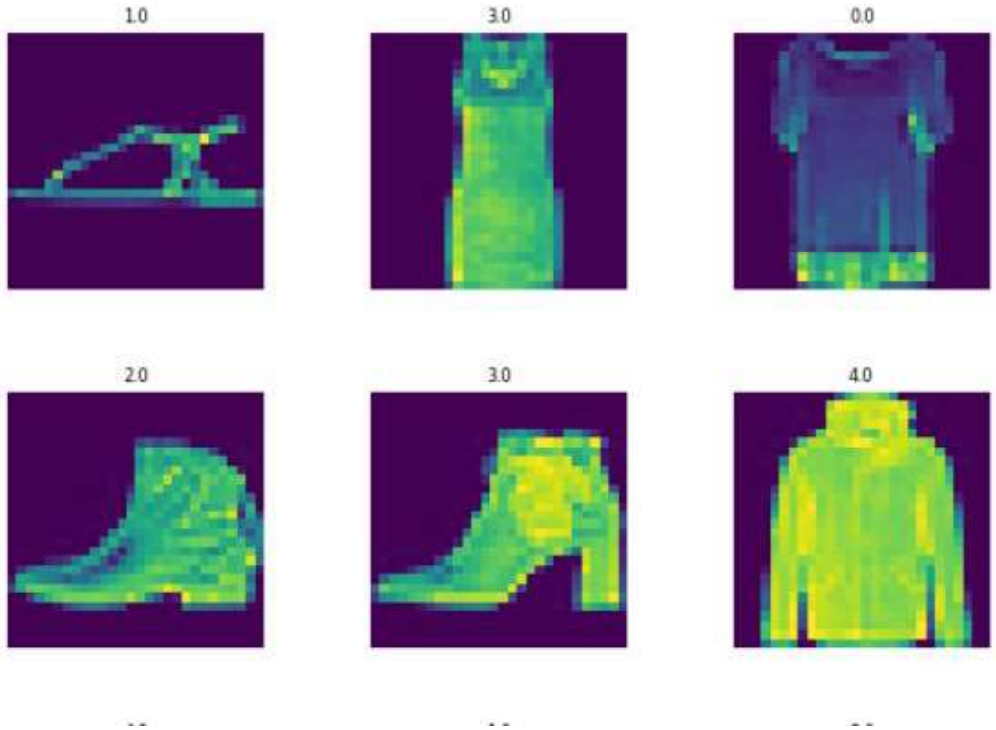
Question 1: Classification: Feature Extraction + Classical Methods (40 points)

1.1: Explanation of Design and Implementation Choices of your Model (15 points)
Describe the underlying algorithms you are using, how they work and why you chose them.

In order to design and implement the ML algorithm to solve image classification problems, it is important to look at the dataset first. The dataset and corresponding features are explained below:

Understanding the Dataset:
The assignment dataset contains Fashion MNIST (twisted) images with 5 (0,1,2,3 and 4) different categories. A sample image from the dataset would have 784 features (i.e. 28x28 pixels). Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers corresponding to darker color. This pixel-value is an integer between 0 and 255.The pixels converted into images are shown below:



Looking at the above images, it can be said that a nonlinear Machine learning algorithm would be better at classifying these images to the corresponding categories compared to the linear ML algorithms. Furthermore, not all of the pixels (i.e. features) of a given dataset might be necessary to classify the images (The pixels at the boundary of images seems less important, compared to features in the middle regions). In addition to this, use of fewer transformed independent features will be computationally economical for application of ML classification algorithm.These dataset properties provide an intuition for application of feature extraction techniques such as Principal component analysis (PCA), along with use of non-linear classical ML algorithms such as support vector machines (SVM). The working of PCA and SVM are explained below:

PCA:
For dimensionality reduction and feature selection, principal component analysis (PCA) is used. Principal Component Analysis (PCA) is a popular dimensionality reduction technique used in Machine Learning applications. PCA condenses information from a large set of variables into fewer variables by transforming these components into independent components. The goal of the PCA is to obtain independent components to reduce the dimensionality of feature space. The principal components can be obtained by

the eigendecomposition of the covariance matrix of our data. The dimensionality is then reduced by projecting the data onto the largest eigenvectors. PCA gives the orthogonal features explaining maximum variance. These extracted features can be used to solve the classification problem.

There are several classification machine learning algorithms which can be used for solving problems which exhibit nonlinear relationships among sample features. The methods such as K-Nearest Neighbours, Tree-Based algorithms support vector machines (SVM) and neural network based approaches have been implemented extensively for solving such problems. Classical Support vector machines based [1] approach is one of most popular methods that can deal with non-linear features of data and hence is used in this assignment to solve the problem of image classification.

SVM:

SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It creates a decision boundary to separate data points of different classes. Linear SVM is used to separate classes using linear boundaries whereas Non-linear SVM uses nonlinear boundaries.We have used Non-linear SVM(RBF based) i.e. non-linear kernel to classify image data. Non-linear SVM means that the boundary that the algorithm calculates does not have to be a straight line. The benefit is that it can capture much more complex relationships between data points.

1.2: Implementation of your Design Choices (5 points)

Show some of the important code blocks to implement your model. We will also consult your full code on LEARN, so this is your chance to guide us to understand your code and how you achieved your result.

1. **Splitting the data**

```
#Spliting the data into training and validation
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size= 0.1, random_state=42)
```

The complete training dataset is splitted in the two parts i.e. Training dataset (90%) and validation (10%) dataset. The performance on validation set is used to compare results of various algorithms and architecture used.

2. **Normalization of data**

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler(with_std=False)
X_train =scalar.fit_transform(X_train)
X_val = scalar.transform(X_val)
```

Features are centered before being fed to the PCA by using the “StandardScalar” function. The function is fitted on the training dataset (training dataset is transformed as well) and is used to transform validation dataset.

3. **Feature Extraction**

```
#Feature Extraction
from sklearn.decomposition import PCA
pca = PCA()
X_r = pca.fit(X_train).transform(X_train)
sum_cum = np.cumsum(pca.explained_variance_ratio_)
```

PCA is used to fit and transform training data into independent components, PCA explained ratio is used to understand the total variance explained by PCA components.

4. n_components selection

```
for k in range(0,784):
  if sum_cum[k] >= 0.9:
    h = k
    break
  else:
    continue
h
```

```
83
```

Above loop is used to gather the number of PCA components required to explain 90%(fixed threshold) of total variance explained by all independent PCA features.

5. Data Transformation

```
# PCA n_components = 85
pca_85 = PCA(n_components= 85)
X_train_85 = pca_85.fit(X_train).transform(X_train)
X_val_85 = pca_85.transform(X_val)
```

90% of the total variance explained by the 83 numbers of PCA components. However, the best accuracy of the SVM on validation set is achieved by the 85 number of PCA transformed (90.15% of the total variance explained). Therefore, Training and validation data is transformed into 85 PCA components.

6. Classifier

