CSE 590-04 Homework 3

Aliaa Elshamekh

February 24, 2021

1 Introduction

Image classification is one of the most fundamental problems in Machine Learning. It is the core foundation for bigger problems such as Computer Vision,Face Recognition System,or self-driving car.There are many classification models that can be used for this task;however,it is important for student to fully understand the concepts of each model, and how they perform on different dataset.

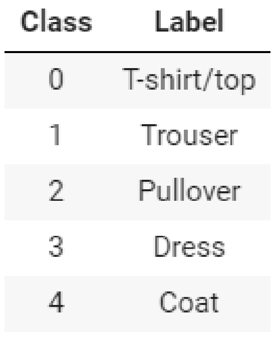
This goal of this project is to build a performant multi-class classifier that successfully classifies the images of the *Fashion-MNIST* dataset.Comprised of five categories of grayscale images (28 x 28), *the Fashion-MNIST* dataset consists of 10,000 training images and 5,000 test samples which is provided with the assignment.

In this project,we will explore and study two classification models and compare their performance on grayscale images.

The two classifiers are :

- Kernal SVM Classifier.
- Multilayer perceptrons MLPs

When building a machine learning model, the correct data preparation and preprocessing is just as important as creating the classification model itself.Prior to any optimization or training of subsequent classification models, the *Fashion-MNIST* dataset needed to be read in and re-shaped before any additional pre-processing steps were carried out.



Class	Label
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat

Figure 1: The five classes available in the data set

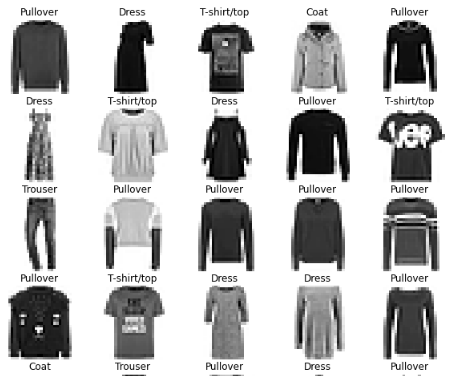


Figure 2: sample from *Fashion-MNIST* dataset.

In order to prevent over fitting of the model,the dataset can be split into both a Train and Validation datasets.As stated in the assignment,we are going to use 4 fold cross validation,as by testing the model against the test set,we could avoid the realistic trap of training our model to very successfully fit only the training data,with poor performance capabilities on the unseen test data.

For both classification models used in this homework,a four fold cross validation was performed,in order to verify the trained models performance and associated hyper-parameters.

The *Fashion-MNIST* dataset is available in a (n, 28, 28) 2-dimensional (2d) array where n represents the number of image records in the source data. A 2d array for each pixel in the image can be assumed to make both logical and structural sense to the human eye, but to a linear machine learning algorithm, which is expecting a 1-dimensional

(1d) array of input features (X), this 2d array is difficult to analyze. As a result, the goal of re-shaping the array was to convert the (n, 28, 28) 2-d array to a vector of 784 pixels for both the Support Vector Machine and Multi-layer Perceptron.

We need to shuffle our training data to ensure that we don't miss out any digit in a cross validation fold,we have to get uniform samples for cross validation.

No further work was carried out to rectify any class imbalances that could affect the classification models discussed in this homework,since the classes are balanced.

Both classifiers are requiring sort of data normalization in preprocessing.It was asked explicitly to make some experiments using four kind of scaling in data preprocessing.

- No Scaling.
- Standard Scaling
$$x_{scaled} = \frac{x - \mu_x}{\sigma_x}$$
- Min Max Scaling
$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$
- Robust Scaling
It scales features using statistics that are robust to outliers.This method removes the median and scales the data in the range between 1st quartile and 3rd quartile. i.e., in between 25th quantile and 75th quantile range. This range is also called an Interquartile range.
$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

2 Kernal SVM

SVM algorithms use a set of mathematical functions that are defined as the kernel.

The function of kernel is to take data as input and transform it into the rm. Different SVM algorithms use different types of kernel functions. These functions can be different types.For example linear,nonlinear, polynomial, radial basis function (RBF), and sigmoid. Introduce Kernel functions for sequence data, graphs, text, images, as well as vectors.The most used type of kernel function is RBF. Because it has localized and finite response along the entire x-axis. The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with little computational cost even in very high-dimensional spaces.
Kernel Function is a method used to take data as input and transform into the required form of processing data. "Kernel" is used due to set of mathematical functions used in Support Vector Machine provides the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface is able to transformed to a linear equation in a higher number of dimension spaces. Basically, It returns the inner product between two points in a standard feature dimension.

