Importing Modules

```
!pip install xlrd==2.0.0
```

```
Collecting xlrd==2.0.0
  Downloading xlrd-2.0.0-py2.py3-none-any.whl (95 kB)
     |████████████████████████████████| 95 kB 3.1 MB/s
Installing collected packages: xlrd
  Attempting uninstall: xlrd
    Found existing installation: xlrd 1.1.0
    Uninstalling xlrd-1.1.0:
      Successfully uninstalled xlrd-1.1.0
Successfully installed xlrd-2.0.0
```

```
import pandas as pd
import xlrd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Preparing Datasets

bloodPressureIndia.csv

[ ] ↳ *2 cells hidden*

Datafile.xls

[ ] ↳ *4 cells hidden*

Test Plotting

[ ] ↳ *2 cells hidden*

Data.csv

[ ] ↳ *4 cells hidden*

Test Plotting

↳ *1 cell hidden*

Data (1).csv

```
#Read CSV
df4 = pd.read_csv("/content/data (1).csv")
df4
```

	Nutrients	Units	DNP(Rural)	DNP(Urban)	Pooled	RDA	NNMB (Rural)
0	Energy	Kcal	2321.0	2259.0	2308.0	2425.0	2172.0
1	Protein	gm	70.0	70.0	70.0	60.0	55.8
2	Fats	gm	31.3	39.5	33.0	0.0	31.2
3	Calcium	mg	631.5	673.4	640.1	400.0	528.0
4	Iron	mg	23.2	22.3	23.0	28.0	26.1
5	Thiamine	mg	1.9	1.9	1.9	1.2	1.1
6	Riboflavin	mg	1.0	1.0	1.0	1.4	0.8
7	Niacin	mg	19.7	18.8	19.5	16.0	13.5
8	Vitamin C	mg	55.2	62.4	56.7	40.0	34.5
9	Vitamin A (Retinol)	ug	355.3	356.0	355.4	600.0	288.0

```
df1 = df4.transpose()[2:]
df1
```

	0	1	2	3	4	5	6	7	8	9
DNP(Rural)	2321.0	70.0	31.3	631.5	23.2	1.9	1.0	19.7	55.2	355.3
DNP(Urban)	2259.0	70.0	39.5	673.4	22.3	1.9	1.0	18.8	62.4	356.0
Pooled	2308.0	70.0	33.0	640.1	23.0	1.9	1.0	19.5	56.7	355.4
RDA	2425.0	60.0	0.0	400.0	28.0	1.2	1.4	16.0	40.0	600.0
NNMB (Rural)	2172.0	55.8	31.2	528.0	26.1	1.1	0.8	13.5	34.5	288.0

```
cl = {0 : 'Energy', 1 : 'Protein', 2 : 'Fats', 3 : 'Calcium', 4 : 'Iron',
      5 : 'Thiamine', 6 : 'Riboflavin', 7 : 'Niacin', 8 : 'Vitamin C', 9 : 'Vitamin
df1 = df1.rename(columns = cl)
df1
```

	Energy	Protein	Fats	Calcium	Iron	Thiamine	Riboflavin	Niacin
DNP(Rural)	2321.0	70.0	31.3	631.5	23.2	1.9	1.0	19.7
DNP(Urban)	2259.0	70.0	39.5	673.4	22.3	1.9	1.0	18.8
Pooled	2308.0	70.0	33.0	640.1	23.0	1.9	1.0	19.5

```
dfl = dfl.drop(labels='Pooled', axis=0)
dfl = dfl.drop(labels='Fats', axis=1)
dfl
```

	Energy	Protein	Calcium	Iron	Thiamine	Riboflavin	Niacin	Vitam
DNP(Rural)	2321.0	70.0	631.5	23.2	1.9	1.0	19.7	
DNP(Urban)	2259.0	70.0	673.4	22.3	1.9	1.0	18.8	
RDA	2425.0	60.0	400.0	28.0	1.2	1.4	16.0	
NNMB (Rural)	2172.0	55.8	528.0	26.1	1.1	0.8	13.5	

```
filt = dfl.columns
row = dfl.index
mean = dfl.loc['RDA']
mean
```

```
Energy                2425.0
Protein                 60.0
Calcium                400.0
Iron                    28.0
Thiamine                 1.2
Riboflavin               1.4
Niacin                  16.0
Vitamin C               40.0
Vitamin A (Retinol)    600.0
Name: RDA, dtype: object
```

```
row
```

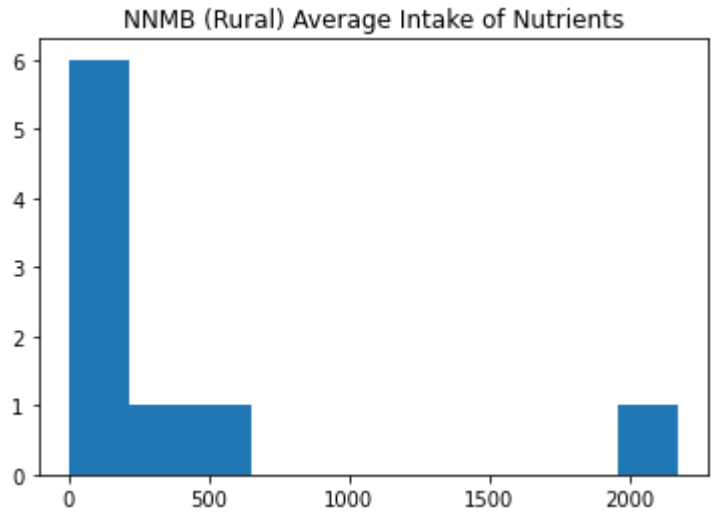
```
Index(['DNP(Rural)', 'DNP(Urban)', 'RDA', 'NNMB (Rural)'], dtype='object')
```

```
p = pd.DataFrame()
r = 0
for i in row:
  k = r = 0
  for j in filt:
    if i == 'RDA':
      break
    if dfl[j][i] < mean[r]:
      k = k + 1
      if k==5:
        p = p.append(dfl.loc[i])
        break
      r = r + 1
```

```
p
```

	Energy	Protein	Calcium	Iron	Thiamine	Riboflavin	Niacin	Vitan
NNMB (Rural)	2172.0	55.8	528.0	26.1	1.1	0.8	13.5	

```
plt.hist(p.loc['NNMB (Rural)'])
plt.title('NNMB (Rural) Average Intake of Nutrients')
plt.show()
```



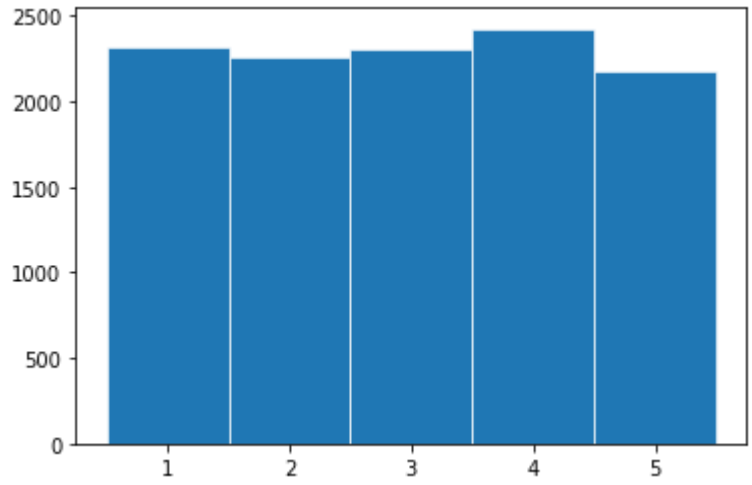
Confusion Matrix

[ ] ↳ *4 cells hidden*

Energy values in each area

```
col = df4["Nutrients"]
col
dft = df4.T
dft = dft.drop(["Nutrients", "Units"], axis=0)
dft

dict = {}
for i in range(0,10):
  dict[i] = col[i]

  dict

dft = dft.rename(dict, axis=1)
dft

x = [1,2,3,4,5]
y = dft["Energy"]
# plot
```

```
fig, ax = plt.subplots()

ax.bar(x, y, width=1, edgecolor="white", linewidth=0.7)

plt.show()
```




```
#Check for null values
df4.isnull().values.any()


df4.shape


#Rechecking for null values
df4.isnull().values.any()


df4.info()
```

Data (2).csv

```
#Read CSV
df1 = pd.read_csv("/content/data (2).csv")
df1.head()
```

	States/UTs	Area	Marasmus	Kwashiorkor	Bitots Spot	Corneal Xerosis	Cor Opa
0	Haryana	R	0.11	0.01	0.04	0.01	
1	Himachal Pradesh	R	0.08	0.01	0.01	0.00	
2	Punjab	R	0.00	0.00	0.12	0.00	
3	Rajasthan	C	0.03	0.01	0.22	0.16	
4	Rajasthan	R	0.04	0.01	0.25	0.18	

```
#Check for null values
dfl.isnull().values.any()
```

```
False
```

```
dfd = pd.DataFrame(columns = dfl.columns)
dfd
```

States/UTs	Area	Marasmus	Kwashiorkor	Bitots Spot	Corneal Xerosis	Corn Opac

```
filt = dfl['States/UTs'].unique()
filt
```

```
array(['Haryana', 'Himachal Pradesh', 'Punjab', 'Rajasthan', 'Chandigarh',
       'Delhi', 'Bihar', 'Sikkim', 'Orissa', 'Arunachal Pradesh', 'Assam',
       'Manipur', 'Meghalaya', 'Mizoram', 'Nagaland', 'Tripura',
       'Dardra & Nagar Haveli', 'Daman & Diu', 'Goa', 'Gujarat',
       'Maharashtra', 'Kerala', 'Karnataka', 'Tamil Nadu',
       'Andhra Pradesh', 'Madhya Pradesh*'], dtype=object)
```

```
#Pooling values of same states
df5 = pd.DataFrame()
for j in filt:
  d = dfl[dfl['States/UTs'].str.contains(j)]
  d = d.drop(['Area'], axis = 1)
  p = pd.DataFrame()
  p = pd.concat([p, d.sum(axis = 0, numeric_only=True)/len(d)], axis=0)
  p = p.transpose()
  p['States/UTs'] = j
  df5 = df5.append(p, ignore_index = True)

df5.head()
```

	Marasmus	Kwashiorkor	Bitots Spot	Corneal Xerosis	Corneal Opacity	Keratomal
0	0.110000	0.010000	0.040000	0.010000	0.00	0.00
1	0.080000	0.010000	0.010000	0.000000	0.00	0.00
2	0.000000	0.000000	0.120000	0.000000	0.00	0.01
3	0.023333	0.016667	0.186667	0.136667	0.02	0.07
4	0.030000	0.000000	0.000000	0.000000	0.00	0.00

```
df5.shape
```

```
(26, 12)
```

```
X = df5.loc[:, df5.columns != "States/UTs"]
```

```
Y = df5["States/UTs"]
```

```
from sklearn.preprocessing import StandardScaler
# Standardizing the features
X = StandardScaler().fit_transform(X)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents, columns = ['PC1', 'PC2'])
principalDf.head()
```

	PC1	PC2
0	0.568797	-0.590931
1	-0.972710	-0.113285
2	-1.414724	-0.279315
3	-0.832111	-0.838095
4	-1.529412	-0.824578

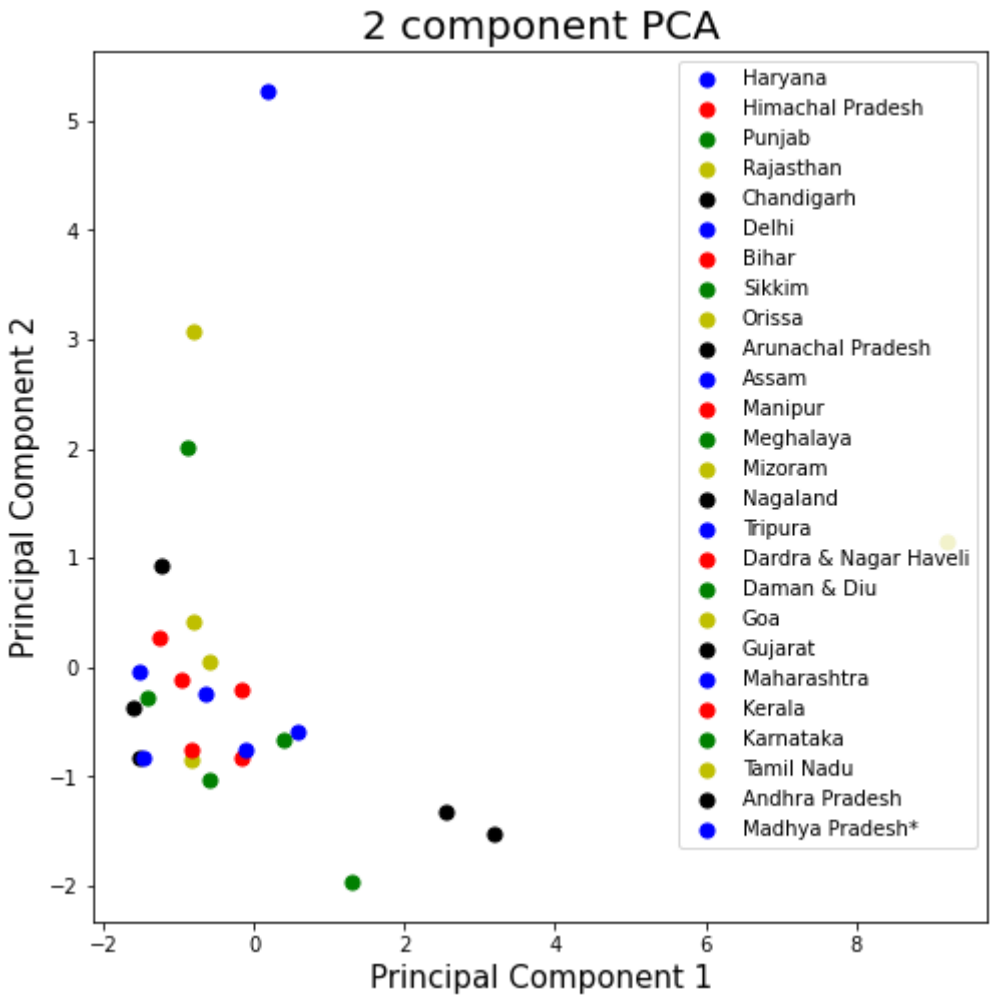
```
finalDf = pd.concat([principalDf, Y], axis = 1)
finalDf.head()
```

	PC1	PC2	States/UTs
0	0.568797	-0.590931	Haryana
1	-0.972710	-0.113285	Himachal Pradesh
2	-1.414724	-0.279315	Punjab
3	-0.832111	-0.838095	Rajasthan
4	-1.529412	-0.824578	Chandigarh

```
from itertools import cycle, islice
```

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
```

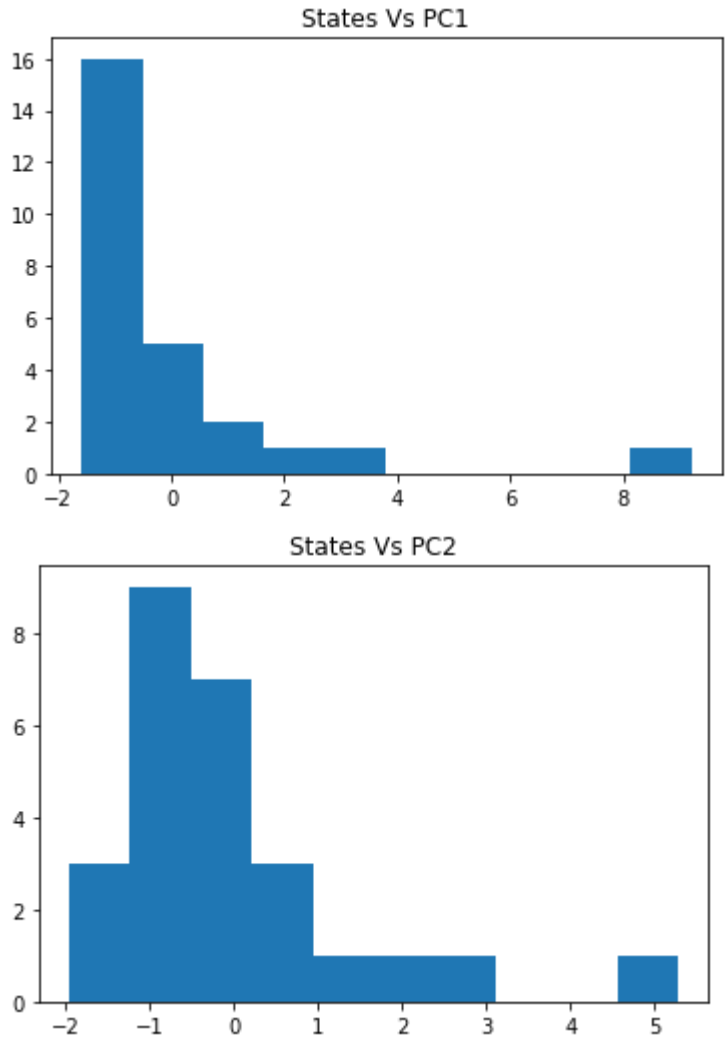
```
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = Y
colors = list(islice(cycle(['b', 'r', 'g', 'y', 'k']), None, len(df5)))
for target, color in zip(targets,colors):
  indicesToKeep = finalDf['States/UTs'] == target
  ax.scatter(finalDf.loc[indicesToKeep, 'PC1'], finalDf.loc[indicesToKeep, 'PC2'],
  ax.legend(targets)
  ax.grid()
```



From the above graph, Haryana has a much higher value and can be considered as an outlier. So the prevalence of nutritional deficiency in Haryana is much higher than other states.

Geographic

```
plt.hist(finalDf['PC1'])
plt.title('States Vs PC1')
plt.show()
plt.title('States Vs PC2')
plt.hist(finalDf['PC2'])
plt.show()
```



Data (3).csv

```
df6 = pd.read_csv("/content/data (3).csv")
df6.head()
```

	States/UTs	Area	Energy Kcal	Protein g.	Fats g.	Calcium Gm.	Phos. G
0	RDA	NaN	2425	60.0	NaN	400	Na
1	Haryana	R	2336	72.0	49.0	886	Na
2	Himachal Pradesh	R	2323	74.0	41.0	640	Na
3	Punjab	R	2341	77.0	34.0	966	Na
4	Rajasthan	C	2386	77.0	47.0	734	213(

```
#Check for null values
df6.isnull().values.any()
```

```
filt = df6['States/UTs']
filt
```

```
#Replace null values with 0
df6 = df6.fillna(0)
df6.head()
```