```
def SVC_C_85(C):
  model = SVC(C)
  start = time.time()
  model.fit(X_train_85,y_train)
  end = time.time()
  start_1 = time.time()
  y_pred = model.predict (X_val_85)
  end_1 = time.time()
  y_pred_training = model.predict(X_train_85)
  accuracy = accuracy_score(y_val, y_pred)
  accuracy_training = accuracy_score(y_train, y_pred_training)
  print("Validation Accuracy : %f" % accuracy)
  print("Training Accuracy : %f" % accuracy_training)
  precision = precision_score(y_val, y_pred, average = 'micro')
  print('Precision: %f' % precision)
  recall = recall_score(y_val, y_pred, average = 'micro')
  print('Recall: %f' % recall)
  f1 = f1_score(y_val, y_pred, average = 'micro')
  print('F1 score: %f' % f1)
  print('Fitting Time: %f' % (end - start))
  print('Testing Time: %f' % (end_1 - start_1))
  ROC(y_val, y_pred)
```

SVM classifiers with different C values are trained with transformed training data and its performance is predicted on validation data. Various metrics such as precision, recall, F1 score and ROC curve are also explored to understand the performance of this kernel based SVM classifier on validation dataset.

1.3: Kaggle Competition Score (5 points)

Report the highest score your submissions received on Kaggle. If you have any explanations for varying performance by different teams explain it here.

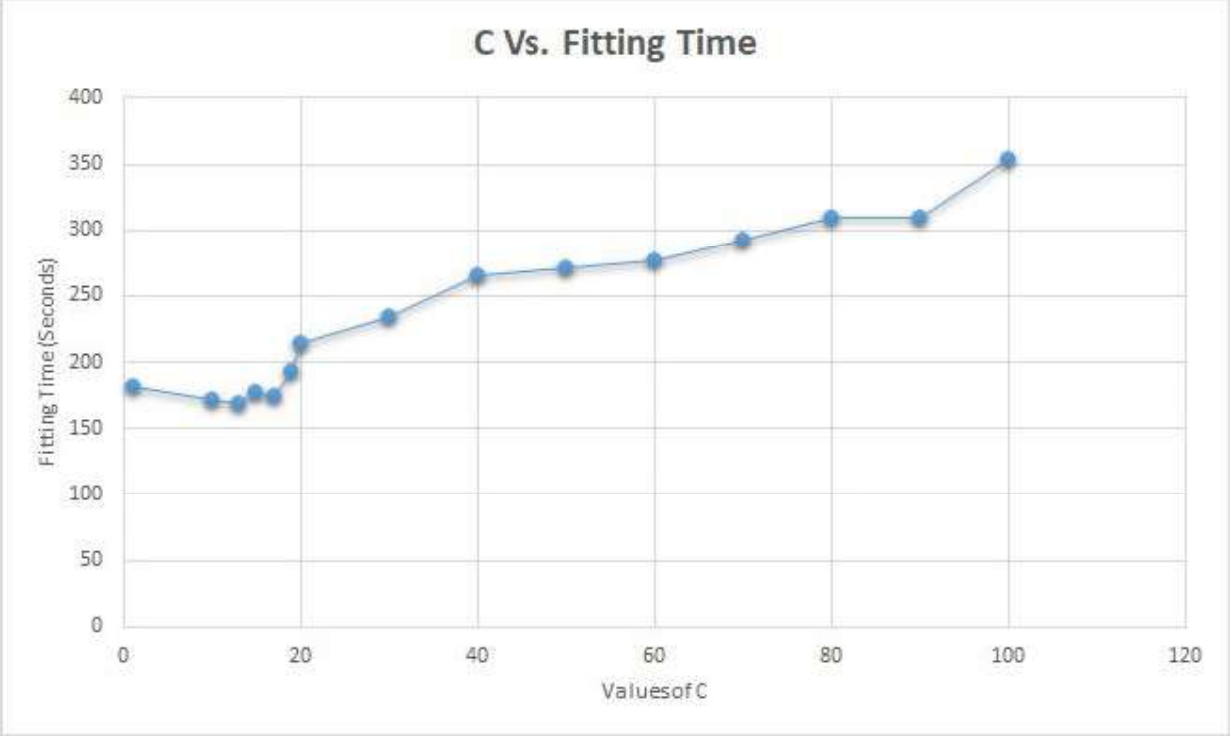
89.78% Accuracy is achieved on the leaderboard implementing the PCA as feature extraction technique and kernel SVM as the classifier. The model is trained on the complete training data set provided to achieve this accuracy.

There are various parameters that can be changed to improve accuracy scores like extracted number of features from PCA, value of C, gamma in SVM. Due to limited access to the computational power, we could not implement the algorithm for n_components more than 100. These hyperparameters of SVM can further be tuned for different numbers of n_components.

1.4: Results Analysis (15 points)

• Runtime performance for training and testing.

The below plot shows the time taken by the kernel SVM model to fit the training data. Time required to fit the training data with hyperparameter value less than 20 is small and almost equal between 11 to 17. With further increase in C results in longer time to fit the training data. Time taken to fit the best SVM model (i.e. with n_components = 85, C = 15) on the training dataset is approximately 177.76 seconds and time taken to test the validation data is approximately 16.06 seconds. The training time and testing time is dependent on RAM and CPU core availability and varies dynamically as system configuration is made available by google collaboratory at the given time.



• Comparison of the different algorithms and parameters you tried.

For this classification problem, we have tried K-Nearest Neighbour (KNN), Tree-based algorithms like Random forest and support vector machines (SVM). All these algorithms are trained and validated for different hyperparameters. The below table shows the best performance of each algorithm with their run time.

Algorithm	Training Time (seconds)	Testing Time (seconds)	Validation Accuracy (%)
KNN	0.75	9.92	86.2
Random Forest	98.15	0.187	81.56
SVM	177.76	16.06	89.78
Logistic Regression	5.41	0.001	66.71

