|  |
| --- |
| **Project 4: Reinforcement Learning** |

**UBIT Name: Prashi Khurana**

**UBIT Number: 50316796**

prashikh@buffalo.edu

**Abstract**

In this project, we will learn how to train and perform cluster analysis on fashion MNIST dataset using unsupervised learning. Cluster analysis is one of the unsupervised machine learning technique which doesn't require labeled data.

**1 What is Reinforcement Learning**

**1.1 Formal Definition**

Reinforcement learning is an area of Machine Learning. Reinforcement. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience. [1]

**1.2 Understanding Reinforcement Leaning in Basic Terms**

The main concept is make mistakes and learn. In very lame concepts this can also be considered as make mistakes and learn. Consider the very case for humans, if we make a mistake and we get hurt we are probable to never do that agian. A machine on the other hand will get rewards for every action it takes and the basic goal is to maximise the rewards.

**1.3 Advantages and disadvantages of Reinforcement Learning**

Some **advantages**:

Reinforcement learning can be used to solve very complex problems that cannot be solved by conventional techniques

This technique is preferred to achieve long-term results which are very difficult to achieve

This learning model is very similar to the learning of human beings. Hence, it is close to achieving perfection

The model can correct the errors occurred during the training process

Once an error is corrected by the model, the chances of occurring the same error are very less. It can create the perfect model to solve a particular problem.

Some **disadvantages**:

Reinforcement learning as a framework is wrong in many different ways, but it is precisely this quality that makes it useful.

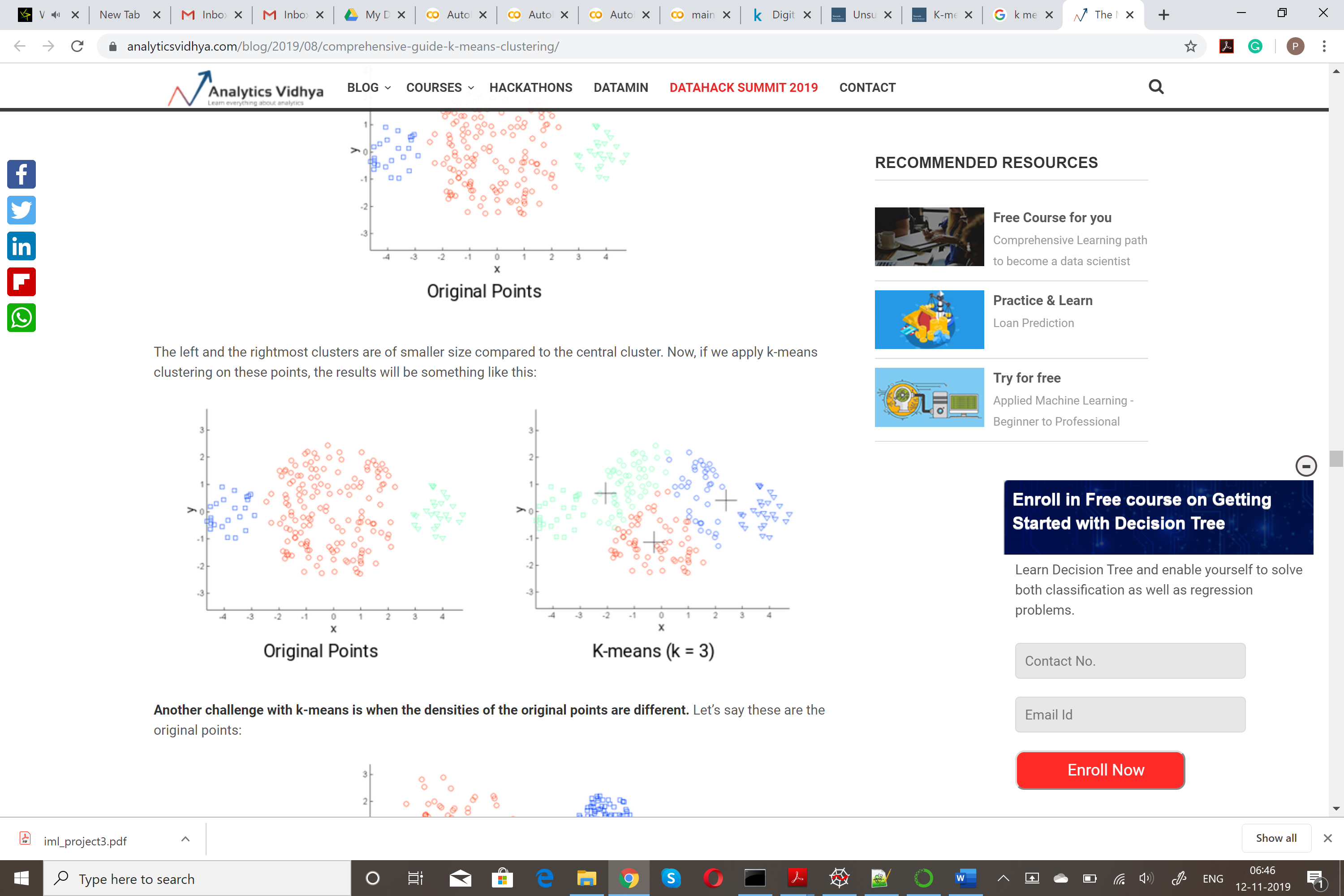
Too much reinforcement learning can lead to an overload of states which can diminish the results.

Reinforcement learning is not preferable to use for solving simple problems.

Reinforcement learning needs a lot of data and a lot of computation. It is data-hungry. That is why it works really well in video games because one can play the game again and again and again, so getting lots of data seems feasible[2]

**2 Understanding Reinforcement Leaning in terms of Mathematics**

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into ‘K’ pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster’s centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.[2]

[3]

**2.2 GMM:**

In real life, many datasets can be modeled by Gaussian Distribution (Univariate or Multivariate). So it is quite natural and intuitive to assume that the clusters come from different Gaussian Distributions. Or in other words, it is tried to model the dataset as a mixture of several Gaussian Distributions. This is the core idea of this model. [6]

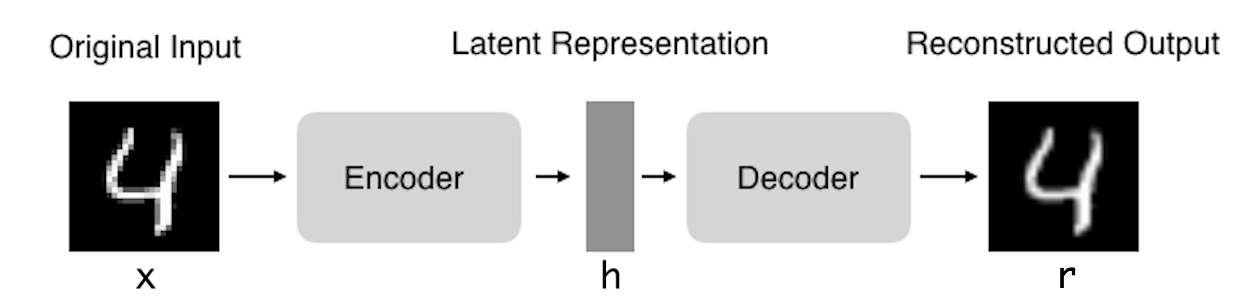
**2.3 Auto Encoder:**

Let us first try to understand what are auto-encoders.

Autoencoders (AE) are neural networks that aims to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. This kind of network is composed of two parts :

*Encoder*: This is the part of the network that compresses the input into a latent-space representation. It can be represented by an encoding function h=f(x).

*Decoder*: This part aims to reconstruct the input from the latent space representation. It can be represented by a decoding function r=g(h).



What is the benefit of using auto encoders ?

Autoencoders are learned automatically from data examples. It means that it is easy to train specialized instances of the algorithm that will perform well on a specific type of input and that it does not require any new engineering, only the appropriate training data. [4]

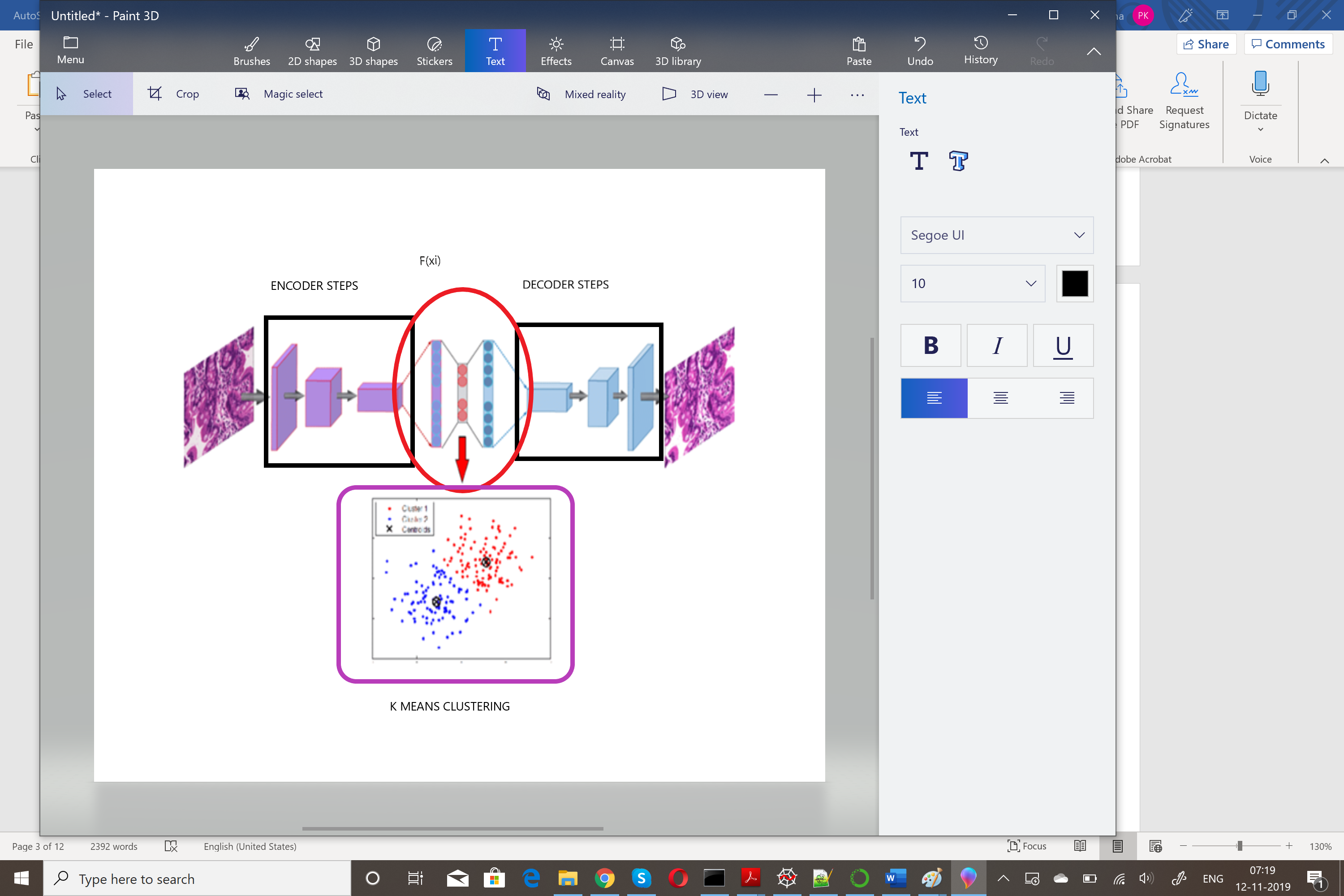
The main benefit of auto encoders are they can be used for dimensionality reduction and denoising of images.

**2.3.1 Auto Encoders with KMeans:**

KMeans is a clustering algorithm which is used to create clusters. But if the input has too many dimensions KMeans will get confused and would not be able to give a very good output. So we use an Auto-Encoder to reduce the dimensionality of the input and then pass it through KMeans.

Steps for the above algorithm:

1. Encode the input image
2. Get F(xi)
3. Do KMeans Clustering for the F(xi)

[5]

**2.3.2 Auto Encoder and GMM:**

The same as above we will use Auto Encoder but we will now use the GMM clustering algorithm.

**3 Dataset Definition**

The dataset is *Fashion-MNIST clothing images* . A few instances of data set:



Training Input size -> (60000, 784)

Training Output -> (60000,1)

Test Input size -> (10000, 784)

Test Output size->(10000,1)

**4. Pre-processing**

**4.1 Preprocessing the data**

The data which is generally used for training models might be inconsistent, incomplete and needs pre-processing. In our project, the following pre-processing functions were required:

**4.1.1 Normalize the data**

Normalization is an important part of the pre-processing of the data with machine learning. Normalization is the process of getting all the features of the of the instance in a common range. Gradient descent converges much faster when the features are normalized. The equation of normalization is X\_train = X\_train/255

**5 Implementing the underlying algorithms in on our dataset**

Now we will learn how to implement these models on our dataset.

**5.1 KMeans**

Since we have an output of 10 classes, we will be using 10 clusters. Here if we use say more than 10 clusters we will get a better accuracy but at what cost?

Here we find a useful method called the ‘elbow’ method where we find a graph for the SSE(sum of squared error) vs the number of clusters, and where we get the elbow point we use those number of clusters as the best clusters. But since the input size here has 10 clusters, we will use 10 clusters.

**5.2 Auto Encoder:**

Model: "model\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

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

input\_1 (InputLayer) (None, 784) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 32) 25120

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 784) 25872

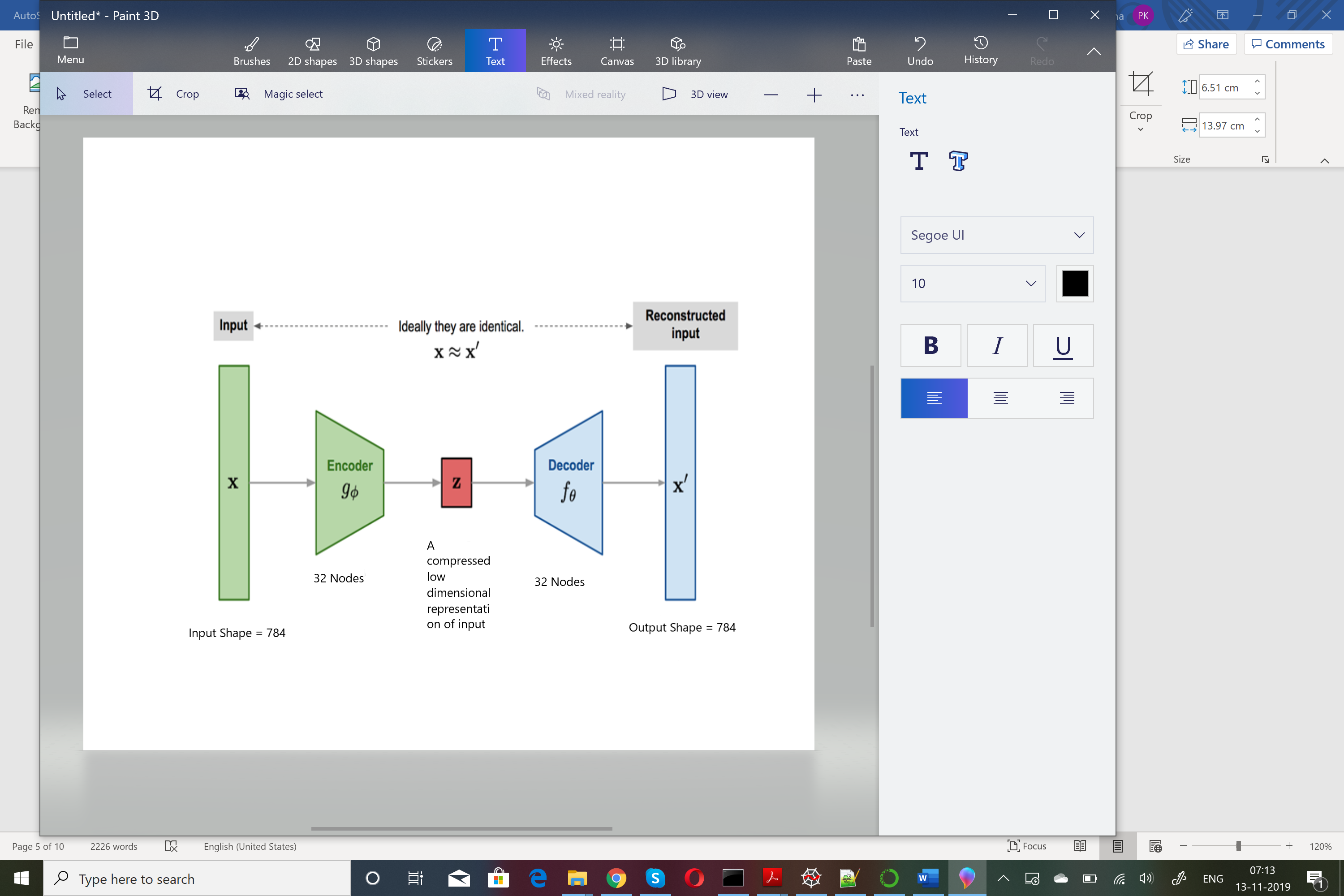
=================================================================

Total params: 50,992

Trainable params: 50,992

Non-trainable params: 0

Architecture of Auto Encoder:



The function f(xi) will compress an image of 784 size to 32. As we can see in the output of the code we are getting a size of : EncodedImagesSHape (10000, 32), which is then passed on to the clustering algorithm.

**5.2.1 Auto Encoder with KMeans**

We will use our auto encoder with K means. This means we will train our model and then use the encoded layer as an input to the KMeans. As mentioned in section 2.3.1 we will then use this f(xi) -> (encoded layer from the autoencoder) as the input for the KMeans.

**5.2.2 Auto Encoder with GMM**

We will use our auto encoder with GMM. This means we will train our model and then use the encoded layer as an input to the GMM. As mentioned in section 2.3.2 we will then use this f(xi) -> (encoded layer from the autoencoder) as the input for the GMM.

**6 Actual Implementation**

**6.1 KMeans**

**6.1.1 KMeans with Normalized Mutual Info Score:**

1. Pre-process the data (3)
2. Assign a KMeans model with the cluster size as 10
3. Fit the KMeans algo with the data X
4. Predict the clusters with KMeans.predict
5. Find the normalized\_mutual\_info\_score of the KMeans Model

**6.1.2 KMeans with Accuracy:**

1. Pre-process the data (3)
2. Assign a KMeans model with the cluster size as 10
3. Fit the KMeans algo with the data X
4. Use infer\_cluster\_labels to Associates most probable label with each cluster in KMeans model. Infer\_Cluster\_Labels from the KMeans ,Y
5. Use infer\_data\_labels Determines label for each array, depending on the cluster it has been assigned to.
6. Predict the X\_clusters from kmeans.predict(X)
7. Get the predicted labels as predicted\_labels = infer\_data\_labels(X\_clusters, cluster\_labels)and this returns predicted labels for each array
8. Get the accuracy score, accuracy\_score from the KMeans.[8]
   1. **Auto-Encoder**
9. Create a model as mentioned in the section 5.2.
10. encoding\_dim = 32
11. input\_img = Input(shape=(784,))
12. encoded = Dense(encoding\_dim,activation='selu',kernel\_regularizer=regularizers.l2(0.01))(input\_img)
13. decoded = Dense(784, activation='sigmoid')(encoded)
14. autoencoder = Model(input\_img,decoded)
15. encoder = Model(input\_img,encoded)
16. encoded\_input = Input(shape=(encoding\_dim,))
17. decoder\_layer = autoencoder.layers[-1]
18. decoder = Model(encoded\_input, decoder\_layer(encoded\_input))
19. autoencoder.compile(optimizer='adam', loss='binary\_crossentropy')
20. autoencoder.fit(x\_train,x\_train, epochs=60, batch\_size=2056, shuffle=True, validation\_data=(x\_test,x\_test))
21. autoencoder.summary()
22. encoded\_images = encoder.predict(x\_test)
    1. **Auto Encoder with KMeans**
23. Once the auto-encoder is set(as in section 6.2), pass the f(xi) to the KMeans algorithm (as in section 6.1)
    1. **Auto Encoder with GMM**
24. Once the auto-encoder is set(as in section 6.2), pass the f(xi) to the GMM algorithm
25. clf = GmmMml()
26. clf.fit(encoded\_image)
27. clf.predict(encoded\_image)
28. Once you predict the encoded\_images we can get the normalized\_mutual\_info\_score

**7 Results**

**7.1 Understanding Normalized Mutual Info Score:**

Normalized Mutual Information (NMI) is a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation). In this function, mutual information is normalized by some generalized mean of H(labels\_true) and H(labels\_pred)), defined by the average\_method. This measure is not adjusted for chance. Therefore adjusted\_mutual\_info\_score might be preferred. This metric is independent of the absolute values of the labels: a permutation of the class or cluster label values won’t change the score value in any way. This metric is furthermore symmetric: switching label\_true with label\_pred will return the same score value. This can be useful to measure the agreement of two independent label assignments strategies on the same dataset when the real ground truth is not known.[7]

**Now we need to know the difference between accuracy and NMIS.**

If we need to calculate the accuracy we need to know the labels assigned to the clusters. NMIS takes that into account so for our case, we can do with NMIS instead of the accuracy.

**7.1.1 Understanding the results**

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **NMIS** |
| **Kmeans** | 0.56 | 0.5123 |
| **Auto Encoder with Kmeans** |  | 0.5323 |
| **Auto Encoder with GMM** |  | 0.5152 |

**7.2 Confusion Matrix**

**7.2.1 KMeans with NMIS:**

[[ 29 0 587 6 244 1 5 94 0 34]

[890 0 50 0 29 0 0 22 0 9]

[ 4 0 19 4 342 0 4 61 0 566]

[503 0 277 2 111 0 3 94 0 10]

[ 27 0 136 4 159 0 5 42 0 627]

[ 0 45 0 0 6 227 0 650 72 0]

[ 12 0 189 15 358 0 0 115 0 311]

[ 0 2 0 0 0 784 0 62 152 0]

[ 6 1 3 353 35 39 408 84 10 61]

[ 0 423 0 0 4 23 2 29 519 0]]

CLASSIFICATION REPORT

precision recall f1-score support

0 0.47 0.58 0.52 1000

1 0.02 0.02 0.02 1000

2 0.27 0.34 0.30 1000

3 0.34 0.50 0.41 1000

4 0.00 0.00 0.00 1000

5 0.00 0.00 0.00 1000

6 0.19 0.31 0.24 1000

7 0.20 0.15 0.17 1000

8 0.96 0.41 0.57 1000

9 0.02 0.02 0.02 1000

accuracy 0.23 10000

macro avg 0.25 0.23 0.22 10000

weighted avg 0.25 0.23 0.22 10000

**7.2.2 KMeans with Accuracy:**

[[588 24 36 0 7 66 268 1 10 0]

[ 49 889 2 0 11 19 30 0 0 0]

[ 14 4 313 0 289 48 326 0 6 0]

[281 490 3 0 12 74 135 0 5 0]

[102 25 188 0 510 36 132 0 7 0]

[ 0 0 0 0 0 690 8 220 3 79]

[185 11 144 0 194 97 356 0 13 0]

[ 0 0 0 0 0 84 0 865 1 50]

[ 3 4 61 0 7 69 56 48 752 0]

[ 0 0 1 0 1 36 7 107 4 844]]

CLASSIFICATION REPORT

precision recall f1-score support

0 0.47 0.59 0.52 1000

1 0.61 0.89 0.72 1000

2 0.00 0.00 0.00 1000

3 0.00 0.00 0.00 1000

4 0.39 0.63 0.48 1000

5 0.52 0.65 0.58 1000

6 0.28 0.36 0.31 1000

7 0.73 0.79 0.76 1000

8 0.94 0.76 0.84 1000

9 0.77 0.94 0.85 1000

accuracy 0.56 10000

macro avg 0.47 0.56 0.51 10000

weighted avg 0.47 0.56 0.51 10000

**7.2.3 Auto Encoder with KMeans with NMIS**:

[[ 5 4 6 0 0 28 258 2 651 46]

[ 1 0 10 0 0 900 36 0 51 2]

[ 3 3 410 0 0 4 309 0 17 254]

[ 2 1 14 0 0 538 181 0 260 4]

[ 2 5 605 0 0 28 137 0 119 104]

[ 2 0 0 206 134 0 111 547 0 0]

[ 11 2 263 0 0 12 370 3 216 123]

[ 1 0 0 165 2 0 0 832 0 0]

[360 392 7 9 0 5 108 45 5 69]

[ 1 0 1 565 400 0 8 23 2 0]]

CLASSIFICATION REPORT

precision recall f1-score support

0 0.17 0.24 0.20 1000

1 0.01 0.01 0.01 1000

2 0.00 0.00 0.00 1000

3 0.34 0.49 0.40 1000

4 0.00 0.00 0.00 1000

5 0.39 0.58 0.46 1000

6 0.00 0.00 0.00 1000

7 0.00 0.00 0.00 1000

8 0.92 0.41 0.56 1000

9 0.00 0.00 0.00 1000

accuracy 0.17 10000

macro avg 0.18 0.17 0.16 10000

weighted avg 0.18 0.17 0.16 10000

**7.2.4 Auto Encoder with GMM with NMIS**:

[[ 26 822 82 59 11 0 0 0 0 0]

[ 1 55 16 926 2 0 0 0 0 0]

[122 76 775 2 25 0 0 0 0 0]

[ 13 382 19 584 2 0 0 0 0 0]

[ 57 191 718 16 18 0 0 0 0 0]

[402 1 0 0 51 546 0 0 0 0]

[131 334 497 15 23 0 0 0 0 0]

[ 5 0 0 0 13 982 0 0 0 0]

[110 69 67 0 741 13 0 0 0 0]

[ 67 0 0 0 627 306 0 0 0 0]]

CLASSIFICATION REPORT

precision recall f1-score support

0 0.03 0.03 0.03 1000

1 0.03 0.06 0.04 1000

2 0.36 0.78 0.49 1000

3 0.36 0.58 0.45 1000

4 0.01 0.02 0.01 1000

5 0.30 0.55 0.38 1000

6 0.00 0.00 0.00 1000

7 0.00 0.00 0.00 1000

8 0.00 0.00 0.00 1000

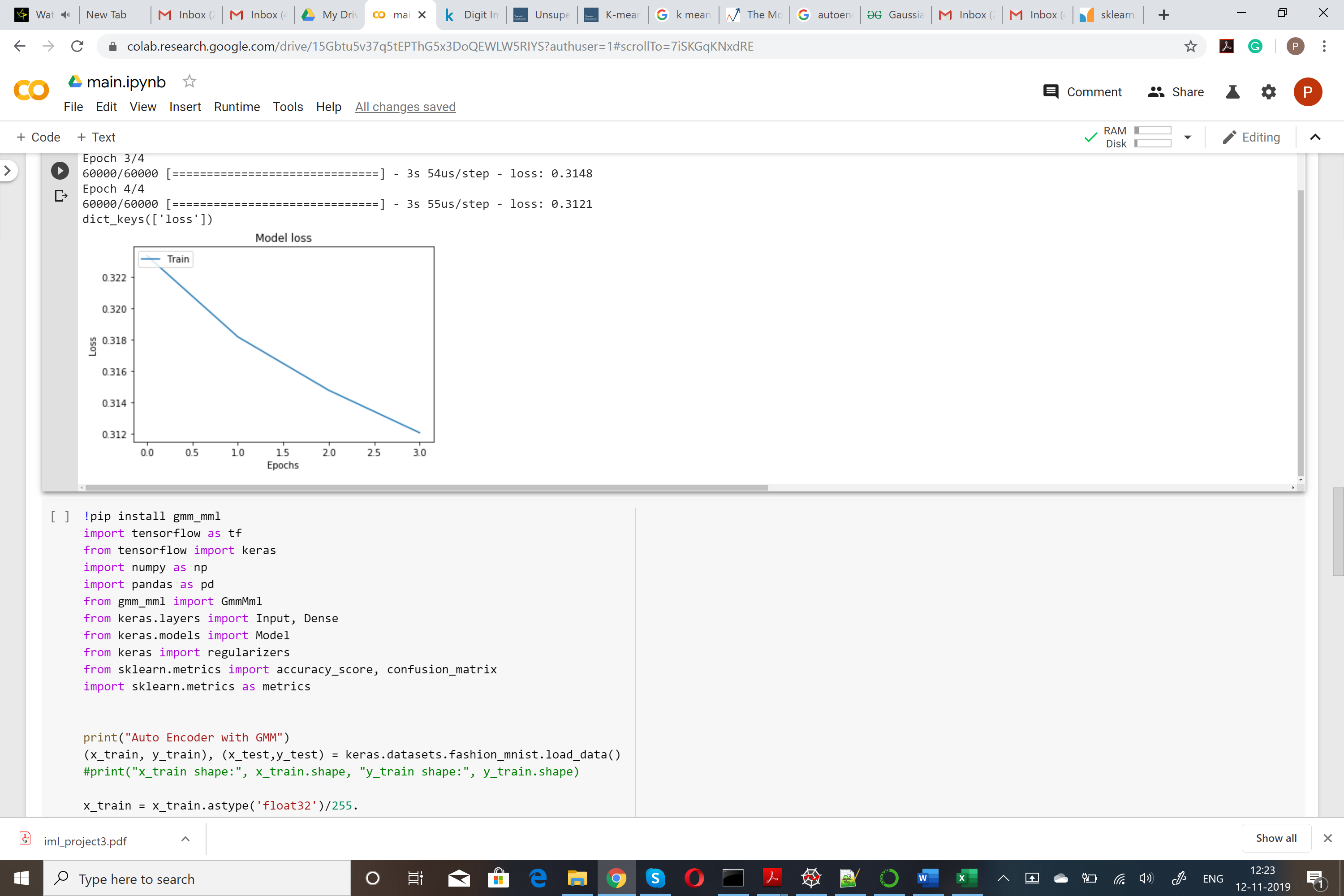
9 0.00 0.00 0.00 1000

accuracy 0.20 10000

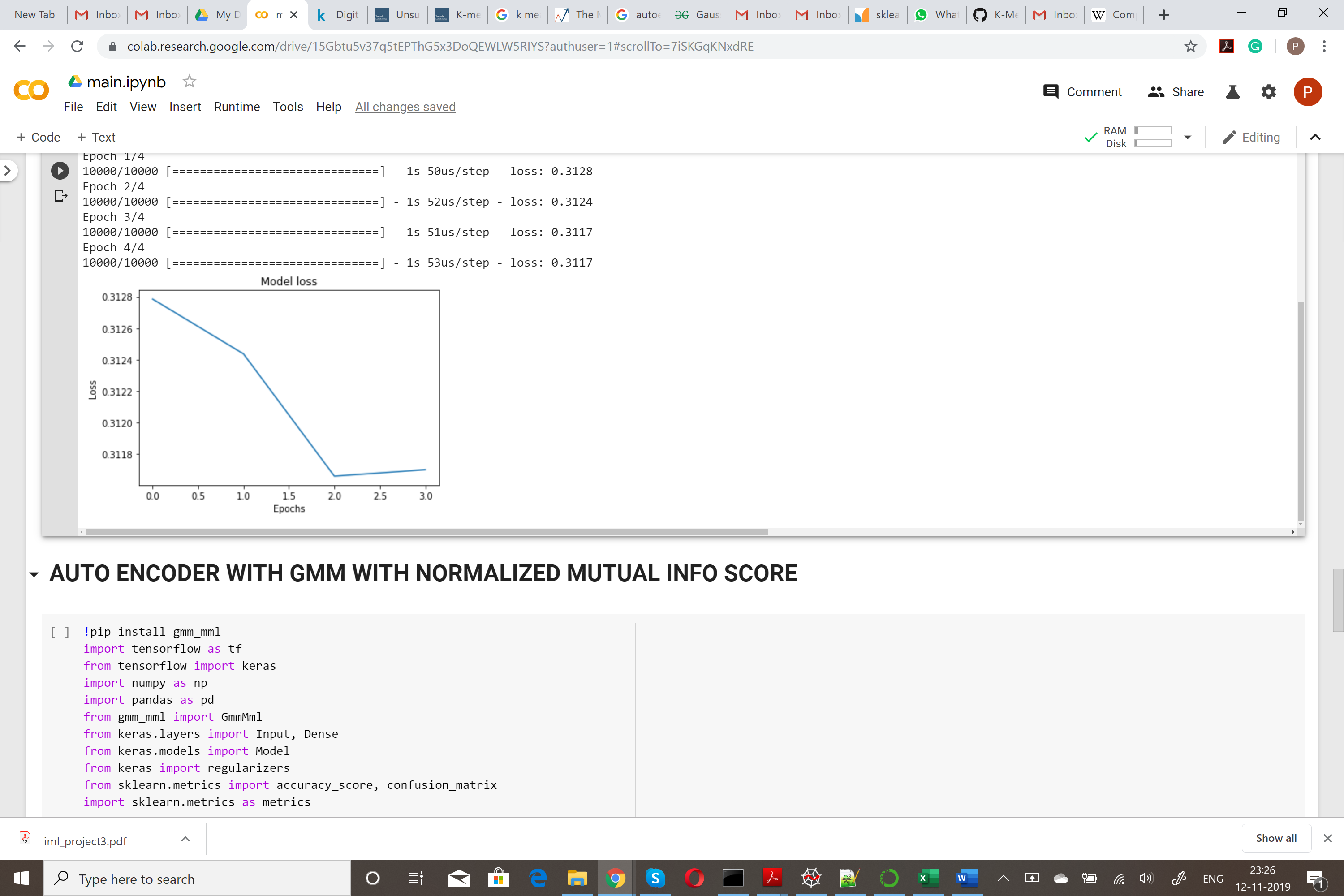
macro avg 0.11 0.20 0.14 10000

weighted avg 0.11 0.20 0.14 10000

**7.3 Graphs for Training Dataset Loss:**



**7.4 Graphs for Validation Dataset Loss**



**8 Conclusion**

Once we have trained the logistic regression model, we need to understand the results of the model.

* In this project, we are trying to see the difference of applying different clustering algorithms
* Initially we will use the KMeans Clustering algorithm to see the accuracy of the input size which is around 56%, while the NIMS is around 51%. We can use the NMIS score instead of the accuracy as it performs the same operation.
* Auto encoder is used to reduce the dimensionality of the input
* After passing through the encoder we get a reduced dimensionality and then use KMeans, we can train them faster.
* We also see that the validation and training loss decrease as we increase the epochs.

**References**

[1] https://www.geeksforgeeks.org/supervised-unsupervised-learning/

[2]https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

[3] https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

[4] https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

[5] https://www.researchgate.net/figure/Structure-of-clustering-model-with-autoencoder-and-K-means-combination\_fig2\_332368916

[6] https://www.geeksforgeeks.org/gaussian-mixture-model/

[7]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized\_mutual\_info\_score.html

[8]https://github.com/xoraus/K-Means-Clustering-for-Imagery-Analysis/blob/master/KMeans%20Clustering%20for%20Imagery%20Analysis%20(Jupyter%20Notebook).ipynb