Olimov Bekhzod (올리모브벡조드)

Kyungpook National University

Principal Component Analysis using numpy

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline
```

Import Iris dataset

In [2]:

```
iris = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data', hea
der=None)
iris.head()
```

Out[2]:

	0	1	2	3	4
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 [3]:

```
iris.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
iris.dropna(how='all', inplace=True)
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 7.0+ KB
```

Data Visualization

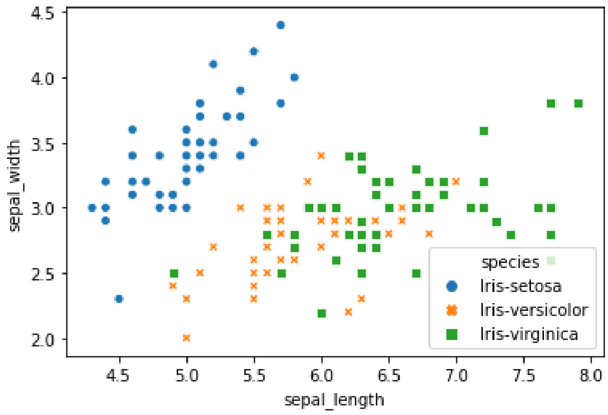
In [4]:

```
sns.scatterplot(x = iris.sepal_length, y = iris.sepal_width,
               hue = iris.species, style = iris.species)
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24b40aa4438>
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24b49aa44438>
```



Data Standardization

In [5]:

```python
# Extracting features and labels
X = iris.iloc[:, :4].values
y = iris.species.values
print(f"There are {len(X)} number of instances and {len(np.unique(y))} classes in the dataset")

# Data Standardization
print("\nMean and standard deviation before standardization: {:.3f}, {:.3f}".format(X.mean(),
X.std()))
mean = np.mean(X)
std = np.std(X)
X = (X - mean) / std
print("Mean and standard deviation after standardization: {:.3f}, {:.3f}".format(X.mean(), X.std())
)
```

```
There are 150 number of instances and 3 classes in the dataset

Mean and standard deviation before standardization: 3.464, 1.974
Mean and standard deviation after standardization: 0.000, 1.000
```

Eigenvectors and eigenvalues

In [6]:

```python
# Convariance Matrix
cov_matrix = np.cov(X.T)
print(f"Covariance matrix:\n\n {cov_matrix}")

# Eigen Values and Eigen Vectors
eigen_values, eigen_vectors = np.linalg.eig(cov_matrix)
print(f"\nEigenvalues:\n {eigen_values}")
print(f"\nEigenvectors:\n {eigen_vectors}")
```

```
Covariance matrix:

 [[ 0.17596865 -0.01007741  0.32686347  0.13265237]
 [-0.01007741  0.04824723 -0.08256073 -0.03027737]
 [ 0.32686347 -0.08256073  0.79893127  0.33269027]
 [ 0.13265237 -0.03027737  0.33269027  0.14946425]]

Eigenvalues:
 [1.08421551 0.06216666 0.02015149 0.00607775]

Eigenvectors:
 [[ 0.36158968 -0.65653988 -0.58099728  0.31725455]
 [-0.08226889 -0.72971237  0.59641809 -0.32409435]
 [ 0.85657211  0.1757674   0.07252408 -0.47971899]
 [ 0.35884393  0.07470647  0.54906091  0.75112056]]
```

Explained Variance

In [7]:

```
explained_variance = [eigvalue / sum(eigen_values) * 100 for eigvalue in eigen_values]

for index, value in enumerate(explained_variance):
    print("{} component(s) retain(s) {:.2f}% variance".format(index + 1, value))
```

```
1 component(s) retain(s) 92.46% variance
2 component(s) retain(s) 5.30% variance
3 component(s) retain(s) 1.72% variance
4 component(s) retain(s) 0.52% variance
```

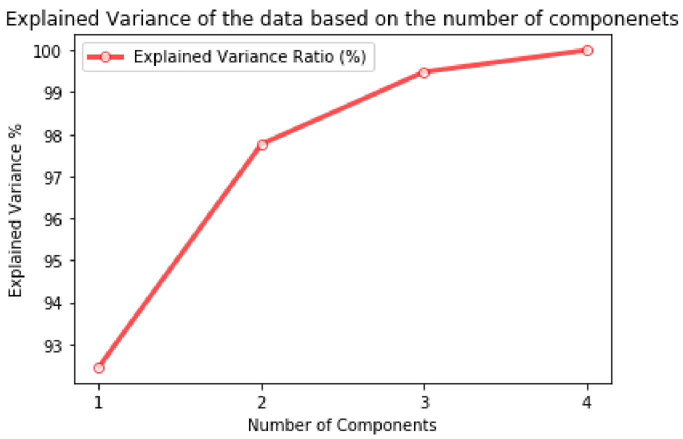
In [8]:

```
cum_explained_variance = np.cumsum(explained_variance)

plt.plot(np.arange(1, 5, 1), cum_explained_variance, label='Explained Variance Ratio (%)',
         marker='o', ms=6, markerfacecolor='w', linewidth=3, c="r", alpha=0.7)
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance %')
plt.title('Explained Variance of the data based on the number of componenets')
plt.xticks(np.arange(1,5,1), labels=['1', '2', '3', '4'])
plt.legend()
```

Out[8]:

```
<matplotlib.legend.Legend at 0x24b4adf2940>
```



Data Projection

In [9]:

```
proj_matrix = (eigen_vectors.T[:][:])[:2].T
print(f'Projection matrix:\n {proj_matrix}')
```

```
Projection matrix:
 [[ 0.36158968 -0.65653988]
 [-0.08226889 -0.72971237]
 [ 0.85657211  0.1757674 ]
 [ 0.35884393  0.07470647]]
```

2D representation of the Iris dataset

In [10]:

```
X_transformed = X.dot(proj_matrix)

for label in ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']:
```

```
    plt.scatter(X_transformed[y == label, 0],
                X_transformed[y == label, 1], label=label)
    plt.legend()
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
```