KNN: KNN classifier model was fitted for the various values of the k from 1 to 100 in to get optimal k (number of neighbours) value corresponding to the maximum validation accuracy. The validation maximum accuracy obtained is 86.2% with k = 9. Moreover, the training time of the K-nearest neighbor is minimum with intermediate testing time compared to SVM and random forest.

Random Forest: Random forest model is fitted for the various values of max depth and number of estimators. In order to check the accuracy, the range of max depth is kept from 3 to 50 and the number of estimators are kept as 50, 100, 150, 200, 250. The best validation set accuracy of 81.56% achieved for max_depth = 11 and number of estimators = 200. Further increase in max_depth and number of estimators resulted in decreased accuracy. In contrast with the KNN, random forest takes intermediate time to train and minimum time to predict the validation data.

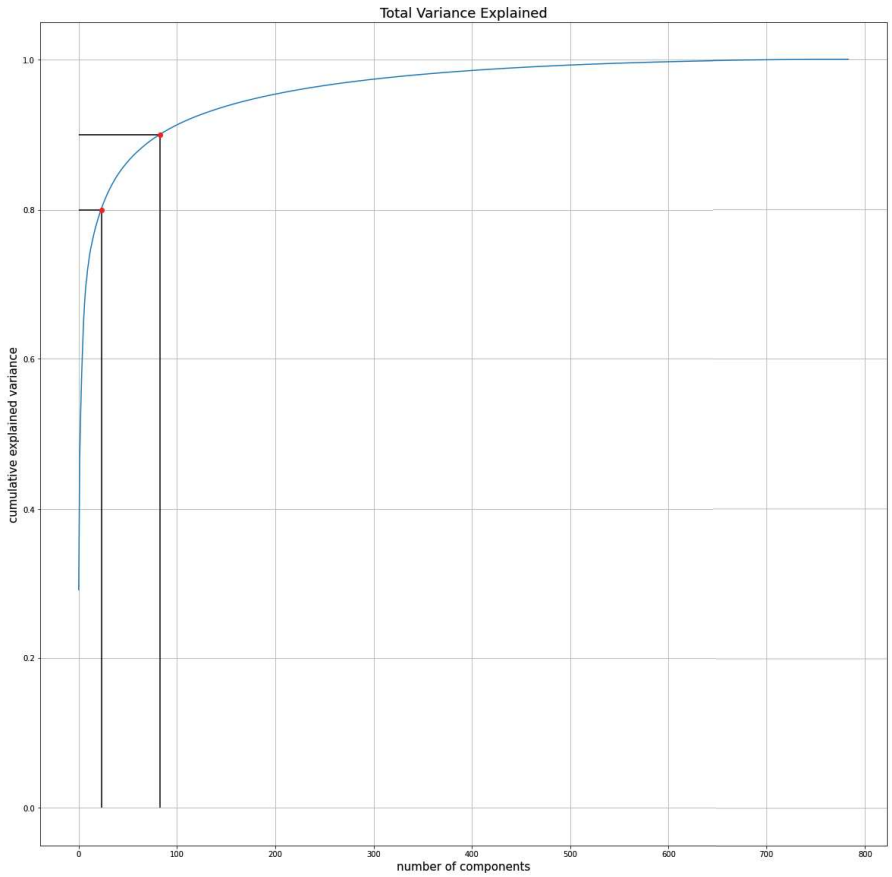
SVM: SVM fitted various values of C and gamma for different types of kernels. Linear, polynomial, sigmoid and rbf kernels are used for the various runs, among these kernels performance of radial basis kernel (RBF) has been the best. C values from 1 to 100 have been used. The best accuracy achieved using SVM is 89.78 which is better than KNN and random forest. However, training and validation time of SVM is higher than the other two algorithms.

Logistic Regression: The performance of logistic regression classifiers on this non-linear dataset is poor compared to the rest of the SVM, random forest and KNN. However, time required to predict the data is less.

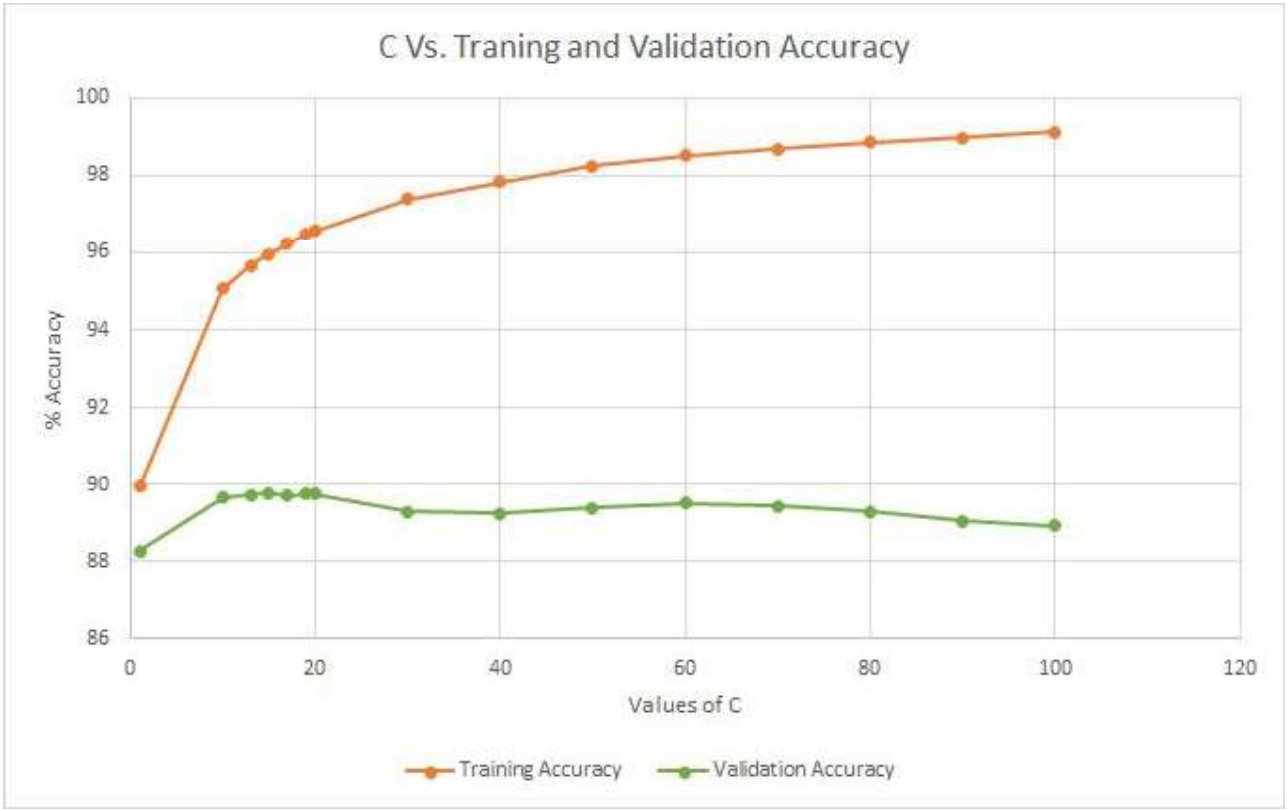
