ASSIGNMENT 3.3

on

Unsupervised Machine Learning

Submitted by:

Haseebullah Shaikh (2303.KHI.DEG.015)

and

Faiza Gulzar Ahmed (2303.khi.deg.001)

Task 01: Perform k-means clusterization on the Iris dataset. Repeat the procedure on the dataset reduced with PCA, and then compare the results.

Solution:

jupyter Assignment 3.3_Updated Last Checkpoint: 33 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import adjusted_rand_score
import pandas as pd
import numpy as np
```

In [2]:
```
iris = datasets.load_iris()
```

Performing Exploratory data analysis to gain insights on data.

In [3]:
```
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['Species'] = iris.target
iris_df.head()
```

Out[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [4]:
```
x = iris_df.iloc[:,:4]
y= iris_df.iloc[:,4]
```

In [5]:
```
iris_df.shape
```

Out[5]: (150, 5)

In [6]:
```
iris_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   Species            150 non-null    int32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

There is no any null entry in the dataset :)

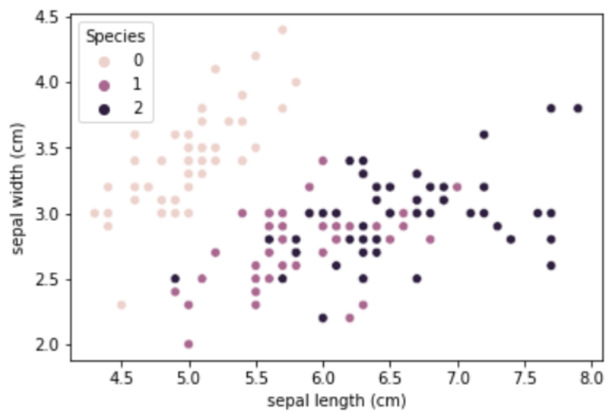
In [7]:
```
x.describe()
```

Out[7]:

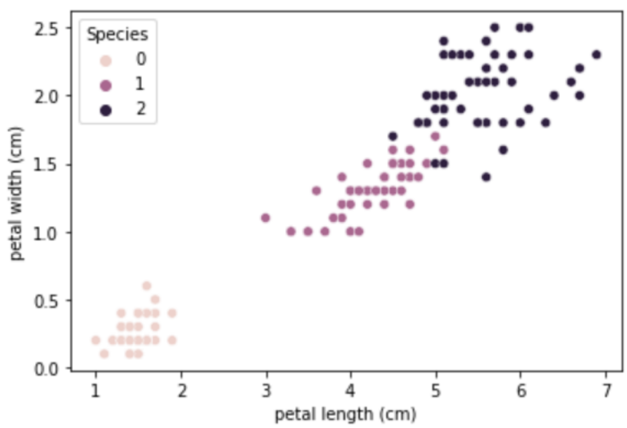
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Evaluating relationship between features based on species

In [8]:
```
sns.scatterplot(data=iris_df,x='sepal length (cm)', y ='sepal width (cm)', hue='Species')
plt.show()
```



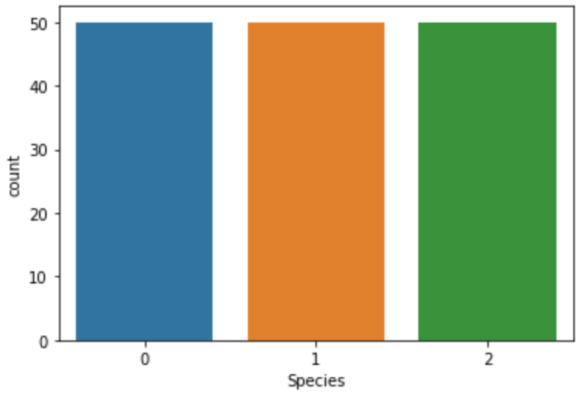
In [9]:
```
sns.scatterplot(data=iris_df,x='petal length (cm)', y ='petal width (cm)', hue='Species')
plt.show()
```



