1. **Data:**
   The data that you need for this coursework is in a single CSV file called **Iris.csv**. Within the CSV file, each row is formatted as follows:
   1. Sepal length
   2. Sepal width
   3. Petal length
   4. Petal width
   5. Class

2. **Task**
   2.2. PCA
   In this task you have to apply the PCA on the provided dataset and you have to explain the terms like how you find number of k-components?
   2.3. Clustering
   Apply K-mean clustering on the provided dataset and find the ideal number of clusters, also explain the terminology how you choose the value of K? Visualize the cluster after training.
3. **Distribution of Marks**

   The assignment will be evaluated on the basis of your performance and errors. The general distribution will be as follows:

1. Task 1.	(10)
2. Task 2.	(10)

Task 2.2:

The function of PCA works as following:

1. The data points are centered by subtracting the mean value from them

2. The covariance matrix is generated by calculating the variances and covariances of the centered data.

3. Eigenvalues and eigenvectors are generated from the covariance matrix and the following equations: $| A - (\Omega \times I) | = 0$ and $A - (\Omega \times I)$.

4. Eigenvalues are sorted and the largest K eigenvalues are picked as these values will retain the most amount of data after transformation.

5. Pick the corresponding eigenvectors of the selected eigenvalues and then take the dot product of the transpose of the eigenvectors with the transpose of the centered data points.

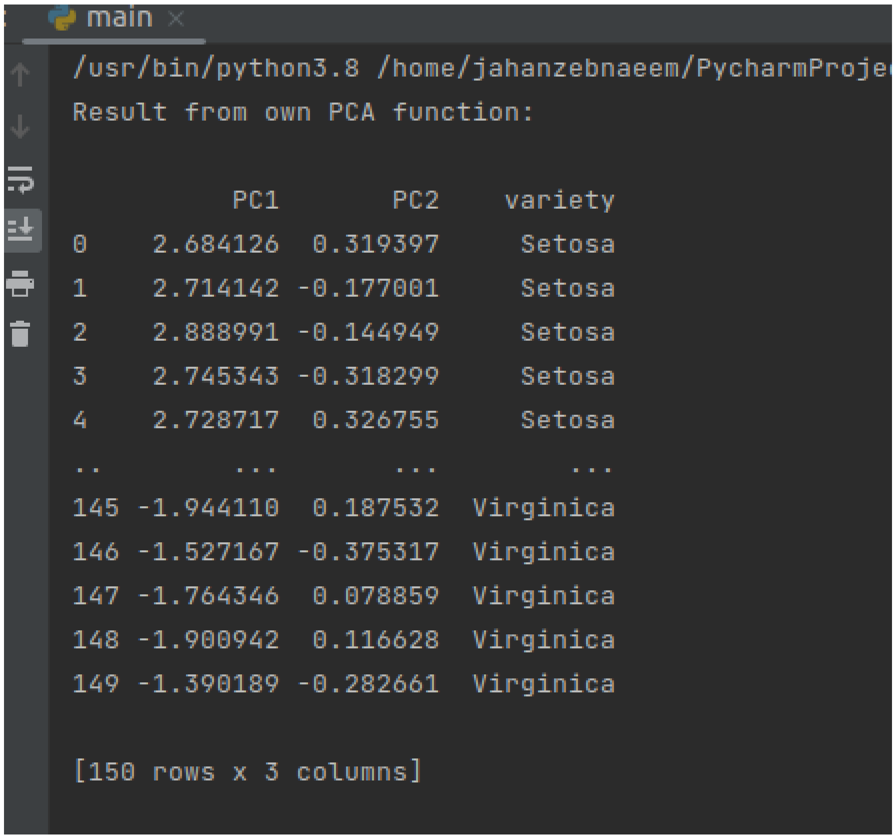
6. resultant data points will be the principle component values.

The K components in PCA are selected by check how many attributes exist in the original dataset and how many attributes we want to have in our transformed dataset. If we want 1 column, then K will be 1, if we want 2 columns, then K will be 2, and so on.

