Machine Learning

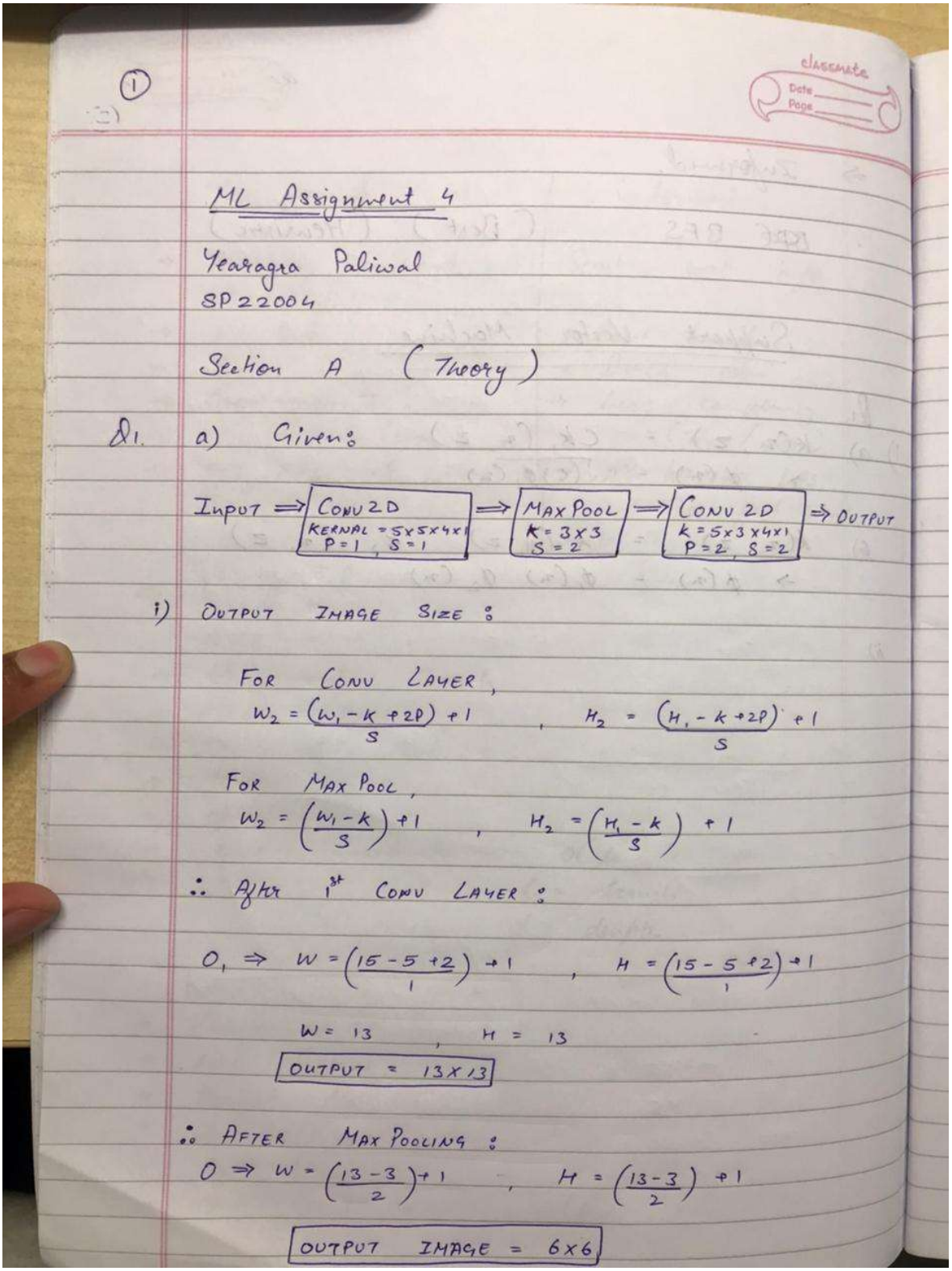
Assignment 4

Submitted By:

Yearagra Paliwal

(SP22004)

Question1:



ML Assignment 4

Yeasagra Paliwal
SP22004

Section A (Theory)

Q1. a) Given:

Input ⇒ [Conv 2D: Kernal = 5x5x4x1, P=1, S=1] ⇒ [Max Pool: K=3x3, S=2] ⇒ [Conv 2D: k=5x3x4x1, P=2, S=2] ⇒ Output

i) Output Image Size:

For Conv Layer,

$W_2 = \frac{(W_1 - K + 2P)}{S} + 1$, $H_2 = \frac{(H_1 - K + 2P)}{S} + 1$

For Max Pool,

$W_2 = \left(\frac{W_1 - K}{S}\right) + 1$, $H_2 = \left(\frac{H_1 - K}{S}\right) + 1$

∴ After 1st Conv Layer:

$O_1 \Rightarrow W = \left(\frac{15 - 5 + 2}{1}\right) + 1$, $H = \left(\frac{15 - 5 + 2}{1}\right) + 1$

$W = 13$, $H = 13$

Output = 13x13

∴ After Max Pooling:

$O \Rightarrow W = \left(\frac{13 - 3}{2}\right) + 1$, $H = \left(\frac{13 - 3}{2}\right) + 1$

Output Image = 6x6

∴ AFTER FINAL CONV LAYER :

$$O \Rightarrow W = \left(\frac{6-5+(2\times2)}{2}\right)+1 \quad , \quad H = \left(\frac{6-3+(2\times2)}{2}\right)+1$$

$$W = 3.5 \quad , \quad H = 4.5$$

OUTPUT IMAGE = 3.5×4.5

OR

OUTPUT = 3×4

ii) SIGNIFICANCE OF CNN :

→ VARIANCE MAY GET ACHIEVE IN TRANSLATION.
→ EFFECT OF NOISE & DISTORTION WILL GET REDUCE.
→ SPARSE INTERACTION : INPUT OF DIMENSION GET REDUCE. DUE TO KERNAL & HENCE INPUT COMPLEXITY REDUCES.
→ PARAMETER SHARING.

iii) FOR CONV LAYER :

n = KERNAL SIZE + (1) × KERNAL
↳ FOR BIAS

IGNORING BIAS

n = KERNAL SIZE × NO. OF KERNAL

FOR POOLING : NO. OF LEARNLISE PARAMETER = 0

∴ FOR CONV LAYER 1 : $n_1 = (5 \times 5 \times 4 \times 1) \times 1$

$n_1 = 100$

∴ FOR CONV LAYER 2 : $n_2 = (5 \times 3 \times 4 \times 1) \times 1$

$n_2 = 60$

TOTAL LEARNALISE PARAMETER = 100 + 60

= 160

③

b.) Pts :
(3, 12) , (3, 7) , (9, 6) , (6, 10) , (8, 7) , (7, 6) , (2, 13)

INITIAL CENTER : (3, 12) , (8, 7) , (2, 13)
c_1 c_2 c_3

ITERATION I :

Pt	Dist FROM c_1 (3, 12)	Dist FROM c_2 (8, 7)	Dist FROM c_3 (2, 13)	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
(8, 7)	10	0	12	c_2
(7, 6)	10	2	12	c_2
(2, 13)	2	12	0	c_3

RECALCULATING CENTER USING K-MEAN

$$c_1 = \left(\frac{3+3}{2} , \frac{12+7}{2}\right)$$

$$\boxed{c_1 = (3, 9.5)}$$

$$c_2 = \left(\frac{9+6+8+7}{4} , \frac{6+10+7+6}{4}\right)$$

$$\boxed{c_2 = 7.5 , 7.25}$$

$$\boxed{c_3 = 2, 13}$$

ITERATION II :

P_7	Dist FROM $c_1(3, 9.5)$	Dist FROM $c_2(7.5, 7.25)$	Dist FROM $c_3(2, 13)$	CLUSTER
(3, 12)	2.5	4.75	2	C_3
(3, 7)	2.5	4.75	7	C_1
(9, 6)	9.5	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	1.75	12	C_2
(2, 13)	4.5	11.75	0	C_3

C_1 = (3, 7) , (6, 10)
C_2 = (9, 6) , (8, 7) , (7, 6)
C_3 = (3, 12) , (2, 13)

~~HER~~ NEW CENTERS :

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

$$C_2 = \left(\frac{9+7+8}{3}, \frac{6+7+6}{3}\right) = (8, 6.3)$$

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

Question3:

Dataset:

```
199523 rows × 40 columns
```

Number of row and columns.

```
[112] columns1=["AAGE", "A

[113] columns2=[ "ACLSWKR"
```

Differentiating in categorical and numerical value.

Preprocessing:

```
#Replace ? with NAN
for q in columns2:
  data[q] = data[q].ma
  data[q] = data[q].ap
  data[q].apply(lambda
```

Replacing ? with NAN

```
#Null Values in Respective column
data.isnull().sum().sort_values(as
```

```
AAGE       0
ACLSWKR    0
HHDFMX     0
HHDREL     0
MIGMTR1    0
MIGMTR3    0
MIGMTR4    0
MIGSAME    0
MIGSUN     0
NOEMP      0
dtype: int64
```

Null Values in Respective column.

```
#Checking the missing value and if they have more than 40% will remove them
perc = 40.0
for c in columns2:
   print(c,data[c].isna().sum()/len(data))
min_count =  int(((100-perc)/100)*data.shape[0] + 1)
data = data.dropna( axis=1,
                thresh=min_count)
```

```
ACLSWKR 0.0
```

Checking the missing value and if they have more than 40% will remove them

Imputation, Bucketization, One Hot Encoding:

```
[118] #Droping columns based on ratio with more than 85%
      az=[]
      for col in data:
        print(data[col].value_counts(ascending = True, nor
      for col in data:
        ratio = data[col].value_counts(ascending = True, n
        if(ratio > 0.85):
          az.append(col)

      data=data.drop(az, axis = 1)

      0.01748670579331706 AAGE
      0.5024232795216591 ACLSWKR
```

Dropping columns based on ratio with more than 85%

```
Imputation

[126] mode_append=[]
      for column in data.columns:
          mode_append.append(data[column].mode()[0])
          data[column].fillna(data[column].mode()[0], inplace=True)
      print(mode_append)

      [0.0, ' Not in universe', 0, 0, ' High school graduate', ' Nev
```

Imputation.

Bucketization

```
[127] # Numercial Columns
      numercial_columns=["AAGE","WKSWORK"]

[128] #Bucketizing
      from sklearn.preprocessing import KBinsDiscretizer
      def bucketize(data, c, b):
        Bucketizer=KBinsDiscretizer(n_bins=b, encode='ordinal', strategy='quantile')
        n1data=pd.cut(data[c], bins = b, labels = False).to_numpy()
        d1=Bucketizer.fit_transform(n1data.reshape(-1, 1))
        data[c] = d1

[129] for c in numercial_columns:
        if(c=="AAGE"):
          bucketize(data,c,10)
        else:
          bucketize(data,c,5)
```

Bucketization

One-Hot Encoding

```
[131] dt=data.copy()
      for c in data.columns:

        dt= pd.get_dummies(dt, columns=[c], prefix = [c])

      print(dt)
```

One Hot Encoding

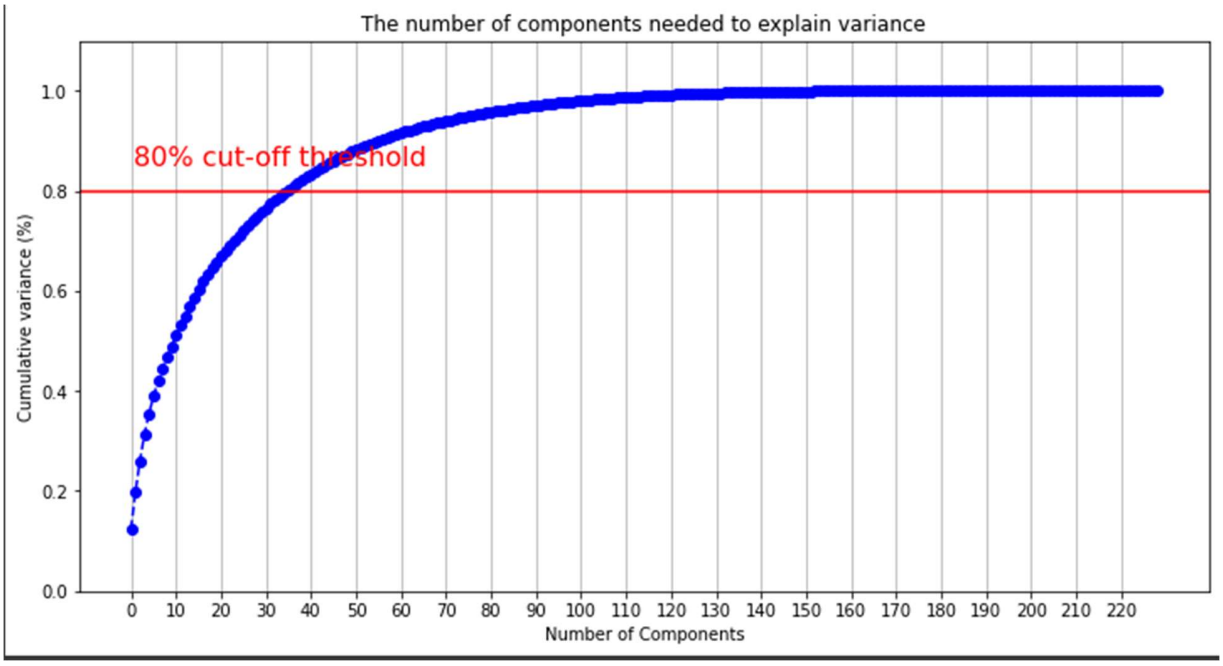
PCA:

```
[149] #PCA for dividng the dataset into samples and labels
      from sklearn.decomposition import PCA
      import matplotlib.pyplot as plt
      pca = PCA().fit(data2)
      plt.rcParams["figure.figsize"] = (12,6)
      fig, ax = plt.subplots()
      xi = np.arange(0, 350, step=1)
      y = np.cumsum(pca.explained_variance_ratio_)

      plt.ylim(0.0,1.1)
      plt.plot(xi, y, marker='o', linestyle='--', color='b'

      plt.xlabel('Number of Components')
      plt.xticks(np.arange(0, 352, step=8)) #change from 0-
```

PCA for dividng the dataset into samples and labels



Above 80% given the chosen variables.

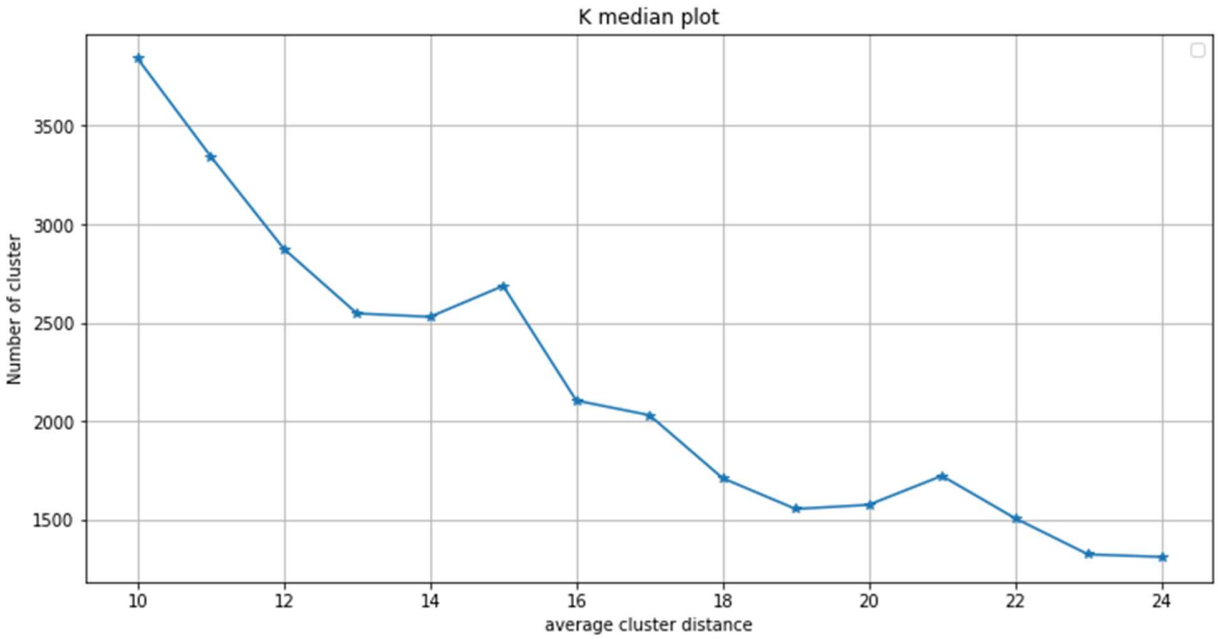
Clustering:

```
from pyclustering.cluster import cluster_visualizer_multidim
from pyclustering.cluster import cluster_visualizer
from pyclustering.cluster.kmedians import kmedians
def KMEDIAN(reduced,value_k):
  np.random.shuffle(reduced)
  k_median1=kmedians(reduced,np.copy(np.unique(reduced, axis=0)[:value_k]
  k_median1.process()
  clus_distance=k_median1.get_total_wce()
  return clus_distance
```

```
[153] # cluster distances store the avg within-cluster distance for k=[10,24]
cluster_distances=[]
idx=9
while(idx<24):
  y=PCA_data.copy()
  np.random.shuffle(y)
```

```
10 171911.61050250547
11 145902.64092623795
12 137395.08227156827
13 137588.85758527793
14 103532.46882942908
15 107360.68290355525
16 96323.8789072522
17 93933.96890667874
18 86427.5566445596
19 77061.94845265696
20 80307.31615346056
21 70047.35093479493
22 68022.22120427842
23 64545.59690010693
24 57102.264721945285
```

Results of cluster k=[10,24]



Graph for the same.

```
# k median clustering on best value of k = 22
y=PCA_data.copy()
np.random.shuffle(y)
k_median1=kmedians(y,np.copy(np.unique(y, axis=0)[:22]))
k_median1.process()
clus_distance=k_median1.get_total_wce()
cluster = k_median1.get_clusters()
md = k_median1.get_medians()
print(22,clus_distance/22)
```

```
22 68022.22120427842
```

The best came to be K=22

For more than 50K data:

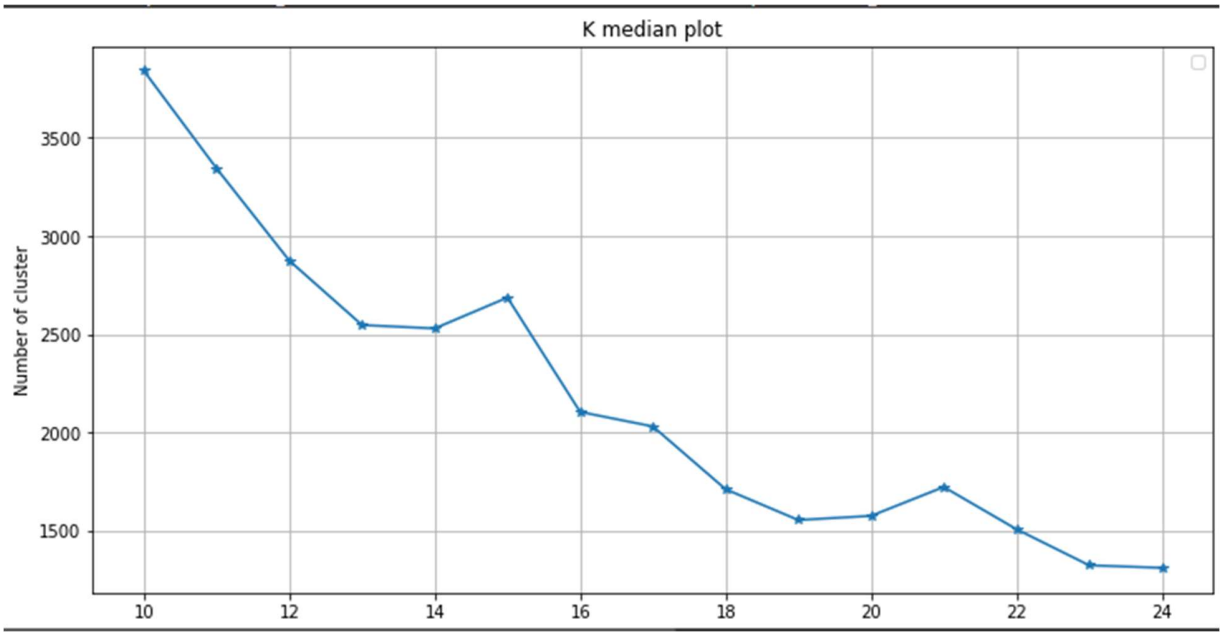
Did the same steps for

- Preprocessing
- Imputation, Bucketization, One Hot Encoding
- PCA

- Clustering

```
10 3840.561721914825
11 3343.884503455742
12 2874.9877779374856
13 2546.991111386026
14 2529.892327082505
15 2686.847915781711
16 2104.8624489308318
17 2031.0309795734836
18 1711.405723975964
19 1554.763595205172
20 1575.8041349436185
21 1722.9399564778537
22 1508.024250148708
23 1324.3417428943005
24 1311.5131007461061
```

Results of cluster k=[10,24]



Graph for the same.

```
cluster1 = k_median1.get_
md1 = k_median1.get_media
print(22,clus_distance/22
```

```
22 1508.024250148708
```

The best came for the k=22.