- Standard Kernel Function
$$\begin{aligned} K(\bar{x}) &= 1, \quad ||\bar{x}|| \leq 1 \\ K(\bar{x}) &= 0, \quad o.w \end{aligned}$$
- Gaussian Kernel Radial Basis Function (RBF)
$$K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2} \qquad \text{for} \quad \gamma > 0$$
- Sigmoid kernel
$$K(x, y) = \tanh(\alpha x^T y + c)$$

In general,if data can be perfectly separated using a hyperplane,then there will exist an finite number of such hyper planes.SVM chooses the maximal margin hyperplane,such that the hyperplane has the furthest minimum distance to the training observations.The maximal margin hyperplane depends directly on the support vectors,which is only a small subset of training points along margins.This means SVM is quite memory efficient,which is one advantage of using SVM.

There are cases when data can not be perfectly separated.In this case,rather than seeking the largest possible margin so that every observation is on the correct side of the hyperplane,it is allowed to be violated such that some training points are allowed to be on the incorrect side of margin. This refers to a soft margin.

There are other popular kernels but I just mentioned the kernel I am going to make my experiments on.SVMs are sensitive to the feature scales, as we are gonna see later on

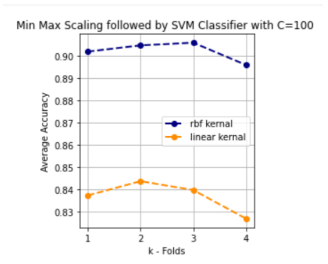
In practice,the boundary between multiple classes is not always linear.In such case,a particular technique called kernel trick is suggested to create non-linear classifiers.When the feature space gets enlarged, the decision boundary would actually become linear.There are many different ways to enlarge the feature space by using different kernel functions,e.g. polynomial, radial basis function,sigmoid.It is tricky to make the right choice of kernels and kernel parameters,as it is very easy to overfit the model due to the variance of model selection criteria. The Radial Basis Function kernel is a good default kernel.There are a few parameters need to be specified in sklearn's support vector machine function:a regularization value (C),which controls the strictness of soft margin,and a kernel of radial bias function are specified.The choice of kernel and kernel parameters are optimised by a cross-validation based model selection through sklearn's gridsearch method.

In table 1 , we can see the SVM classifier performance sensitivity with changing of the kind of prepossessing to the training data

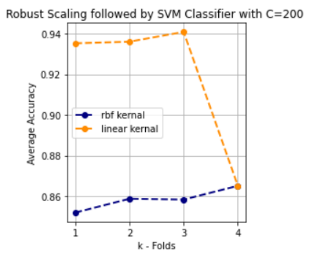
	No Scaling	Standard Scaling	Min Max Scaling	Robust Scaling
average training accuracy	0.98	0.818	0.92	0.75
Test accuracy	0.89	0.815	0.90	0.72

Table 1: Accuracy Scores for SVM classifier with kernal rbf

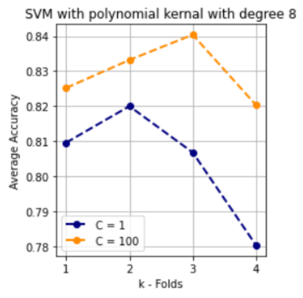
Also we can see in Figure 3 , The average training score for each fold while experimenting different hyperparamters ,such as different kernel, regularization ,...etc. We can see the fluctuation in average accuracy for each case in the figure 3,sigmoid kernel gives the worst model and from the preprocessing prespective the Robust Scaling does not suit this kind of data at all, as It makes the performance worse.



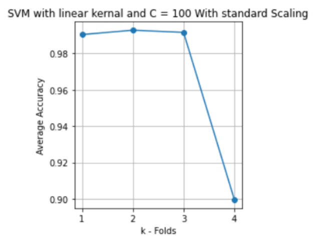
(a)



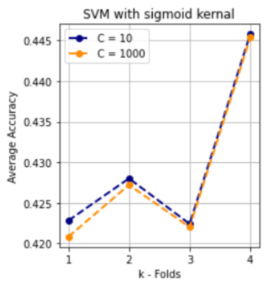
(b)



(c)



(d)

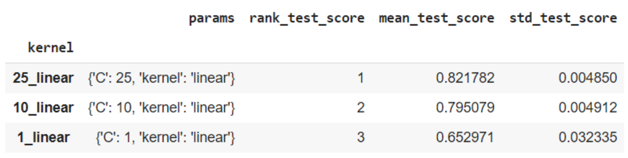


(e)

Figure 3: (a) Min Max Scaling followed by SVM Classifier with C=100 (b) Robust Scaling followed by SVM Classifier with C=200 (c) SVM with polynomial kernal with degree 8 (d) SVM with linear kernal and C = 100 With standard Scaling (e) SVM with sigmoid kernal

SVM tries to find the best hyperplane to separate the different classes by maximising the distance between sample points and the hyperplane.Parameter tuning is performed in order to achieve good test accuracy.In Scikit learn, GridSearchCV takes care of all the hard work of looping through all combinations of parameters and find the best average classification result by cross validation.From our experiment,it is found that non-linear kernel Radial Basis Function (RBF) tend to give better classification accuracy.Due to the limited resource and time, it is not possible to test all possible combinations of parameters.The experiment focuses on parameter C,which is the penalty parameter and and γ which is the regularization.The higher the C value,the more strict the rule to accept training points to be classified to a wrong label.

From all experiments performed in figure3,I have to decided to exclude the sigmoid and polynomial kernel from my parameter tuning of SVM classifier.
In figure 5 , we can see the performance of each intermediate model in this search,and the optimal C is 25.for the optimal linear svm , we can see the model performance on each fold in figure 4



kernel	params	rank_test_score	mean_test_score	std_test_score
25_linear	{'C': 25, 'kernel': 'linear'}	1	0.821782	0.004850
10_linear	{'C': 10, 'kernel': 'linear'}	2	0.795079	0.004912
1_linear	{'C': 1, 'kernel': 'linear'}	3	0.652971	0.032335

Figure 4: Intermediate models performance during parameter tuning for linear kernal

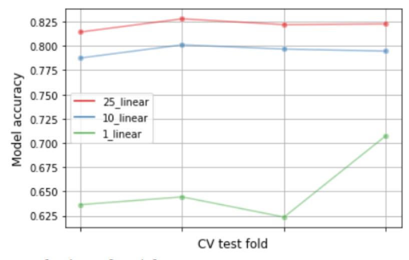
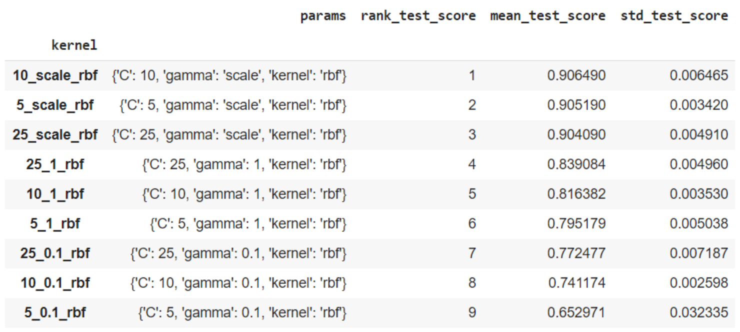


Figure 5: testing accuracy for each fold for intermediate models

Hence,linear kernel is best for a large number of features (> 1000) because it is more likely that the data is linearly separable in high dimensional space.
The following tuning process was for C and γ with *rbf* kernal.We see how much the performance improved and reach almost 90% accuracy,which makes sense because we have high dimension data that building a linear boundary between classes is very hard task.
γ:Kernel coefficient for 'rbf','poly' and 'sigmoid'.Higher the value of gamma,will try to exact fit the as per training data set i.e. generalization error and cause over-fitting problem.
C:Penalty parameter C of the error term.It also controls the trade-off between smooth decision boundaries and classifying the training points correctly.It worth mentioning that this processes takes almost 4 hours to be executed,which is very time consuming.