Code:

```
import pandas as pd
import numpy as np

def PCA_func(X, num_components):

    #Step 1: centering data
    X_centered = X - np.mean(X, axis=0)

    #Step 2: calculate covariance matrix
    cov_mat = np.cov(X_centered, rowvar=False)

    #Step 3: find eigen values and corresponding eigen vectors
    eigen_values, eigen_vectors = np.linalg.eigh(cov_mat)

    #Step 4: sorting eigen values and eigen vectors to find N most significant eigen values and their corresponding
eigen vectors
    sorted_index = np.argsort(eigen_values)[::-1]          #Descending order
    sorted_eigenvalue = eigen_values[sorted_index]
    sorted_eigenvectors = eigen_vectors[:, sorted_index]

    #Step 5: selecting 'num_components' subset from the eigen vectors
    eigenvector_subset = sorted_eigenvectors[:, 0:num_components]

    #Step 6: performing dot product to reduce data
    X_reduced = np.dot(eigenvector_subset.transpose(), X_centered.transpose()).transpose()

    return X_reduced


path = r'/home/jahanzebnaeem/Downloads/DM/Project-20092022-124505am/iris.csv'
data = pd.read_csv(path)

x = data.iloc[:, 0:4]
target = data.iloc[:, 4]
reduced_matrix = PCA_func(x, 2)

final_df = pd.DataFrame(reduced_matrix, columns=['PC1', 'PC2'])
final_df = pd.concat([final_df, pd.DataFrame(target)], axis=1)
print('Result from own PCA function:\n')
print(final_df, '\n\n')
```

Screenshot:



```
/usr/bin/python3.8 /home/jahanzebnaeem/PycharmProje
Result from own PCA function:

          PC1       PC2    variety
0    2.684126  0.319397     Setosa
1    2.714142 -0.177001     Setosa
2    2.888991 -0.144949     Setosa
3    2.745343 -0.318299     Setosa
4    2.728717  0.326755     Setosa
..        ...       ...        ...
145 -1.944110  0.187532  Virginica
146 -1.527167 -0.375317  Virginica
147 -1.764346  0.078859  Virginica
148 -1.900942  0.116628  Virginica
149 -1.390189 -0.282661  Virginica

[150 rows x 3 columns]
```

Task 2.2

In the visualizations, it can be seen that 1 and 2 are too low for the K value because the points clearly show that there are more clusters in the dataset. With 3 as the K value, we can see that the value is just right because intra class similarity is maximized while inter class similarity is minimized (using distance as the measure). With 4 and 5 as K value we can see that the clusters are really close to each other and there's even some overlap which means there are high chances of wrong cluster assignment.

Code:

