Principal Component Analysis

Load python library

In [107]:
```
# Importing pandas to perform operations using DataFrames
import pandas as pd

# Importing numpy to perform Matrix operations
import numpy as np

# Importing matplotlib to plot graphs
import matplotlib.pyplot as plt
import seaborn as sns

# Importing the following libraries for preprocessing
from sklearn.preprocessing import StandardScaler

# Importing the library for PCA
from sklearn.decomposition import PCA
```

In [108]:
```
# importing IRIS data set
iris_data = pd.read_csv('Iris_data.csv')
```

In [109]:
```
# Information on the data
print (iris_data.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
None
```

In [110]:
```
iris_data.head()
```

Out[110]:

	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

In [111]:
```
# dropping column 'Id'
iris_data = iris_data.drop(['Id'],axis=1)
iris_data.head()
```

Out[111]:

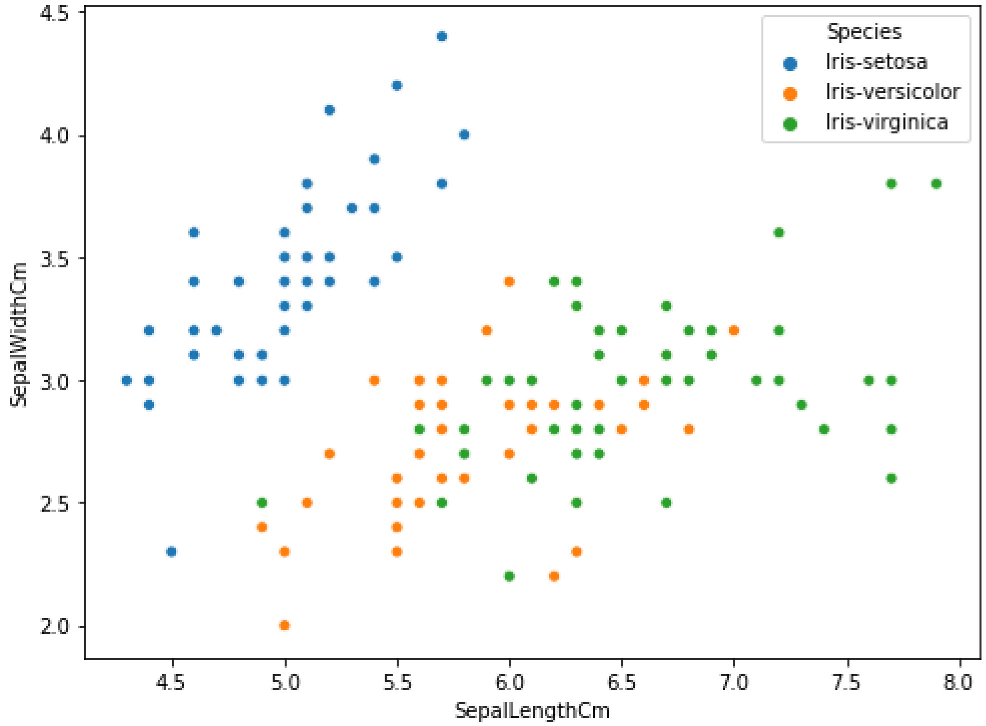
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [112]:
```
# Sort the columns of X
input_columns = list(iris_data.iloc[:,:4].columns)
input_columns.sort()
input_data=iris_data[input_columns]
input_data.head()
```

Out[112]:

	PetalLengthCm	PetalWidthCm	SepalLengthCm	SepalWidthCm
0	1.4	0.2	5.1	3.5
1	1.4	0.2	4.9	3.0
2	1.3	0.2	4.7	3.2
3	1.5	0.2	4.6	3.1
4	1.4	0.2	5.0	3.6

In [113]:
```
plt.subplots(figsize=(8,6))
sns.scatterplot(x='SepalLengthCm', y='SepalWidthCm', hue='Species', data=iris_data)
plt.show()
```



In [115]:
```
# Scaling data using (x-mu)
scaler = StandardScaler(with_std=False)
input_data = scaler.fit_transform(input_data)
input_data = pd.DataFrame(input_data,columns=input_columns)
input_data.head()
```

Out[115]:

	PetalLengthCm	PetalWidthCm	SepalLengthCm	SepalWidthCm
0	-2.358667	-0.998667	-0.743333	0.446
1	-2.358667	-0.998667	-0.943333	-0.054
2	-2.458667	-0.998667	-1.143333	0.146
3	-2.258667	-0.998667	-1.243333	0.046
4	-2.358667	-0.998667	-0.843333	0.546

In [134]:
```
u, s, v = np.linalg.svd(input_data)  #decomposing using SVD
exp_var=s**2/np.sum(s**2)*100  # Explained variance by each eigen value/PC
pc=iris_data[input_columns].dot(v.T) # Rotating and trnsforming from sample space to feature space
pc.columns=['PC1','PC2','PC3','PC4']
pc['Species']= iris_data.Species
pc.head()
```

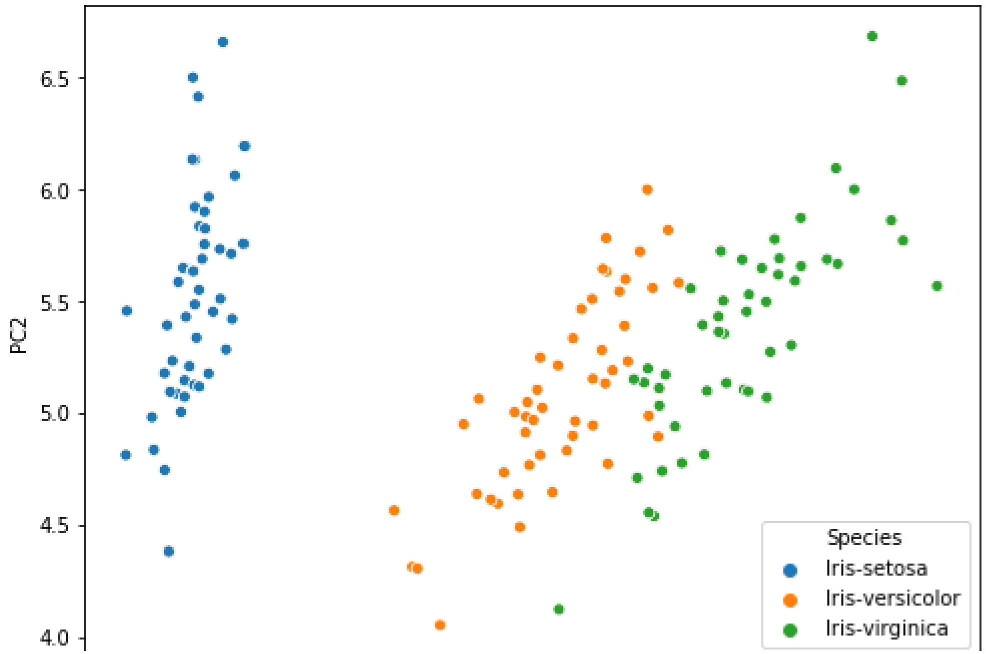
Out[134]:

	PC1	PC2	PC3	PC4	Species
0	2.827136	5.641331	-0.664277	0.037715	Iris-setosa
1	2.795952	5.145167	-0.846287	-0.060882	Iris-setosa
2	2.621524	5.177378	-0.618056	0.019416	Iris-setosa
3	2.764906	5.003599	-0.605093	0.114676	Iris-setosa
4	2.782750	5.648648	-0.546535	0.101849	Iris-setosa

In [132]:
```
exp_var
```

Out[132]:
```
array([92.46162072,  5.30155679,  1.71851395,  0.51830855])
```

In [118]:
```python
plt.subplots(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='Species', data=pc)
plt.show()
```



In [116]:
```python
# computing covariance using scaled data (renamed the data as 'input_data')
covariance_matrix           = input_data.cov()
print (covariance_matrix)
```

```
               PetalLengthCm  PetalWidthCm  SepalLengthCm  SepalWidthCm
PetalLengthCm       3.113179      1.296387       1.273682     -0.321713
PetalWidthCm        1.296387      0.582414       0.516904     -0.117981
SepalLengthCm       1.273682      0.516904       0.685694     -0.039268
SepalWidthCm       -0.321713     -0.117981      -0.039268      0.188004
```

In [119]:

```
# Computing Eigen values and Eigen vectors of the Covariance Matrix
eig_vals, eig_vecs = np.linalg.eig(covariance_matrix.values)
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i])for i in range(len(eig_vals))]
eig_pairs
```

Out[119]:

```
[(4.224840768320115,
  array([ 0.85657211,  0.35884393,  0.36158968, -0.08226889])),
 (0.24224357162751567,
  array([ 0.1757674 ,  0.07470647, -0.65653988, -0.72971237])),
 (0.023683027126002221,
  array([-0.47971899,  0.75112056,  0.31725455, -0.32409435])),
 (0.07852390809415481,
  array([ 0.07252408,  0.54906091, -0.58099728,  0.59641809]))]
```

In [120]:

```
#abs - absolute value
eig_pairs.sort(key = lambda x: x[0], reverse=True)# sort eig_pairs in descending order based on the eigen values
#false for ascending order

print('Eigenvalues in descending order:')
for i in eig_pairs:
    print(i[0])
```

```
Eigenvalues in descending order:
4.224840768320115
0.24224357162751567
0.07852390809415481
0.023683027126002221
```

In [121]:
```python
# setting threshold as '95% variance'
threshold = 0.95
# Computing number of PCS required to captured specified variance
print('Explained variance in percentage:\n')
cumulative_variance = 0.0
count          = 0
eigv_sum       = np.sum(eig_vals)
for i,j in enumerate(eig_pairs):
    variance_explained = (j[0]/eigv_sum).real
    print('eigenvalue {}: {}'.format(i+1, variance_explained*100 ))
    cumulative_variance     += variance_explained
    count                = count+1
    if (cumulative_variance>=threshold):
        break
print('\nCumulative variance=',cumulative_variance*100)
```

```
Explained variance in percentage:

eigenvalue 1: 92.46162071742681
eigenvalue 2: 5.3015567850535055

Cumulative variance= 97.76317750248033
```

In [122]:
```python
print('Total no. of eig vecs =',len(eig_vecs),'\nselected no. of eig vecs =',count)
```

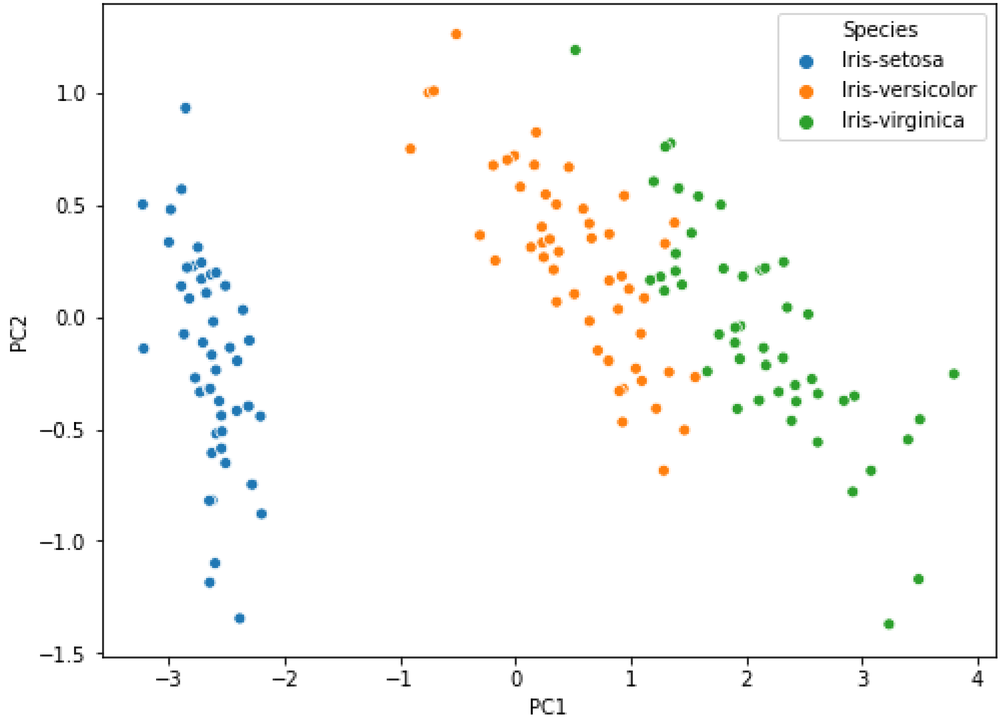
```
Total no. of eig vecs = 4
selected no. of eig vecs = 2
```

In [123]:
```python
# select required PCs based on the count  - projection matrix w=d*k
reduced_dimension   = np.zeros((len(eig_vecs),count))
for i in range(count):
    reduced_dimension[:,i]= eig_pairs[i][1]

# Projecting the scaled data onto the reduced space (using eigen vectors)
projected_data = input_data.values.dot(reduced_dimension)
projected_dataframe = pd.DataFrame(projected_data,
                                   columns=['PC1','PC2'])

projected_dataframe_with_class_info = pd.concat([projected_dataframe,
                                            iris_data.Species],axis=1)
```

In [124]:
```
plt.subplots(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='Species', data=projected_dataframe_with_class_info)
plt.show()
```



In [125]:
```
# Choosing the extent of variance to be covered by the PCs
PCA_Sklearn = PCA(n_components=0.95)

# Transforming the iris data input_columns
Projected_data_sklearn= PCA_Sklearn.fit_transform(iris_data.iloc[:,:4])

# Storing the PCs in the data frame
Projected_data_sklearn_df = pd.DataFrame(Projected_data_sklearn,
                                         columns=['PC1','PC2'])
```

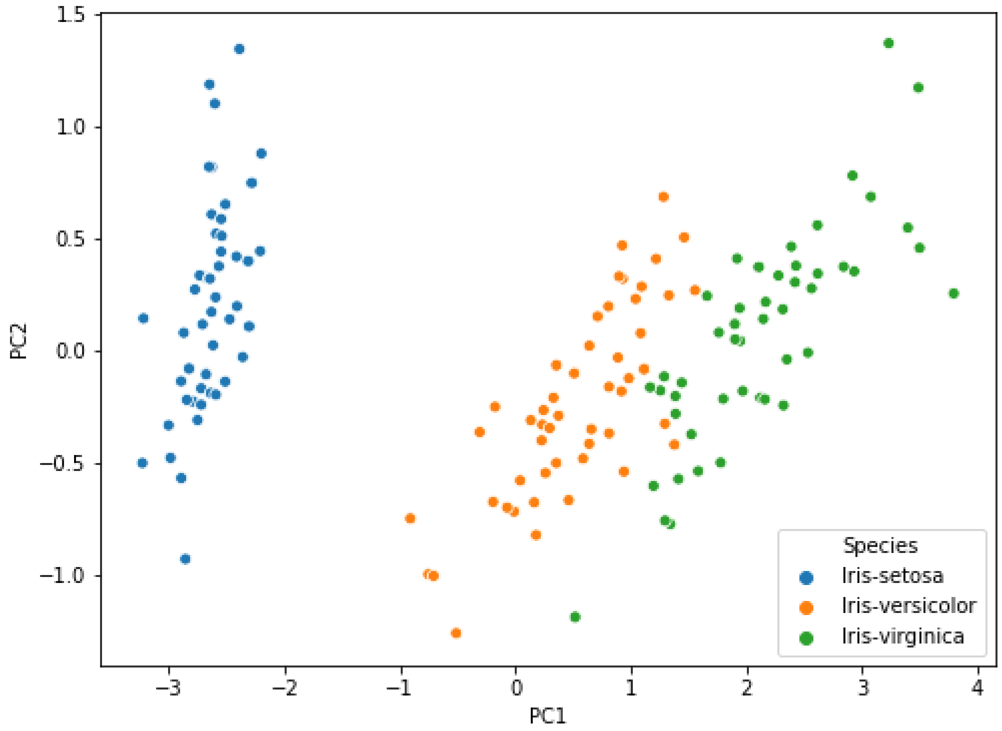
In [126]:
```
# Storing the PCs in the data frame along with class label
Projected_data_sklearn_df_with_class_info=pd.concat([Projected_data_sklearn_df,
                                                     iris_data.Species],axis=1)

print('Explained variance :\n')
print(PCA_Sklearn.explained_variance_ratio_)
```

```
Explained variance :

[0.92461621 0.05301557]
```

In [127]:
```
plt.subplots(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='Species', data=Projected_data_sklearn_df_with_class_info)
plt.show()
```



END OF SCRIPT