GRIP : The Sparks Foundation

Data Science and Business Analytics Intern

Batch : March 2022

Author : Pragati Dilip Gawale

Task 2 : Prediction Using Unsupervised ML

In this task it is required to predict the optimum number of clusters for the iris data set consists of 3 types of flower namely Iris-setosa Iris-versicolour and Iris-virginica

In [1]:
```
# Importing required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import decomposition
from sklearn.cluster import KMeans
import seaborn as sns
```

Reading the data

In [2]:
```
iris=datasets.load_iris()
df=pd.DataFrame(iris.data,columns=iris.feature_names)
```

Data Exploration

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
df.tail()
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

In [5]:
```
df.columns
```

Out[5]:
```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 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
dtypes: float64(4)
memory usage: 4.8 KB
```

In [7]:
```
df.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

In [8]:
```
# checking for any missing values

df.isnull().sum()
```

Out[8]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

In [9]:
```
# Remove the duplicate values
df.duplicated().sum()
```

Out[9]:
```
1
```

In [10]:
```
df.drop_duplicates(inplace=True)
df.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
# Correlation between multiple values
corr=df.corr()
corr
```

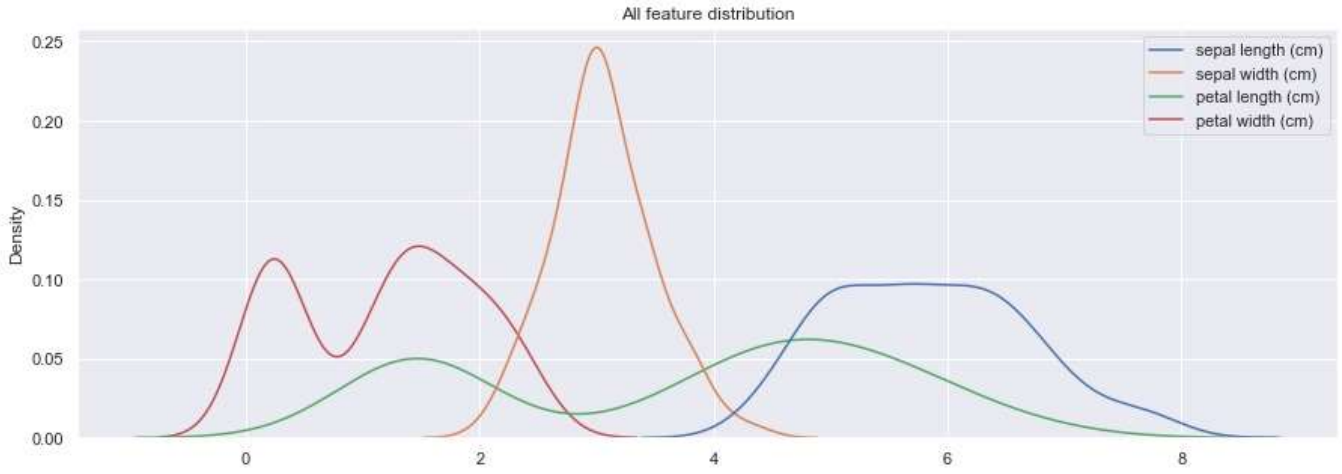
Out[11]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	1.000000	-0.118129	0.873738	0.820620
sepal width (cm)	-0.118129	1.000000	-0.426028	-0.362894
petal length (cm)	0.873738	-0.426028	1.000000	0.962772
petal width (cm)	0.820620	-0.362894	0.962772	1.000000

Plotting the distribution

In [12]:
```
sns.set(rc={"figure.figsize":(15,5)})
g = sns.kdeplot(data=df)
g.set(title="All feature distribution")
```

Out[12]:
```
[Text(0.5, 1.0, 'All feature distribution')]
```



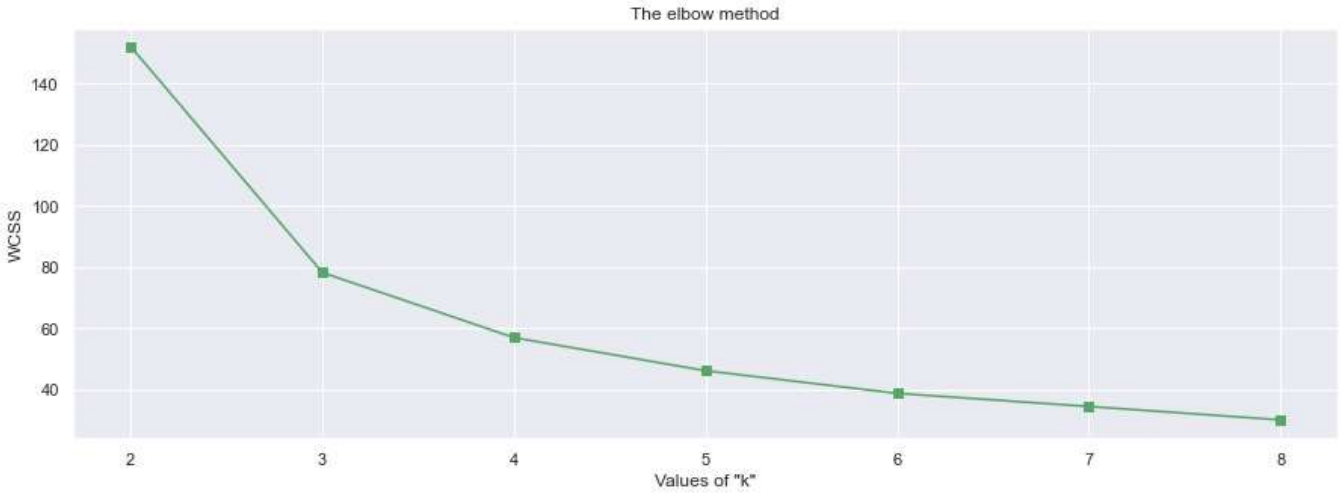
In [13]:
```
### Finding the maximum number of the clusters
limit=int((df.shape[0]//2)**0.5)
limit
```

Out[13]:
```
8
```

In [14]:
```
# Using elbow method to determine the optimal number of clusters for iris dataset
```

```
wcss= {}
for k in range(2,limit+1):
    model=KMeans(n_clusters=k)
    model.fit(df)
    wcss[k]=model.inertia_
```

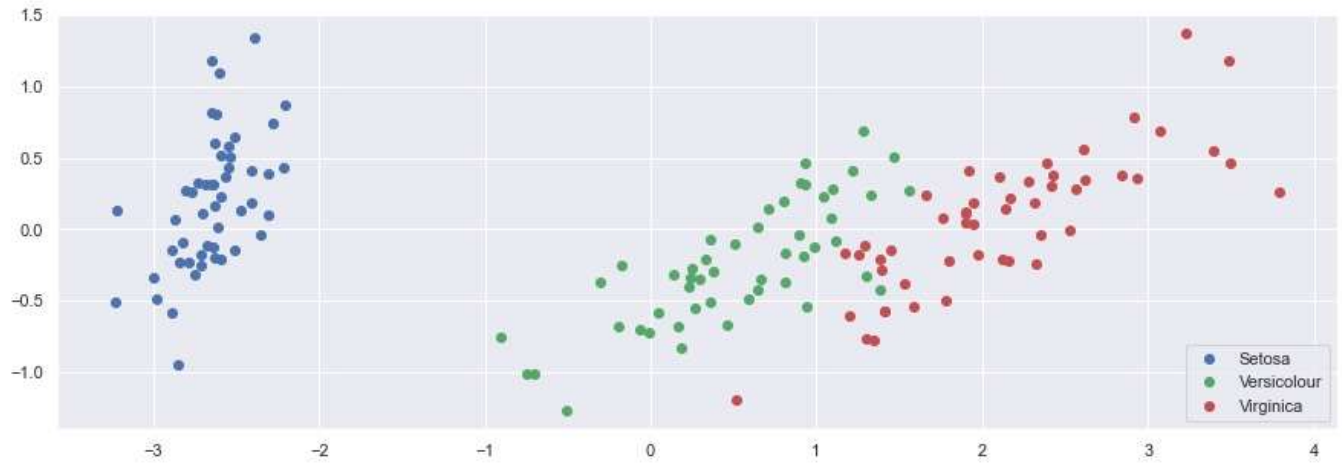
In [15]:
```
plt.plot(wcss.keys(),wcss.values(),'gs-')
plt.xlabel('Values of "k"')
plt.ylabel('WCSS')
plt.title('The elbow method')
plt.show()
```



optimal number of clusters for k-means clustering is 3

In [16]:
```
# Using principal component analysis(PCA)
X=iris.data
y=iris.target
pca=decomposition.PCA(n_components=2)
X_centered = X-X.mean(axis=0)
pca.fit(X_centered)
X_pca = pca.transform(X_centered)
```

In [17]:
```
# Plotting the results of PCA
plt.plot(X_pca[y == 0, 0],X_pca[y ==0,1],'bo',label='Setosa')
plt.plot(X_pca[y == 1, 0],X_pca[y ==1,1],'go',label='Versicolour')
plt.plot(X_pca[y == 2, 0],X_pca[y ==2,1],'ro',label='Virginica')
plt.legend(loc=0);
```

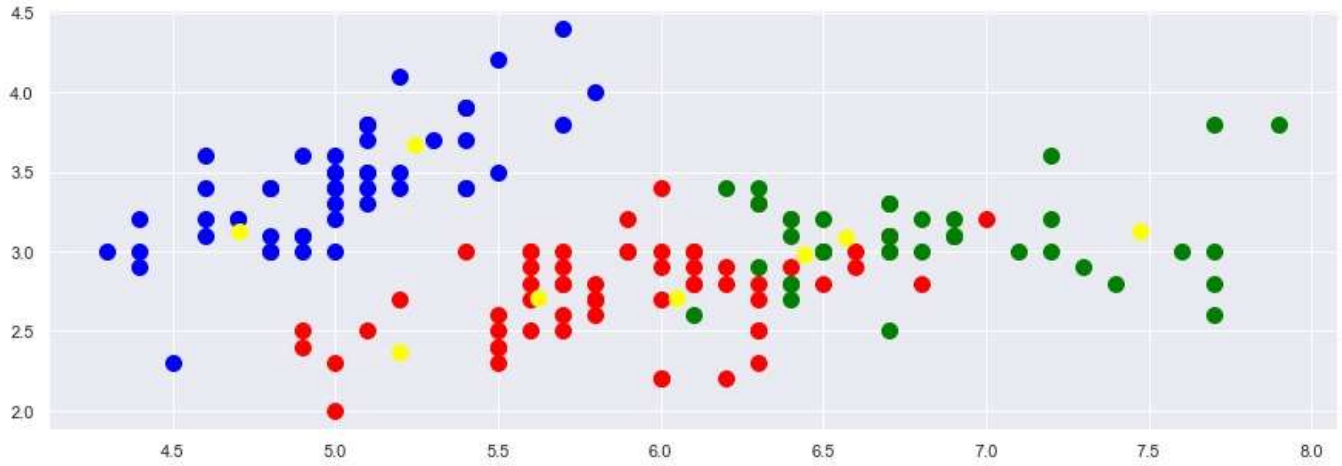


In [18]:
```
# Applying Kmeans to the dataset(Creating the kmeans classifier)
kmeans = KMeans(n_clusters=3,init='k-means++',max_iter=300,n_init=10,random_state=0)
y_kmeans=kmeans=kmeans.fit_predict(X)
```

In [23]:
```
# Plotting the centroids of the clusters
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s=100,c='red',label='Iris-setosa')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s=100,c='blue',label='Iris-versicolour')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s=100,c='green',label='Iris-virginica')

plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],s=100,c='yellow',label='Centroid')
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x1c9f574a800>
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js