# Implementing Principal Component Analysis from scratch with Python in OOP

Principial Component Analysis (PCA) is an unsupervised machine learning method that is often used to reduce the dimentionsality of the dataset by transforming a large set into a lower dimensional set that still contains most of the information of the large set.

PCA finds a new set of dimensions such that all the dimensions are orthogonal (and hence linearly independent) and ranked according to the variance of data along them.

- The transfomed features should be linearly independent
- Dimensionality can be reduced by taking only the dimensions with the highest importance
- Dimensions should minimize projection error
- Projected points should have maximum spread (i.e., maximum variance)

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

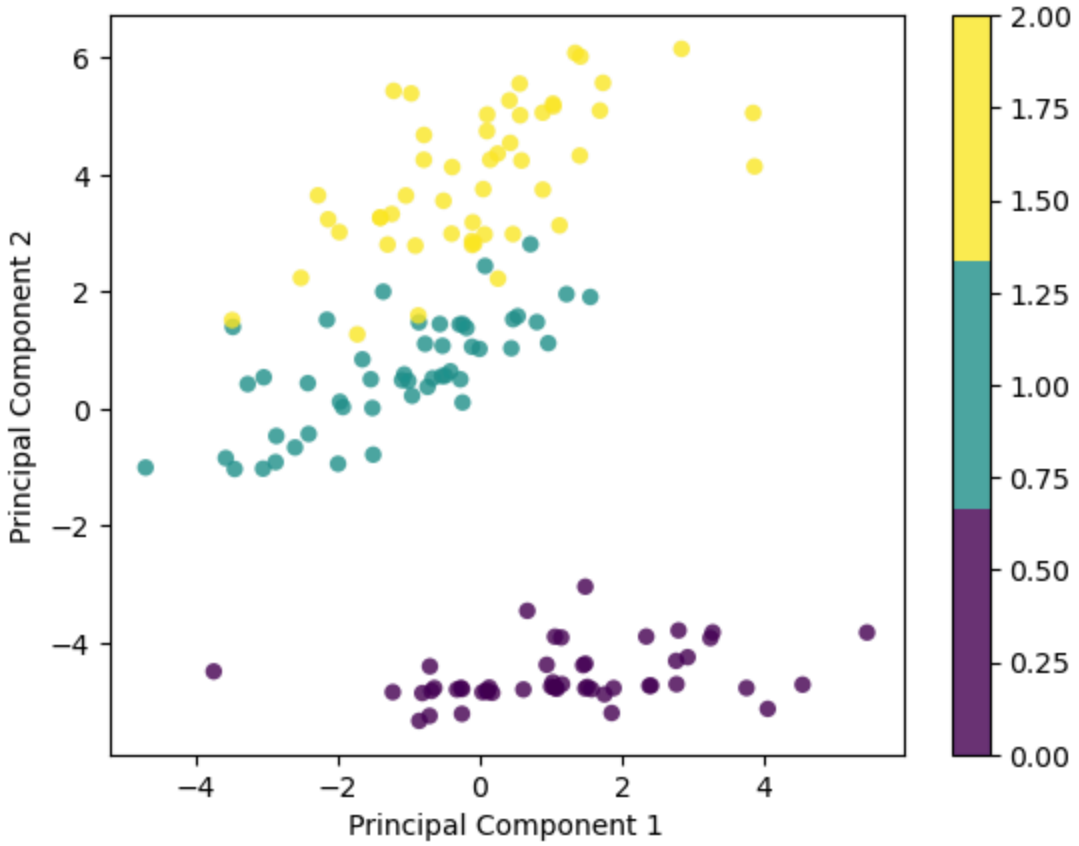
In [5]:
```
class PCA:
    def __init__(self, n_components):
        self.n_components = n_components
        self.components = None
        self.mean = None

    def fit(self, X): # only need X, no class labels because unsupervised
        # Mean centering
        self.mean = np.mean(X, axis=0)
        X = X - self.mean

        # Covariance, function needs samples as columns
        cov = np.cov(X.T)

        # Calculate eigenvectors and eigenvalues
        eigenvectors, eigenvalues = np.linalg.eig(cov)
        # Transpose eigenvectors, v = [:,i] column vector
        eigenvectors = eigenvectors.T

        # Sort the eigenvectors according to eigenvalues
        indxs = np.argsort(eigenvalues)[::-1] # in decreasing order
        eigenvalues = eigenvalues[indxs]
        eigenvectors = eigenvectors[indxs]

        # Only save first k components
        self.components = eigenvectors[:self.n_components]

    def transform(self,X):
```

```
        X = X - self.mean

        # Project the data
        return np.dot(X, self.components.T)
```

In [7]:
```
# Testing
if __name__ == "__main__":
    # Import data
    data = datasets.load_iris()
    X = data.data
    y = data.target

    # Project the data onto the 2 primary principal components
    pca = PCA(2)
    pca.fit(X)
    X_projected = pca.transform(X)

    print(f"Shape of X: {X.shape}")
    print(f"Shape of transformed X: {X_projected.shape}")

    # Extract first 2 dimensions of projected data and plot
    x1 = X_projected[:,0]
    x2 = X_projected[:,1]

    plt.scatter(x1, x2, c=y, edgecolor="none", alpha=0.8, cmap=plt.cm.get_cm
    plt.xlabel("Principal Component 1")
    plt.ylabel("Principal Component 2")
    plt.colorbar()
    plt.show()
```

```
Shape of X: (150, 4)
Shape of transformed X: (150, 2)
```

```
/var/folders/cx/jsmdsr392b16bs81k3s1s4n00000gn/T/ipykernel_89303/2975043885.
py:20: MatplotlibDeprecationWarning: The get_cmap function was deprecated in
Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]
`` or ``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
  plt.scatter(x1, x2, c=y, edgecolor="none", alpha=0.8, cmap=plt.cm.get_cmap
("viridis",3))
```


Principal Component 2
Principal Component 1
6
4
2
0
−2
−4
−4
−2
0
2
4
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00