PCA on Iris Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
%matplotlib inline

iris=pd.read_csv("https://media.githubusercontent.com/media/neurospin/pystatsml/refs/heads/master/

iris.head()

iris.info()

iris.describe()

sns.heatmap(iris.drop(['species'], axis=1).corr(), annot=True, cmap="coolwarm")
plt.show()

pca = PCA()
x_new1 = pca.fit_transform(iris.drop(['species'], axis=1))
x_new1[:5]

explained_variance=pca.explained_variance_ratio_
explained_variance

plt.figure(figsize=(10,6))
plt.bar(range(4), explained_variance, alpha=0.5, align='center', label="Individual Explained Variance",
color="darkred")
plt.ylabel("Explained Variance Ratio")
plt.xlabel("Principal Component")
plt.legend(loc="best")
plt.tight_layout()
plt.show()


pca = PCA(n_components=3)
x_new = pca.fit_transform(iris.drop(['species'], axis=1))
x_new[:5]

categ_num={"variety": {"setosa":0,"versicolor":1,"virginica":2}}
iris1 = iris.replace(categ_num)
columns= list(iris.columns[:4])

fig, axes = plt.subplots(3, 4, figsize=(15, 10))
```

```
for i in range(3):  # Loop over 3 PCA components
    for j in range(4):  # Loop over 4 original features
        axes[i, j].scatter(x_new[:, i], iris[columns[j]], c="r")
        axes[i, j].set_xlabel(f'PC {i+1}')
        axes[i, j].set_ylabel(columns[j])
plt.tight_layout()
plt.show() explain the above code word by word
```

The code you provided performs **PCA (Principal Component Analysis)** on the Iris dataset and visualizes the results. Here's a detailed breakdown of the code step by step:

---

1. Importing Necessary Libraries

```
python


  import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  import seaborn as sns
  from sklearn.decomposition import PCA
  %matplotlib inline
```

- `numpy`: Used for numerical computations.
- `pandas`: Handles data manipulation and analysis, especially DataFrames.
- `matplotlib.pyplot`: For visualizations.
- `seaborn`: Provides advanced visualizations and better aesthetics.
- `sklearn.decomposition.PCA`: Library for performing PCA.
- `%matplotlib inline`: Enables inline plotting in Jupyter Notebooks.

---

2. Loading and Inspecting the Dataset

```
python


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

- Downloads the **Iris dataset** from the provided URL and loads it into a pandas DataFrame called `iris`.

```
python


  iris.head()
```

- Displays the first five rows of the DataFrame.

```
python


  iris.info()
```

- Shows the data types and non-null count for each column in the dataset.

```
python


  iris.describe()
```

- Provides summary statistics (mean, standard deviation, min, max, etc.) for each numeric column.

---

3. Correlation Heatmap

```
python


  sns.heatmap(iris.drop(['species'], axis=1).corr(), annot=True, cmap="coolwarm")
  plt.show()
```

- `iris.drop(['species'], axis=1)`: Removes the `species` column (categorical variable) to focus only on numeric data for correlation analysis.
- `.corr()`: Computes the correlation matrix for the numeric columns.
- `sns.heatmap()`: Visualizes the correlation matrix as a heatmap.
  - `annot=True`: Displays the correlation values within each cell.
  - `cmap="coolwarm"`: Sets the color scheme to highlight positive and negative correlations.
- `plt.show()`: Displays the heatmap.

---

4. Performing PCA

(a) Without Specifying Number of Components

```
python


  pca = PCA()
  x_new1 = pca.fit_transform(iris.drop(['species'], axis=1))
  x_new1[:5]
```

- `PCA()`: Initializes the PCA object.
- `pca.fit_transform()`: Fits PCA to the numeric data (excluding `species`) and transforms it into the new principal component space.
  - `x_new1`: Contains the transformed data in the new PCA space.
- `x_new1[:5]`: Displays the first five rows of the transformed data.

```
python


  explained_variance = pca.explained_variance_ratio_
  explained_variance
```

- `pca.explained_variance_ratio_`: Extracts the proportion of variance explained by each principal component.

(b) Visualizing Explained Variance

```
python


  plt.figure(figsize=(10,6))
  plt.bar(range(4), explained_variance, alpha=0.5, align='center', label="Individual Explained
  Variance", color="darkred")
  plt.ylabel("Explained Variance Ratio")
  plt.xlabel("Principal Component")
  plt.legend(loc="best")
  plt.tight_layout()
  plt.show()
```

- `plt.bar()`: Creates a bar plot of explained variance for each principal component.
- `range(4)`: Represents the 4 principal components (equal to the number of original features).
- `alpha=0.5`: Makes the bars slightly transparent.
- Labels and titles make the plot informative.

---

5. Reducing Dimensions to 3 Components

```
python


  pca = PCA(n_components=3)
  x_new = pca.fit_transform(iris.drop(['species'], axis=1))
  x_new[:5]
```

- `n_components=3`: Limits the PCA transformation to 3 principal components.
- `x_new`: Contains the transformed data in 3D PCA space.
- `x_new[:5]`: Displays the first five rows of the reduced data.

---

6. Data Preparation and Visualization

```
python


  categ_num = {"variety": {"setosa": 0, "versicolor": 1, "virginica": 2}}
  iris1 = iris.replace(categ_num)
```

- `categ_num`: Creates a mapping to convert the `species` categorical variable into numeric values (not used further in this example).

- `iris1`: Replaces the `species` column in the DataFrame with its numeric equivalent.

```
python


  columns = list(iris.columns[:4])
```

- Extracts the names of the first 4 columns (features) into a list called `columns`.

Subplot of PCA vs Original Features

```
python


  fig, axes = plt.subplots(3, 4, figsize=(15, 10))
  for i in range(3):  # Loop over 3 PCA components
      for j in range(4):  # Loop over 4 original features
          axes[i, j].scatter(x_new[:, i], iris[columns[j]], c="r")
          axes[i, j].set_xlabel(f'PC {i+1}')
          axes[i, j].set_ylabel(columns[j])
  plt.tight_layout()
  plt.show()
```

- `plt.subplots(3, 4)`: Creates a grid of 3 rows and 4 columns for subplots.
- Nested `for` loops:
  - **Outer loop (`i`)**: Iterates over the 3 PCA components (PC1, PC2, PC3).
  - **Inner loop (`j`)**: Iterates over the 4 original features.
- `axes[i, j].scatter()`: Plots scatter plots between a PCA component and an original feature.
- Labels are dynamically set using `f'PC {i+1}'` and the feature names.

Purpose of Visualization

This step explores how each original feature is distributed in relation to the reduced PCA components, giving insight into the influence of individual features on the principal components.