GMM Clustering and PCA

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.mixture import GaussianMixture
from sklearn.metrics.cluster import adjusted_rand_score
import warnings

warnings.filterwarnings("ignore")

# Load the Iris dataset
iris =
pd.read_csv("https://media.githubusercontent.com/media/neurospin/pystatsml/refs/heads/master/data
print(iris)

# Data preprocessing
x = iris.iloc[:, :4]
y = iris.iloc[:, -1]
sc = StandardScaler()
sc.fit(x)
std_array = sc.transform(x)
X = pd.DataFrame(std_array, columns=x.columns)

# Gaussian Mixture Model clustering
cluster = GaussianMixture(n_components=3)
cluster.fit(X)
y_pred = cluster.predict(X)
score = adjusted_rand_score(y, y_pred)
print("Adjusted Rand Score:", score)

# PCA for dimensionality reduction
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca_array = pca.fit_transform(X)  # Use the standardized data
pca_df = pd.DataFrame(pca_array, columns=["PC1", "PC2"])

col_code = {0: "yellow", 1: "darkblue", 2: "green"}
label = {0: "setosa", 1: "versicolor", 2: "virginica"}

pca_df["labels"] = pd.DataFrame(y_pred)
groups = pca_df.groupby('labels')

# Plotting
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
for name, group in groups:
    ax.plot(group.PC1, group.PC2, color=col_code[name], label=label[name], marker='o', linestyle="",
ms=10)

ax.legend()
plt.title("PCA of Iris Dataset with GMM Clusters")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()  explain the above code word by word
```

Explanation of the Code:

Imports and Warnings

```
python


  import pandas as pd
  import numpy as np
  import seaborn as sns
  import matplotlib.pyplot as plt
  from sklearn.preprocessing import StandardScaler
  from sklearn.mixture import GaussianMixture
  from sklearn.metrics.cluster import adjusted_rand_score
  import warnings
```

1. **`pandas` (`pd`)**: Used for handling and analyzing data in tabular form.
2. **`numpy` (`np`)**: Provides support for numerical operations, particularly on arrays.
3. **`seaborn` (`sns`)**: A visualization library (not directly used here but often complements `matplotlib`).
4. **`matplotlib.pyplot` (`plt`)**: Used for data visualization.
5. **`sklearn.preprocessing.StandardScaler`**: A module for standardizing data to have a mean of 0 and a standard deviation of 1.
6. **`sklearn.mixture.GaussianMixture`**: Implements Gaussian Mixture Models (GMM) for clustering.
7. **`sklearn.metrics.cluster.adjusted_rand_score`**: A clustering evaluation metric to measure similarity between predicted and true cluster labels.
8. **`warnings.filterwarnings("ignore")`**: Suppresses warnings to avoid cluttered output.

Loading the Dataset

```
python


  iris = pd.read_csv("https://media.githubusercontent.com/media/neurospin/pystatsml/refs/head
  s/master/datasets/iris.csv")
  print(iris)
```

1. **`pd.read_csv()`**: Loads the Iris dataset directly from the provided URL.

2. `print(iris)`: Displays the dataset to verify successful loading.

Data Preprocessing

```
python


 x = iris.iloc[:, :4]
 y = iris.iloc[:, -1]
```

1. `iris.iloc[:, :4]`: Selects the first four columns of the dataset (features: `sepal length`, `sepal width`, `petal length`, `petal width`).
2. `iris.iloc[:, -1]`: Selects the last column (target labels: `species`).

```
python


 sc = StandardScaler()
 sc.fit(x)
 std_array = sc.transform(x)
 X = pd.DataFrame(std_array, columns=x.columns)
```

3. `StandardScaler`: Standardizes the feature columns (`x`) to ensure all variables are on the same scale.
4. `sc.fit(x)`: Learns the mean and standard deviation from the data.
5. `sc.transform(x)`: Applies the standardization to the data.
6. `pd.DataFrame()`: Converts the standardized array back into a DataFrame with the original column names.

Gaussian Mixture Model Clustering

```
python


 cluster = GaussianMixture(n_components=3)
 cluster.fit(X)
 y_pred = cluster.predict(X)
```

1. `GaussianMixture(n_components=3)`: Creates a Gaussian Mixture Model with 3 clusters, matching the 3 species in the dataset.
2. `cluster.fit(X)`: Fits the model to the standardized data.
3. `cluster.predict(X)`: Predicts the cluster labels for the dataset.

```
python


 score = adjusted_rand_score(y, y_pred)
 print("Adjusted Rand Score:", score)
```

4. `adjusted_rand_score(y, y_pred)`: Compares the predicted cluster labels (`y_pred`) with the true labels (`y`) to calculate the Adjusted Rand Index, a measure of clustering performance.

Principal Component Analysis (PCA)

```
python


 from sklearn.decomposition import PCA
 pca = PCA(n_components=2)
 pca_array = pca.fit_transform(X)
 pca_df = pd.DataFrame(pca_array, columns=["PC1", "PC2"])
```

1. `PCA(n_components=2)`: Reduces the dimensionality of the standardized data to 2 components for visualization.
2. `pca.fit_transform(X)`: Fits the PCA model and transforms the data into 2 principal components.
3. `pd.DataFrame()`: Converts the PCA result into a DataFrame for easy handling.

Mapping and Grouping Data

```
python


 col_code = {0: "yellow", 1: "darkblue", 2: "green"}
 label = {0: "setosa", 1: "versicolor", 2: "virginica"}
```

1. `col_code` and `label`: Maps cluster indices to specific colors and species names for plotting.

```
python


 pca_df["labels"] = pd.DataFrame(y_pred)
 groups = pca_df.groupby('labels')
```

2. `pca_df["labels"]`: Adds the predicted cluster labels (`y_pred`) as a new column in the PCA DataFrame.
3. `pca_df.groupby('labels')`: Groups the PCA DataFrame by the cluster labels for grouped visualization.

Plotting

```
python


 fig, ax = plt.subplots(1, 1, figsize=(15, 10))
```

1. `plt.subplots(1, 1, figsize=(15, 10))`: Creates a single plot with a figure size of 15x10 inches.

```python
for name, group in groups:
    ax.plot(group.PC1, group.PC2, color=col_code[name], label=label[name], marker='o', lines
tyle="", ms=10)
```

2. `for name, group in groups`: Iterates through each cluster group.
3. `ax.plot()`: Plots the PCA coordinates (`PC1`, `PC2`) for each group with the specified color, label, and style.

```python
ax.legend()
plt.title("PCA of Iris Dataset with GMM Clusters")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

4. `ax.legend()`: Adds a legend to distinguish clusters.
5. `plt.title()`, `plt.xlabel()`, `plt.ylabel()`: Adds title and axis labels for context.
6. `plt.show()`: Displays the final plot.

Outcome:

1. The code clusters the Iris dataset into 3 groups using Gaussian Mixture Models.
2. The PCA reduces the dataset to 2 dimensions for visualization.
3. The resulting plot shows the clusters in PCA space, colored and labeled according to the predicted cluster labels.