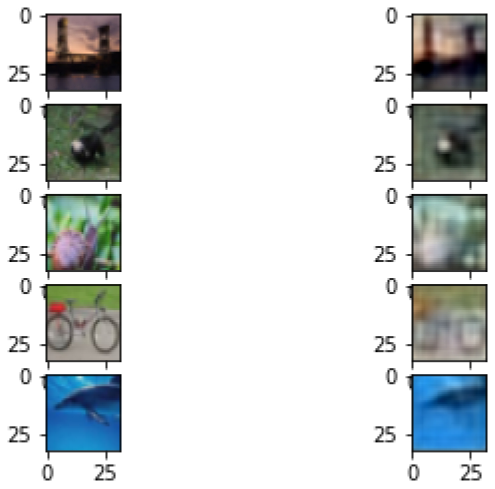
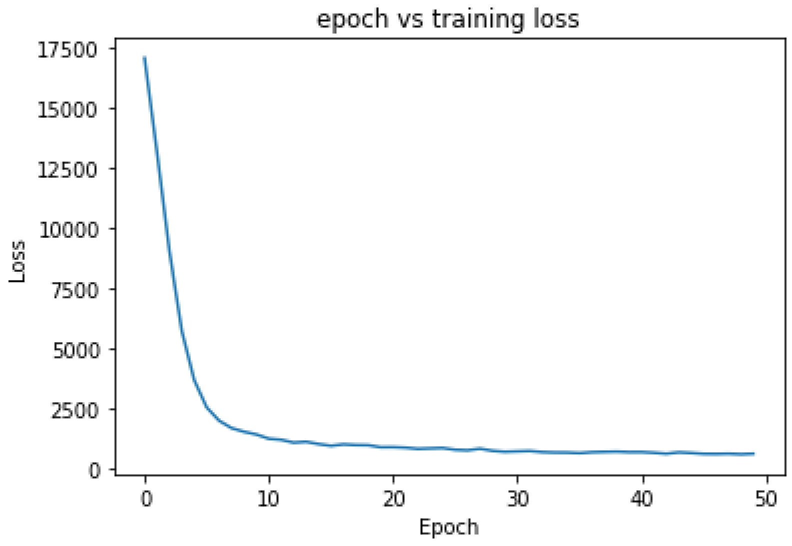
Ans 2 : In this problem, initially I am loading the dataset using the link provided and then I am unpickling the data. After that I am generating 5 random superclasses and then randomly obtaining the train as well as test labels and images with equal proportion for each selected superclass. Then I am defining my autoencoder mode in which I have done 2d convolution with depth 16 and filter 3x3 and then used max pooling and then batch normalization. Then I have done 2d convolution with depth 64 and filter 3x3 and then used max pooling and then batch normalization and then again 2d convolution with depth 128 and filter 3x3 and then again max pooling and then batch normalization. Then in decoder size I have done conv2dtranspose with depth 128 and then max pooling and then batch normalization and then conv2dtranspose with depth 64 followed by max pooling and then batch normalization and then conv2dtranspose with depth 16 followed by max pooling and then batch normalization. After that I am obtaining the output layer with depth same as input layer. Then I am predicting the decoded image and then plotting the 5x2 grid with a test image of each 5 superclasses and the corresponding obtained decoded image. Then I am plotting the epoch vs training loss curve.

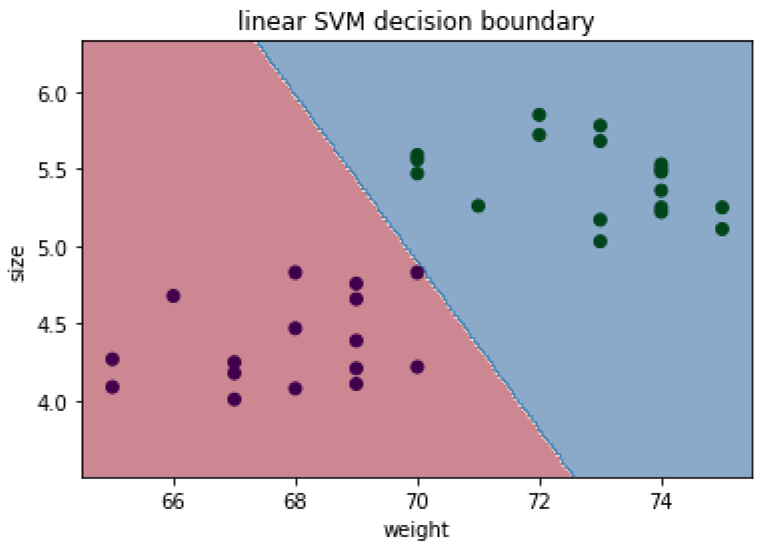




Then for 5 class classification, I am obtaining the train and test embeddings and using these embeddings I am doing the classification. For classification, I am using kernel SVM.

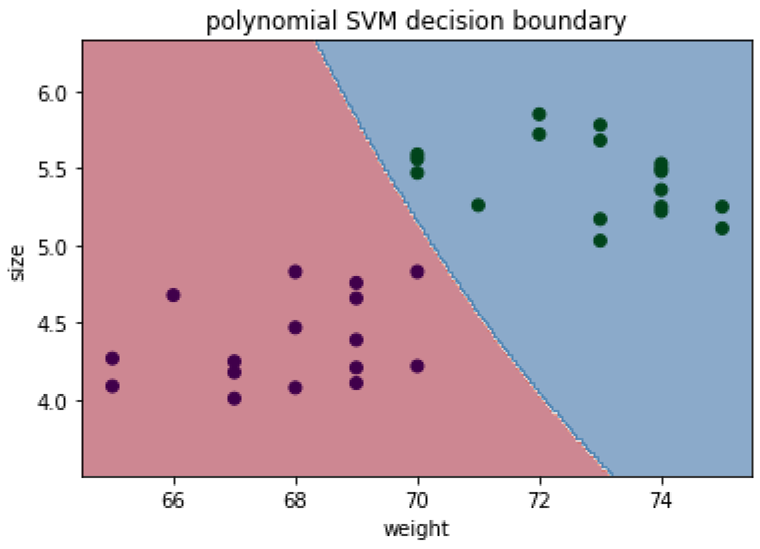
Ans 3 :
In this problem, I am loading the dataset given. Then I am separating the features and classes into different dataframes. Then I am using different inbuilt SVMs for classification. I have used linear SVM and the accuracy came out to be 100%. Similarly for polynomial SVM I have taken the degree as 10 and accuracy came out to be 100% as well. For kernel SVM I have taken gamma to be 1.3 and accuracy came out to be 100% as well.

Linear SVM



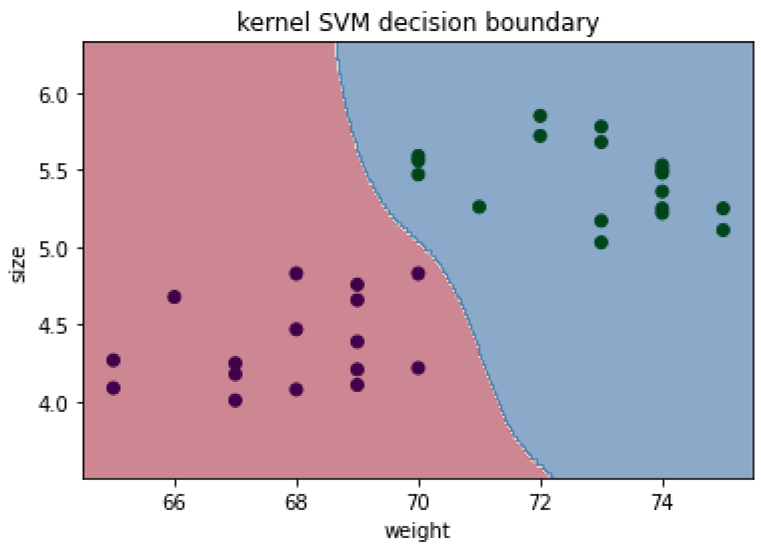
Since the data was separable, the accuracy came out to be 100% in linear SVM. The decision boundary is just at the point and is not separating the two classes with equal spaces.

Polynomial SVM



In polynomial SVM, the accuracy came out to be 100% and the decision boundary clearly separates the two classes with equal spaces for degree = 10.

Kernel SVM



In polynomial SVM, the accuracy came out to be 100% and the decision boundary clearly separates the two classes with equal spaces for gamma = 1.3. Here as I was increasing the gamma, the decision boundary was becoming more defined.

According to me, kernel SVM is better classifier than polynomial and linear SVM at least for this dataset because by tuning the value of gamma I was able to get more defined boundaries whereas in linear the boundary is fixed and in polynomial even after changing the value of degree the boundary wasn't as defined as it is for kernel SVM.

Ans 4 :

In this problem, initially I have loaded the dataset and converted the input image to 28x28 images to 56x56x3 by stacking the images in depth and resizing it to 56x56. Then I am using the pretrained VGG16, VGG19, ResNet50V2, MobileNet and EfficientNetB0 models. In each model, I am removing the output layer of the pretrained models. Then setting the trainable layers to false. Then I am defining the input and output layers. After that I am flattening the output layer of already trained model and then creating a dense layer of 256 neurons after that a dense layer of 64 neurons and then again a dense layer of 64 neurons and then output softmax layer of 10 size. Here I am using adam optimizer, categorical cross entropy and metrics as accuracy.

VGG16 accuracy : 86.9% (277s)
VGG19 accuracy : 86.36% (334s)
ResNet50V2 accuracy : 88.2% (253s)
MobileNet accuracy : 54.9% (113s)
EfficientNetB0 accuracy : 89.7% (181s)

For VGG16 and VGG19 the accuracy came out to be higher (approx. 86%). These models are simple models but are highly computationally expensive as compared to other models.
For ResNet50V2, the accuracy came out to be higher than VGG16 and VGG19 models and the computation time also reduced.
For MobileNet, the computation time reduced significantly but the accuracy was also sacrificed.
For EfficientNetB0, the computation time is less as compared to other models like VGG16, VGG19 and ResNet50V2 and the accuracy is also the highest.

According to me EfficicentNetB0 is a good model atleast for this dataset as it less computationally expensive and also has higher accuracy. MobileNet, even though is least computationally expensive but then it also sacrifices the accuracy.