Clustering and Classification on Twisted Fashion MNIST dataset

Objectives Achieved:

Twisted Fashion MNIST Dataset

The label was originally an assigned clothing type represented by an integer from 0-9. The training

set contains 60,000 examples and the test set 10,000 examples. For this project, we are providing you the same input features, but the labels you will be using will be a new mystery label. We have created a new label based on the data and calculated to be associated with each entry. The y train and y test files provide this modified label which is a category numbered 0-4. What does the number mean? That's for you to figure out through your analysis!

• x train.csv - the training set for your model. The training file contains vectors of size 784 representing pixel values of a 28x28 image.

• y train.csv - is the "mystery label", a numeric target obtained using our formulation.

• x test.csv - are the features for the test set to use for final evaluation.

• y test.csv - the "mystery label" for the test data.

My Analysis and Results:

- Pre-Processing of Data:
    - The preprocessing code represents a data preprocessing function that takes in a raw data set of images and returns the processed data in a specific format
    - Reshaping: The function reshapes the raw data from a 1D array to a 4D array of shape (num_images, 28, 28, 1). This is because the input data is assumed to be a set of grayscale images with dimensions 28x28. In order to process this data using certain machine learning algorithms, such as convolutional neural networks (CNNs), we need to reshape the data into a 4D tensor that represents the images in the correct format.
    - Normalization:The function also normalizes the pixel values of the images by dividing each value by 255. This is because pixel values in grayscale images typically range from 0 to 255, and normalizing the values to a range of 0 to 1 can help improve the performance of certain machine learning algorithms.
    - Compatibility: The function processes the raw data into a format that is compatible with certain machine learning frameworks and libraries, such as TensorFlow and Keras. These libraries typically expect image data to be in the format (num_images, height, width, channels), where channels refers to the color channels of the image (e.g. 1 for grayscale, 3 for RGB).

Overall, the data preprocessing function helps to ensure that the input data is in the correct format and normalized appropriately for use in certain machine learning algorithms, and that it is compatible with specific libraries and frameworks.

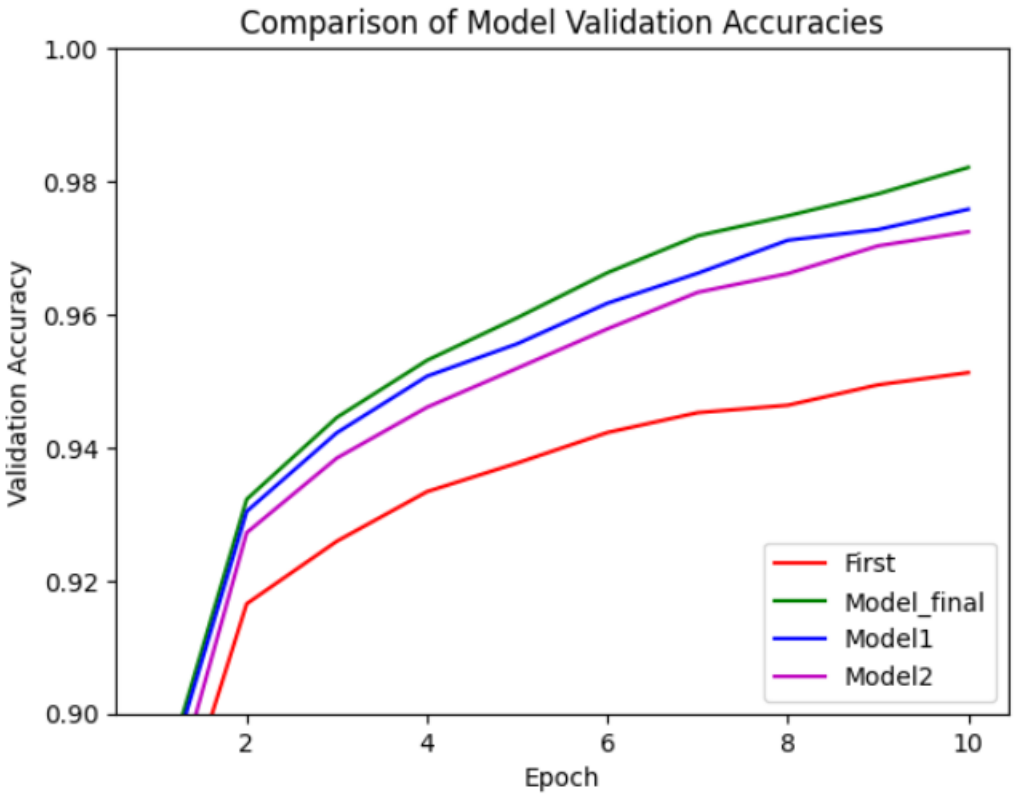
- Classify the data using a Convolutional Neural Network with the following architecture:
    - input features x
    - convolutional layer with 32, 3x3 filters, stride 1, padding 1x1
    - max pooling layer 2,2
    - convolutional layer with 32, 3x3 filters, stride 1, padding 1x1
    - flatten the output
    - output layer: one fully connected layer with five output values
    - a softmax layer to transform the outputs into a multi-class probability distribution for classification
    - activation functions: internal layers all use ReLU activation
    - optimizer : Stochastic Gradient Descent

