GRIP@The Spark Foundation- Data Science & Business Analytics Internship

Author - Shweta Pamane

Task 2: Prediction using Unsupervised ML

Dataset used: Iris dataset

It can be downloaded through the following link - [https://bit.ly/3kXTdox (https://bit.ly/3kXTdox)](https://bit.ly/3kXTdox)

Problem Statement(s) :

*** Predict the optimum number of clusters and represent it visually.

Import necessary libraries

In [2]:
```
# Importing Libraries required for data analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Read the data from Dataset

In [3]:
```
#Reading the data from Dataset
df = pd.read_csv("Iris.csv")
df
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [4]:
```
df.shape
```

Out[4]:
```
(150, 6)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
Id                 150 non-null int64
SepalLengthCm      150 non-null float64
SepalWidthCm       150 non-null float64
PetalLengthCm      150 non-null float64
PetalWidthCm       150 non-null float64
Species            150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [6]:
```
# dropping Id column
df.drop('Id', axis=1, inplace=True)
df.columns
```

Out[6]:
```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [7]:
```
print(df.isnull().sum(), '\n\nNumber of duplicate rows: ' , df.duplicated().sum
())
```

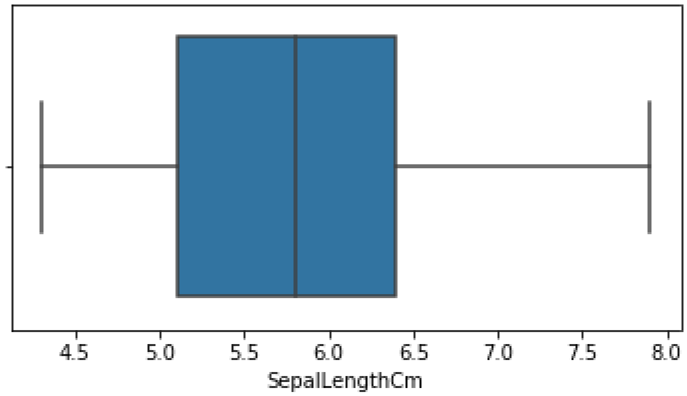
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64

Number of duplicate rows:  3
```

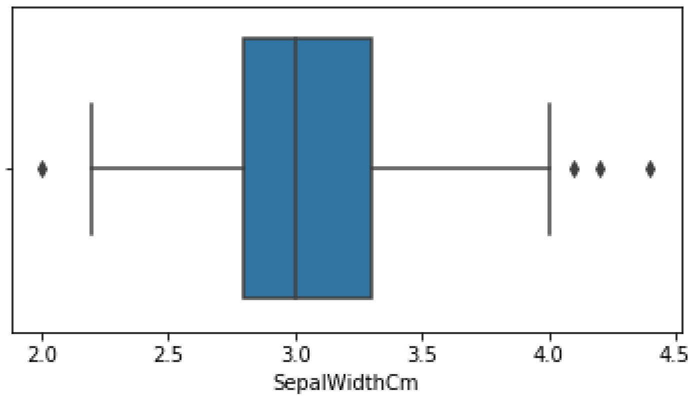
Drop duplicate rows

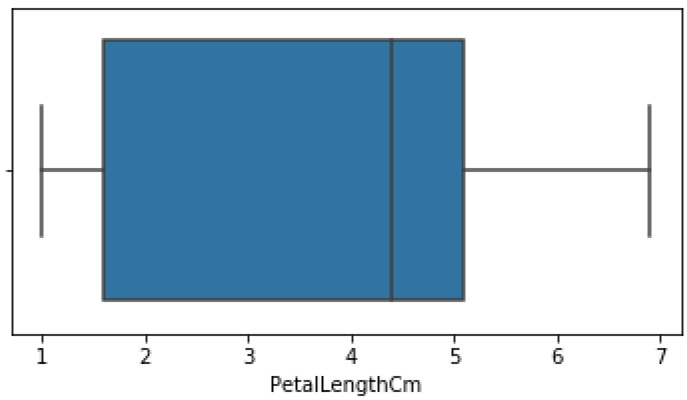
In [9]:
```
# Drop duplicate rows
df.drop_duplicates(inplace=True)
df.shape[0] # gives number of rows. Similarly, data.shape[1] will give number o
f columns

