Name: Mantvydas Luksas

Student ID: R00150390

Task1

To get the number of samples for sneakers the total count was calculated using the count function. The same was done for the number of samples for ankle boots.

```
"""Calculating the number of samples that are images of sneakers"""

numberOfSneakers = sneakerLabels.count();

print("\nThe number of samples that are images of sneakers is:", numberOfSneakers, "\n");

"""Calculating the number of samples that are images of ankle boots"""

numberOfAnkleBoots = ankleBootLabels.count();
```

Images were plotted using the plt.imshow()

```
for i in range(2):
    ax = plt.subplot(1, 2, i+1);

    if i == 0:
        plt.imshow(sneakers[0].reshape(28, 28));
        ax.set_title("An image of sneaker class");

    elif i == 1:
        plt.imshow(ankleBoots[0].reshape(28, 28));
        ax.set_title("An image of ankle boot class")
```

Samples to be used were calculated by multiplying the length by a decimal to get a percentage amount of the sample to be used for the data.

```
samples = []

"""70 % of samples to be used for perceptron classifier"""
numberOfSamples = int(len(df) * 0.70)

samples.append(numberOfSamples)

"""20 % of samples to be used for perceptron classifier"""
numberOfSamples = int(len(df) * 0.20)

samples.append(numberOfSamples)

"""
```

Task 2 (Similar to Task 3)

For every sample in the amount of samples in the sample array the perceptron classifier was trained using the specific sample. The train and the test data were produced from the sample that was used.

```
for sampleAmount in samples:

    featureVectors = featureVectors.head(sampleAmount)

    features = np.array(featureVectors)

    train_data = features[0:int(0.8*len(features))]

    labels = labels.head(sampleAmount)

    labelVectors = np.array(labels)

    train_target = labelVectors[0:int(0.8*len(labelVectors))]

    test_data = features[int(0.8*len(features)):len(features)]

    test_target = labelVectors[int(0.8*len(labels)):len(labelVectors)]
```

This classifier was then trained and the prediction was retrieved for every split of KFold.

This was done similarly for the Svm classifier.