```
import pandas as pd
import numpy as np



#-----import dataset
datasales = pd.read_csv('Sales_Transactions_Dataset_Weekly.csv')


datasales = datasales[1:53]


print(datasales)
```

```
   Product_Code  W0  W1  ...  Normalized 49  Normalized 50  Normalized 51
1            P2   7   6  ...           0.10           0.60           0.00
2            P3   7  11  ...           0.45           0.45           0.36
3            P4  12   8  ...           0.35           0.29           0.35
4            P5   8   5  ...           0.53           0.33           0.40
5            P6   3   3  ...           0.27           0.91           0.55
6            P7   4   8  ...           0.40           0.20           0.10
7            P8   8   6  ...           0.08           0.50           0.50
8            P9  14   9  ...           0.60           0.27           0.67
9           P10  22  19  ...           0.58           0.38           0.46
10          P11  15   7  ...           0.31           0.31           1.00
11          P12   3   4  ...           0.44           0.67           0.33
12          P13  12  10  ...           0.58           0.47           0.26
13          P14  14  12  ...           0.45           0.41           0.68
14          P15  19  45  ...           0.45           0.58           0.03
15          P16  30  27  ...           0.28           0.03           0.00
16          P17  49  40  ...           0.37           0.26           0.23
17          P18  40  38  ...           0.32           0.32           0.11
18          P19  26  31  ...           0.34           0.28           0.34
19          P20  13  17  ...           0.27           0.67           0.47
                                       0.75           0.17           0.17
                                       0.18           0.59           0.24
                                       0.22           0.44           0.00
23          P24  36  42  ...           0.67           0.46           0.00
24          P25  26  28  ...           0.34           0.21           0.03
25          P26  14  14  ...           0.44           0.39           1.00
26          P27  44  34  ...           0.29           0.42           0.23
27          P28  34  32  ...           0.18           0.55           0.14
28          P29  13  10  ...           0.33           0.29           0.57
29          P30  46  36  ...           0.40           0.07           0.17
30          P31   7  17  ...           0.44           0.19           0.38
31          P32  15  13  ...           0.23           0.23           0.31
32          P33  15  12  ...           0.80           0.87           0.47
33          P34  47  42  ...           0.69           0.56           0.09
34          P35  34  37  ...           0.04           0.76           0.16
35          P36  41  32  ...           0.21           0.40           0.00
36          P37  36  39  ...           0.15           0.44           0.21
37          P38  37  36  ...           0.37           0.10           0.10
38          P39  31  21  ...           0.35           0.38           0.15
39          P40  41  27  ...           0.07           0.17           0.00
40          P41  35  27  ...           0.19           0.00           0.26
41          P42  42  27  ...           0.42           0.00           0.03
42          P43  28  43  ...           0.24           0.22           0.17
43          P44  34  27  ...           0.32           0.38           0.00
44          P45  40  29  ...           0.32           0.32           0.11
```

Saved successfully!

```
45          P46  27  46  ...          0.51          0.44          0.00
46          P47  40  42  ...          0.28          0.38          0.03
47          P48  29  51  ...          0.40          0.51          0.29
48          P49  37  28  ...          0.04          0.43          0.07
49          P50  12   3  ...          0.36          0.43          0.43
50          P51  19  14  ...          0.29          0.57          0.00
51          P52  40  44  ...          0.72          0.14          0.03
52          P53   2   5  ...          0.50          0.30          0.90

[52 rows x 107 columns]
```

```
datasales = datasales.drop(['Product_Code'], axis=1)
datasales.head()
```

	W0	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W1
1	7	6	3	2	7	1	6	3	3	3	2	2	6	2	0	6	2	7	
2	7	11	8	9	10	8	7	13	12	6	14	9	4	7	12	8	7	11	1
3	12	8	13	5	9	6	9	13	13	11	8	4	5	4	15	7	11	9	1
4	8	5	13	11	6	7	9	14	9	9	11	18	8	4	13	8	10	15	
5	3	3	2	7	6	3	8	6	6	3	1	1	5	4	3	5	3	5	1

