In [1]:

```
# Tejas Acharya
# 06-06-2023
# EE-541
# Homework 02
# Problem 02
```

In [2]:

```
#Importing Libraries
import re
import csv
import numpy as np
from numpy.linalg import norm
import matplotlib.pyplot as plt
from scipy.cluster.vq import kmeans
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

In [3]:

```
#Constants
MICKEY_FILENAME = './mickey.csv'
K = 3
KMEANS_LABELS_FILENAME = '../gmm/kmeans_label.csv'
```

In [4]:

```
def get_data(filename):
    data = []
    labels_dict = {'head' : 2, 'ear_left' : 1, 'ear_right' : 3}

    with open(filename, 'r') as f:
        reader = csv.reader(f)

        for row in reader:
            if not re.search('^#', row[0]):
                actual_row = row[0].split()
                actual_row[0] = float(actual_row[0])
                actual_row[1] = float(actual_row[1])
                actual_row[-1] = labels_dict[actual_row[-1].lower()]
                data.append(actual_row)

    data = np.array(data)
    X = data[:, :-1]
    y = data[:, -1]

    return (X, y)
```

In [5]:

```
class KMeans():
    def __init__(self):
        self.centroids = None

    def fit(self, X, k):
        self.centroids, _ = kmeans(X, k)
        sorted_centroids = []
        for i in np.sort(self.centroids[:, 0], axis=0):
            sorted_centroids.append(list(self.centroids[self.centroids[:, 0

        sorted_centroids = np.array(sorted_centroids)
        self.centroids = sorted_centroids

        return

    def predict(self, X):
        y_predict = []

        for i in X:
            distance = np.empty((self.centroids.shape[0], ))
            for j in range(len(self.centroids)):
                distance[j] = norm(i - self.centroids[j])
            y_predict.append(np.argmin(distance) + 1)

        y_predict = np.array(y_predict)

        return y_predict
```

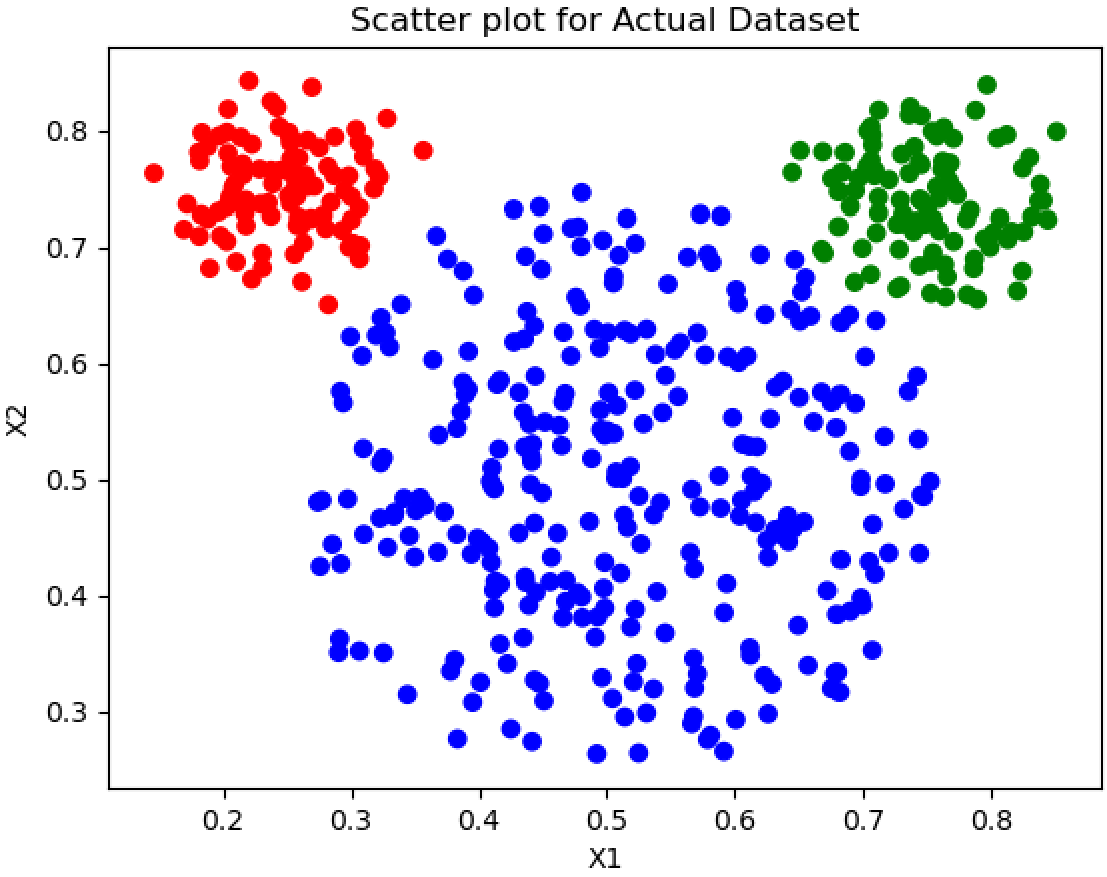
In [6]:
```
def get_label_colors(y):
    colors_dict = {2 : 'b', 1 : 'r', 3 : 'g'}
    y_colors = []

