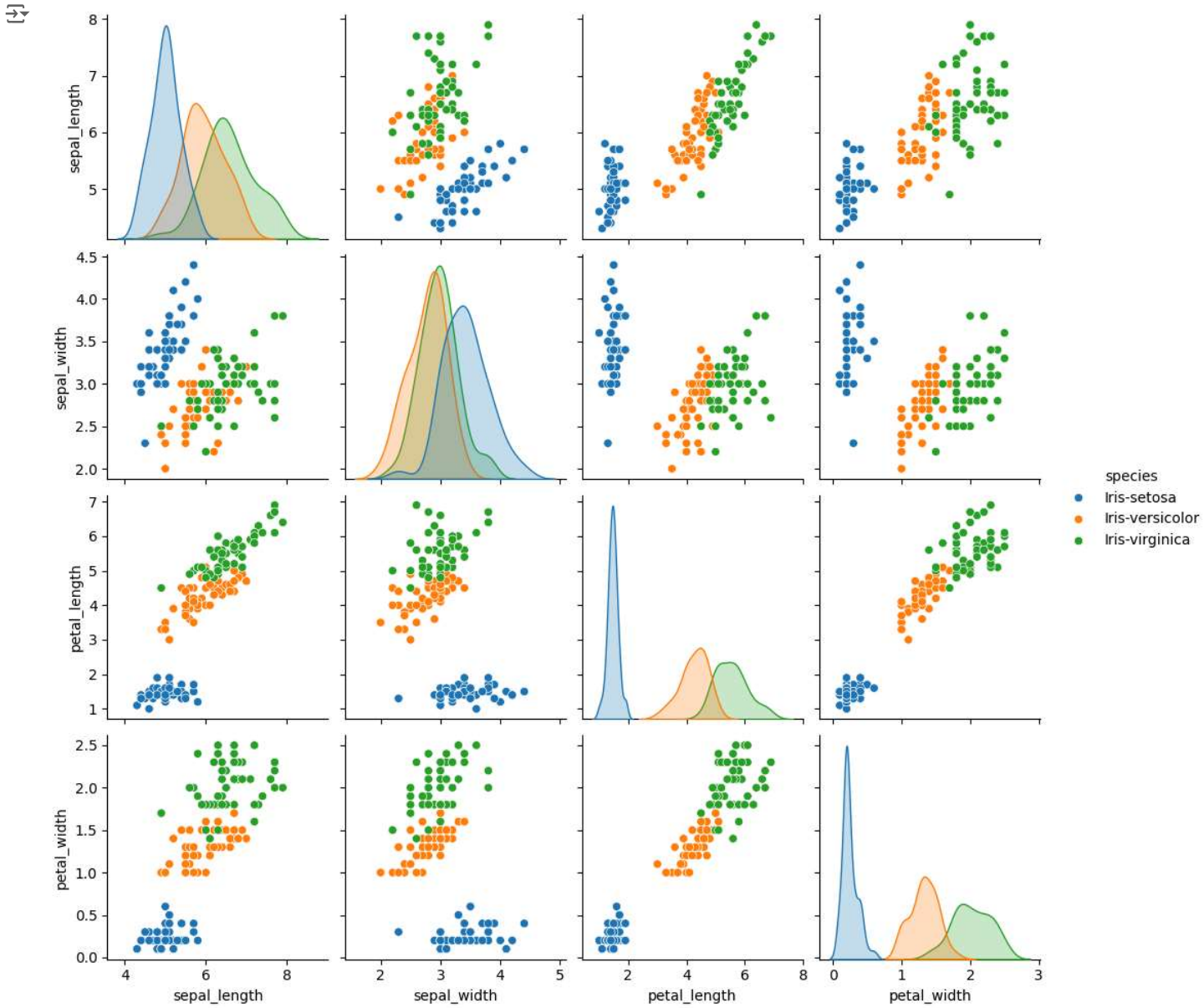
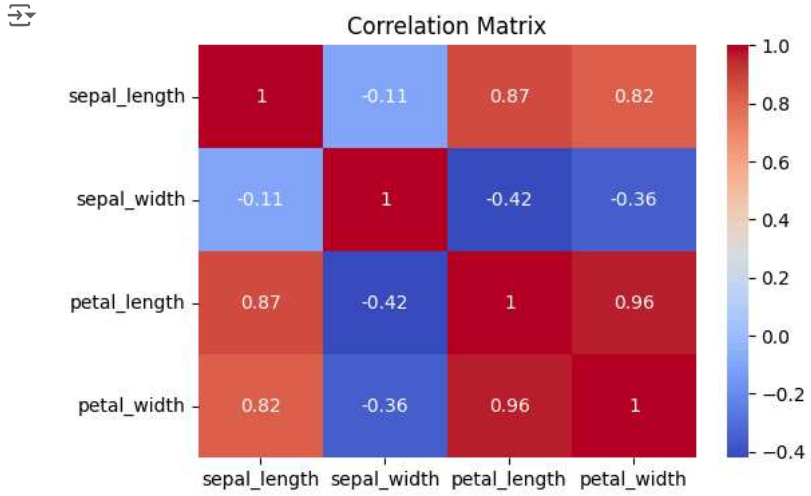
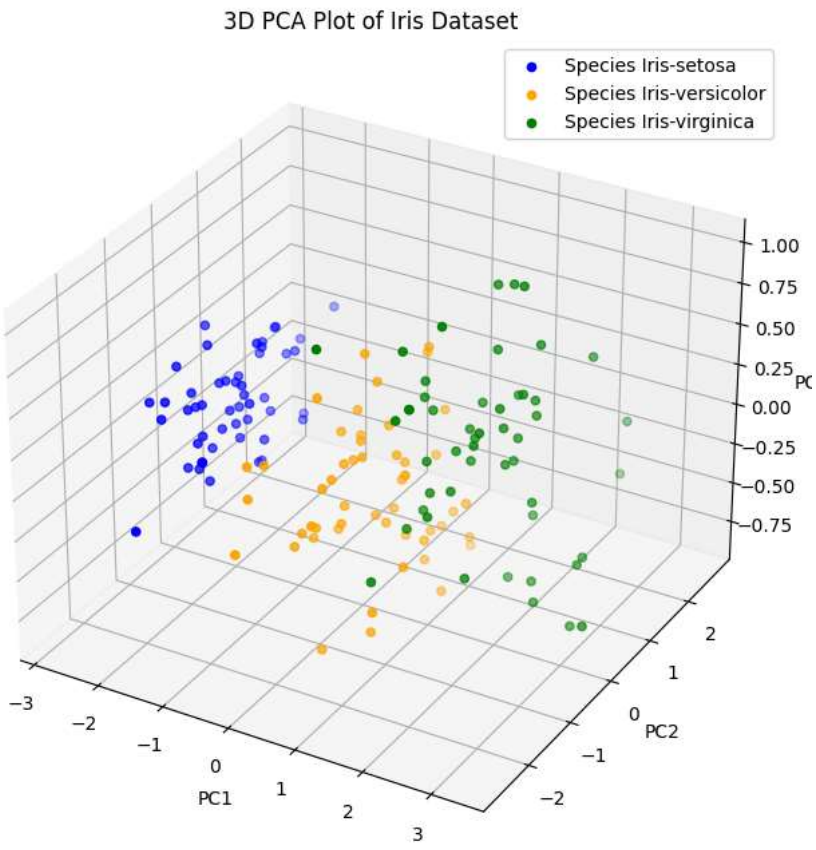
```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d


df = pd.read_csv("IRIS.csv")
# Define features and target variable
X = df.drop(columns=['species'])
y = df['species']

#Visualizing pairplot of the features of the data
sns.pairplot(df, hue='species')
plt.show()
```



```
#Visualizing the correlation of the features
df_numeric = df.drop(columns=['species'])
correlation_matrix = df_numeric.corr()
plt.figure(figsize=(6,4))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



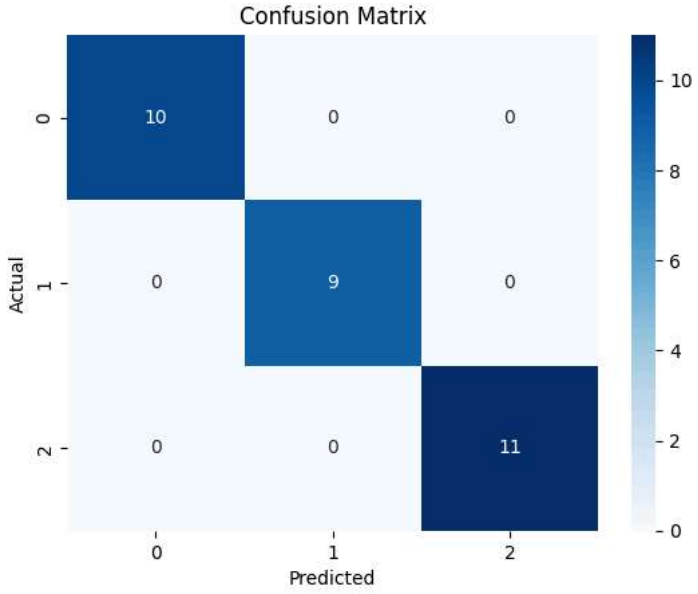

```
# Standardize the dataset
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA to reduce to 3 principal components
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)


pca_df = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2', 'PC3'])
pca_df['species'] = y

# Plotting
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot
colors = ['blue', 'orange', 'green']  # Colors for each species (setosa, versicolor, virginica)
for species_id, color in zip(pca_df['species'].unique(), colors):
    species_subset = pca_df[pca_df['species'] == species_id]
    ax.scatter(species_subset['PC1'], species_subset['PC2'], species_subset['PC3'], c=color, label=f'Species {species_id}')

# Set labels and title
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('3D PCA Plot of Iris Dataset')

# Add legend
ax.legend()

# Show plot
plt.show()
```




```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train Logistic Regression on the PCA-reduced data
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Make predictions
y_pred = logreg.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}\n")
print("Confusion Matrix:")
print(conf_matrix)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
print()
print("Classification Report:")
print(class_report)
```

Accuracy: 1.00

Confusion Matrix:
[[10 0 0]
 [0 9 0]
 [0 0 11]]



Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	1.00	1.00	1.00	9
Iris-virginica	1.00	1.00	1.00	11