5 rows × 106 columns

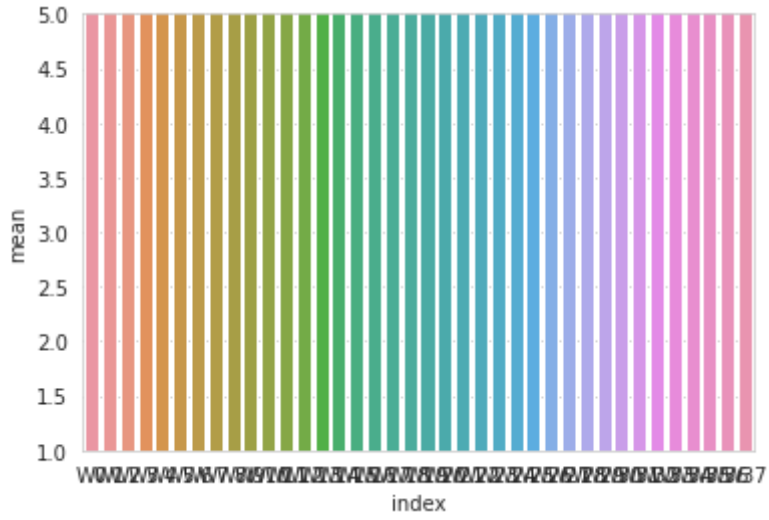
```
datasales.describe()
```

	W0	W1	W2	W3	W4	W5	W6	
			.000000	52.000000	52.000000	52.000000	52.000000	52.(
mean	23.673077	22.846154	22.788462	24.884615	24.884615	22.653846	24.711538	24.2
std	14.283847	14.439033	14.404978	16.104626	16.141110	15.385102	14.898791	15.(
min	2.000000	3.000000	1.000000	2.000000	3.000000	1.000000	2.000000	3.(
25%	12.000000	9.750000	9.750000	8.000000	9.750000	7.000000	9.750000	11.7
50%	24.000000	24.000000	25.000000	27.500000	24.000000	23.000000	27.000000	26.(
75%	36.250000	36.000000	37.500000	38.250000	38.000000	36.250000	38.250000	36.2
max	49.000000	51.000000	47.000000	55.000000	61.000000	50.000000	53.000000	58.(

8 rows × 106 columns


```
import seaborn as sns
sns.set_style('whitegrid')
import matplotlib.pyplot as plt
```

```
X_sales = datasales.iloc[:,0:38]
sales_means = X_sales.mean(axis = 0)
sales_means = sales_means.to_frame('mean')
sales_means.reset_index(level=0, inplace=True)
sns.barplot(x="index", y="mean", data=sales_means)
plt.ylim(1,5)
plt.show()
```




```
X = datasales.iloc[:,1:38]
X
```

Saved successfully!

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18
1	6	3	2	7	1	6	3	3	3	2	2	6	2	0	6	2	7	7
2	11	8	9	10	8	7	13	12	6	14	9	4	7	12	8	7	11	10
3	8	13	5	9	6	9	13	13	11	8	4	5	4	15	7	11	9	15
4	5	13	11	6	7	9	14	9	9	11	18	8	4	13	8	10	15	6
5	3	2	7	6	3	8	6	6	3	1	1	5	4	3	5	3	5	10
6	8	3	7	8	7	2	3	10	3	5	2	3	4	5	3	7	10	0
7	6	10	9	6	8	7	5	10	10	8	8	15	9	5	11	10	7	13
8	9	10	7	11	15	12	7	13	12	15	15	16	10	9	9	13	8	10
9	19	19	29	20	16	26	20	24	20	31	22	23	19	15	19	22	23	20
10	7	15	14	17	7	10	16	11	8	8	10	10	12	10	16	13	10	13
11	4	1	6	4	3	7	3	5	3	5	6	5	0	4	0	7	1	5
12	10	9	6	10	11	18	8	10	17	11	12	11	13	10	8	9	10	9
13	12	9	11	13	12	8	12	13	10	10	17	14	14	25	18	13	22	12
14	45	47	42	29	44	43	36	25	52	39	42	43	42	43	51	40	44	30
15	27	27	43	29	32	49	41	49	38	42	30	43	43	54	48	34	36	44
16	40	40	28	40	47	44	45	39	33	39	37	33	52	29	45	34	43	40
17	38	39	38	39	33	28	44	36	36	23	38	38	41	43	27	38	31	43
18	31	45	36	31	28	28	34	42	40	43	35	30	33	40	45	48	35	30
19	17	11	10	7	11	17	8	12	10	8	9	8	10	10	13	11	10	7
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	6	12	13	10	15	12	8	12	12	9	12
21	14	8	9	17	6	17	15	11	13	9	18	5	8	12	6	17	9	10
22	5	4	3	3	2	5	4	5	5	7	7	5	4	6	6	6	8	4
23	42	27	33	40	48	38	39	41	39	44	35	53	52	43	45	41	42	43
24	28	33	32	20	33	42	29	24	32	45	41	35	39	32	36	31	32	48
25	14	9	8	9	7	9	11	10	14	16	7	15	8	18	8	11	12	9
26	34	33	39	34	30	47	27	45	39	47	39	35	47	29	45	52	38	51
27	32	36	41	31	31	32	29	43	33	37	44	37	27	36	38	24	36	30
28	10	12	17	17	11	15	16	10	12	18	6	14	12	11	10	19	25	9
29	36	45	34	35	36	43	28	26	33	46	42	41	42	37	38	40	35	29
30	17	6	7	9	7	11	10	13	9	4	15	10	7	10	9	11	8	14
31	13	10	4	13	7	10	12	11	15	5	15	8	10	7	16	11	10	7
32	12	11	17	10	18	11	16	8	6	11	10	18	10	14	13	13	12	14
33	42	24	55	42	23	41	51	45	29	51	40	28	48	35	40	38	36	53

Saved successfully!

33	42	24	55	42	23	41	51	45	29	51	40	28	48	35	40	38	36	53
34	37	26	27	49	48	36	34	28	41	41	35	29	42	41	31	38	27	36
35	32	39	45	38	40	39	39	27	48	47	35	61	34	37	47	30	46	41
36	39	43	42	38	37	31	26	36	29	31	36	46	25	51	44	42	52	40

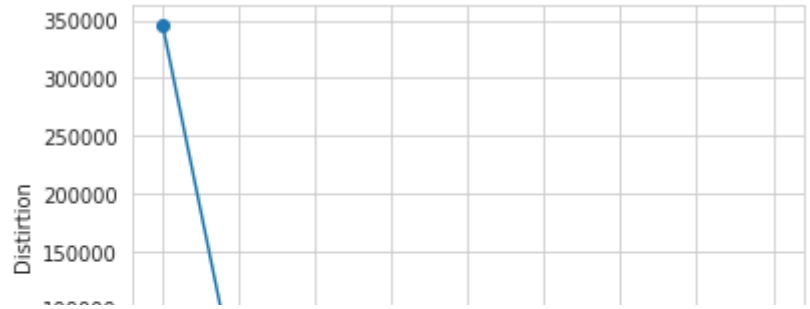
```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2, random_state=1)
X_pca = pca.fit_transform(X)
print('Explained Variance Ratio : ' + str(pca.explained_variance_ratio_.cumsum()[1]))
```

```
Explained Variance Ratio : 0.8841665661344688
```

```
from sklearn.cluster import KMeans
import collections
from sklearn.metrics import silhouette_samples
from matplotlib import cm

distortions = []
K_to_try = range(1, 10)
for i in K_to_try:
    model = KMeans(
            n_clusters=i,
            init='k-means++',
            random_state=1)
    model.fit(X_pca)
    distortions.append(model.inertia_)
plt.plot(K_to_try, distortions, marker='o')
plt.xlabel('Number Of Clusters (k)')
plt.ylabel('Distirtion')
plt.show()
```

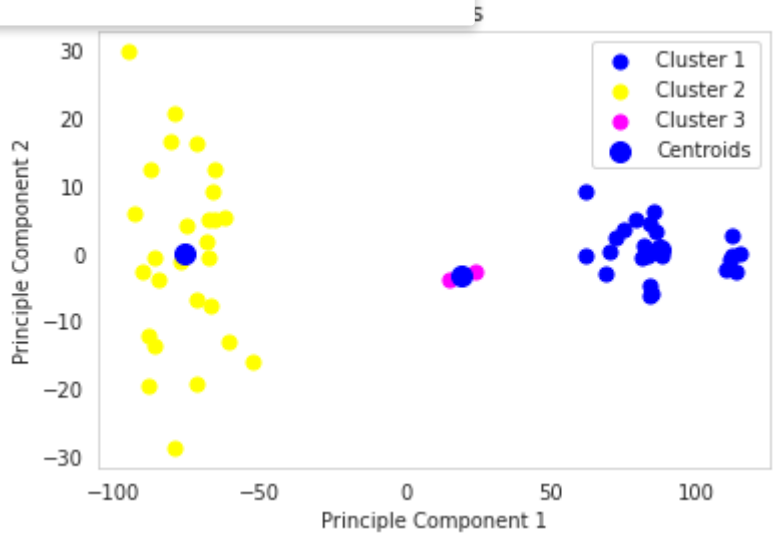
Saved successfully!

```
print(collections.Counter(y))
```




```
model = KMeans(
    n_clusters=3,
    init='k-means++',
    random_state=1)

model = model.fit(X_pca)

y = model.predict(X_pca)

plt.scatter(X_pca[y == 0, 0], X_pca[y ==0, 1], s = 50, c = 'blue', label = 'Cluster 1')
plt.scatter(X_pca[y == 1, 0], X_pca[y ==1, 1], s = 50, c = 'yellow', label = 'Cluster 2')
plt.scatter(X_pca[y == 2, 0], X_pca[y ==2, 1], s = 50, c = 'magenta', label = 'Cluster 3')

plt.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:, 1], s = 100, c = 'blue
plt.title('Clusters')
plt.xlabel('Principle Component 1')
plt.ylabel('Principle Component 2')
plt.legend()
plt.grid()
plt.show()

print('K Means Result : ')
print(collections.Counter(y))
```

Saved successfully!



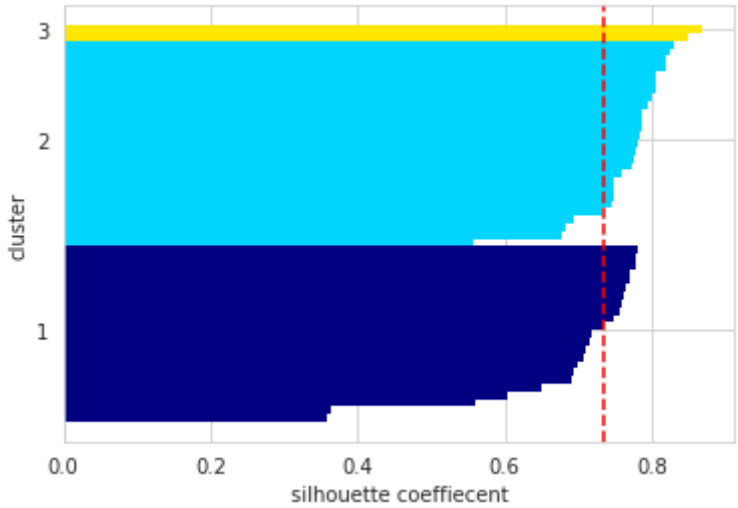

```
K Means Result :
Counter({1: 27, 0: 23, 2: 2})
```

```
cluster_labels = np.unique(y)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(X_pca, y, metric='euclidean')
```

```
y_ax_lower, y_ax_upper = 0,0
yticks = []

for i, c in  enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper),
          c_silhouette_vals,
          height=1.0,
          edgecolor='none',
          color=color)
    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)
silhouette_avg = np.mean(silhouette_vals)

plt.axvline(silhouette_avg, color="red", linestyle="--")
plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('cluster')
plt.xlabel('silhouette coeffiecent')
plt.show()

#menggunakan k dari metode elbow
model_k = KMeans(
  n_clusters=3,
  init='k-means++',
  random_state=1)

#fit with x instead of x_pca
model_k = model_k.fit(X)
```

Saved successfully!

```
print('Final K Meeans Resulting (No PCA): ')
print((collections.Counter(y_final)))

y_final = pd.DataFrame(y_final, columns=['cluster'])

raw_result = pd.concat([X, y_final], axis=1)

y = pd.DataFrame(y, columns=['cluster'])
raw_result_pca = pd.concat([X, y], axis=1)

mean1 = raw_result[raw_result['cluster']==0].iloc[:, 1:38].mean(axis = 1)
mean2 = raw_result[raw_result['cluster']==1].iloc[:, 1:38].mean(axis = 1)
mean3 = raw_result[raw_result['cluster']==1].iloc[:, 1:38].mean(axis = 1)

print('Mean Cluster 1 : ' + str(mean1.mean()) + ',STD :' + str(mean1.std()))
print('Mean Cluster 2 : ' + str(mean2.mean()) + ',STD :' + str(mean2.std()))
print('Mean Cluster 3 : ' + str(mean3.mean()) + ',STD :' + str(mean3.std()))
```




```
Final K Meeans Resulting (No PCA):
Counter({1: 27, 0: 23, 2: 2})
Mean Cluster 1 : 14.307873090481786,STD :11.62226082217493
Mean Cluster 2 : 29.347347347347345,STD :10.089557125481385
Mean Cluster 3 : 29.347347347347345,STD :10.089557125481385
```

Saved successfully!

✓ 0s completed at 5:12 PM