## now number of rows left 147, earlier there were 150 rows.
```

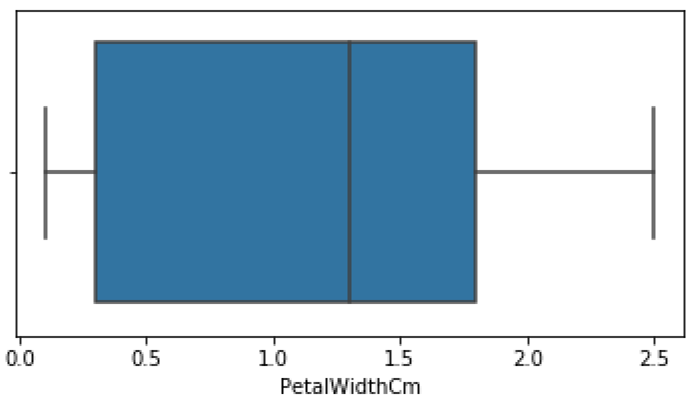
Out[9]:
```
147
```

In [10]:
```
# Check for any outliers in the numeric data
for i in df.columns:
    if df[i].dtype=='float64':
        plt.figure(figsize=(6,3))
        sns.boxplot(df[i])
        plt.show()
```



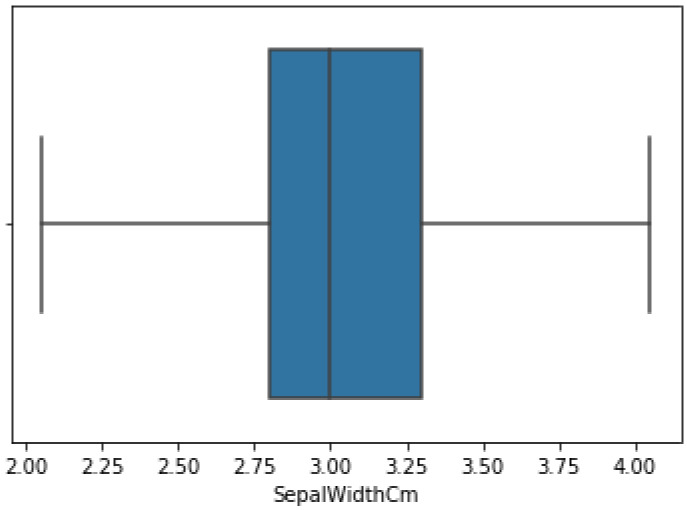






In [11]:
```
# Treating outliers present in the SepalWidthCm column
q1,q3 = np.percentile(df['SepalWidthCm'],[25,75])
iqr = q3-q1
lower_fence = q1 - (1.5*iqr)
upper_fence = q3 + (1.5*iqr)
df['SepalWidthCm'] = df['SepalWidthCm'].apply(lambda x: upper_fence if x>upper_
fence
                                                         else lower_fence if x<lower_f
ence else x)
```

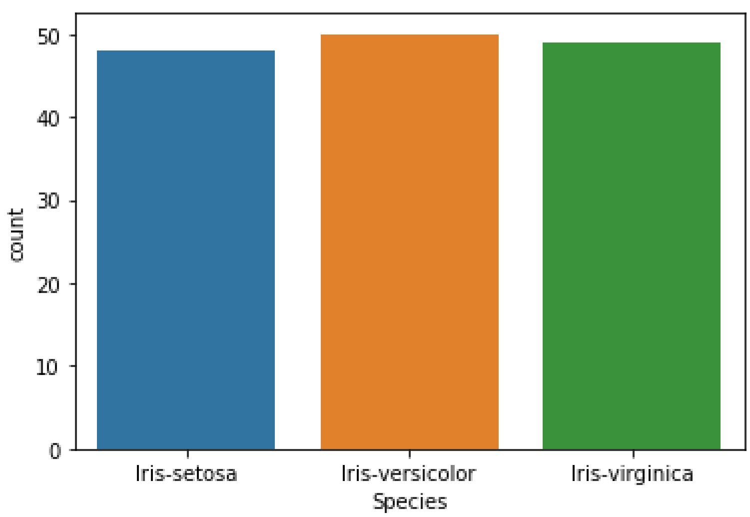
In [12]:
```
sns.boxplot(df['SepalWidthCm']);
```



Understanding the data

In [13]:
```
# Target class
print(df.Species.value_counts())
sns.countplot(df.Species);
```