- **Explanation of your model (algorithms, design choices, numbers of parameters)**

The model used for the fashion MNIST dataset classification problem comprises the feature extraction technique and the supervised machine learning classifier. As described in the above question 1.1, the PCA is used for feature extraction and the features are classified by the Support vector machine.

Number of PCA components (i.e. Transformed orthogonal components) and C value for SVM are both hyperparameters for our approach. Usually the number of PCA components is decided using knee point(from scree plot) or the total variance explained (80%, 90% or 95% threshold value). We used 90% of the total variance explained to select the number of components. The below plot shows the cumulative variance explained by the number of components.



Knee in scree plot is found at 3 principal components, however 80% and 90% variance explanation is achieved at 23 and 83 PCA components.The below plot shows that validation accuracy of the SVM classifier for different values of C.



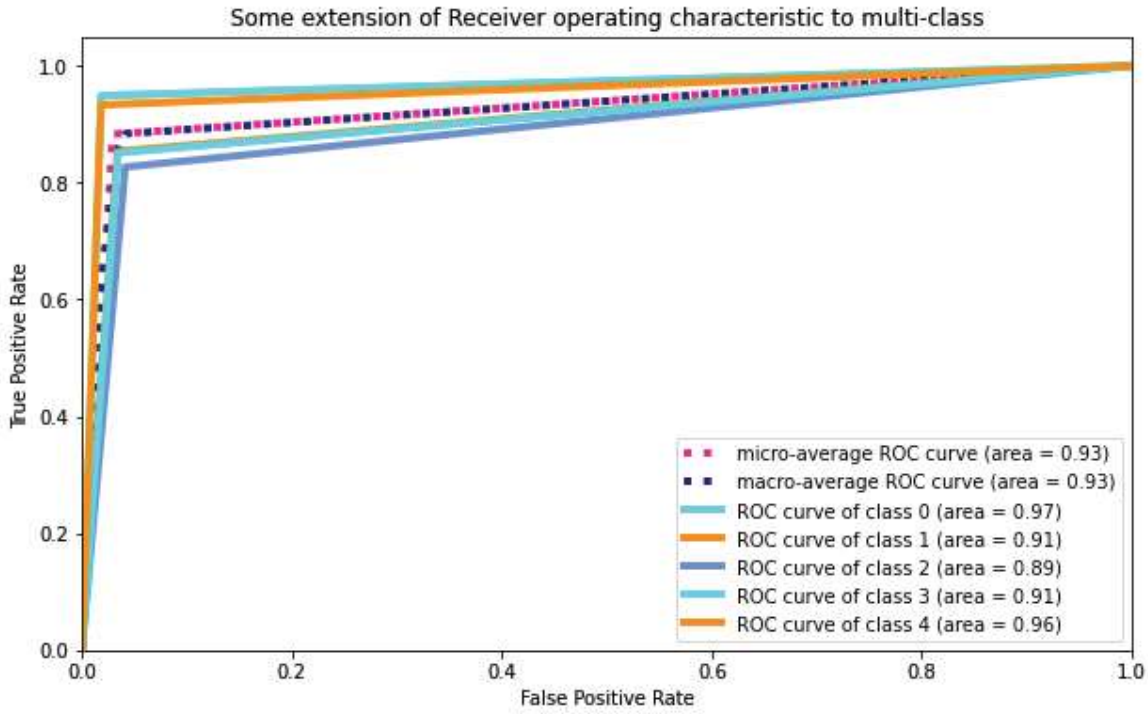
Looking at the above plot it can be said that with increase in the value of C the training accuracy increases however, The accuracy of the SVM on the validation dataset increases with increase in the value of C till 15. With further increase in the value of C, the validation accuracy decreases indicating the over fitting of the model. The below plot shows the fitting time of the SVM model for different values of C. The best SVM accuracy on validation data with similar C values were achieved at 85 PCA components (validation accuracy = 89.78% at C=15 , kernel = rbf).

