ASSIGNMENT-10: PRINCIPAL COMPONENT ANALYSIS

```
Objectives
  1. Understand the concept and application of PCA.
  2. Implement PCA for dimensionality reduction.
  3. Visualize and interpret the results of PCA.
Dataset
  Use the Wine dataset available from the UCI Machine Learning Repository.
```

Task 1: Data Exploration and Preprocessing

```
  1 Load the dataset and display the first few rows.
  2 Perform basic statistical analysis to understand the distribution of the features.
  3 Check for missing values and handle them appropriately.
  4 Standardize the features if necessary.
```

```
#Answer  1. Load the dataset and display the first few rows.
from sklearn.datasets import load_wine
import pandas as pd

# Load the Wine dataset
wine_data = load_wine()
wine_df = pd.DataFrame(wine_data.data, columns=wine_data.feature_names)
wine_df['target'] = wine_data.target

# Display the first few rows of the dataset
print("1. First 5 rows of the dataset:\n")
print(wine_df.head())
```

```
1. First 5 rows of the dataset:

   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80
1    13.20        1.78  2.14               11.2      100.0           2.65
2    13.16        2.36  2.67               18.6      101.0           2.80
3    14.37        1.95  2.50               16.8      113.0           3.85
4    13.24        2.59  2.87               21.0      118.0           2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04
1        2.76                  0.26             1.28             4.38  1.05
2        3.24                  0.30             2.81             5.68  1.03
3        3.49                  0.24             2.18             7.80  0.86
4        2.69                  0.39             1.82             4.32  1.04

   od280/od315_of_diluted_wines  proline  target
0                          3.92   1065.0       0
1                          3.40   1050.0       0
2                          3.17   1185.0       0
3                          3.45   1480.0       0
4                          2.93    735.0       0
```

```
wine_df['target'].value_counts()
```

target	count
1	71
0	59
2	48

dtype: int64

```
#2.  Perform basic statistical analysis
print("2. Statistical summary of the dataset:\n")
print(wine_df.describe())
```

```
2. Statistical summary of the dataset:

          alcohol  malic_acid         ash  alcalinity_of_ash   magnesium  \
count  178.000000  178.000000  178.000000         178.000000  178.000000   
mean    13.000618    2.336348    2.366517          19.494944   99.741573   
std      0.811827    1.117146    0.274344           3.339564   14.282484   
min     11.030000    0.740000    1.360000          10.600000   70.000000   
25%     12.362500    1.602500    2.210000          17.200000   88.000000   
50%     13.050000    1.865000    2.360000          19.500000   98.000000   
75%     13.677500    3.082500    2.557500          21.500000  107.000000   
max     14.830000    5.800000    3.230000          30.000000  162.000000   

       total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
count     178.000000  178.000000            178.000000       178.000000   
mean        2.295112    2.029270              0.361854         1.590899   
std         0.625851    0.998859              0.124453         0.572359   
min         0.980000    0.340000              0.130000         0.410000   
25%         1.742500    1.205000              0.270000         1.250000   
50%         2.355000    2.135000              0.340000         1.555000   
75%         2.800000    2.875000              0.437500         1.950000   
max         3.880000    5.080000              0.660000         3.580000   

       color_intensity         hue  od280/od315_of_diluted_wines      proline  \
count       178.000000  178.000000                    178.000000   178.000000   
mean          5.058090    0.957449                      2.611685   746.893258   
std           2.318286    0.228572                      0.709990   314.907474   
min           1.280000    0.480000                      1.270000   278.000000   
25%           3.220000    0.782500                      1.937500   500.500000   
50%           4.690000    0.965000                      2.780000   673.500000   
75%           6.200000    1.120000                      3.170000   985.000000   
max          13.000000    1.710000                      4.000000  1680.000000   

           target  
count  178.000000  
mean     0.938202  
std      0.775035  
min      0.000000  
25%      0.000000  
50%      1.000000  
75%      2.000000  
max      2.000000  
```

```
#3. Check for missing values
print("3. Missing values in the dataset:\n")
print(wine_df.isnull().sum())
```

```
3. Missing values in the dataset:

alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
target                          0
dtype: int64
```

```
# 4. Standardize the features (excluding the target column)
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
standardized_data = scaler.fit_transform(wine_df.iloc[:, :-1])  # Exclude the target column
standardized_df = pd.DataFrame(standardized_data, columns=wine_data.feature_names)
standardized_df['target'] = wine_df['target']

print("4. First 5 rows of the standardized dataset:\n")
print(standardized_df.head())
```

```
4. First 5 rows of the standardized dataset:

    alcohol  malic_acid       ash  alcalinity_of_ash  magnesium  \
0  1.518613   -0.562250  0.232053          -1.169593   1.913905   
1  0.246290   -0.499413 -0.827996          -2.490847   0.018145   
2  0.196879    0.021231  1.109334          -0.268738   0.088358   
3  1.691550   -0.346811  0.487926          -0.809251   0.930918   
4  0.295700    0.227694  1.840403           0.451946   1.281985   
```

```
   total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
0       0.808997    1.034819             -0.659563         1.224884
1       0.568648    0.733629             -0.820719        -0.544721
2       0.808997    1.215533             -0.498407         2.135968
3       2.491446    1.466525             -0.981875         1.032155
4       0.808997    0.663351              0.226796         0.401404

   color_intensity       hue  od280/od315_of_diluted_wines   proline  target
0         0.251717  0.362177                      1.847920  1.013009       0
1        -0.293321  0.406051                      1.113449  0.965242       0
2         0.269020  0.318304                      0.788587  1.395148       0
3         1.186068 -0.427544                      1.184071  2.334574       0
4        -0.319276  0.362177                      0.449601 -0.037874       0
```

```
wine_df.corr()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_ph
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	-0.1
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	0.2
ash	0.211545	0.164045	1.000000	0.443367	0.286587	0.128980	0.115077	0.1
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0.321113	-0.351370	0.3
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.214401	0.195784	-0.2
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.000000	0.864564	-0.4
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.864564	1.000000	-0.5
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0.449935	-0.537900	1.0
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.612413	0.652692	-0.3
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	-0.055136	-0.172379	0.1
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	0.433681	0.543479	-0.2
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	0.699949	0.787194	-0.5
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	0.498115	0.494193	-0.3
target	-0.328222	0.437776	-0.049643	0.517859	-0.209179	-0.719163	-0.847498	0.4

```
wine_df.shape
```

```
(178, 14)
```

Task 2: Implement PCA

1. Perform PCA on the standardized dataset to reduce dimensionality.
2. Determine the number of principal components to retain by analyzing the explained variance ratio.

```
# Step 2: Perform PCA
from sklearn.decomposition import PCA
pca = PCA()
pca_result = pca.fit_transform(standardized_data)
```

```
pca_result.shape
```

```
(178, 13)
```

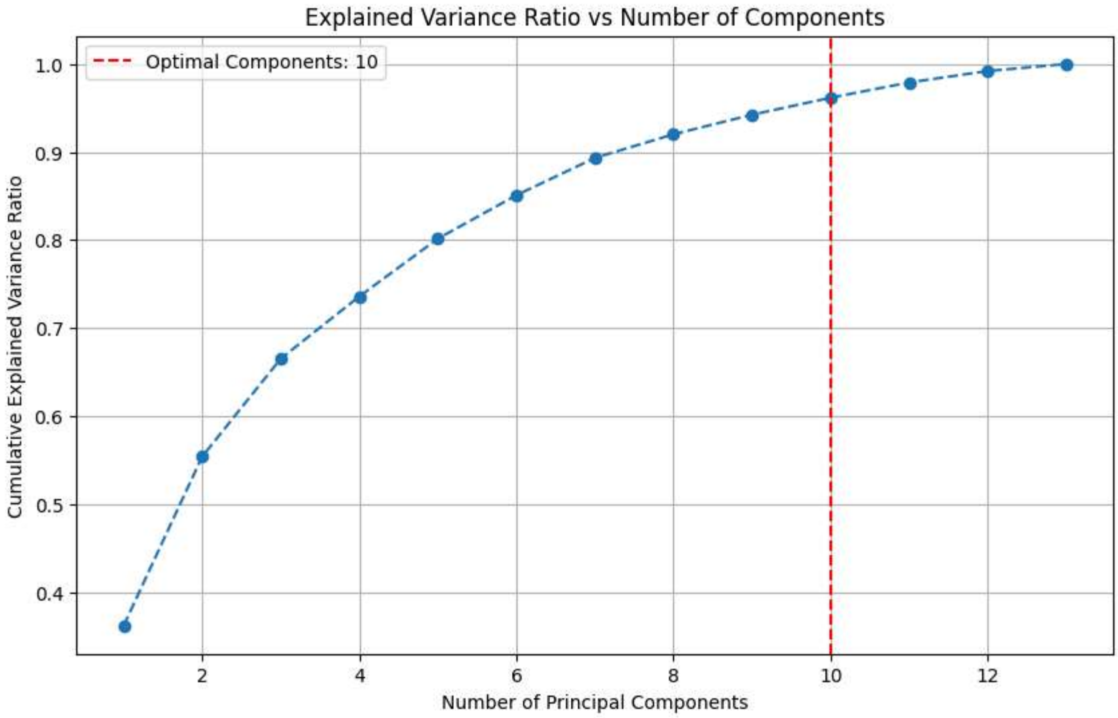
```
# Step 3: Analyze explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance_ratio = explained_variance_ratio.cumsum()



print("Explained Variance Ratio for Each Component:\n", explained_variance_ratio)
print("\nCumulative Explained Variance Ratio:\n", cumulative_variance_ratio)


# Step 4: Determine optimal number of components (95% variance)
optimal_components = (cumulative_variance_ratio >= 0.95).argmax() + 1  # Add 1 as index is 0-based
```

```
# Step 5: Plot cumulative explained variance ratio
from matplotlib import pyplot as plt
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance_ratio) + 1), cumulative_variance_ratio, marker='o', linestyle='--')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.title('Explained Variance Ratio vs Number of Components')
plt.axvline(optimal_components, color='r', linestyle='--', label=f'Optimal Components: {optimal_components}')
plt.legend()
plt.grid()
plt.show()
```




```
# Output results
print("Explained Variance Ratio for Each Component:\n", explained_variance_ratio)
print("\nCumulative Explained Variance Ratio:\n", cumulative_variance_ratio)
print(f"\nOptimal Number of Components to Retain (95% Variance): {optimal_components}")
```

```
Explained Variance Ratio for Each Component:
 [0.36198848 0.1920749  0.11123631 0.0706903  0.06563294 0.04935823
 0.04238679 0.02680749 0.02222153 0.01930019 0.01736836 0.01298233
 0.00795215]

Cumulative Explained Variance Ratio:
 [0.36198848 0.55406338 0.66529969 0.73598999 0.80162293 0.85098116
 0.89336795 0.92017544 0.94239698 0.96169717 0.97906553 0.99204785
 1.        ]

Optimal Number of Components to Retain (95% Variance): 10
```

Task 3: Visualization of Principal Components

1. Visualize the data in the new principal component space using scatter plots.
2. Color-code the scatter plots by the wine cultivars to see if the PCA helps in distinguishing between the classes.