```
Iris-versicolor    50
Iris-virginica     49
Iris-setosa        48
Name: Species, dtype: int64
```



In [14]:
```
df.describe()
```

Out[14]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	147.000000	147.000000	147.000000	147.000000
mean	5.856463	3.052381	3.780272	1.208844
std	0.829100	0.426331	1.759111	0.757874
min	4.300000	2.050000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.050000	6.900000	2.500000

In [15]:
```
df.Species.unique()
```

Out[15]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [23]:
```
## Correlation Matrix

df.corr()
```

Out[23]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.110155	0.871305	0.817058
SepalWidthCm	-0.110155	1.000000	-0.420140	-0.355139
PetalLengthCm	0.871305	-0.420140	1.000000	0.961883
PetalWidthCm	0.817058	-0.355139	0.961883	1.000000

In [28]:
```
plt.figure(figsize=(10,5))
sns.heatmap(abs(df.corr()), cmap='GnBu', annot=True);
```

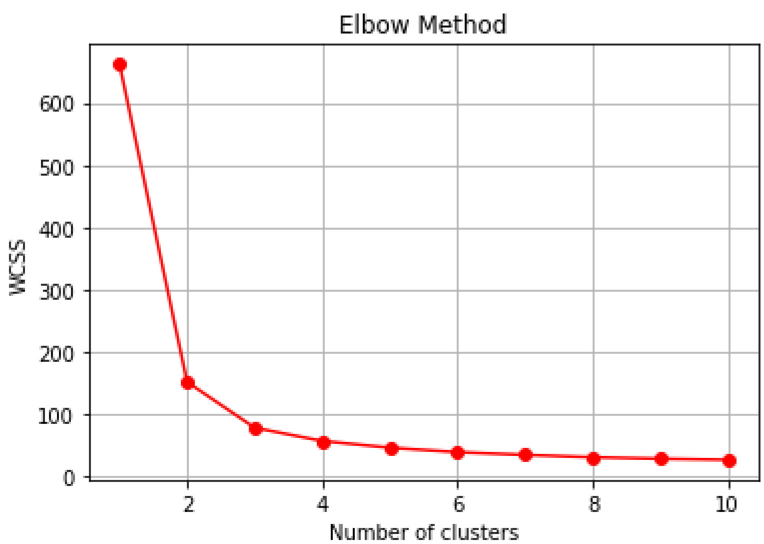
K-means clustering

In [29]:
```
X = df.iloc[:, [0,1,2, 3]].values
```

In [30]:
```
# find optimal number of clusters using elbow method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, ra
ndom_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

Finding optimal number of clusters using elbow method

In [31]:
```
#find optimal number of clusters using elbow method
plt.plot(range(1, 11), wcss, 'go-', color='red')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.grid()
plt.show()
```