Count Species

Count Species

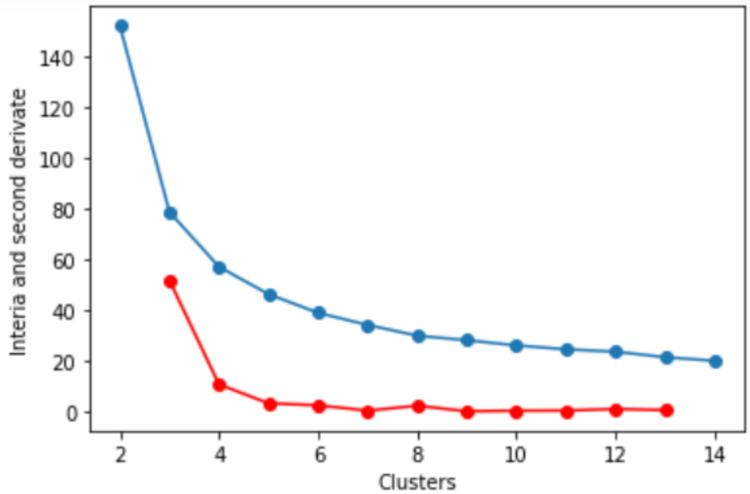
In [10]:
```
sns.countplot(data=iris_df, x='Species')
plt.show()
```



Finding optimal numbers of clusters using elbow method

In [11]:
```

k_values = []
intertia_scores = []

for k in range(2,15):
    model = KMeans(n_clusters=k)
    model.fit(x)
    intertia_scores.append(model.inertia_)
    k_values.append(k)

module_of_second_derivative = np.abs(np.diff(np.diff(intertia_scores)))
```

In [12]:
```
plt.plot(k_values, intertia_scores)
plt.scatter(k_values, intertia_scores)
plt.plot(k_values[1:-1], module_of_second_derivative, color='red')
plt.scatter(k_values[1:-1], module_of_second_derivative, color='red')
plt.xlabel("Clusters")
plt.ylabel("Interia and second derivate")
plt.show()
```



Elbow point can be seen at value = 3, therefore optimal number of clusters will be 3

Training the model on 4 features using Kmeans clustering with optimal clusters k = 3

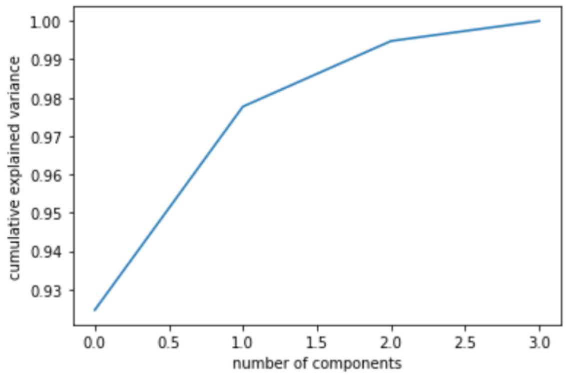
In [13]:
```
model = KMeans(n_clusters=3, n_init=1, max_iter=100)
model.fit(x)

all_predictions = model.predict(x)
centroids = model.cluster_centers_
centroids
```

Out[13]:
```
array([[5.88360656, 2.74098361, 4.38852459, 1.43442623],
       [5.006     , 3.428     , 1.462     , 0.246     ],
       [6.85384615, 3.07692308, 5.71538462, 2.05384615]])
```

finding optimal pca components

In [14]:
```
pca = PCA().fit(x)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
```

Out[14]: Text(0, 0.5, 'cumulative explained variance')



as it can be seen in above graph there is 99% variance in first two components, therefore selecting 2 components

---

In [15]:
```
pca = PCA(n_components=2)
x_reduced = pca.fit_transform(x)

x_reduced.shape
```

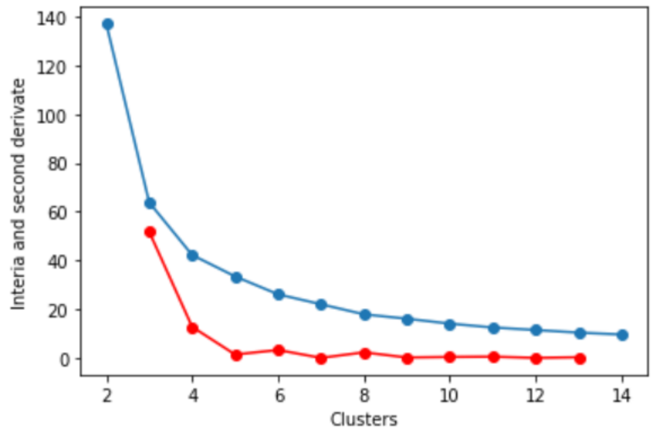
Out[15]: (150, 2)

Training model on reduced dataset having 2 features

finding optimal clusters for reduced dataset

In [16]:
```

k_values = []
intertia_scores = []

for k in range(2,15):
    model = KMeans(n_clusters=k)
    model.fit(x_reduced)
    intertia_scores.append(model.inertia_)
    k_values.append(k)

module_of_second_derivative = np.abs(np.diff(np.diff(intertia_scores)))
```

In [17]:
```
plt.plot(k_values, intertia_scores)
plt.scatter(k_values, intertia_scores)
plt.plot(k_values[1:-1], module_of_second_derivative, color='red')
plt.scatter(k_values[1:-1], module_of_second_derivative, color='red')
plt.xlabel("Clusters")
plt.ylabel("Interia and second derivate")
plt.show()
```

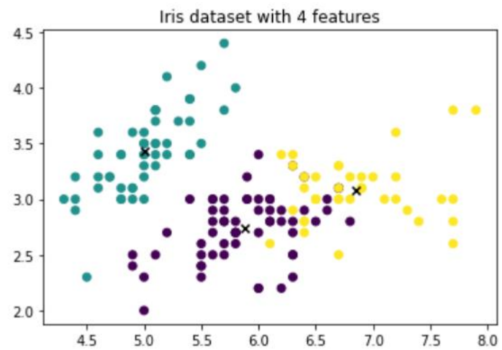


Elbow point can be seen at value = 3, therefore optimal number of clusters will be 3

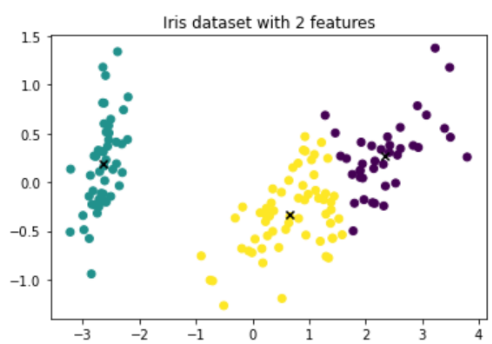
In [18]:
```
model_2 = KMeans(n_clusters=3, n_init=1, max_iter=100)
model_2.fit(x_reduced)

all_predictions_2 = model_2.predict(x_reduced)
centroids_2 = model_2.cluster_centers_
centroids_2
```

Out[18]:
```
array([[ 2.34652659,  0.27393856],
       [-2.64241546,  0.19088505],
       [ 0.66567601, -0.3316042 ]])
```

In [19]:
```
plt.scatter(x.iloc[:,0], x.iloc[:,1], c=all_predictions)
plt.scatter(centroids[:,0], centroids[:,1], marker='x', color="black")
plt.title("Iris dataset with 4 features")
plt.show()
```



[20]:
```
plt.scatter(x_reduced[:,0], x_reduced[:,1], c=all_predictions_2)
plt.scatter(centroids_2[:,0], centroids_2[:,1], marker='x', color="black")
plt.title("Iris dataset with 2 features")
plt.show()
```



As it can be seen in first graph, the data has been speparated in 3 clusters, but it's quite not clear due to thedata points are overlaping within the other groups, there are also some outliers whears in the second graph with reduced data set. The data separation in three cluster is pretty good, there is no overlapping data points between the group but there are few outlires

Implementing External Validation

In [21]:
```
score = adjusted_rand_score(y, all_predictions)
print(score)
```

```
0.7163421126838475
```

In [22]:
```


score = adjusted_rand_score(y, all_predictions_2)
print(score)
```

```
0.7163421126838475
```

we have got same validation accuray in both dataset

It can be concluded, Kmeans algorithm is giving pretty same results on all features and reduced features with 3 optimal number of clusters but the cluster separation is pretty good in reduced dataset :)