```
# Import necessary library for visualization
import seaborn as sns

# Step 1: Reduce the dataset to the first two principal components
pca_10 = PCA(n_components=10)
pca_10_result = pca_10.fit_transform(standardized_data)
```

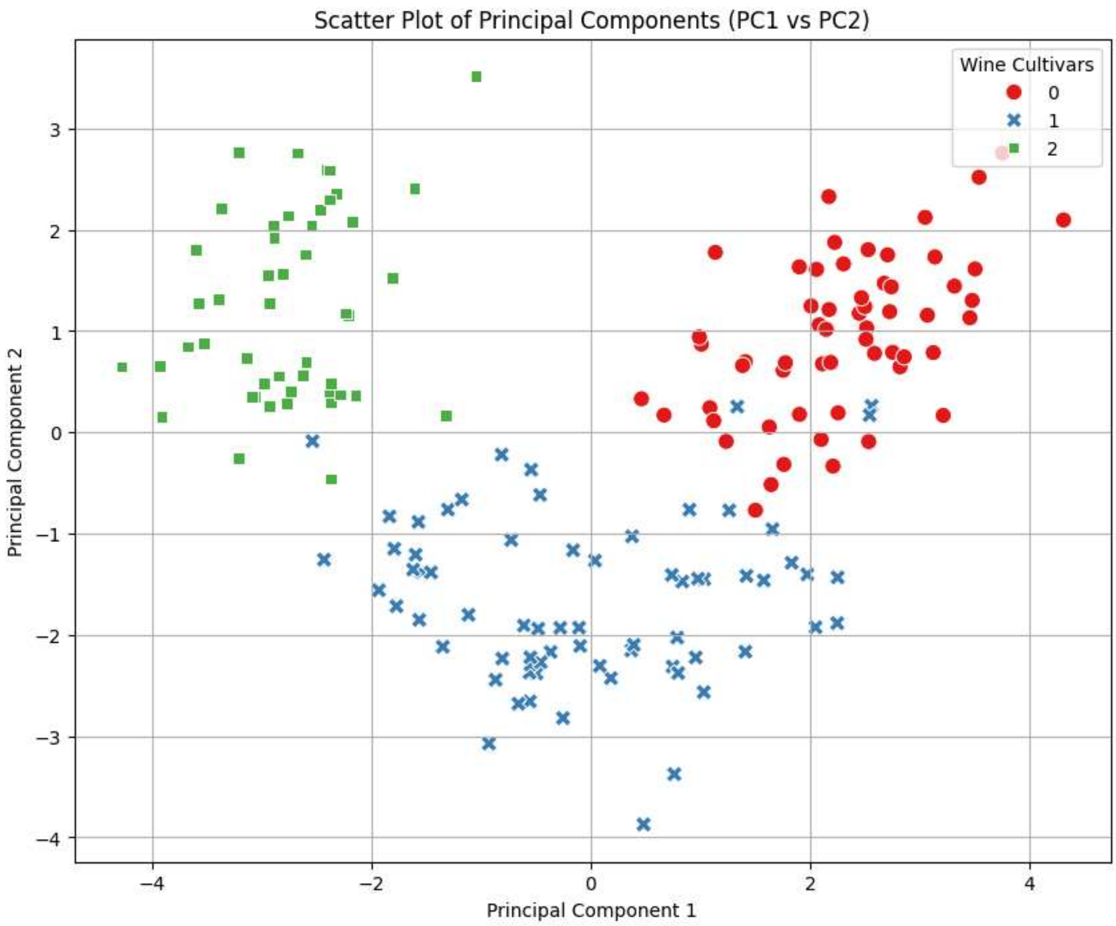
```
pca_10_result.shape
```

```
(178, 10)
```

```
pca_df = pd.DataFrame(pca_result[:, :10], columns=[f'PC{i+1}' for i in range(10)])  # First 10 PCs
pca_df['target'] = wine_df['target']


# Step 2: Visualize the data in the first two principal components
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.scatterplot(
    x=pca_df['PC1'],
    y=pca_df['PC2'],
    hue=pca_df['target'],
    palette='Set1',
    style=pca_df['target'],
    s=80
)
plt.title('Scatter Plot of Principal Components (PC1 vs PC2)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Wine Cultivars', loc='upper right')
plt.grid()
plt.show()
```



Task 4: Interpretation of Results

1. Analyze the loadings (coefficients) of the original features on the principal components.
2. Discuss how the principal components can be interpreted based on the loadings.

```
# Step 1: Analyze the loadings
loadings = pd.DataFrame(
    pca.components_,
    columns=wine_data.feature_names,
    index=[f'PC{i+1}' for i in range(len(pca.components_))]
)

# Display the loadings for the first few principal components
print("Loadings (Principal Component Coefficients):\n")
print(loadings.head())

# Step 2: Identify key contributors to each PC
```

```
top_features_per_pc = loadings.abs().idxmax(axis=1)  # Features with the highest loading for each PC
print("\nTop contributing features to each Principal Component:\n")
print(top_features_per_pc)
```

```
Loadings (Principal Component Coefficients):

      alcohol  malic_acid       ash  alcalinity_of_ash  magnesium  \
PC1  0.144329   -0.245188 -0.002051          -0.239320   0.141992
PC2  0.483652    0.224931  0.316069          -0.010591   0.299634
PC3 -0.207383    0.089013  0.626224           0.612080   0.130757
PC4 -0.017856    0.536890 -0.214176           0.060859  -0.351797
PC5 -0.265664    0.035214 -0.143025           0.066103   0.727049

     total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
PC1       0.394661    0.422934             -0.298533         0.313429
PC2       0.065040   -0.003360              0.028779         0.039302
PC3       0.146179    0.150682              0.170368         0.149454
PC4       0.198068    0.152295             -0.203301         0.399057
PC5      -0.149318   -0.109026             -0.500703         0.136860

     color_intensity       hue  od280/od315_of_diluted_wines   proline
PC1        -0.088617  0.296715                      0.376167  0.286752
PC2         0.529996 -0.279235                     -0.164496  0.364903
PC3        -0.137306  0.085222                      0.166005 -0.126746
PC4         0.065926 -0.427771                      0.184121 -0.232071
PC5        -0.076437 -0.173615                     -0.101161 -0.157869

Top contributing features to each Principal Component:

PC1                       flavanoids
PC2                  color_intensity
PC3                              ash
PC4                       malic_acid
PC5                        magnesium
PC6                       malic_acid
PC7             nonflavanoid_phenols
PC8                              hue
PC9                          proline
PC10    od280/od315_of_diluted_wines
PC11                         proline
PC12                 color_intensity
PC13                      flavanoids
dtype: object
```

Task 5: Classification Using Principal Components

1. Use the principal components as features to train a classification model (e.g., logistic regression, decision tree).
2. Evaluate the classification performance and compare it with the performance using the original features.

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# Prepare data for classification
X_pca = pca_result[:, :10]  # Use the first 10 principal components
X_original = standardized_data  # Original features
y = wine_df['target']

# Split data into train and test sets (80-20 split)
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=42)
X_train_original, X_test_original, _, _ = train_test_split(X_original, y, test_size=0.2, random_state=42)

# Train and evaluate using PCA features
logreg_pca = LogisticRegression(max_iter=1000)
logreg_pca.fit(X_train_pca, y_train)
y_pred_pca = logreg_pca.predict(X_test_pca)

# Train and evaluate using original features
logreg_original = LogisticRegression(max_iter=1000)
logreg_original.fit(X_train_original, y_train)
y_pred_original = logreg_original.predict(X_test_original)

# Evaluate performance
print("Performance Using Principal Components:\n")
print(classification_report(y_test, y_pred_pca))

print("Performance Using Original Features:\n")
print(classification_report(y_test, y_pred_original))
```

```
Performance Using Principal Components:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         8

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```