SVM can be used with different kernels such as linear, RBF, polynomial and sigmoid kernels. Our choice of kernel is RBF as even if the features have non-linear relation, it transforms the data to the linear dimension and tries to find the linear boundary among it whereas linear kernel tries to find the linear relation among the data the non-linear data. Moreover, the training time of SVM with linear kernel is larger than with the rbf kernel.

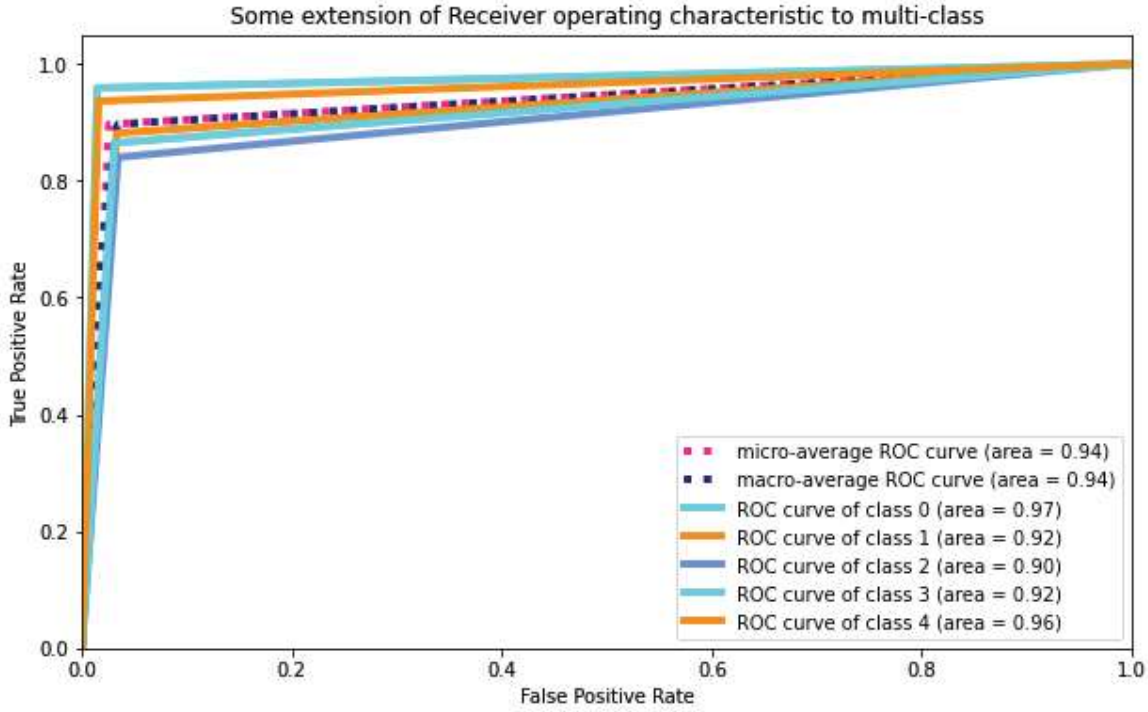
- **Use a ROC curve used for some method in your initial or results analysis such as exploring the impact on accuracy of some parameter.**

The effect of different values of hyperparameter C on a support vector machine can be visualized using the receiver operating characteristic graph. The below images show the ROC curve for different C values.

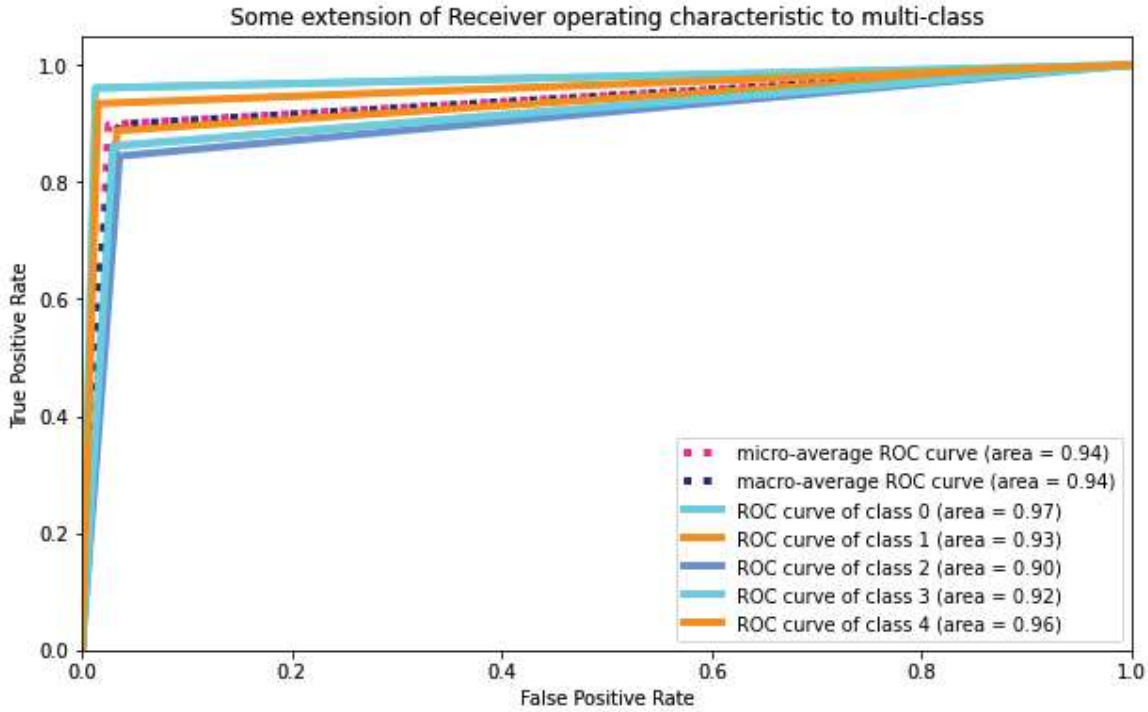
ROC Curve for C = 1 [2]



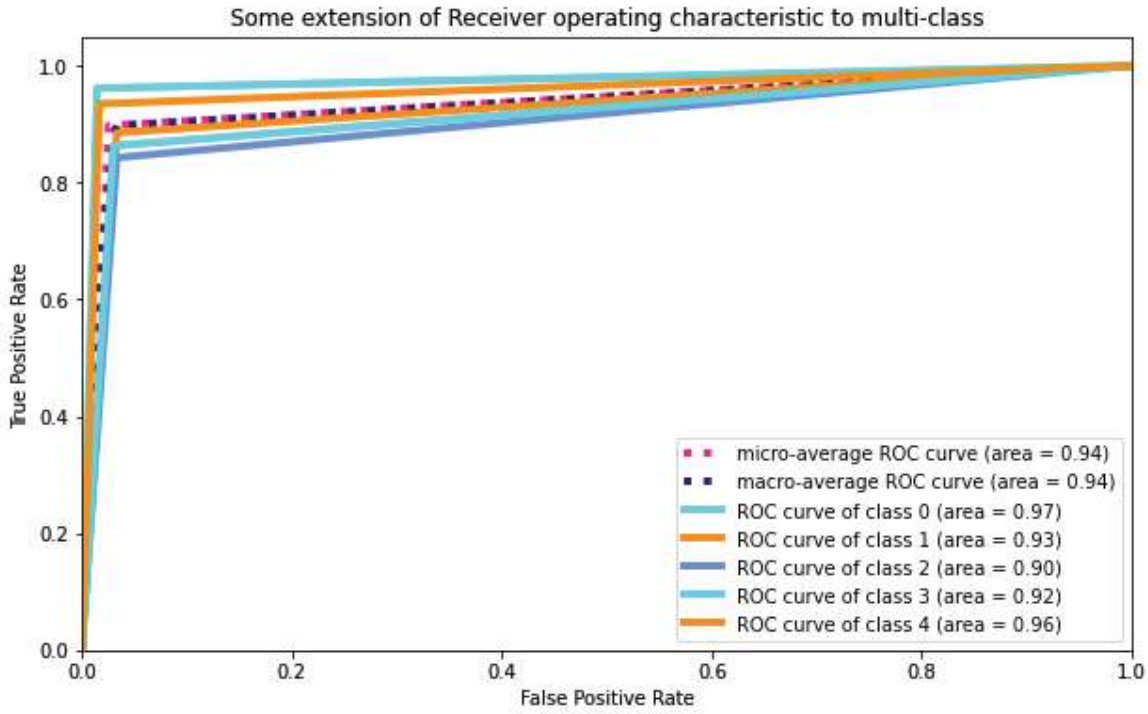
ROC Curve for C = 10



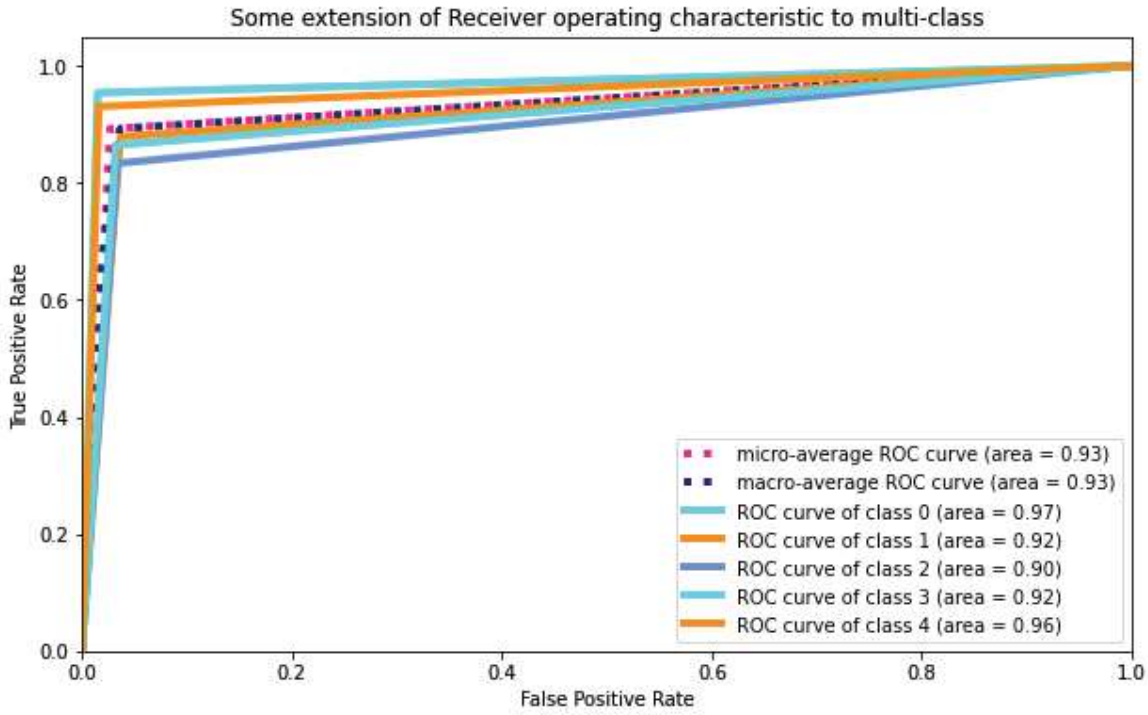
ROC Curve for C = 15



ROC Curve for C = 20



ROC Curve for C = 30



Looking at the above plots following observations can be made,

1. Increasing the value of C from 1 to 15, area under the curve for class 1, 2 and 3 increases slightly while for remaining class 0 and 4 it remains constant.

2. For C = 15, SVM has the minimum misclassification for all the five classes of fashion MNIST data compared to the other values of C (1,10 and 30). While for C = 20, the area under the curve for all the classes is the same as the value for C = 15.
3. For further increased value of C (i.e. 30), area under the curve for the micro and macro averaged method is slightly reduced.
4. All of the above curves are overlapped and close to the y-axis representing high true positive rate.

• **Evaluate your code with other metrics on the training data (by using some of it as test data) and argue for the benefit of your approach.**

The model can be evaluated on the different accuracy matrices such as accuracy, F1-score, ROC AUC etc. Values for different accuracy matrices are shown in the table below.

Accuracy	89.78%
Precision	89.78%
Recall	89.78%
F1-Score	89.78%
ROC AUC	0.94

To understand the performance of SVM on validation better, various metrics such as Accuracy, precision, recall, F1-Score and Area under the AUC are used as well. The precision value tells about the sensitivity of the model whereas recall value explains true positive rate. F1-score shows the balance between precision and recall.Similar high values for precision (89.78%) and recall (89.78%) on validation data confirms the model's excellent ability to distinguish between different classes (true positives, false positive and false negative values are taken into consideration to calculate precision, recall and F1 score) classification. High value of F1 score shows the model's balance between precision and recall ability. Furthermore high value (.94, i.e. close to 1) of area under the ROC (Receiver Operating Curve) shows high true positive rate in the model's ability to classify images into respective categories.

References:

1. https://community.alteryx.com/t5/Data-Science-Blog/Why-use-SVM/ba-p/138440 [Accessed on: April, 20,2020]
2. https://stackabuse.com/understanding-roc-curves-with-python/ [Accessed on: April, 21,2020]