**Part 1:**

Use the **Fashion-MNIST** dataset for this question.

1) Load the dataset and perform splitting into training and validation sets with 70:30 ratio.

> Do we need to normalise data? [If so Does it make any difference?]

2) Implement the K Means algorithm. You need to find the optimal number of clusters using the elbow method and silhouette method.

3) Define the initial clusters' centroids using:

> i) Forgy

> ii) Random Partition

4) Experiment with different distance measures[Euclidean distance, Manhattan distance].

5) Plot the error vs number of clusters graph while using the elbow method and silhouette method. Report the optimal number of clusters found.

6) Report the training and the validation accuracy and Compare your trained model with a model trained by the scikit-learn

7) Visualize the dataset to depict the clusters formed. #Prefer T-SNE

8) Implement K-means++, and repeat task 1 to task 7 again.

**Part 2:**

In this task, you will perform operations on `[data.csv]` `(https://drive.google.com/file/d/15NPkfXFoTkiRBlcI4ffe_Lp_BFOyf8UY/view?usp=sharing)`, data.csv is a latent space representation of Fashion-MNIST, before doing this task please read about latent space representation.

9) Load the data.csv file and apply Kmeans and Kmeans++, You need to find the optimal number of clusters using the elbow method and silhouette method.

10) Visualize the dataset to depict the clusters formed. # Prefer T-SNE

11) From these experiments(Part 1 and Part 2), compare accuracy or error, and report which one is better and why?

**Note:** If the model takes a lot of time to train you can use MiniBatchKMeans.

```
#implement elbow method from scratch
def elbow():
```

```

```

```
#implement silhouette method from scratch
def silhouette():



```

```
#implement Kmeans from scratch
class Kmeans:

  def __init__(self):



```

```
#implement Kmeans++ from scratch
class Kmeansplusplus:

  def __init__(self):



```

```
from keras.datasets import fashion_mnist
(trainX, trainy), (testX, testy) = fashion_mnist.load_data()
```