    for i in y:
        y_colors.append(colors_dict[int(i)])

    return y_colors
```

In [7]:
```
def write_kmeans_label(filename, y):
    with open(filename, 'a') as file:
        for i in y:
            file.write(f'{i},')
    return
```

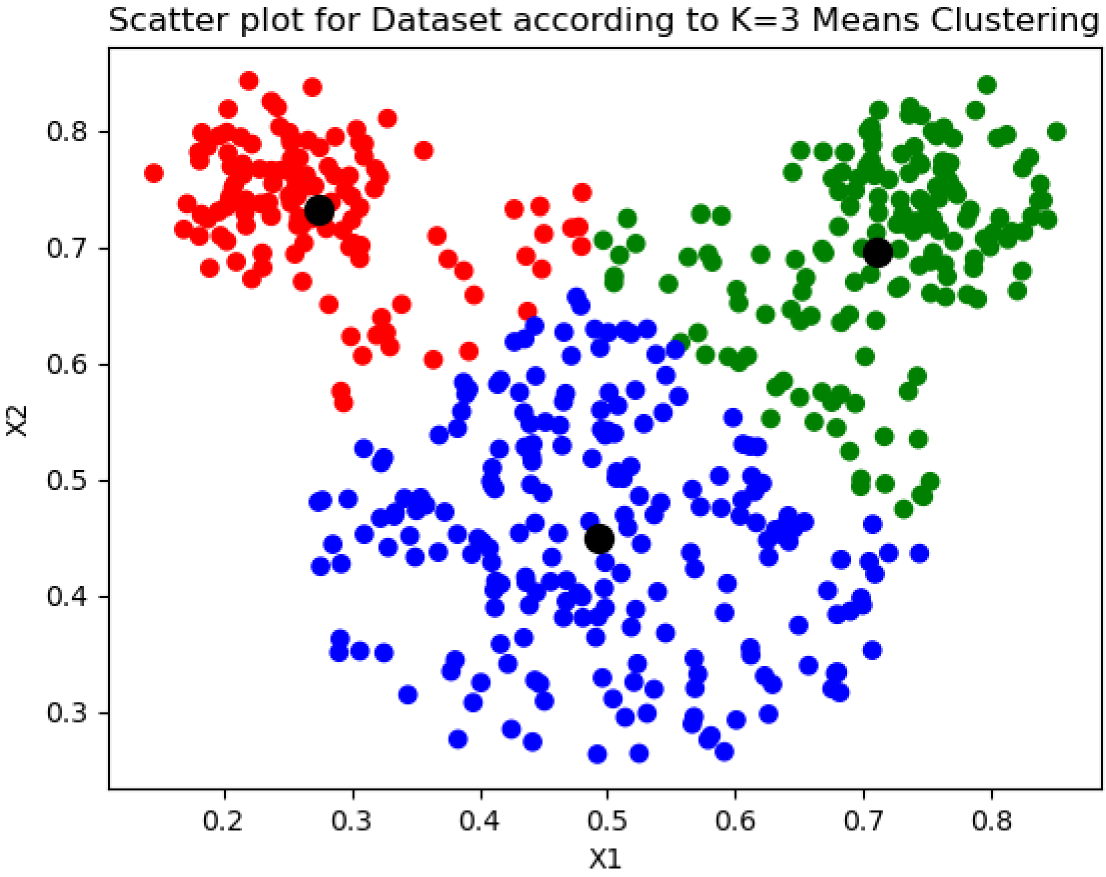
In [8]:
```
def main():
    X, y = get_data(MICKEY_FILENAME)
    model = KMeans()
    model.fit(X, K)
    y_predict = model.predict(X)

    centroids = model.centroids

    print('#' * 50)

    plt.figure()
    plt.scatter(X[:, 0], X[:, 1], c = get_label_colors(y))

    plt.xlabel('X1')
    plt.ylabel('X2')
    plt.title('Scatter plot for Actual Dataset')

    plt.show()

    print('#' * 50)

    plt.figure()
    plt.scatter(X[:, 0], X[:, 1], c = get_label_colors(y_predict))
    plt.scatter(centroids[:, 0], centroids[:, 1], c='k', s=100)

    plt.xlabel('X1')
    plt.ylabel('X2')
    plt.title(f'Scatter plot for Dataset according to K={K} Means Clusterin

    plt.show()

    print('#' * 50)

    cMatrix = confusion_matrix(y_true=y, y_pred=y_predict, normalize='true'
    print(f'Confusion Matrix: \n{cMatrix}')

    plt.figure()
    cm_display = ConfusionMatrixDisplay(confusion_matrix=cMatrix, display_l
    print('#' * 50)

    print('Confusion Matrix for K-Means Clustering\n')
    cm_display.plot()
    plt.show()

    write_kmeans_label(KMEANS_LABELS_FILENAME, y_predict)
```

In [9]:
```
main()
```

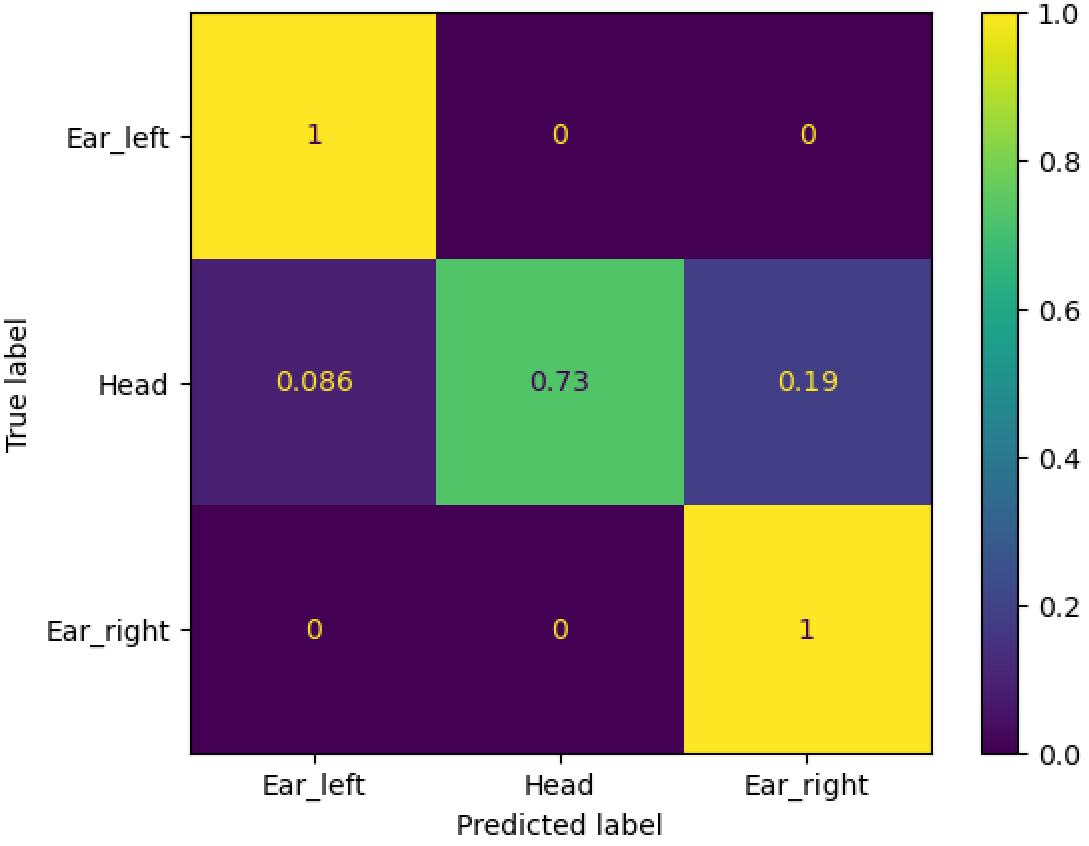
```
##################################################
```



```
##################################################
```



```
##################################################
Confusion Matrix:
[[1.         0.         0.        ]
 [0.0862069  0.72758621 0.1862069 ]
 [0.         0.         1.        ]]
##################################################
Confusion Matrix for K-Means Clustering

<Figure size 640x480 with 0 Axes>
```


Ear_left
Head
Ear_right
True label
Ear_left
Head
Ear_right
Predicted label