In [ ]:
```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA
```

In [ ]:
```
# Step 1: Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
```

In [ ]:
```
# Step 2: Standardize the data
X_standardized = (X - np.mean(X, axis=0)) / np.std(X, axis=0)
```

In [ ]:
```
# Step 3: Calculate the covariance matrix
cov_matrix = np.cov(X_standardized, rowvar=False)

# Step 4: Compute the eigenvectors and eigenvalues of the covariance matrix
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
```

In [ ]:
```
# Step 5: Sort the eigenvectors by decreasing eigenvalues
sorted_indices = np.argsort(eigenvalues)[::-1]
sorted_eigenvalues = eigenvalues[sorted_indices]
sorted_eigenvectors = eigenvectors[:, sorted_indices]

k = 2
principal_components = sorted_eigenvectors[:, :k]
```

In [ ]:
```
# Step 7: Transform the original data
X_pca = np.dot(X_standardized, principal_components)

# Step 8: Split the Data
X_train1, X_test1, y_train1, y_test1 = train_test_split(X_pca, y, test_size=0.2,
```

In [ ]:
```
#
#  Step 9: Train a Model on Original Data
X_train, X_test, y_train, y_test = train_test_split(X_standardized, y, test_size
model_original = LogisticRegression()
model_original.fit(X_train, y_train)

# Step 10: Evaluate Model Performance on Original Data
y_pred_original = model_original.predict(X_test)
accuracy_original = accuracy_score(y_test, y_pred_original)
print("Accuracy on original data:", accuracy_original)
```

```
Accuracy on original data: 1.0
```

In [ ]:
```
# Step 11: Train a Model on PCA-Transformed Data
model_pca = LogisticRegression()
model_pca.fit(X_train1, y_train1)

# Step 12: Evaluate Model Performance on PCA-Transformed Data
y_pred_pca = model_pca.predict(X_test1)
accuracy_pca = accuracy_score(y_test1, y_pred_pca)
print("Accuracy after PCA:", accuracy_pca)
```

```
Accuracy after PCA: 0.9
```