In [1]:

```
#K-MEANS CLUSTERING FOR IRIS DATASET
```

In [1]:

```
#Multi-layer perceptron for iris dataset
#Loading the libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
```

In [4]:

```
#Loading the dataset
df4 = pd.read_csv('C:\\Users\\G.Madhu mitha\\Downloads\\Iris.csv')
print(df4.head())
#Splitting the dataset into data and target attributes
X = df4.iloc[:, :-1]
y = df4.iloc[:, -1]
```

```
   sepal length  sepal width  petal length  petal width      Species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

In [5]:

```
le = LabelEncoder()
y = le.fit_transform(y)
```

In [7]:

```
#Splitting the data into train ans test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.20)
```

In [8]:

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [9]:

```
#Training the perceptron model
mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
mlp.fit(X_train, y_train)
```

Out[9]:

```
MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
```

In [11]:

```
#Making predictions on test data
predictions = mlp.predict(X_test)
```

In [13]:

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
```

```
[[12  0  0]
 [ 0  8  1]
 [ 0  1  8]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.89      0.89      0.89         9
           2       0.89      0.89      0.89         9

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30
```

In [15]:

```
#K-means
import pandas as pd
df = pd.read_csv("C:\\Users\\G.Madhu mitha\\OneDrive\\Documents\\.WINTER SEM 2022-23\\ML
df
```

Out[15]:

	Point	x	y
0	P1	2	5
1	P2	3	4
2	P3	3	5
3	P4	4	4
4	P5	4	5
5	P6	5	3
6	P7	5	4

In [30]:

```
x = df.iloc[:,1:2]
x
```

Out[30]:

	x
0	2
1	3
2	3
3	4
4	4
5	5
6	5

In [31]:

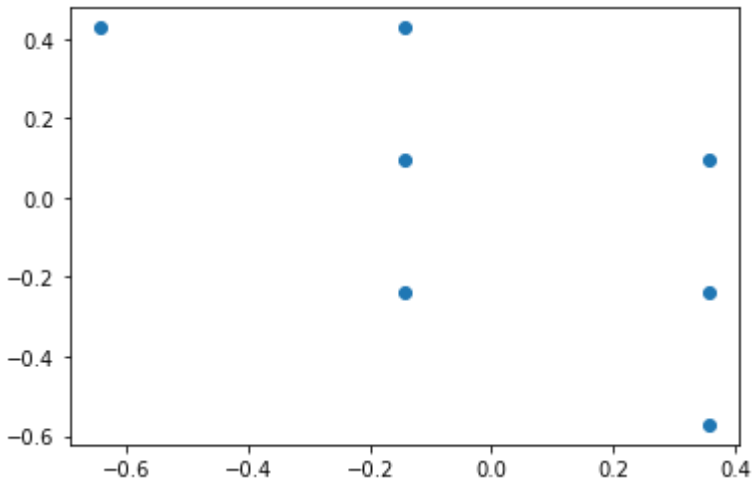
```
y = df.iloc[:,2:3]
y
```

Out[31]:

	y
0	5
1	4
2	5
3	4
4	5
5	3
6	4

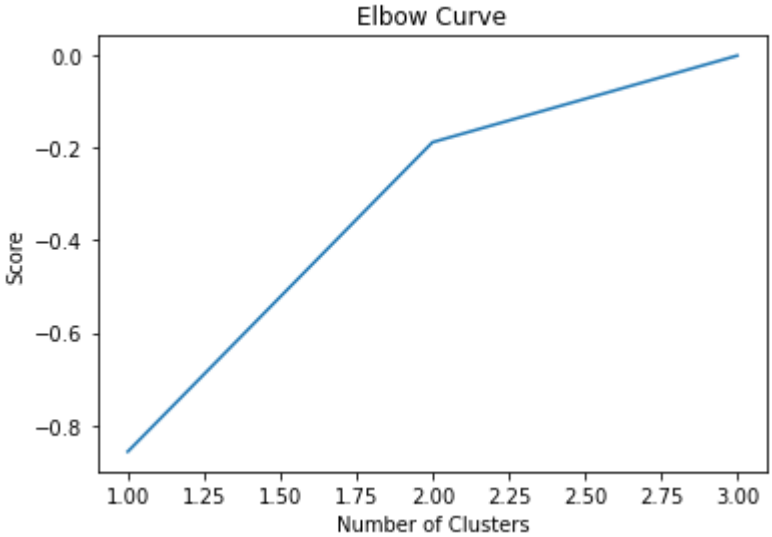
In [32]:

```
import pylab as pl
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
x_norm = (x - x.mean()) / (x.max() - x.min())
y_norm = (y - y.mean()) / (y.max() - y.min())
pl.scatter(y_norm,x_norm)
pl.show()
```



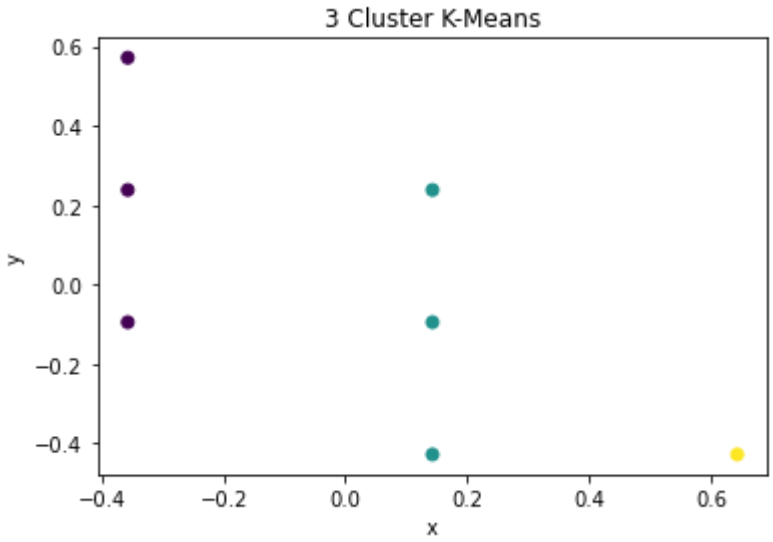
In [35]:

```
Nc = range(1, 4)
kmeans = [KMeans(n_clusters=i) for i in Nc]
kmeans
score = [kmeans[i].fit(y_norm).score(y_norm) for i in range(len(kmeans))]
score
pl.plot(Nc,score)
pl.xlabel('Number of Clusters')
pl.ylabel('Score')
pl.title('Elbow Curve')
pl.show()
```



In [40]:

```
#Principal Component Analysis and K-Means
pca = PCA(n_components=1).fit(y_norm)
pca_d = pca.transform(y_norm)
pca_c = pca.transform(x_norm)
kmeans=KMeans(n_clusters=3)
kmeansoutput=kmeans.fit(y_norm)
kmeansoutput
pl.figure('6 Cluster K-Means')
pl.scatter(pca_d[:, 0], pca_c[:, 0], c=kmeansoutput.labels_)
labels=kmeansoutput.labels_
labels
pl.xlabel('x')
pl.ylabel('y')
pl.title('3 Cluster K-Means')
pl.show()
```



In [ ]: