In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import Normalizer
from sklearn.pipeline import make_pipeline
```

In [2]:
```
df = pd.read_csv('Documents/Iris.csv')
```

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

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

EDA

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

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

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

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

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

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

In [7]:
```
df.corr()
```

Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

Looks like our data doesn't need preprocessing. Let's check for outliers.

In [8]:
```
def outlier_detection(df):
    ### Written by: Sujay Rittikar
    # Detecting the Null or NaN values and remo ving them first
    # to ensure that the numerical columns can be detected correctly.
    r = []
    for col in df.columns:
        for i in df.index:
            if df.loc[i, col]=='Null' or df.loc[i, col] == np.nan:
                r.append(i)
    df = df.drop(list(set(r)))
    df = df.reset_index()
    df = df.drop('index', axis=1)

    # Finding out the columns having numerical values.
    num_cols = []
    for col in df.columns:
        if df[col].dtype == 'object':
            try:
                df[col] = pd.to_numeric(df[col])
                num_cols.append(col)
            except ValueError:
                pass

    # Removing the rows having values which can be called outliers
    # on the basis of their z-scores of >3 or <-3
    count = 0
    t = []
    for i in num_cols:
        z = np.abs(stats.zscore(df[i]))
        for j in range(len(z)):
```

```
            if z[j]>3 or z[j]<-3:
                t.append(j)
                count+=1
    df = df.drop(list(set(t)))
    df = df.reset_index()
    df = df.drop('index', axis=1)
    print(count)
    return df
```

In [9]:
```
df = outlier_detection(df)
```

```
0
```

No Outliers!

Data Visualizations

In [12]:
```
sns.countplot(df['Species'])
```

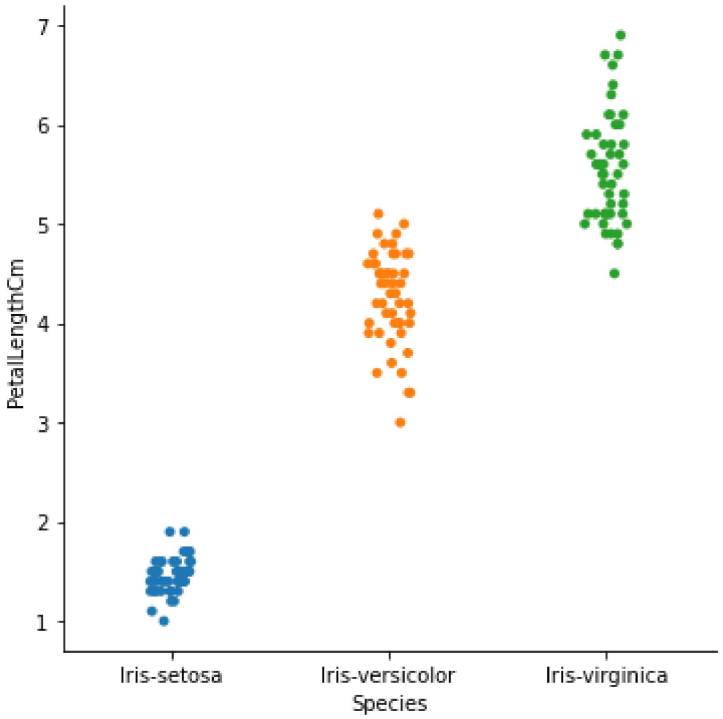
```
C:\Users\LEN\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
```

Out[12]: <AxesSubplot:xlabel='Species', ylabel='count'>

In [11]:
```
sns.catplot("Species", "PetalLengthCm", data = df)
```

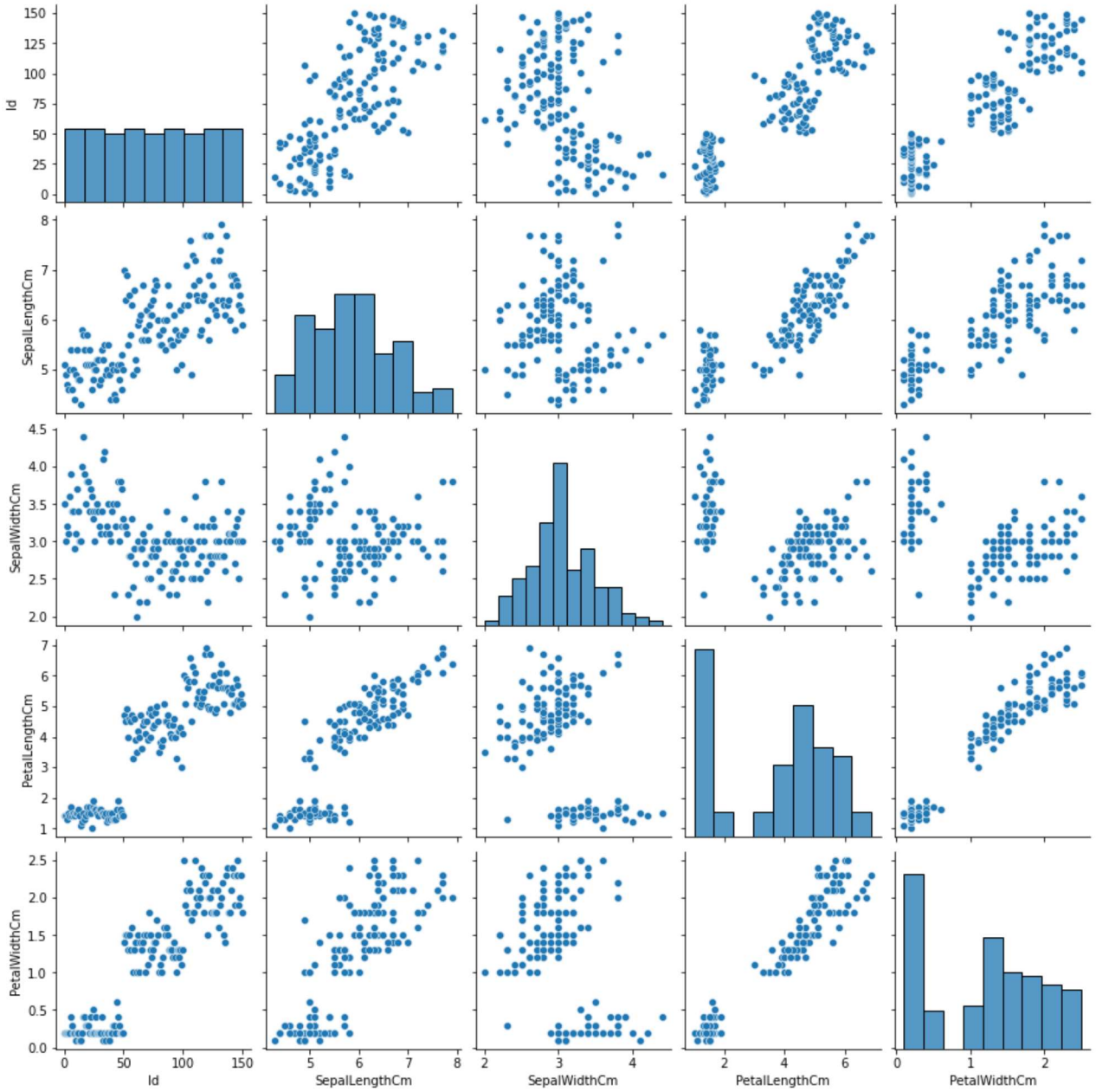
```
C:\Users\LEN\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  warnings.warn(
```

Out[11]: <seaborn.axisgrid.FacetGrid at 0x275f755ccd0>



In [13]:
```
sns.pairplot(data=df)
```

Out[13]: <seaborn.axisgrid.PairGrid at 0x275f7d1bee0>



Let's look at the Elbow graph to find k

In [14]:

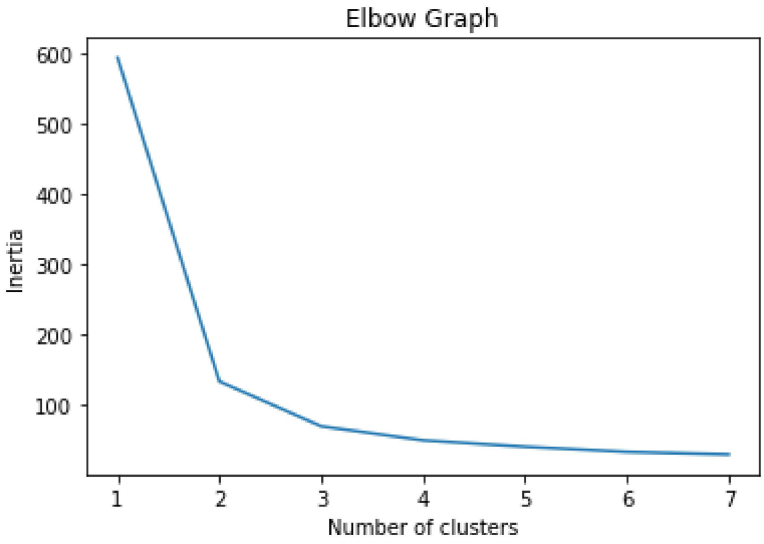
```
x = df.iloc[:, [1, 2, 3]].values
inertias = []

for i in range(1, 8):
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(x)
    inertias.append(kmeans.inertia_)

plt.plot(range(1, 8), inertias)
plt.title('Elbow Graph')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

```
C:\Users\LEN\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KM
```

```
eans is known to have a memory leak on Windows with MKL, when there are less chunks than
available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=
1.
  warnings.warn(
```



In [15]:
```
x
```

Out[15]:
```
array([[5.1, 3.5, 1.4],
       [4.9, 3. , 1.4],
       [4.7, 3.2, 1.3],
       [4.6, 3.1, 1.5],
       [5. , 3.6, 1.4],
       [5.4, 3.9, 1.7],
       [4.6, 3.4, 1.4],
       [5. , 3.4, 1.5],
       [4.4, 2.9, 1.4],
       [4.9, 3.1, 1.5],
       [5.4, 3.7, 1.5],
       [4.8, 3.4, 1.6],
       [4.8, 3. , 1.4],
       [4.3, 3. , 1.1],
       [5.8, 4. , 1.2],
       [5.7, 4.4, 1.5],
       [5.4, 3.9, 1.3],
       [5.1, 3.5, 1.4],
       [5.7, 3.8, 1.7],
       [5.1, 3.8, 1.5],
       [5.4, 3.4, 1.7],
       [5.1, 3.7, 1.5],
       [4.6, 3.6, 1. ],
       [5.1, 3.3, 1.7],
       [4.8, 3.4, 1.9],
       [5. , 3. , 1.6],
       [5. , 3.4, 1.6],
       [5.2, 3.5, 1.5],
       [5.2, 3.4, 1.4],
       [4.7, 3.2, 1.6],
       [4.8, 3.1, 1.6],
       [5.4, 3.4, 1.5],
       [5.2, 4.1, 1.5],
       [5.5, 4.2, 1.4],
       [4.9, 3.1, 1.5],
       [5. , 3.2, 1.2],
       [5.5, 3.5, 1.3],
       [4.9, 3.1, 1.5],
```

```
[4.4, 3. , 1.3],
[5.1, 3.4, 1.5],
[5. , 3.5, 1.3],
[4.5, 2.3, 1.3],
[4.4, 3.2, 1.3],
[5. , 3.5, 1.6],
[5.1, 3.8, 1.9],
[4.8, 3. , 1.4],
[5.1, 3.8, 1.6],
[4.6, 3.2, 1.4],
[5.3, 3.7, 1.5],
[5. , 3.3, 1.4],
[7. , 3.2, 4.7],
[6.4, 3.2, 4.5],
[6.9, 3.1, 4.9],
[5.5, 2.3, 4. ],
[6.5, 2.8, 4.6],
[5.7, 2.8, 4.5],
[6.3, 3.3, 4.7],
[4.9, 2.4, 3.3],
[6.6, 2.9, 4.6],
[5.2, 2.7, 3.9],
[5. , 2. , 3.5],
[5.9, 3. , 4.2],
[6. , 2.2, 4. ],
[6.1, 2.9, 4.7],
[5.6, 2.9, 3.6],
[6.7, 3.1, 4.4],
[5.6, 3. , 4.5],
[5.8, 2.7, 4.1],
[6.2, 2.2, 4.5],
[5.6, 2.5, 3.9],
[5.9, 3.2, 4.8],
[6.1, 2.8, 4. ],
[6.3, 2.5, 4.9],
[6.1, 2.8, 4.7],
[6.4, 2.9, 4.3],
[6.6, 3. , 4.4],
[6.8, 2.8, 4.8],
[6.7, 3. , 5. ],
[6. , 2.9, 4.5],
[5.7, 2.6, 3.5],
[5.5, 2.4, 3.8],
[5.5, 2.4, 3.7],
[5.8, 2.7, 3.9],
[6. , 2.7, 5.1],
[5.4, 3. , 4.5],
[6. , 3.4, 4.5],
[6.7, 3.1, 4.7],
[6.3, 2.3, 4.4],
[5.6, 3. , 4.1],
[5.5, 2.5, 4. ],
[5.5, 2.6, 4.4],
[6.1, 3. , 4.6],
[5.8, 2.6, 4. ],
[5. , 2.3, 3.3],
[5.6, 2.7, 4.2],
[5.7, 3. , 4.2],
[5.7, 2.9, 4.2],
[6.2, 2.9, 4.3],
[5.1, 2.5, 3. ],
[5.7, 2.8, 4.1],
[6.3, 3.3, 6. ],
[5.8, 2.7, 5.1],
[7.1, 3. , 5.9],
```

```
       [6.3, 2.9, 5.6],
       [6.5, 3. , 5.8],
       [7.6, 3. , 6.6],
       [4.9, 2.5, 4.5],
       [7.3, 2.9, 6.3],
       [6.7, 2.5, 5.8],
       [7.2, 3.6, 6.1],
       [6.5, 3.2, 5.1],
       [6.4, 2.7, 5.3],
       [6.8, 3. , 5.5],
       [5.7, 2.5, 5. ],
       [5.8, 2.8, 5.1],
       [6.4, 3.2, 5.3],
       [6.5, 3. , 5.5],
       [7.7, 3.8, 6.7],
       [7.7, 2.6, 6.9],
       [6. , 2.2, 5. ],
       [6.9, 3.2, 5.7],
       [5.6, 2.8, 4.9],
       [7.7, 2.8, 6.7],
       [6.3, 2.7, 4.9],
       [6.7, 3.3, 5.7],
       [7.2, 3.2, 6. ],
       [6.2, 2.8, 4.8],
       [6.1, 3. , 4.9],
       [6.4, 2.8, 5.6],
       [7.2, 3. , 5.8],
       [7.4, 2.8, 6.1],
       [7.9, 3.8, 6.4],
       [6.4, 2.8, 5.6],
       [6.3, 2.8, 5.1],
       [6.1, 2.6, 5.6],
       [7.7, 3. , 6.1],
       [6.3, 3.4, 5.6],
       [6.4, 3.1, 5.5],
       [6. , 3. , 4.8],
       [6.9, 3.1, 5.4],
       [6.7, 3.1, 5.6],
       [6.9, 3.1, 5.1],
       [5.8, 2.7, 5.1],
       [6.8, 3.2, 5.9],
       [6.7, 3.3, 5.7],
       [6.7, 3. , 5.2],
       [6.3, 2.5, 5. ],
       [6.5, 3. , 5.2],
       [6.2, 3.4, 5.4],
       [5.9, 3. , 5.1]])
```

In [16]:

```
kmeans = KMeans(n_clusters = 3, init = 'k-means++',
                max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

In [21]:

```
# Predict the cluster labels: labels
labels = kmeans.predict(x)
```

In [23]:

```
labels
```

Out[23]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

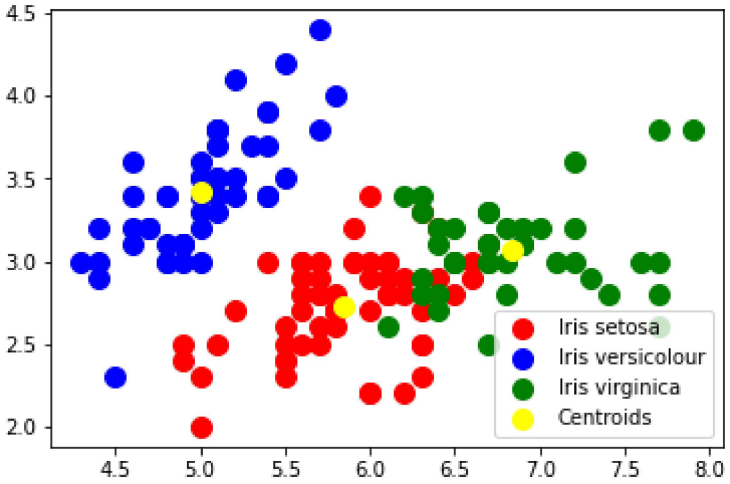
In [20]:

```
plt.scatter(x[labels == 0, 0], x[labels == 0, 1],
            s = 100, c = 'red', label = 'Iris setosa')
plt.scatter(x[labels == 1, 0], x[labels == 1, 1],
            s = 100, c = 'blue', label = 'Iris versicolour')
plt.scatter(x[labels == 2, 0], x[labels == 2, 1],
            s = 100, c = 'green', label = 'Iris virginica')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
            s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```

Out[20]: <matplotlib.legend.Legend at 0x275f8926220>



In [24]:

```
Species = ['Iris-setosa', 'Iris-versicolour','Iris-virginica']
Species_ = []
for i in labels:
    Species_.append(Species[i])
```

In [25]:

```
Species_
```

Out[25]:

```
['Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
```

```
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-versicolour',
'Iris-virginica',
'Iris-setosa',
'Iris-virginica',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
```

```
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-virginica',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-setosa',
'Iris-virginica',
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-setosa',
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-setosa',
'Iris-virginica',
'Iris-setosa',
'Iris-virginica',
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-setosa',
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-virginica',
```

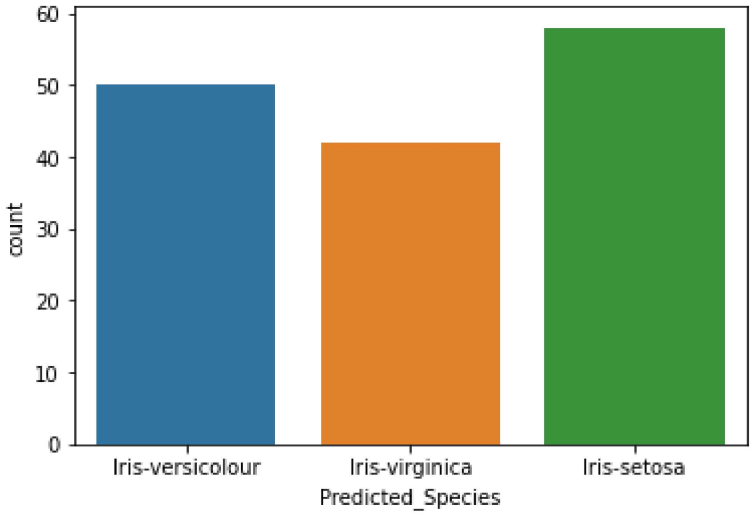
```
'Iris-setosa',
'Iris-virginica',
'Iris-virginica',
'Iris-setosa']
```

In [26]:
```
df['Predicted_Species'] = Species_
```

In [27]:
```
sns.countplot(df['Predicted_Species'])
```

```
C:\Users\LEN\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will re
sult in an error or misinterpretation.
  warnings.warn(
```

Out[27]: <AxesSubplot:xlabel='Predicted_Species', ylabel='count'>



In [ ]: