashion MNIST dataset which just like the regular contains 10 categories but of different clothes. The dataset contained 60,000 images for training and 10, 000 for testing.

In the first task we were asked to print 16 random samples from the training set with the corresponding labels. The picture below shows the samples that were printed



In the next task we were then asked to train the multilayer perceptron. For this task i decided to use the Keras/Tensorflow library to solve it and using the book as my guideline. Two hidden layers were used with 300 units in the first and 100 units in the second. Both used relu as the activation function. In the output layer since there could be 10 possible classifications, an output layer of size 10 was chosen with softmax being the activation function used. Using the settings mentioned gave an accuracy of 88.25% on the test set.

The picture below shows the confusion matrix of the Fashion MNIST dataset



Looking at the confusion matrix above, it seems like T-shirt/top, Pullover, and Shirt seems to be the hardest to classify. T-shirt/top seems to get confused with just normal shirt while pullover seems to get mixed up with Coat and Shirt. Shirt seems to get confused of being T-shirt/top, pullover, Dress and Coat.