Firstly, we should import all needed dependencies.

```
In [2]: from sklearn import datasets
        from sklearn.cluster import KMeans
        import sklearn as sk
        import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        import matplotlib.pyplot as plt
        import warnings
        from sklearn.preprocessing import StandardScaler
        from scipy.spatial.distance import cdist
        from sklearn.decomposition import PCA
        warnings.filterwarnings('ignore')
        %matplotlib inline
```

For this assigment I've chosen standard iris dataset. This dataset contains a sepal and petal length and width.

```
In [13]: data=datasets.load_iris()
         iris=pd.DataFrame(StandardScaler().fit_transform(data.data),columns=data.feature_names)
         iris.head(5)
```

Out[13]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

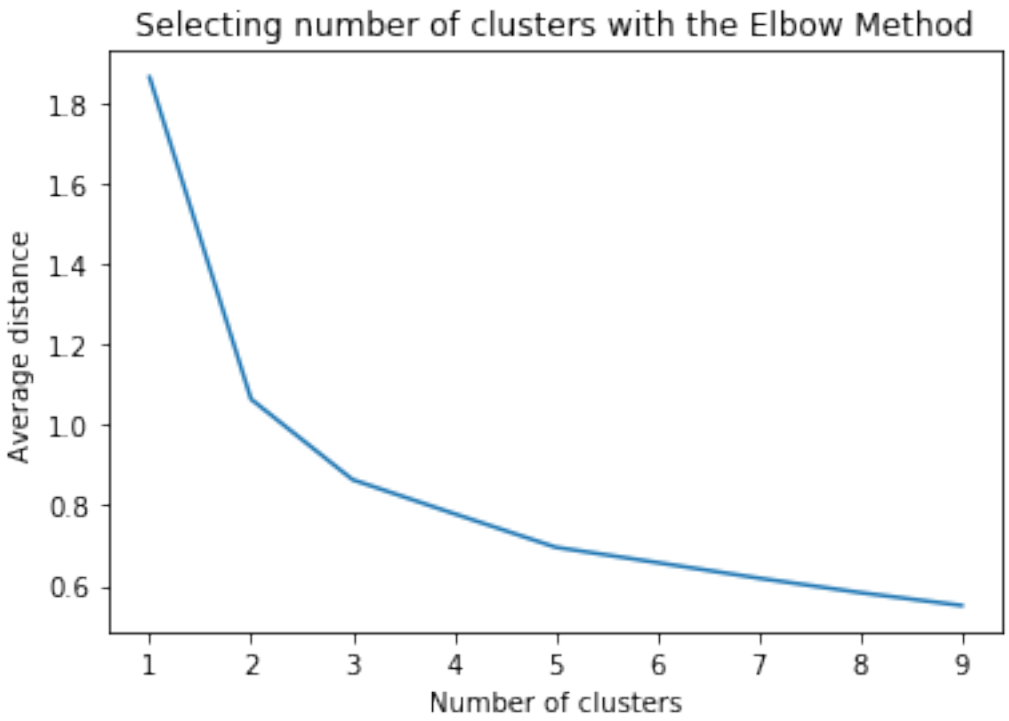
Clustering with number of clusters from 1 to 10:

```
In [20]: clusters=range(1,10)
         meandist=[]
         for k in clusters:
             model=KMeans(n_clusters=k)
             model.fit(iris)
             clusassign=model.predict(iris)
             meandist.append(sum(np.min(cdist(iris, model.cluster_centers_, 'euclidean'), axis=1
```

Comparing different number of clusters:

```
In [21]: plt.plot(clusters, meandist)
         plt.xlabel('Number of clusters')
         plt.ylabel('Average distance')
         plt.title('Selecting number of clusters with the Elbow Method')
```

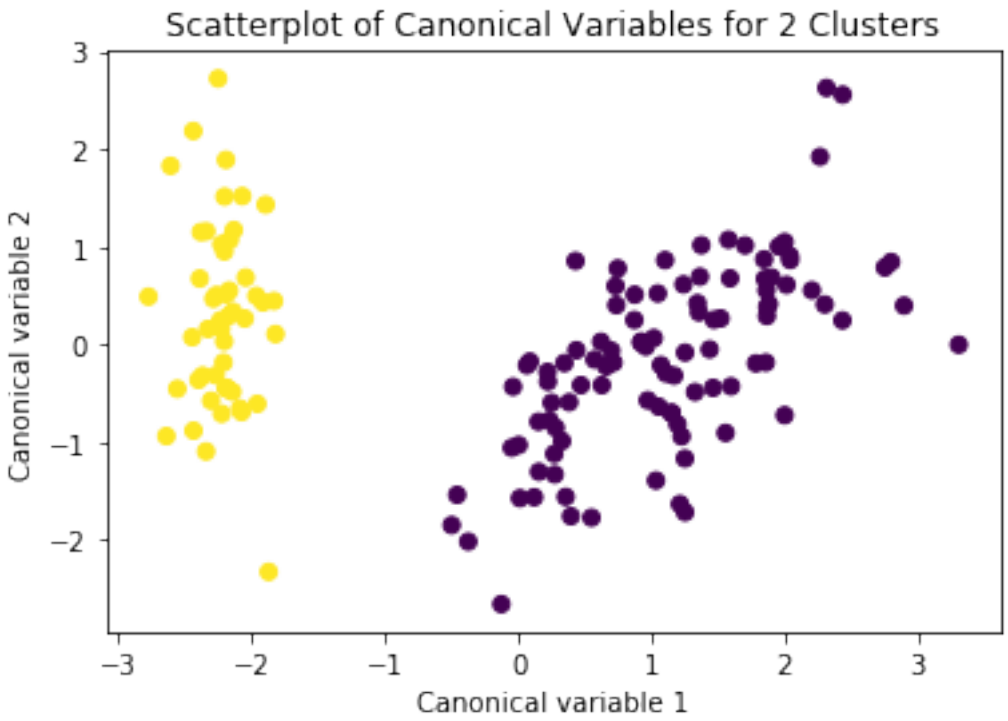
```
Out[21]: Text(0.5,1,'Selecting number of clusters with the Elbow Method')
```



It's not obvious from the plot which a number of clusters are optimal. So, look at the possible variants:

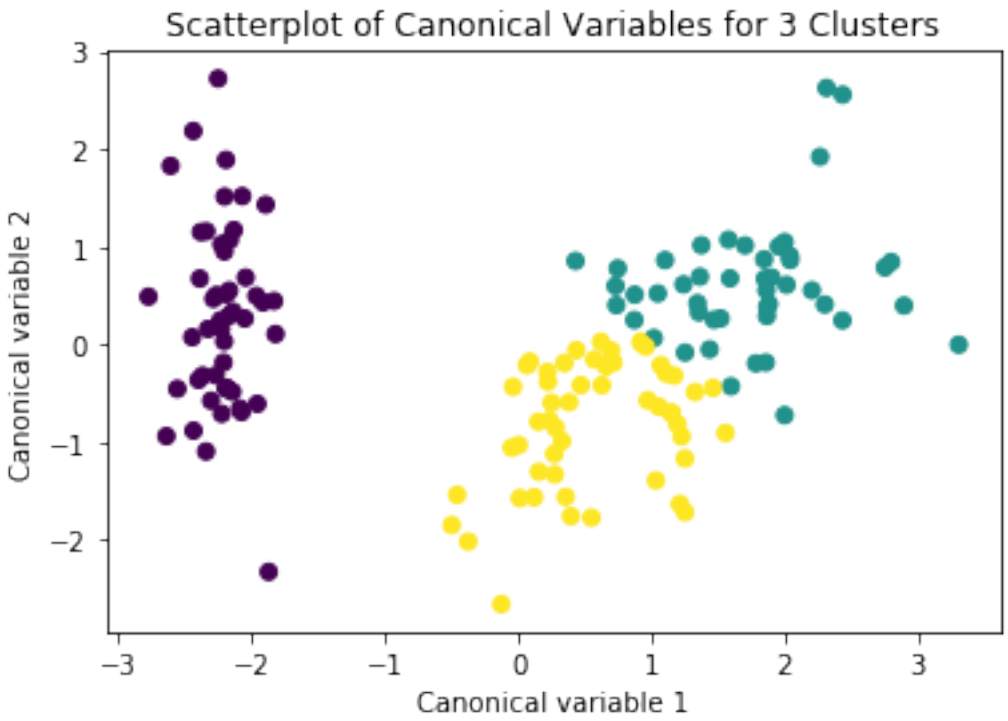
```
In [22]: model2=KMeans(n_clusters=2)
         model2.fit(iris)
         clusassign=model2.predict(iris)

         pca_2 = PCA(2)
         plot_columns = pca_2.fit_transform(iris)
         plt.scatter(x=plot_columns[:,0], y=plot_columns[:,1], c=clusassign)
         plt.title('Scatterplot of Canonical Variables for 2 Clusters')
         plt.xlabel('Canonical variable 1')
         plt.ylabel('Canonical variable 2')
         plt.show()
```



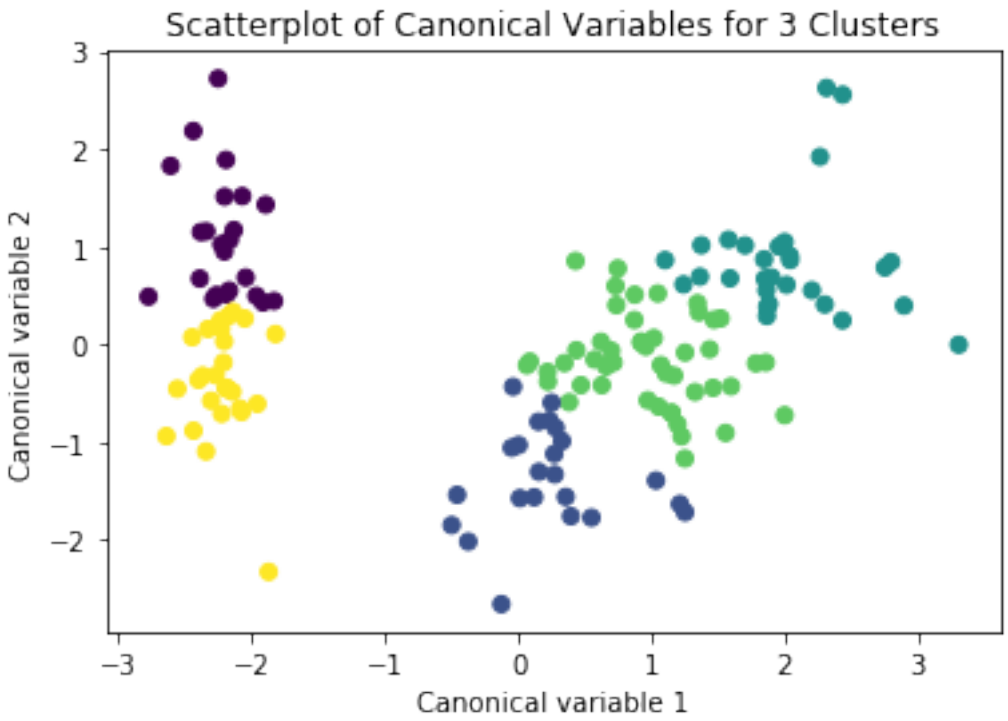
In [6]:
```
model3=KMeans(n_clusters=3)
model3.fit(iris)
clusassign=model3.predict(iris)

pca_2 = PCA(2)
plot_columns = pca_2.fit_transform(iris)
plt.scatter(x=plot_columns[:,0], y=plot_columns[:,1], c=clusassign)
plt.title('Scatterplot of Canonical Variables for 3 Clusters')
plt.xlabel('Canonical variable 1')
plt.ylabel('Canonical variable 2')
plt.show()
```



In [11]:
```
model5=KMeans(n_clusters=5)
model5.fit(iris)
clusassign=model5.predict(iris)

pca_2 = PCA(2)
plot_columns = pca_2.fit_transform(iris)
plt.scatter(x=plot_columns[:,0], y=plot_columns[:,1], c=clusassign)
plt.title('Scatterplot of Canonical Variables for 5 Clusters')
plt.xlabel('Canonical variable 1')
plt.ylabel('Canonical variable 2')
plt.show()
```



For my opininon, 3 clusters is reasonably enough.