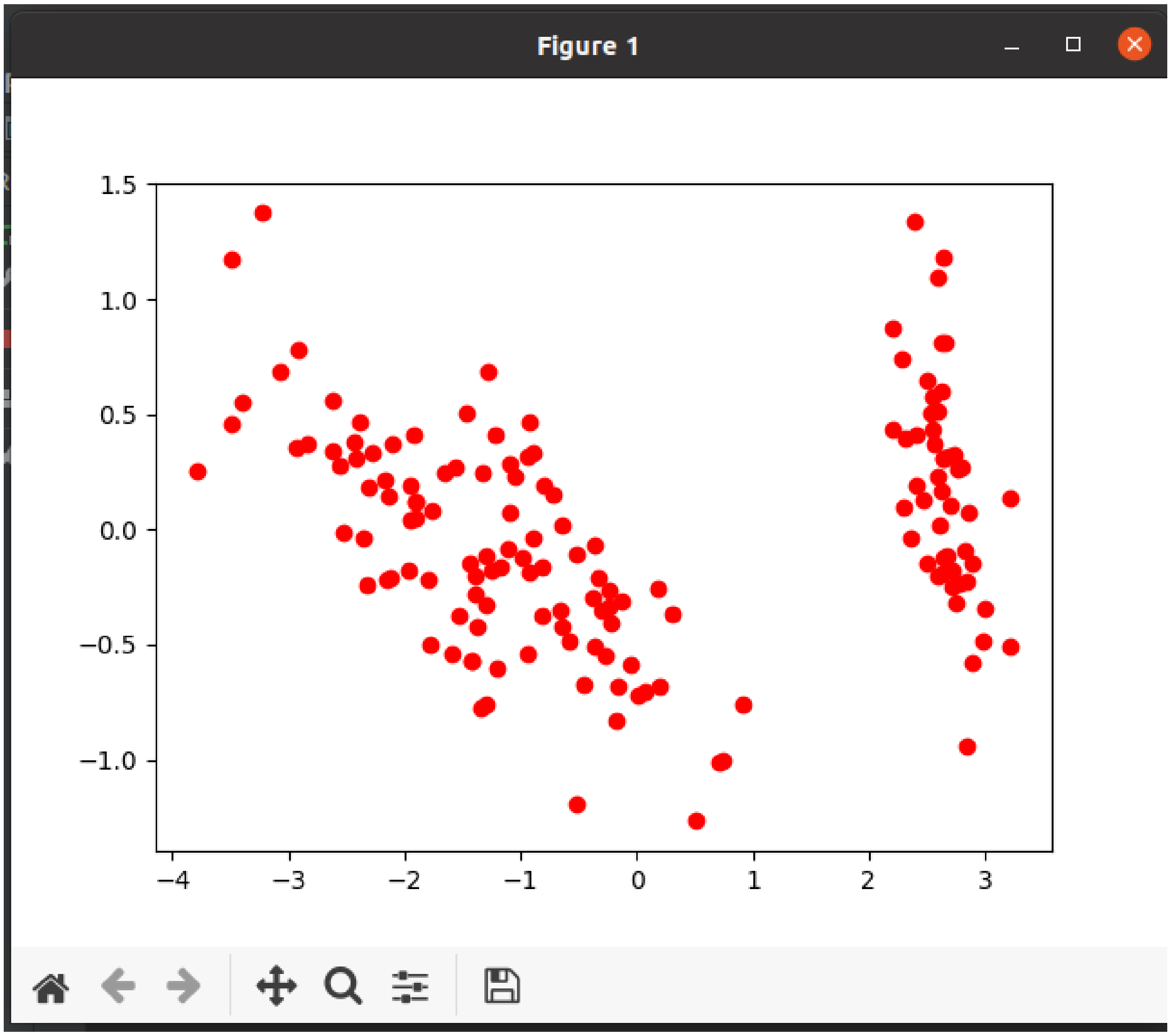
```
import pandas as pd
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt

def PCA_func(X, num_components):

    #Step 1: centering data
    X_centered = X - np.mean(X, axis=0)

    #Step 2: calculate covariance matrix
    cov_mat = np.cov(X_centered, rowvar=False)

    #Step 3: find eigen values and corresponding eigen vectors
    eigen_values, eigen_vectors = np.linalg.eigh(cov_mat)

    #Step 4: sorting eigen values and eigen vectors to find N most significant eigen values and their corresponding
eigen vectors
    sorted_index = np.argsort(eigen_values)[::-1]          #Descending order
    sorted_eigenvalue = eigen_values[sorted_index]
    sorted_eigenvectors = eigen_vectors[:, sorted_index]

    #Step 5: selecting 'num_components' subset from the eigen vectors
    eigenvector_subset = sorted_eigenvectors[:, 0:num_components]

    #Step 6: performing dot product to reduce data
    X_reduced = np.dot(eigenvector_subset.transpose(), X_centered.transpose()).transpose()

    return X_reduced
```

```
path = r'/home/jahanzebnaeem/Downloads/DM/Project-20092022-124505am/iris.csv'
allData = pd.read_csv(path)
data = allData.iloc[:, 0:4]
reduced_data= PCA_func(data, 2)                    # reducing to 2 dimensions so that we can plot easily
colors = ['red', 'green', 'blue', 'black', 'yellow']

for i in range(1, 6):
    labels = KMeans(n_clusters=i, random_state=3).fit_predict(X=reduced_data)
    unique_labels = list(set(labels))

    for j in unique_labels:
        filtered_label = reduced_data[labels == int(j)]
        plt.scatter(filtered_label[:, 0], filtered_label[:, 1], color=colors[int(j)])
    plt.show()
```

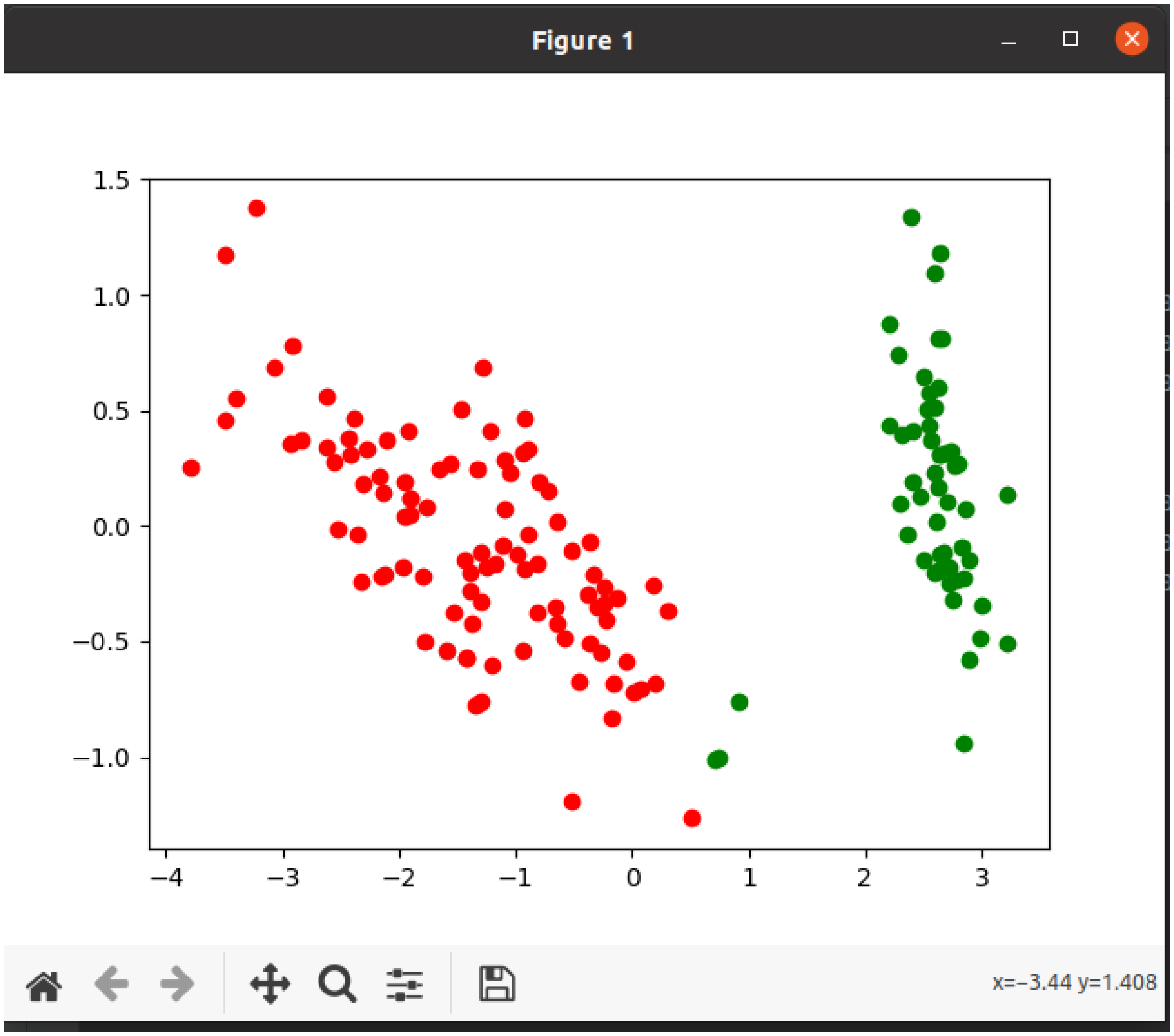
Screenshots:

(on next page)

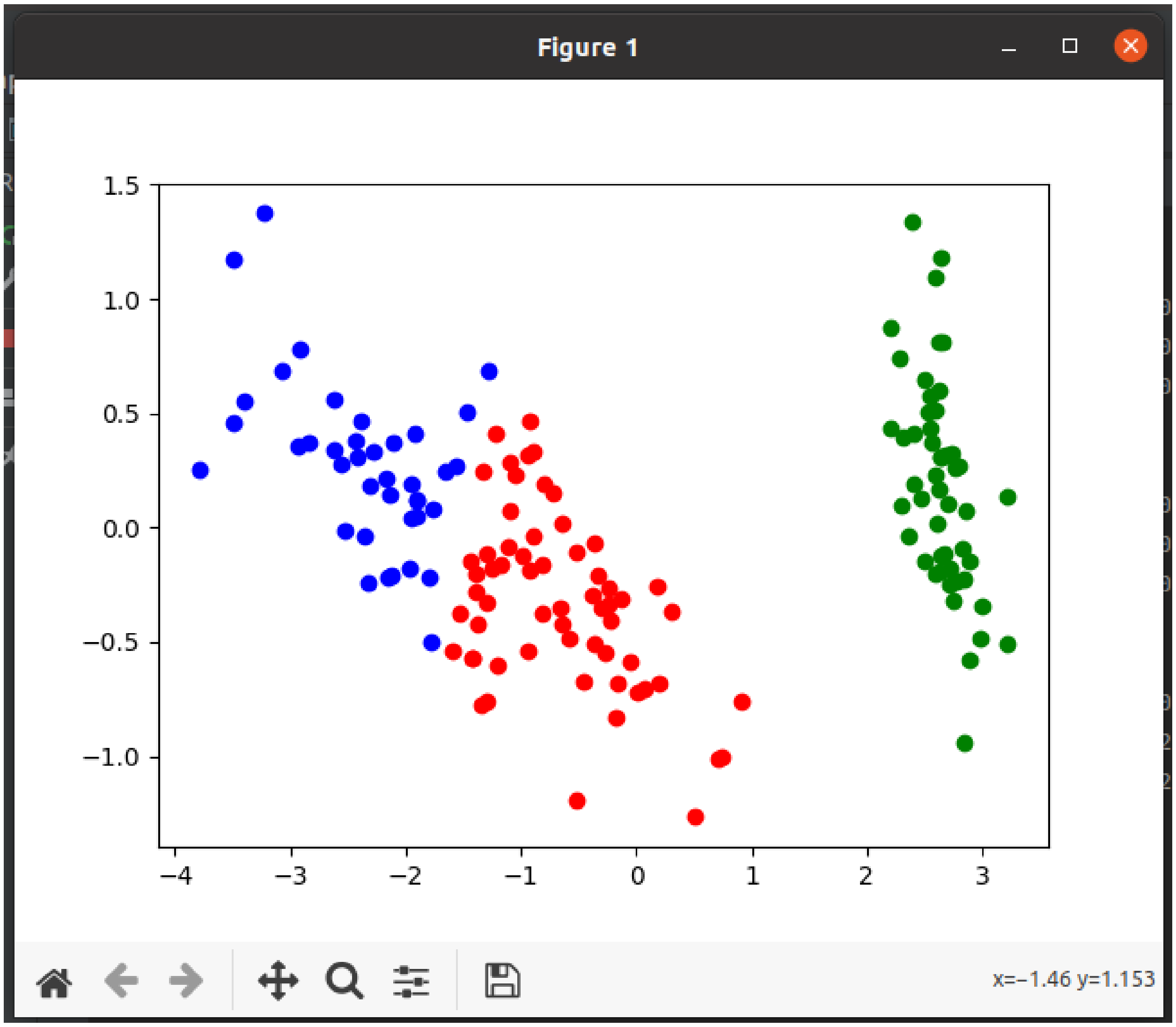
Visualization with 1 cluster:



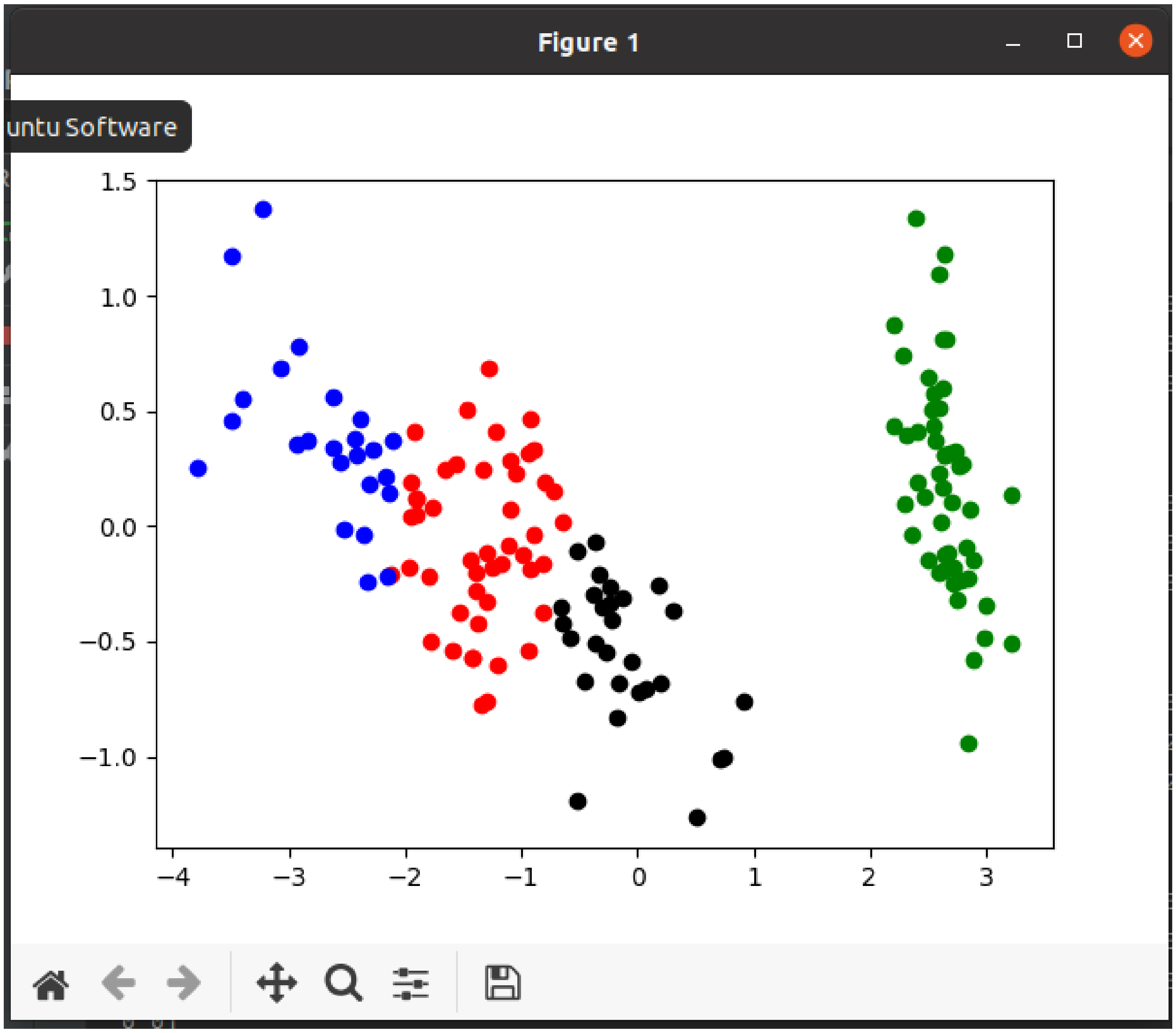
Visualization with 2 clusters:



Visualization with 3 clusters:



Visualization with 4 clusters:



Visualization with 5 clusters:

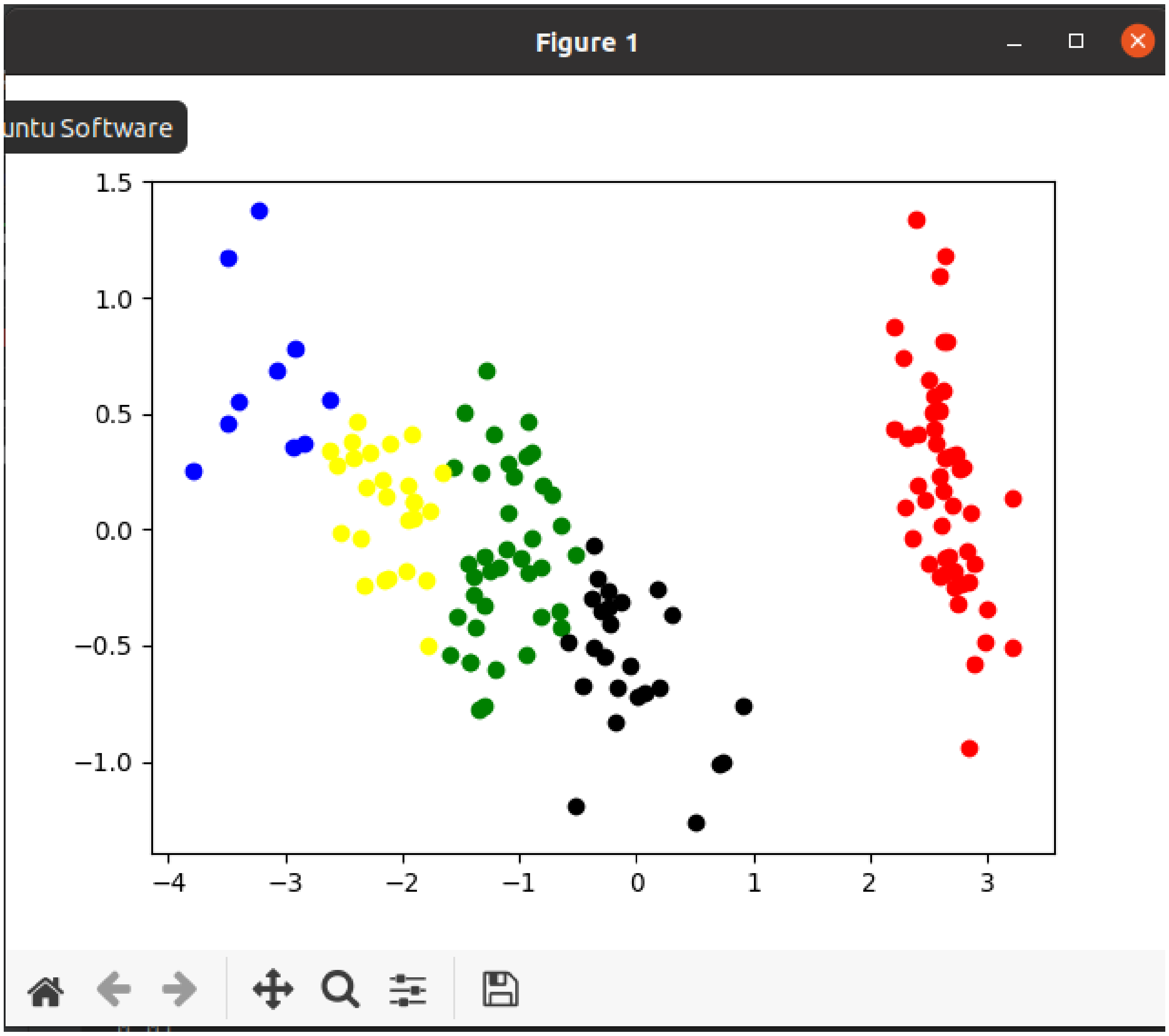


Figure 1
untu Software