	States/UTs	Area	Energy Kcal	Protein g.	Fats g.	Calcium Gm.	Phos. G
0	RDA	0	2425	60.0	0.0	400	0
1	Haryana	R	2336	72.0	49.0	886	0
2	Himachal Pradesh	R	2323	74.0	41.0	640	0
3	Punjab	R	2341	77.0	34.0	966	0
4	Rajasthan	C	2386	77.0	47.0	734	213(

```
df6 = df6[['States/UTs',    'Area', 'Energy Kcal', 'Protein g.', 'Fats g.', 'Calciu
df6.head()
```

	States/UTs	Area	Energy Kcal	Protein g.	Fats g.	Calcium Gm.	Iron Gm
0	RDA	0	2425	60.0	0.0	400	28.
1	Haryana	R	2336	72.0	49.0	886	26.
2	Himachal Pradesh	R	2323	74.0	41.0	640	23.
3	Punjab	R	2341	77.0	34.0	966	28.
4	Rajasthan	C	2386	77.0	47.0	734	31.

```
dfl = pd.DataFrame()
for j in filt:
  d = df6[df6['States/UTs'].str.contains(j)]
  d = d.drop(['Area'], axis = 1)
```

```
    p = pd.DataFrame()
    p = pd.concat([p, d.sum(axis = 0, numeric_only=True)/len(d)], axis=0)
    p = p.transpose()
    p['States/UTs'] = j
    dfl = dfl.append(p, ignore_index = True)

dfl = dfl.drop_duplicates()
dfl
```

	Energy Kcal	Protein g.	Fats g.	Calcium Gm.	Iron Gm.	Thiamin Gm.	Ribo
0	2425.000000	60.000000	0.000000	400.000000	28.000000	1.200000	1.2(
1	2336.000000	72.000000	49.000000	886.000000	26.000000	2.400000	1.1(
2	2232.000000	74.000000	41.000000	640.000000	22.000000	2.100000	0.9(

```
filt = ['Energy Kcal', 'Protein g.', 'Fats g.', 'Calcium Gm.', 'Iron Gm.', 'Thiamin
mean = dfl.loc[0]
mean
```

```
Energy Kcal    2425.0
Protein g.       60.0
Fats g.           0.0
Calcium Gm.     400.0
Iron Gm.         28.0
Thiamin Gm.       1.2
Ribo Gm.          1.2
Niacin Gm.       16.0
Vit.c Gm.        40.0
Vit.A Ug.       600.0
States/UTs        RDA
Name: 0, dtype: object
```

22	2600.333333	77.333333	19.000000	694.000000	26.666667	1.766667	1.5:

```
dfl.shape
```

```
(27, 11)
```

21	2180.000000	76.000000	20.000000	878.000000	22.000000	1.700000	1.5(

```
ar = list(np.arange(27))
dfl = dfl.reset_index()
dfl
```

	index	Energy Kcal	Protein g.	Fats g.	Calcium Gm.	Iron Gm.	Thiamin Gm
0	0	2425.000000	60.000000	0.000000	400.000000	28.000000	1.20000
1	1	2336.000000	72.000000	49.000000	886.000000	26.000000	2.40000
2	2	2323.000000	74.000000	41.000000	640.000000	23.000000	2.10000
3	3	2341.000000	77.000000	34.000000	966.000000	28.000000	2.50000
4	4	2327.333333	74.333333	47.333333	732.333333	29.333333	2.50000
5	7	2470.000000	73.000000	64.666667	964.666667	22.333333	2.10000
6	10	2352.666667	75.000000	49.666667	693.333333	24.666667	2.26666
7	13	2465.666667	70.333333	25.333333	450.000000	22.333333	1.93333
8	16	2181.666667	66.333333	27.333333	564.000000	21.333333	1.60000
9	19	2106.000000	49.000000	13.000000	381.000000	27.000000	0.80000
10	20	1946.000000	91.000000	20.000000	1020.000000	21.000000	1.70000
11	21	1975.000000	51.000000	17.000000	364.000000	12.000000	0.70000
12	22	2600.333333	77.333333	19.000000	694.000000	26.666667	1.76666
13	25	1753.000000	62.000000	31.000000	776.666667	21.666667	1.13333
14	28	2112.333333	69.000000	43.000000	1083.666667	20.333333	1.36666
15	31	2189.000000	76.000000	30.000000	878.000000	23.000000	1.70000
16	32	2198.000000	77.666667	26.000000	1002.000000	24.333333	1.46666

```
dfl['Energy Kcal'][5]
```

```
2470.0
```

19	37	2046.333333	58.333333	40.000000	632.666667	19.000000	1.06666

```
p = pd.DataFrame()
r = 0
for i in range(1, 27):
  k = r = 0
  for j in filt:
    if dfl[j][i] < mean[r]:
      k = k + 1
      if k==5:
        p = p.append(dfl.loc[i], ignore_index = True)
        break
    r = r + 1

p = p.drop(['index'], axis=1)
p
```

	Energy Kcal	Protein g.	Fats g.	Calcium Gm.	Iron Gm.	Thiamin Gm.	Ribo
0	2336.000000	72.000000	49.0	886.000000	26.000000	2.400000	
1	2106.000000	49.000000	13.0	381.000000	27.000000	0.800000	
2	1975.000000	51.000000	17.0	364.000000	12.000000	0.700000	
3	1753.000000	62.000000	31.0	776.666667	21.666667	1.133333	
4	1934.000000	58.000000	25.0	506.000000	23.000000	1.300000	
5	2046.333333	58.333333	40.0	632.666667	19.000000	1.066667	
6	2298.000000	64.100000	44.1	536.000000	26.600000	1.700000	
7	2231.000000	57.100000	58.8	696.000000	22.800000	0.700000	
8	1814.000000	44.400000	20.1	455.000000	20.200000	0.770000	
9	2196.000000	55.500000	24.2	839.000000	30.600000	1.540000	
10	2430.000000	57.600000	28.3	518.000000	26.200000	0.870000	
11	2238.000000	57.900000	17.7	354.000000	27.000000	1.130000	

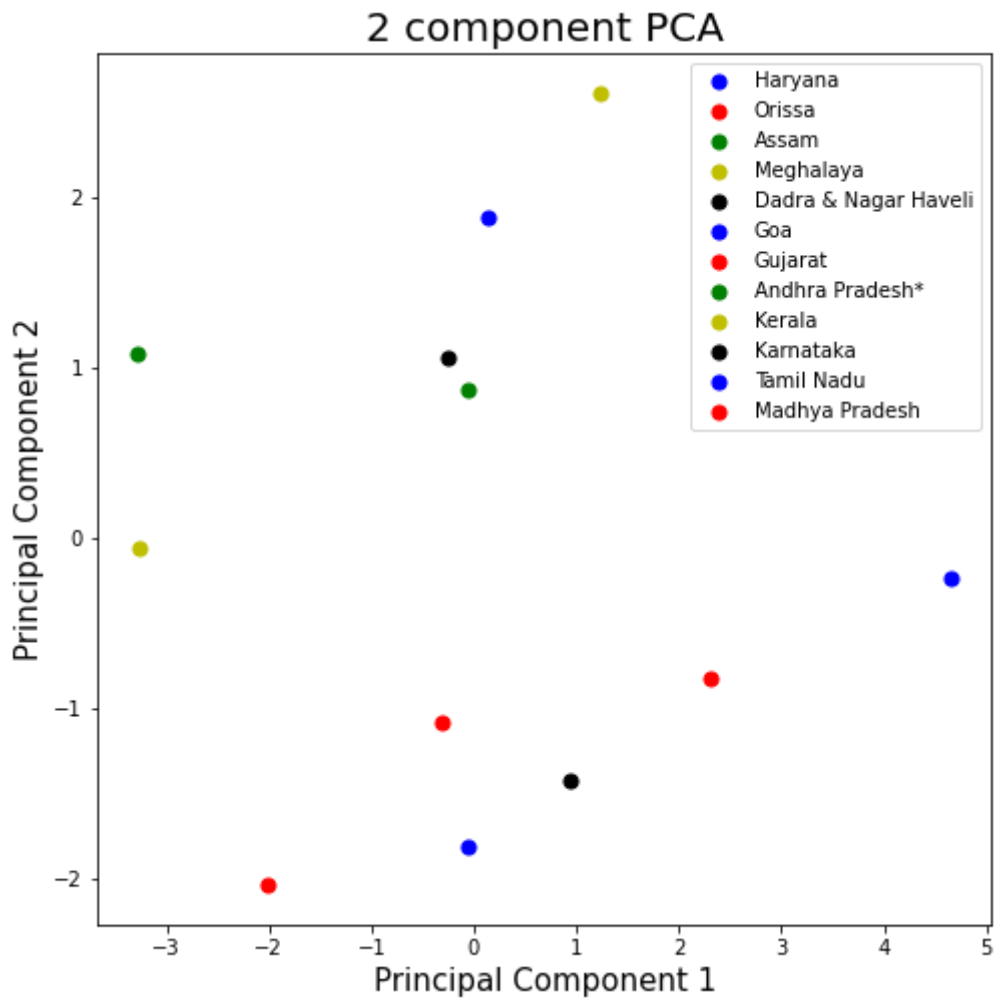
```
X = p.loc[:, p.columns != "States/UTs"]


Y = p['States/UTs']


from sklearn.preprocessing import StandardScaler
# Standardizing the features
X = StandardScaler().fit_transform(X)


from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents, columns = ['PC1', 'PC2'])
principalDf.head()
```

	PC1	PC2
0	4.655780	-0.241459
1	-2.012611	-2.039757
2	-3.280722	1.078474
3	1.229212	2.615224
4	-0.248439	1.056050

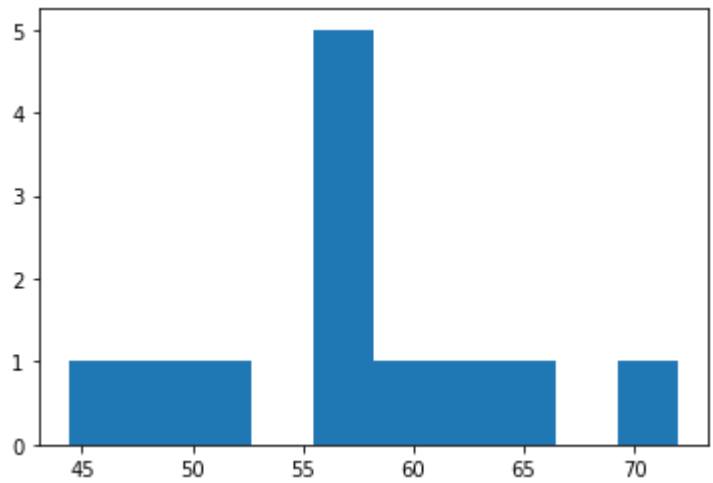
```
finalDf = pd.concat([principalDf, Y], axis = 1)
finalDf.head()
```

	PC1	PC2	States/UTs
0	4.655780	-0.241459	Haryana
1	-2.012611	-2.039757	Orissa
2	-3.280722	1.078474	Assam
3	1.229212	2.615224	Meghalaya

```
from itertools import cycle, islice

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = Y
colors = list(islice(cycle(['b', 'r', 'g', 'y', 'k']), None, len(p)))
for target, color in zip(targets,colors):
  indicesToKeep = finalDf['States/UTs'] == target
  ax.scatter(finalDf.loc[indicesToKeep, 'PC1'], finalDf.loc[indicesToKeep, 'PC2'],
  ax.legend(targets)
  ax.grid()
```



```
plt.hist(p["Protein g."])
plt.show()
```




```
X = df6.drop(["States/UTs"], axis = 1)

for i in range(0, len(df6)):
  if df6["Area"][i] == 'R':
    df6["Area"][i] = 1
  if df6["Area"][i] == 'C':
    df6["Area"][i] = 2
  if df6["Area"][i] == 'U':
    df6["Area"][i] = 3
for i in range(0, len(df6)):
  df6["Area"][i] = str(df6["Area"][i])
Y = df6["States/UTs"]
#+ " " + df6["Area"]
Y
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:11: SettingWithCo
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  # This is added back by InteractiveShellApp.init_path()
0                      RDA
1                  Haryana
2         Himachal Pradesh
3                   Punjab
4                Rajasthan
5                Rajasthan
6                Rajasthan
7                Chandigarh
8                Chandigarh
9                Chandigarh
10                    Delhi
11                    Delhi
12                    Delhi
13                    Bihar
14                    Bihar
15                    Bihar
16                   Sikkim
17                   Sikkim
18                   Sikkim
```

```
19                  Orissa
20       Arunachal Pradesh
21                   Assam
22                 Manipur
23                 Manipur
24                 Manipur
25               Meghalaya
26               Meghalaya
27               Meghalaya
28                 Mizoram
29                 Mizoram
30                 Mizoram
31                Nagaland
32                 Tripura
33                 Tripura
34                 Tripura
35             Daman & Diu
36    Dadra & Nagar Haveli
37                     Goa
38                     Goa
39                     Goa
40                 Gujarat
41             Maharashtra
42         Andhra Pradesh*
43                  Kerala
44               Karnataka
45              Tamil Nadu
46          Madhya Pradesh
Name: States/UTs, dtype: object
```

Model Based Methods

Linear Regression

```
import numpy as np
from sklearn.linear_model import LinearRegression

# y = 1 * x_0 + 2 * x_1 + 3

reg = LinearRegression().fit(X, Y.index)
print(reg.score(X, Y.index))

print(reg.coef_)

print(reg.intercept_)

#reg.predict(np.array([[3, 5]]))
```

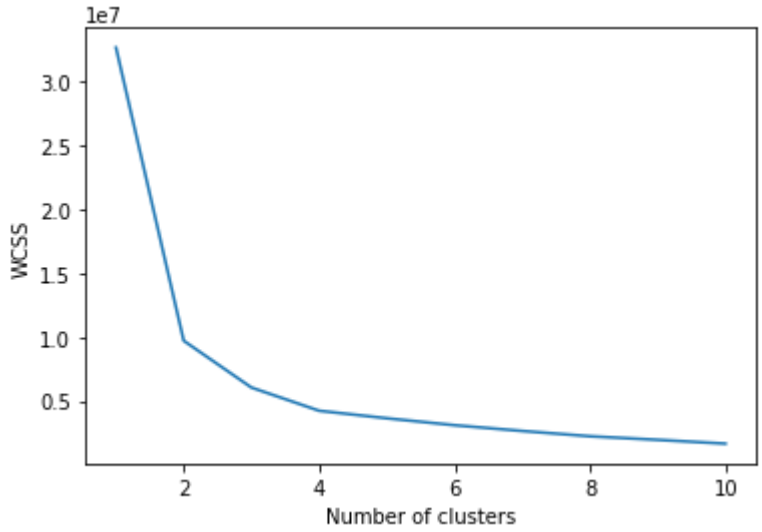
```
0.6952016491813778
[-1.19196305e+00 -1.65679804e-02  1.89262920e-01  3.12322603e-01
 -2.18281678e-03 -2.48301815e-03  1.87151137e+00 -2.29798528e+01
```

```
       8.30491153e+00 -2.23151139e-01  2.51349442e-01 -4.62109013e-02]
     32.955318352164895
```

Elbow Method

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
  kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
  kmeans.fit(X)
  wcss.append(kmeans.inertia_)
```

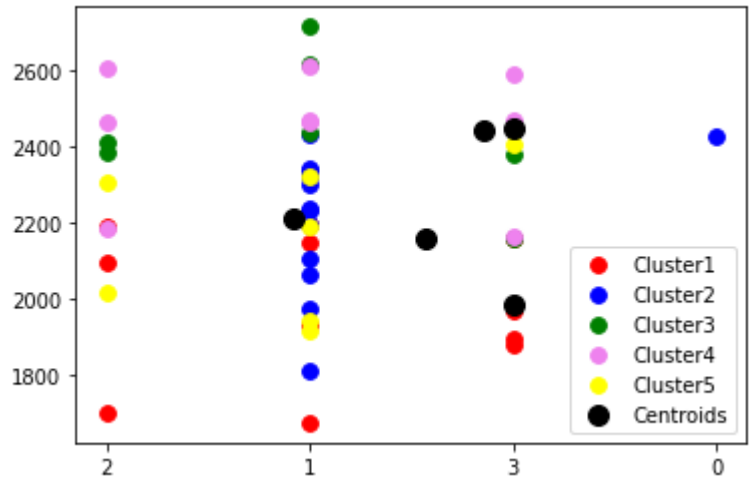
```
plt.plot(range(1, 11), wcss)
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
kmeans = KMeans(n_clusters = 5, init = "k-means++", random_state = 42)
y_kmeans = kmeans.fit_predict(X)
y_kmeans
```

```
array([1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 2, 3, 3, 3, 3, 0, 0, 0, 1, 4, 1,
       3, 3, 3, 0, 0, 0, 4, 4, 4, 4, 4, 4, 0, 3, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1], dtype=int32)
```

```
X= X.values
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 60, c = 'red', label = 'C
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 60, c = 'blue', label = '
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 60, c = 'green', label =
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 60, c = 'violet', label =
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 60, c = 'yellow', label =
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100,
plt.legend()

plt.show()
```




```
df6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   States/UTs   47 non-null     object 
 1   Area         47 non-null     object 
 2   Energy Kcal  47 non-null     int64  
 3   Protein g.   47 non-null     float64
 4   Fats g.      47 non-null     float64
 5   Calcium Gm.  47 non-null     int64  
 6   Phos. Gm.    47 non-null     float64
 7   Iron Gm.     47 non-null     float64
 8   Thiamin Gm.  47 non-null     float64
 9   Ribo Gm.     47 non-null     float64
 10  Niacin Gm.   47 non-null     float64
 11  Vit.c Gm.    47 non-null     float64
 12  Vit.A Ug.    47 non-null     int64  
dtypes: float64(8), int64(3), object(2)
memory usage: 4.9+ KB
```

```
df6.shape
```

```
from sklearn.preprocessing import StandardScaler
# Standardizing the features
X = StandardScaler().fit_transform(X)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents, columns = ['PC1', 'PC2'])
principalDf.head()
```

	PC1	PC2
0	-1.356101	-0.055981
1	0.520031	-1.397566
2	-0.288350	-0.745313
3	1.065647	-1.366777

```
finalDf = pd.concat([principalDf, Y], axis = 1)
finalDf.head()
```

	PC1	PC2	0
0	-1.356101	-0.055981	RDA 0
1	0.520031	-1.397566	Haryana R
2	-0.288350	-0.745313	Himachal Pradesh R
3	1.065647	-1.366777	Punjab R
4	1.995634	-2.319581	Rajasthan C