kernel	params	rank_test_score	mean_test_score	std_test_score
10_scale_rbf	{'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}	1	0.906490	0.006465
5_scale_rbf	{'C': 5, 'gamma': 'scale', 'kernel': 'rbf'}	2	0.905190	0.003420
25_scale_rbf	{'C': 25, 'gamma': 'scale', 'kernel': 'rbf'}	3	0.904090	0.004910
25_1_rbf	{'C': 25, 'gamma': 1, 'kernel': 'rbf'}	4	0.839084	0.004960
10_1_rbf	{'C': 10, 'gamma': 1, 'kernel': 'rbf'}	5	0.816382	0.003530
5_1_rbf	{'C': 5, 'gamma': 1, 'kernel': 'rbf'}	6	0.795179	0.005038
25_0.1_rbf	{'C': 25, 'gamma': 0.1, 'kernel': 'rbf'}	7	0.772477	0.007187
10_0.1_rbf	{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}	8	0.741174	0.002598
5_0.1_rbf	{'C': 5, 'gamma': 0.1, 'kernel': 'rbf'}	9	0.652971	0.032335

Figure 6: Intermediate models performance during parameter tuning for rbf kernal

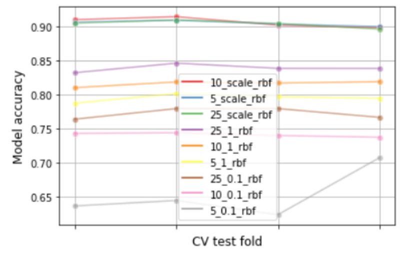


Figure 7: testing accuracy for each fold for intermediate models.

The optimal is Min Max Scalar followed by rbf classifier is with $C = 10$ and $\gamma =$ 'scale' as we see in figure 6 with Testing accuracy $= 0.9$.

The confusion matrix of this classifier is shown in figure 8

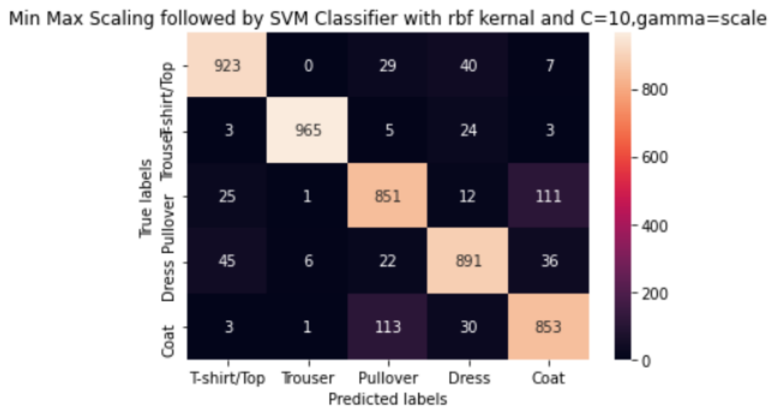


Figure 8: Confusion matrix for the SVM classifier with rbf kernel ,$C = 10$ and γ='scale'

3 Multilayer Perceptrons

The power of neural networks comes from their ability to learn the representation in your training data and how to best relate it to the output variable that you want to predict.In this sense neural networks learn a mapping. Mathematically, they are capable of learning any mapping function and have been proven to be a universal approximation algorithm.

The predictive capability of neural networks comes from the hierarchical or multi-layered structure of the networks.The data structure can pick out (learn to represent) features at different scales or resolutions and combine them into higher-order features.For example from lines,to collections of lines to shapes.

The building block for neural networks are artificial neurons.

These are simple computational units that have weighted input signals and produce an output signal using an activation function.See figure 9

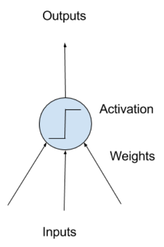


Figure 9: Model of simple neuron

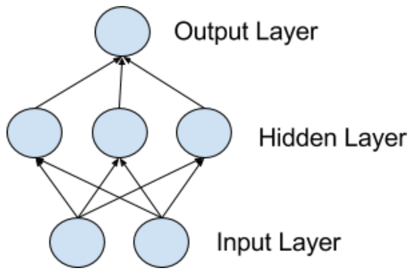


Figure 10: Model of simple network.

Like linear regression, each neuron also has a bias which can be thought of as an input that always has the value 1.0 and it too must be weighted.Weights are often initialized to small random values, such as values in the range 0 to 0.3,although more complex initialization schemes can be used.Like linear regression, larger weights indicate increased complexity and fragility. It is desirable to keep weights in the network small and regularization techniques can be used.

The weighted inputs are summed and passed through an activation function, sometimes called a transfer function.An activation function is a simple mapping of summed weighted input to the output of the neuron.It is called an activation function because it governs the threshold at which the neuron is activated and strength of the output signal.Historically simple step activation functions were used where if the summed input was above a threshold,for example 0.5, then the neuron would output a value of 1.0,otherwise it would output a 0.0. Traditionally non-linear activation functions are used.This allows the network to combine the inputs in more complex ways and in turn provide a richer capability in the functions they can model.Non-linear functions like the logistic also called the sigmoid function were used that output a value between 0 and 1 with an s-shaped distribution,and the hyperbolic tangent function also called tanh that outputs the same distribution over the range -1 to $+1$.

Neurons are arranged into networks of neurons.A row of neurons is called a layer and one network can have multiple layers.The architecture of the neurons in the network is often called the network topology as in figure 10

The performance of a simple MLP classifier for our data is shown for each kind of preprocessing we asked for is in figures 11 and 12

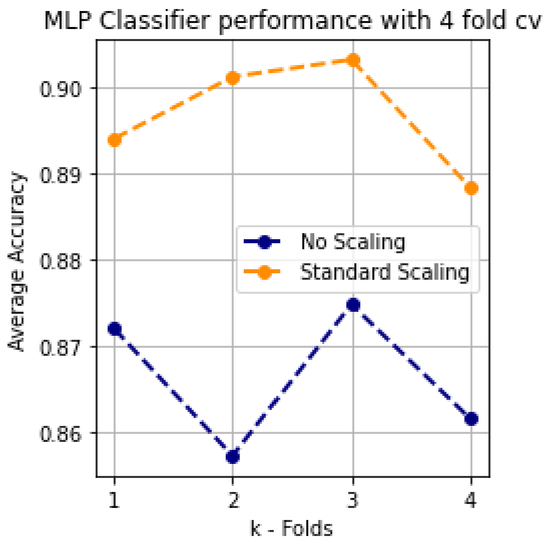


Figure 11: MLP classifier's accuracy for 4 fold cv with no scaling and proceeded by standard scalar

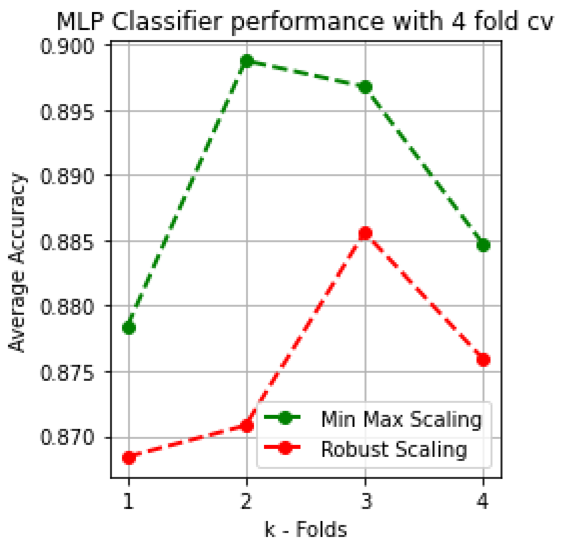


Figure 12: MLP classifier's accuracy for 4 fold cv proceeded by Min Max scalar and Robust scalar.

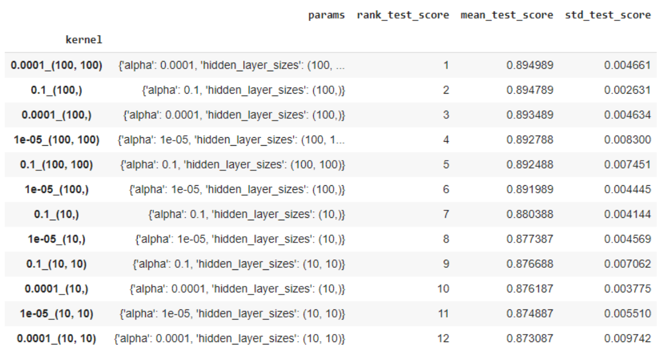
In table 2,we can see that the Min Max classifier get the best test accuracy among all the different kind of reprocessing we do,So I will proceed and make the Min max Scalar is the one used parameter Tuning and final training .

	No Scaling	Standard Scaling	Min Max Scaling	Robust Scaling
average training accuracy	0.94	0.99	0.99	0.98
Test accuracy	0.85	0.89	0.90	0.87

Table 2: Accuracy Scores for MLP Classifier with default parameter

Multi-layer Perceptron allows the automatic tuning of parameters.We will tune these using GridSearchCV.The tunable parameters for me in this homework are the number of hidden layers and α.

We now fit several models, the results are shown in figure , where we can see the highest rank model is for $\alpha = 0.0001$ and *hidden_layer_sizes (100,100).* and we can see in figure 13 the testing score of all the intermediate models (in the grid search) among each fold.



kernel	params	rank_test_score	mean_test_score	std_test_score
0.0001_(100, 100)	{'alpha': 0.0001, 'hidden_layer_sizes': (100, ...	1	0.894989	0.004661
0.1_(100,)	{'alpha': 0.1, 'hidden_layer_sizes': (100,)}	2	0.894789	0.002631
0.0001_(100,)	{'alpha': 0.0001, 'hidden_layer_sizes': (100,)}	3	0.893489	0.004634
1e-05_(100, 100)	{'alpha': 1e-05, 'hidden_layer_sizes': (100, 1...	4	0.892788	0.008300
0.1_(100, 100)	{'alpha': 0.1, 'hidden_layer_sizes': (100, 100)}	5	0.892488	0.007451
1e-05_(100,)	{'alpha': 1e-05, 'hidden_layer_sizes': (100,)}	6	0.891989	0.004445
0.1_(10,)	{'alpha': 0.1, 'hidden_layer_sizes': (10,)}	7	0.880388	0.004144
1e-05_(10,)	{'alpha': 1e-05, 'hidden_layer_sizes': (10,)}	8	0.877387	0.004569
0.1_(10, 10)	{'alpha': 0.1, 'hidden_layer_sizes': (10, 10)}	9	0.876688	0.007062
0.0001_(10,)	{'alpha': 0.0001, 'hidden_layer_sizes': (10,)}	10	0.876187	0.003775
1e-05_(10, 10)	{'alpha': 1e-05, 'hidden_layer_sizes': (10, 10)}	11	0.874887	0.005510
0.0001_(10, 10)	{'alpha': 0.0001, 'hidden_layer_sizes': (10, 10)}	12	0.873087	0.009742

Figure 13: Intermediate models performance during parameter tuning for MLP classifier

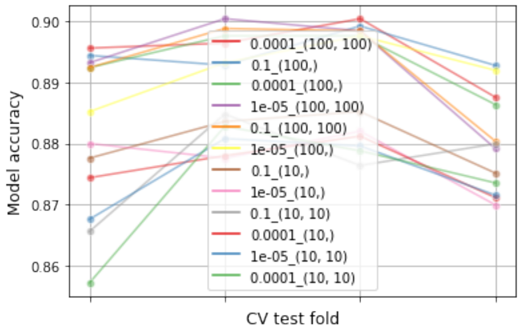


Figure 14: testing accuracy for each fold for intermediate models.

If the cross validation score values for a performance measure (say accuracy) do not vary significantly for various folds (k-folds) then we can say that the model is not overfitting. If the cross validation score values for a performance measure (say accuracy) are not very low for various folds (k-folds) then we can say that the model is not underfitting.

In summary, The optimal model is a Min Max Scalar followed by a MLP classifier with $\alpha = 0.0001$ and *hidden_layer_sizes (100,100)*.The confusion matrix is shown in figure 15

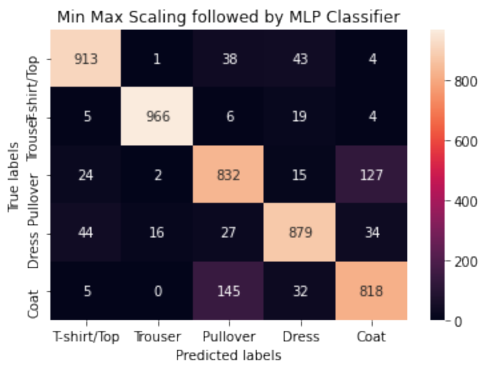


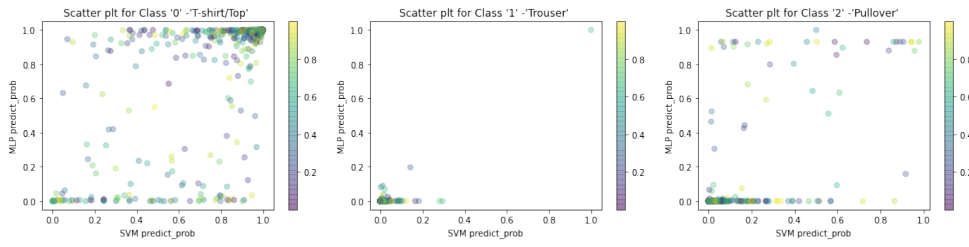
Figure 15: Confusion matrix for the MLP classifier with $\alpha = 0.0001$ and *hidden_layer_sizes (100,100)*

4 Results Comparison and Conclusion

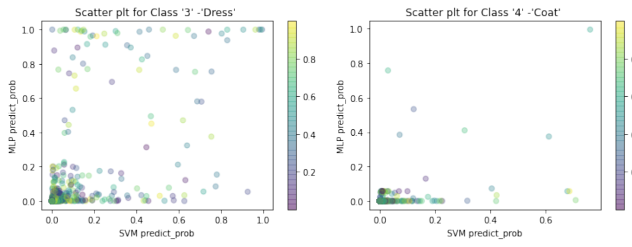
	SVM Classifier	MLP Classifier
Train accuracy	0.98	0.99
Test accuracy	0.9	0.88

Table 3: Summary of the various model performance on testing and training data

Displaying the scatter plot for each classifier predictions' help us to determine the distribution of true classified samples for both of them and the difference of classified samples.This scatter plots can be considered as a special kind of confusion matrix but instead of comparing the predicted output with the true label we compare two predicted value of 2 different classifiers.Ideally,the data has to be on the diagonal which is typically in the case of *class 1 : trousers* which makes much more sense. We take just the first 1000 samples of the test data , we can see both models prediction probabilities are very similar in figure 16(b),(c),(d) and (e)



(a) (b) (c)



(d) (e)

Figure 16: (a) Class '0' -'T-shirt/Top' (b) Class '1' -'Trouser' (c) Class '2' -'Pullover' (d) Class '3' -'Dress' (e) Class '4' -'Coat'

Class 0 performs comparatively very poorly for all models,with a great deal of false negatives and positives.Specifically against classes 4 and 5.

In the figures 17 and 18 we see the top 10 misclassified images for SVM and MLP classifiers respectively.

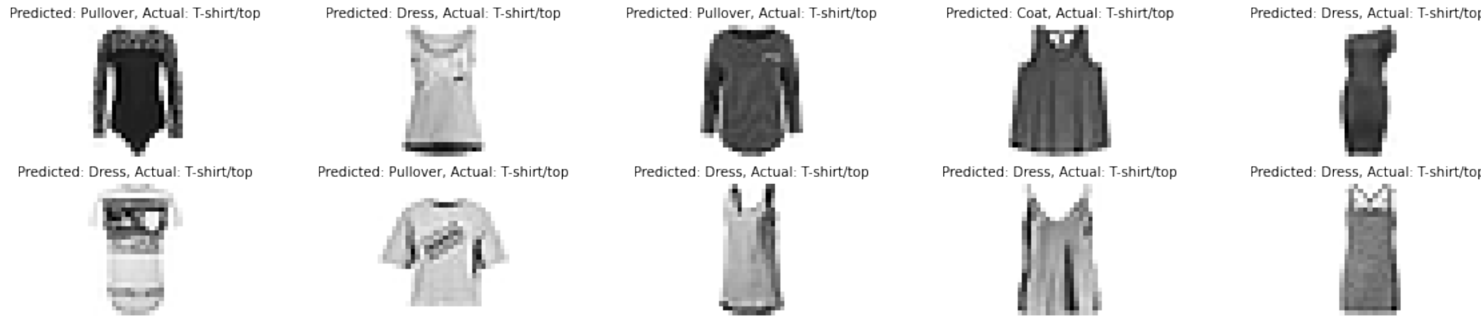

Predicted: Pullover, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Pullover, Actual: T-shirt/top
Predicted: Coat, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Pullover, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top

Figure 17: Some of wrong predictions for SVM classifier

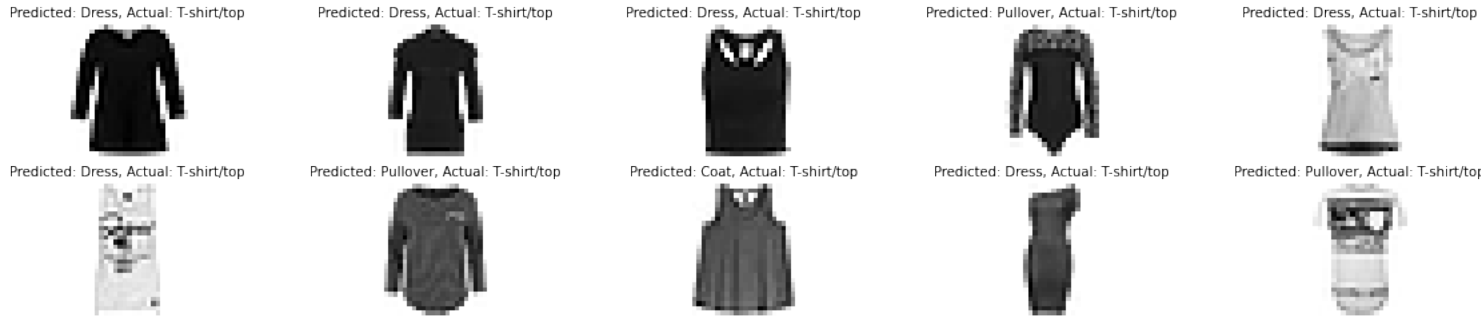

Predicted: Dress, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Pullover, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Pullover, Actual: T-shirt/top
Predicted: Coat, Actual: T-shirt/top
Predicted: Dress, Actual: T-shirt/top
Predicted: Pullover, Actual: T-shirt/top

Figure 18: Some of wrong predictions for MLP classifier

We can see that there are some common wrong predictions between both classifiers.We should note that the MLP classifier may overfit the data since it has a training accuracy of nearly 100%.

Predicted: T-shirt/top, Actual: T-shirt/top
Predicted: T-shirt/top, Actual: T-shirt/top
Predicted: T-shirt/top, Actual: T-shirt/top
Predicted: T-shirt/top, Actual: T-shirt/top
Predicted: T-shirt/top, Actual: T-shirt/top

Figure 19: Some of correct predictions of SVM classifier

In figures 19 and 20 how the models agrees in classifying this images correctly.We can see that the top five images on my list is classified correctly for both models.

Predicted: T-shirt/top, Actual: T-shirt/top
Predicted: T-shirt/top, Actual: T-shirt/top
Predicted: T-shirt/top, Actual: T-shirt/top
Predicted: T-shirt/top, Actual: T-shirt/top
Predicted: T-shirt/top, Actual: T-shirt/top

Figure 20: some of correct predictions of MLP classifier.

One of the samples that are classified wrong using both classifiers is in figure 21 as we see that the top was classified as a Pullover from both classifiers which doesn't match the true label which is T-shirt/top.

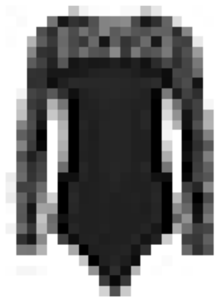


Figure 21: sample of images that is classified wrong by both classifiers.

On the other hand,in figure 22 , we see that,The SVM predicted this image as a Pullover while it is true label is Coat, this was predicted correctly by the MLP classifier.

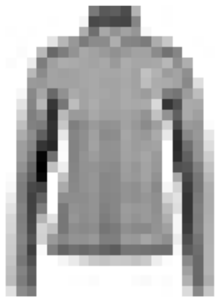


Figure 22: sample of images that is classified wrong using SVM.

Finally, here in figure 23,the image was predicted correctly using SVM but wrong with the MLP classifier,as it has a true label T-shirt/top but the predicted value using MLP is Dress.

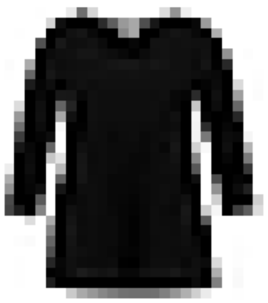


Figure 23: sample of images that is classified wrong using MLP.

In perspective of training time,between the two classifiers and looking at a measure of training time alone,the SVM outperformed by the MLP classifier.