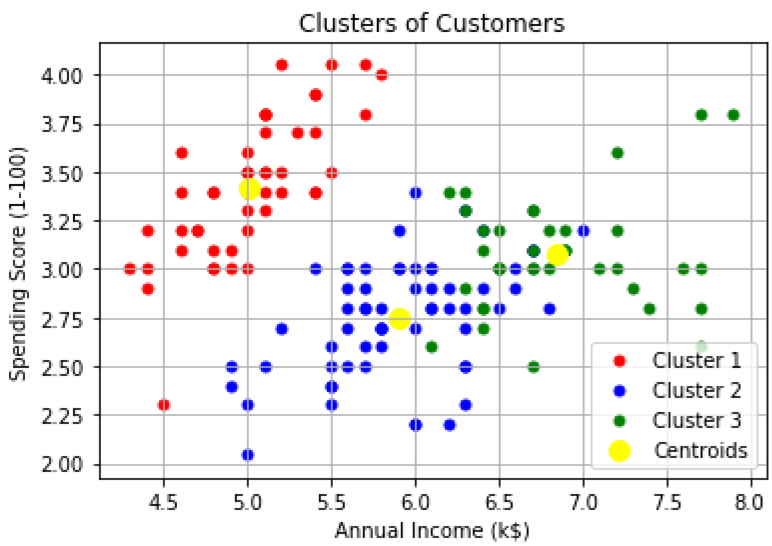
Applying K-means clustering

In [43]:
```
#apply K-means clustering on the data
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random
_state=0)
y_kmeans = kmeans.fit_predict(X)

x = df.iloc[:, [0, 1, 2, 3]].values
```

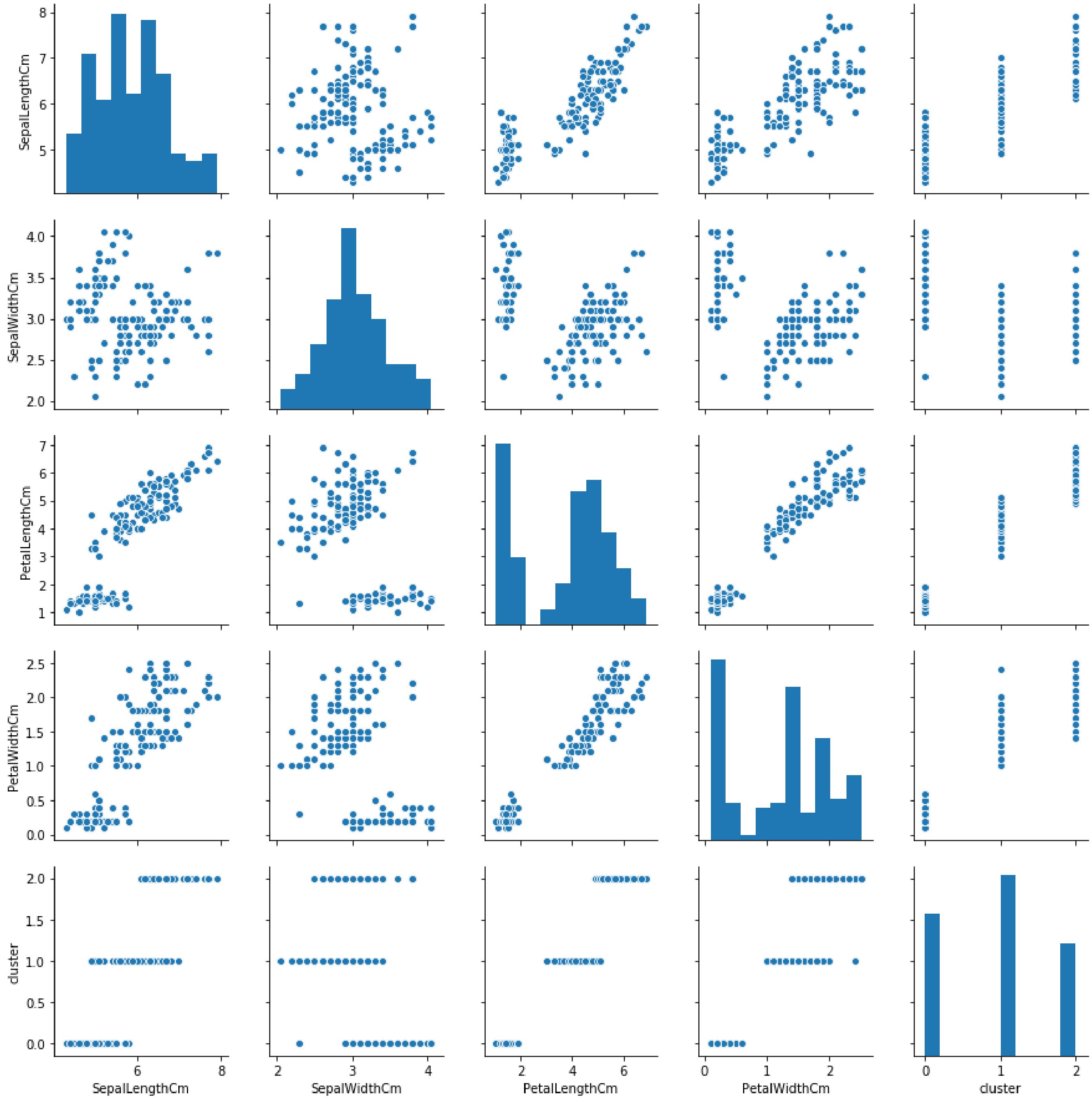
Visualize clusters

In [44]:

```
# visualize clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 25, c = 'red', label
= 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 25, c = 'blue', label
= 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 25, c = 'green', labe
l = 'Cluster 3')

# Plotting the cluster centers
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 1
00, c = 'yellow', label = 'Centroids')
plt.title('Clusters of Customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.grid()
plt.legend()
plt.show()
```

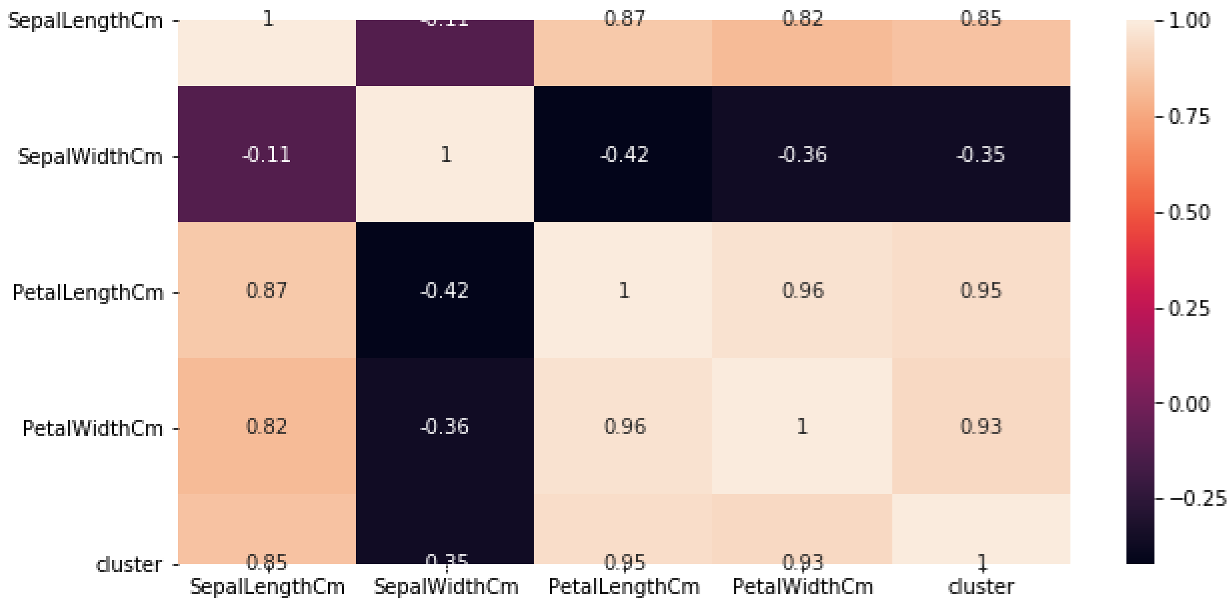


In [45]:
```
sns.pairplot(df,hue=None)
```
Out[45]:
```
<seaborn.axisgrid.PairGrid at 0x215c97e9888>
```



In [46]:
```python
plt.figure(figsize=(10,5))
sns.heatmap(df.corr(), annot=True)
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x215ca3928c8>



In [ ]: