ASSIGNMENT 4
SECTION A

Vasu
2020483

Section - A

a

① * ② Δ ③ Output

i)

~~Input~~ For layer 1

$$W = \frac{(15-5+(2\times1))}{1} + 1 = 13$$

$$h = \frac{(15-5+(2\times1))}{1} + 1 = 13$$

*
13×13

For maxpool layer

$$W_1 = \frac{13-3}{2} + 1 = 6$$

$$h_1 = \frac{13-3}{2} + 1 = 6$$

Δ
6×6

After last conv layer

$$W_2 = \left[\frac{6-5+(2\times2)}{2}\right] + 1 = 3.5 \approx 3$$

$$h_2 = \left[\frac{6-3+(2\times2)}{2}\right] + 1 = 4.5 \approx 4$$

Output 3×4

ii) Significance of pooling in CNN

- It helps in achieving translation invariance
- It reduces the computation cost
- helps in parametre sharing
- Reduces the dimension of the feature map

iii) For layer 1 $\Rightarrow (5 \times 5 \times 4 \times 1) \times 1$

$= 100$

For layer 2 $= (5 \times 3 \times 4 \times 1) \times 1$

$= 60$

Pooling layer = 0 the total parametres w/o bias = 100+60 = 160

b)

Initial clusters $C_1 = (3, 12)$ $C_2 = (8, 7)$ $C_3 = (2, 13)$

Points	Centroid 1	Centroid 2	Centroid 3	Cluster
3, 12	0	10	2	C_1
3, 7	5	5	7	C_1
9, 6	12	2	14	C_2
6, 10	5	5	7	C_2
1, 7	10	6	12	C_2
7, 6	10	2	12	C_2
2, 13	2	12	0	C_3

New centroids for $C_1 = \frac{3+3}{2}, \frac{12+7}{2}$

$C_1 = 3, 9.5$

for C_2

$\left(\frac{9+6+8+7}{4}, \frac{6+10+7+6}{4}\right) = 7.5, 7.25$

For C_3 $(2, 13)$ remains the same

II Iteration

Points	C_1	C_2	C_3	Cluster
3, 12	2.5	9.25	2	C_3
3, 7	2.5	4.75	7	C_1
9, 6	8.5	2.75	14	C_2
6, 10	3.5	4.25	7	C_1
8, 7	7.5	0.75	12	C_2
7, 6	7.5	1.74	12	C_2
2, 13	4.5	11.25	6	C_3

New centroid for C_1 $\left(\frac{3+6}{2}, \frac{7+10}{2}\right) = (4.5, 8.5)$

for C_2 $\left(\frac{9+8+7}{3}, \frac{6+7+6}{3}\right) = (8, 6.33)$

for C_3 $\frac{3+2}{2}$ $\frac{13+12}{2} = (2.5, 12.5)$

SECTION B

We implemented the CNN from scratch using forward and backward propagation, and also we used max-pooling.

In fwd propagation, We begin with one shape input and one shape filter, assuming a number of channels C = 1 and stride = 1, and then perform a convolution operation to obtain our output layer.

In backward propagation, we send the output back to the layers and try to reduce the loss. When performing backpropagation, we usually have an incoming gradient from the following layer as we follow the chain rule.

Max pooling is a pooling operation that selects the largest element from the feature map region covered by the filter.

I performed the CNN from scratch and then used mnist dataset to check my cnn by just checking on the test set. It gave an accuracy above 90%+.

```
incount=0
for i in range(len(test_images)):
    out = convolution.forward((test_images[i] / 255))
    out = mpool.forward(out)
    out = softmax.forward(out)

    if (np.argmax(out) == test_labels[i]):
        count=count+1

    else:
        incount=incount+1


print("Correct Answer percentage ", (incount/(incount+count)*100))
```

```
Correct Answer percentage  91.5
```

SECTION C

Qb)

```
]: # removing data with 30% null values
drop=["MIGMTR1","MIGMTR3","MIGMTR4","MIGSUN"]
for i in drop:
    pdata=pdata.drop(i,axis=1)
```

```
In [56]: print(pdata.isnull().sum())
AAGE            0
ACLSWKR         0
ADTIND          0
ADTOCC          0
AHGA            0
AHRSPAY         0
AHSCOL          0
AMARITL         0
AMJIND          0
AMJOCC          0
ARACE           0
AREORGN         0
ASEX            0
AUNMEM          0
AUNTYPE         0
AWKSTAT         0
CAPGAIN         0
CAPLOSS         0
DIVVAL          0
FILESTAT        0
GRINREG         0
GRINST        708
HHDFMX          0
HHDREL          0
MIGMTR1     99696
MIGMTR3     99696
MIGMTR4     99696
MIGSAME         0
MIGSUN      99696
NOEMP           0
PARENT          0
PEFNTVTY     6713
PEMNTVTY     6119
PENATVTY     3393
PRCITSHP        0
SEOTR           0
VETQVA          0
VETYN           0
WKSWORK         0
YEAR            0
dtype: int64
```

```
In [8]: # removing data with 30% null values
drop=["MIGMTR1","MIGMTR3","MIGMTR4","MIGSUN"]
for i in drop:
    pdata=pdata.drop(i,axis=1)
```

Qc)

```
9]: for col in pdata.columns:
        mode=pdata[col].mode()
        pdata[col].fillna(mode[0],inplace=True)

    print(pdata.isnull().sum())
```

```
AAGE          0
ACLSWKR       0
```

```
In [10]: pdata["AAGE"].max()
         pdata["AAGE"].mean()
         bins=[0,13,18,60,100]
         labels=["Child","Teenager","Adult","Elderly"]
         pdata["AAGE_category"]=pd.cut(pdata["AAGE"],bins,labels=labels)
```

```
In [11]: pdata["AHRSPAY"].max()
         pdata["AHRSPAY"].mean()
         bins=[-1,100,500,10000]
         labels=["low wage","average wage","high Wage"]
         pdata["WAGE_CATEGORY"]=pd.cut(pdata["AHRSPAY"],bins,labels=labels)
```

```
In [12]:
         pdata["CAPGAIN"].mean()
         bins=[-1,100,1000,20000,100000]
         labels=["low ","average ","Good "," very High"]
         pdata["Cap_gain"]=pd.cut(pdata["CAPGAIN"],bins,labels=labels)
```

```
In [17]: from sklearn.preprocessing import OneHotEncoder
         encoder=OneHotEncoder()
         features=encoder.fit_transform(pdata[['ACLSWKR', 'ADTIND', 'ADTOCC', 'AHGA', 'AHSCOL', 'AMARITL
         np.array(features)
         labels=encoder.categories_
         flatlist=[]
         for sublist in labels:
             for element in sublist:
                 flatlist.append(element)
         labels=flatlist
```

```
In [100]: labels
```

We used pandas to find the null rows and then used the mode function to fill the null values, and then we saw that there were no null values in the data.

We were given the description of the data where we saw that 6-7 columns were numerical, so I created bins according to the data and then added a column in the data with categorical data using pd.cut.

Then one hot encoder library was used from SK learn to do the encoding of the data and then stored the data as a new data frame.

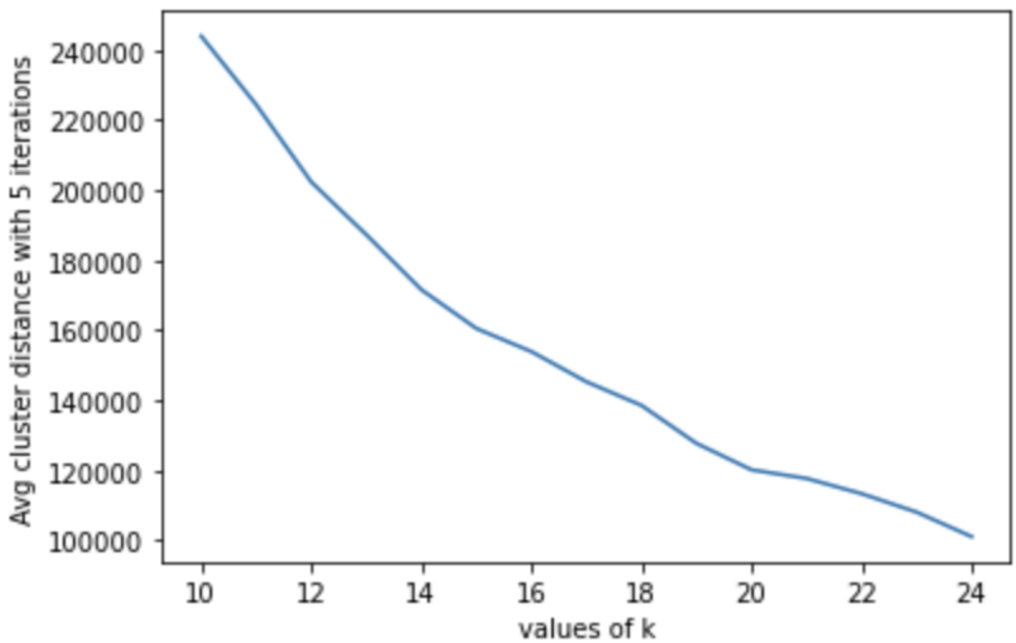
Qd)

K-medians Clustering is a cluster analysis algorithm. It is a k-means clustering variant in which the median is used instead of the mean to determine the centroid of each cluster.

The elbow method is commonly used in cluster analysis to determine the number of clusters in a data set. The method entails plotting the explained variation as a function of cluster count and selecting the curve's elbow as the number of clusters to use.
From the elbow graph, we can see that we can not get a precise value of k, so we can take 15 as there is a slight elbow there.

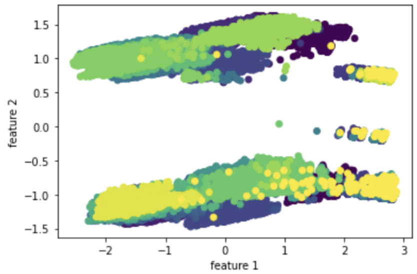
```
plt.xlabel("values of k")
plt.ylabel("Avg cluster distance with 5 iterations")
plt.plot(K,loss_val)
```

```
[<matplotlib.lines.Line2D at 0x7fa328356850>]
```

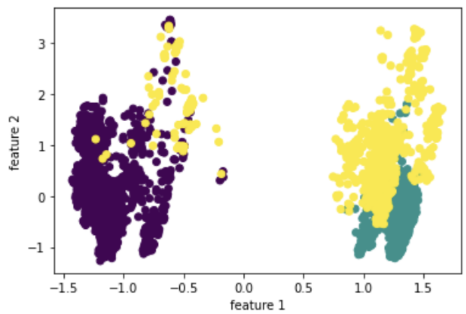


Q f)
We used PCA to reduce the data into 2 dimensions and plotted their clusters.
We observed that cluster 2 comes out to be a subset of cluster 1, and we can also observe that the data is missing below and above the origin in the more than 50k dataset.
The population data shows that the green cluster is dense, whereas the corners are not dense. They are a bit sparse.

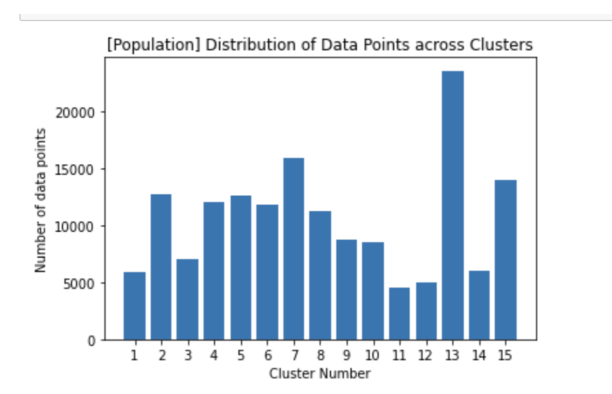
The cluster of population.csv :



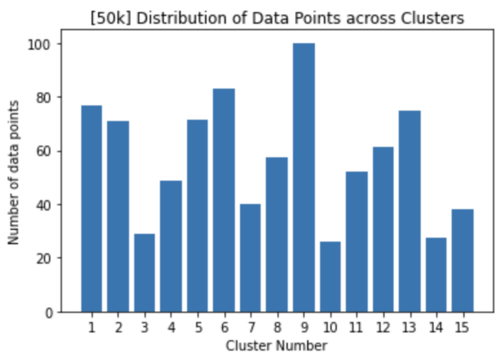
The cluster of more than 50.csv:



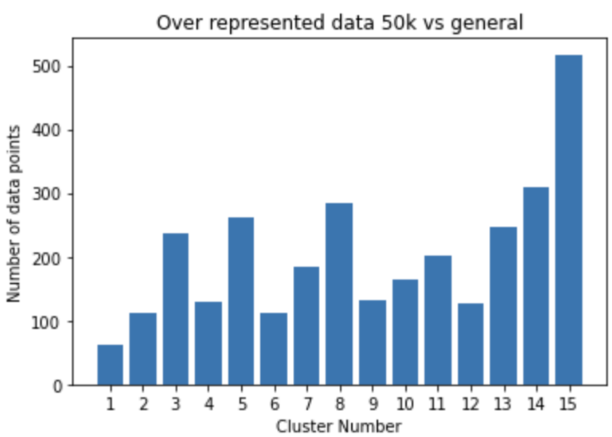
We observed that the general population's highest number of data points was in cluster number 13, and the least were in cluster 11.



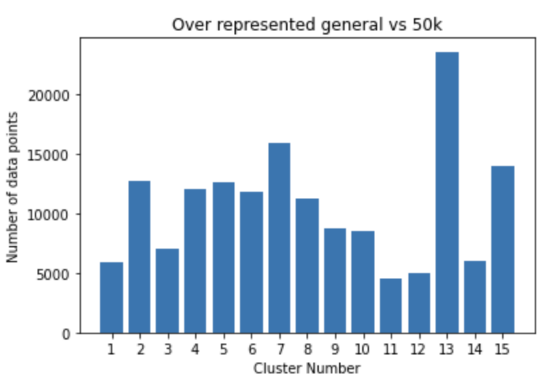
We observed that the highest data points in the more than 50k population were in cluster number 9, and the least were in cluster 10.



Overreprenstation:
For 50k vs general, we can see that 14 and 15 are most overrepresented.



For general vs 50k, we can see that 7 and 13 are the most over represenated.



Comparisons:
The optimal number of clusters we got, in general, was about 15 and 18 in more than 50k data.

The cluster of the data sets was dense in the corners(right and left) for more than 50k, and they were dense in the top and bottom in the case of the general population.

The 50k data set is the subset of the general population dataset.