Accuracy of 92.26% and total loss of 21.87% achieved

- My own designed architecture:
    - input features x
    - activation function:Leaky ReLu
    - max pooling layer 2,2
    - convolutional layer with 32, 3x3 filters, stride 1, padding 1x1
    - max pooling layer 2,2
    - convolutional layer with 32, 3x3 filters, stride 1, padding 1x1
    - max pooling layer 2,2
    - convolutional layer with 32, 3x3 filters, stride 1, padding 1x1
    - activation function:Leaky ReLu
    - max pooling layer 2,2
    - flatten the output
    - Dense layer
    - activation function:Leaky ReLu
    - Dense Layer
    - optimizer : Stochastic Gradient Descent

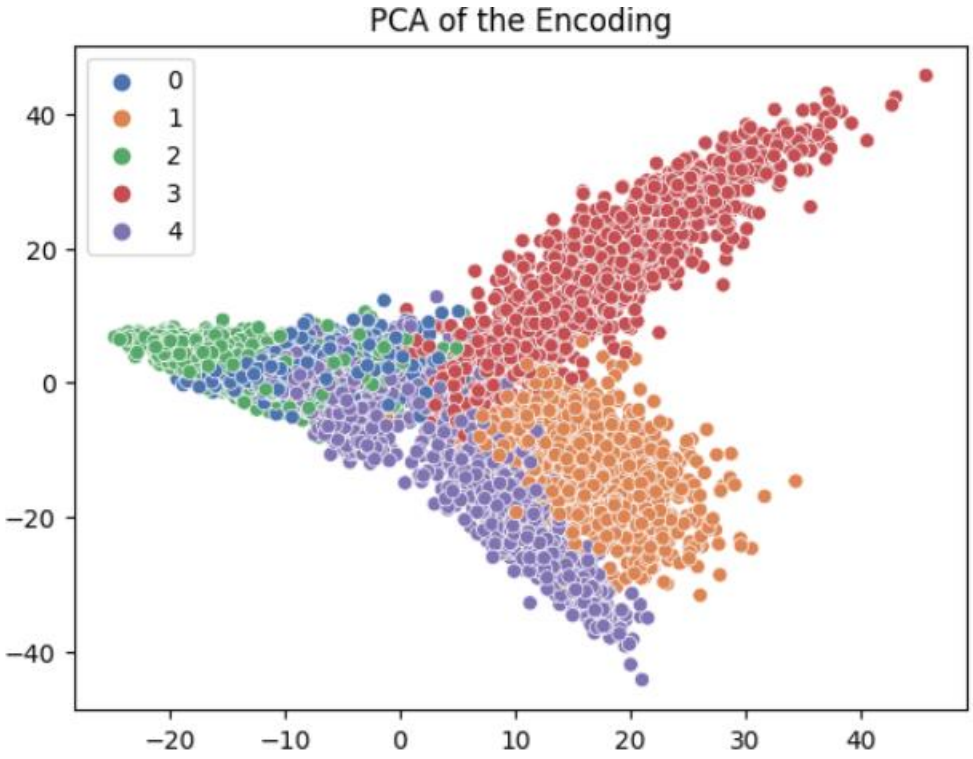
Accuracy loss: 22.35% - accuracy: 94.07% achieved.

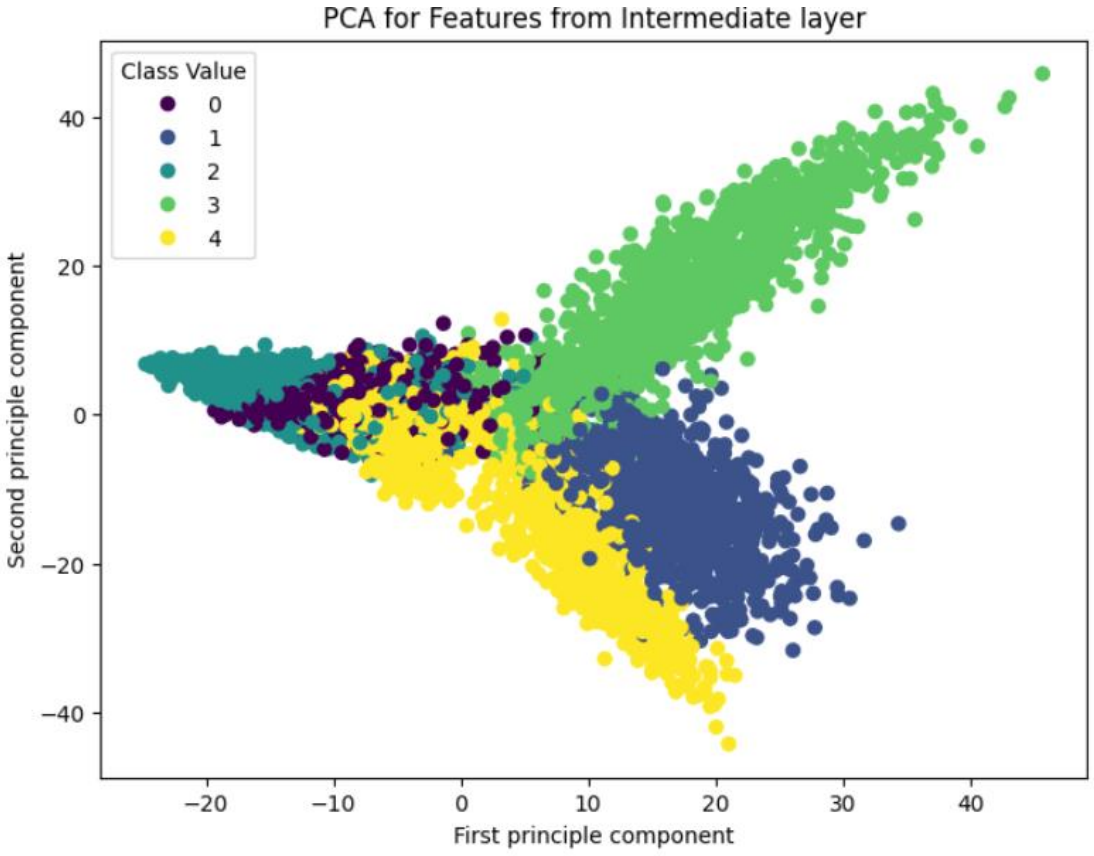
I designed 3 models similar to this and below is the accuracy graph.

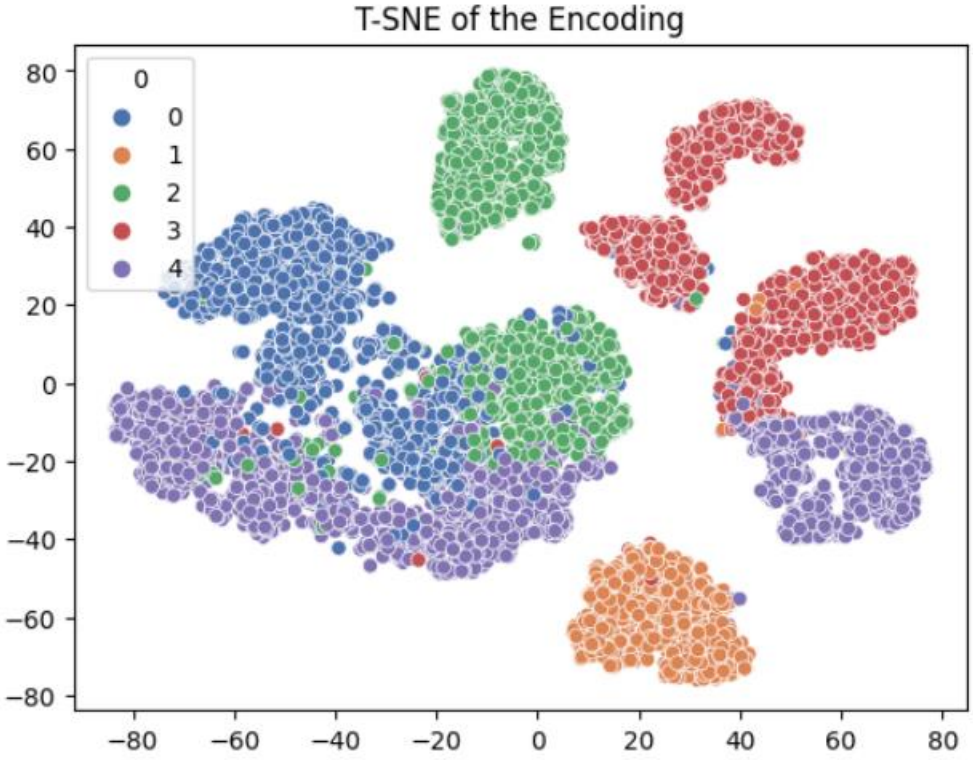


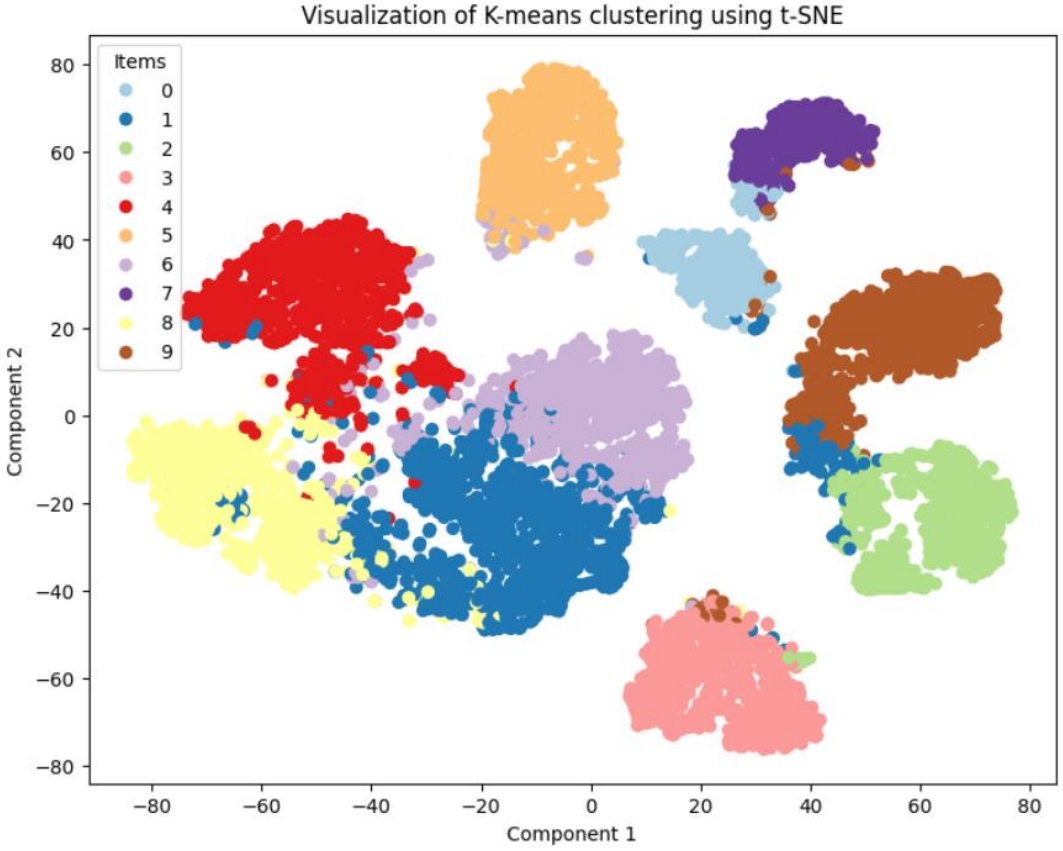
Own Encoding

Define a simple intermediate layer model using one of the later internal layers from your trained network (ie. anything before softmax is possible, but some will work better than others). This is your encoding. Then you will treat the elements of this encoding as features in a dataset for clustering and visualization to help you understand what the mystery label for our dataset might mean.

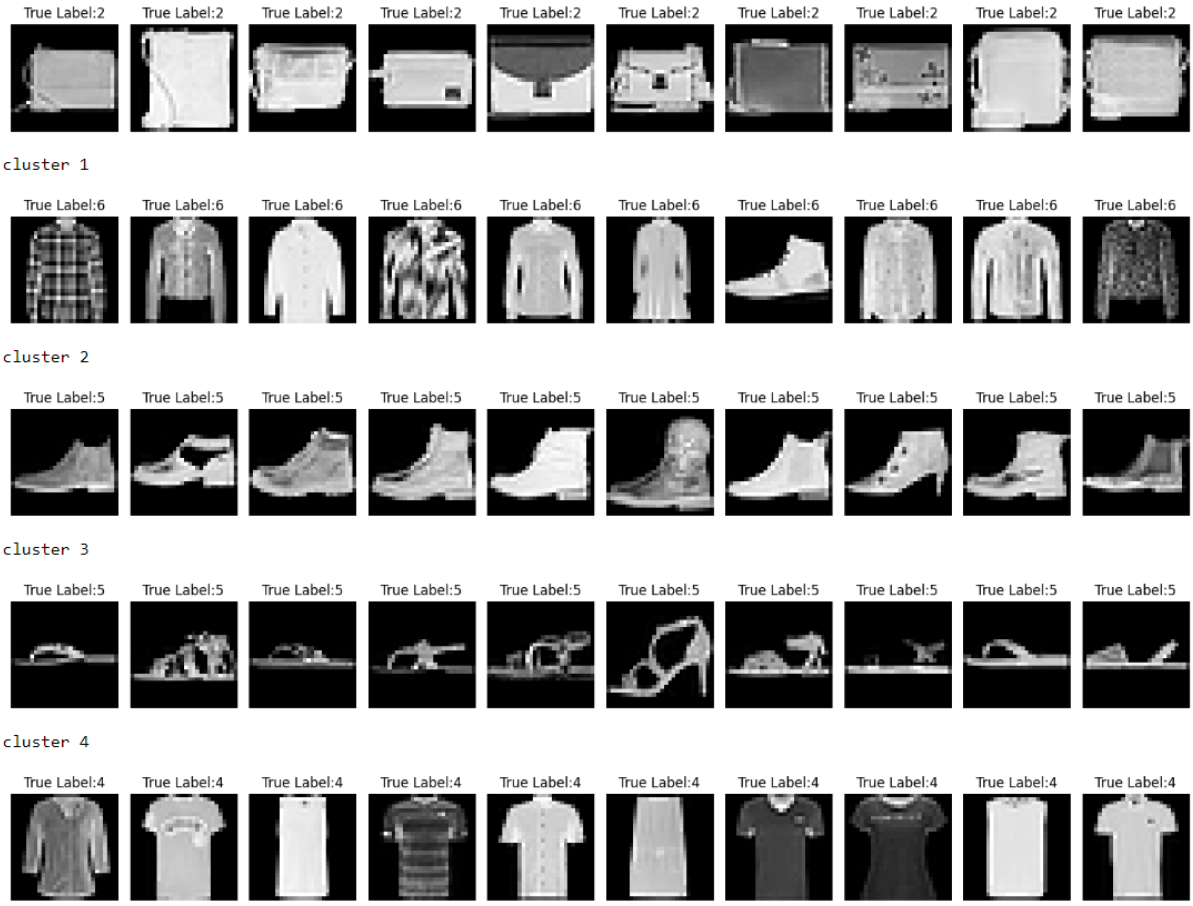

PCA of the Encoding
0
1
2
3
4
40
20
0
−20
−40
−20
−10
0
10
20
30
40


PCA for Features from Intermediate layer
Class Value
0
1
2
3
4
Second principle component
40
20
0
−20
−40
−20
−10
0
10
20
30
40
First principle component

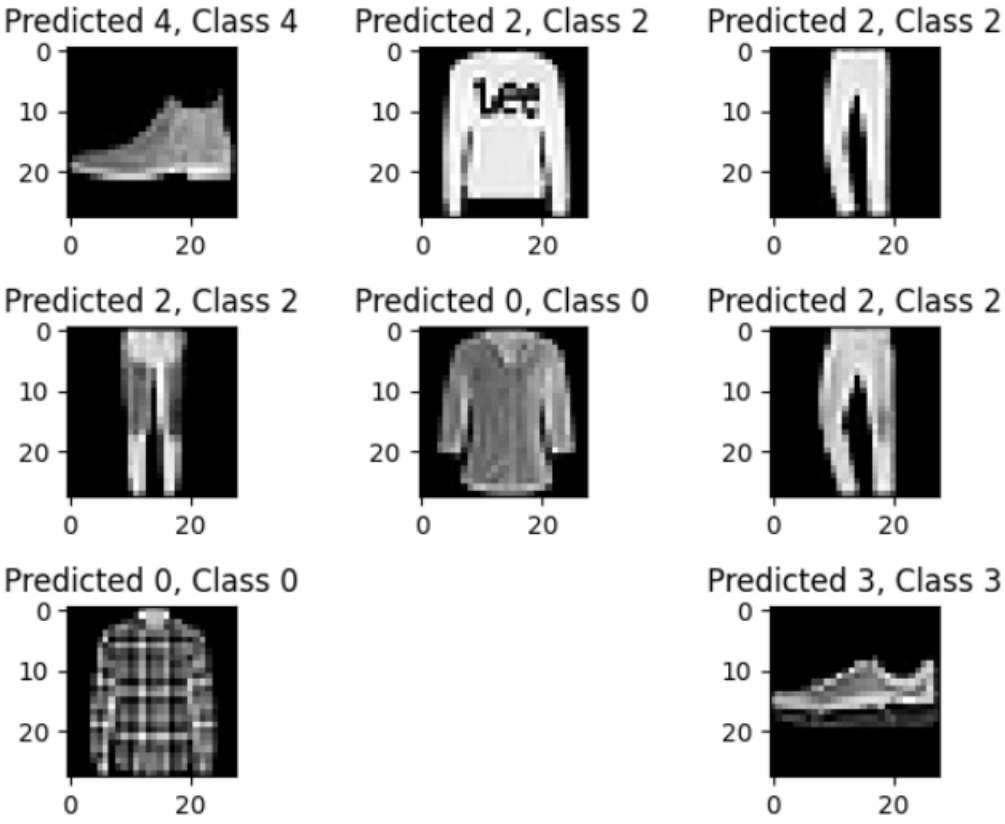

T-SNE of the Encoding
0
0
1
2
3
4
80
60
40
20
0
−20
−40
−60
−80
−80
−60
−40
−20
0
20
40
60
80


Visualization of K-means clustering using t-SNE
Items
0
1
2
3
4
5
6
7
8
9
Component 2
Component 1
80
60
40
20
0
−20
−40
−60
−80
−80
−60
−40
−20
0
20
40
60
80

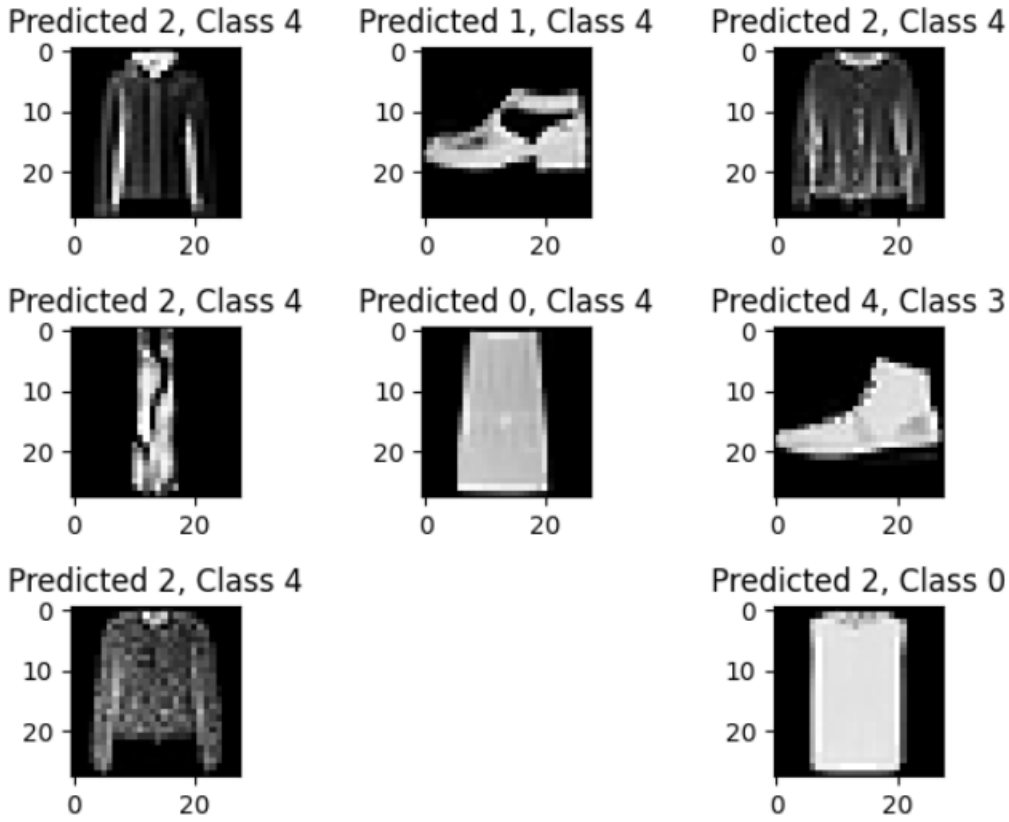
Visualizing images of test data using results from K-Means



Predicting the correct and incorrect labels and visualizing the results of clustering to find the mystery label


Predicted 4, Class 4
Predicted 2, Class 2
Predicted 2, Class 2
Predicted 2, Class 2
Predicted 0, Class 0
Predicted 2, Class 2
Predicted 0, Class 0
Predicted 3, Class 3
0
10
20

In-correct Labels



As we have address the following points in the above analysis, Visualize your encoding with the first two components from PCA, the colour mapping could be the label values.

• Perform DBSCAN and K-means clustering algorithms on the features that you have ex tracted from your own designed model with 5 clusters and visualize the results. Use the resulting clusters as alternate colour mappings for the PCA plot above.

• Apply t-SNE on the features that you have extracted for your own designed model and visualize the results in the same way.

• Based on the results of clustering and t-SNE can you guess what are the labels for the given dataset? It might also help to list out a random selection of data entries (the original images) for each cluster and their label value to help understand the patterns each cluster might represent.

• Feel free to try some other approach to using this encoding in a creative way. -> For this we have done lot of things in above code snippets.

==========
==========

```
We have checked that how many labels are correctly identified and
how many are incorrectly identified?

Correctly identified Lables                : 9373 samples


Incorrectly identified Lables              : 627 samples


Total Labels that we have in Test Data set : 10,000 samples
```

==========

Based on the results of the clustering and also visualizing the images for the mystery label we can observe the following :

1. Category 0 : Shirts/T-Shirts/Top
2. Category 1 : Sandals/Slippers
3. Category 2 : Pullover/Trouser
4. Category 3 : Sneakers/Bags
5. Category 4 : Dress/Ankle Boots/Coat

==========