Assignment No. 5

#Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.decomposition import PCA
from sklearn.ensemble import IsolationForest
from sklearn.svm import OneClassSVM
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split

# Step 1: Dataset Loading and Preprocessing
df = pd.read_csv("kddcup.data_10_percent_corrected", header=None)
categorical_features = [1, 2, 3]
numerical_features = list(set(df.columns) - set(categorical_features)- {41})

# Encoding categorical features
encoder = OneHotEncoder()
categorical_encoded = encoder.fit_transform(df[categorical_features]).toarray()
df[41] = df[41].apply(lambda x: 1 if x.strip() == 'normal.' else 0)

# Normalizing numerical features
scaler = StandardScaler()
numerical_scaled = scaler.fit_transform(df[numerical_features])

# Combining processed features
X = np.hstack((numerical_scaled, categorical_encoded))

# Step 2: Dimensionality Reduction
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], alpha=0.5)
plt.title("PCA Visualization of Network Traffic")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()

# Step 3: Model Development
model = IsolationForest(contamination=0.1, random_state=42)
model.fit(X)
y_pred = model.predict(X)
```

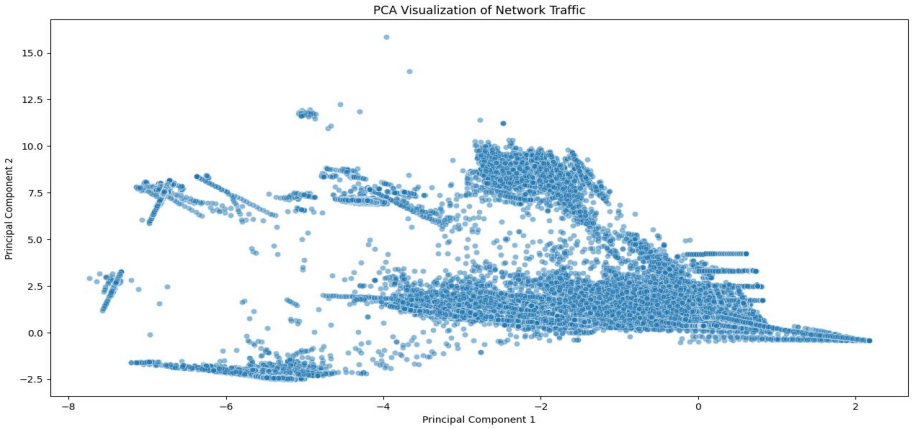
```
y_pred = np.where(y_pred == 1, 0, 1)  # Convert to anomaly labels (1: anomaly, 0: normal)

# Step 4: Evaluation
y_true = np.random.choice([0, 1], size=len(y_pred))  # Placeholder for true labels, replace with actual
print("Classification Report:")
print(classification_report(y_true, y_pred))

print("Confusion Matrix:")
conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Step 5: Visualization - Anomalies in Reduced Space
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=y_pred, palette=["blue", "red"], alpha=0.5)
plt.title("PCA Anomaly Visualization")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title='Anomaly', labels=['Normal', 'Anomalous'])
plt.show()
```

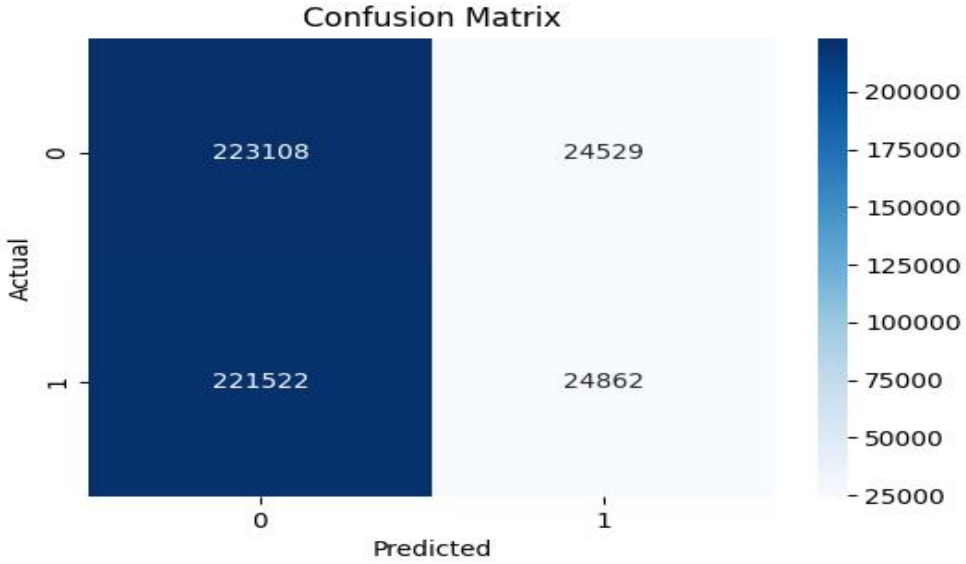
#Output

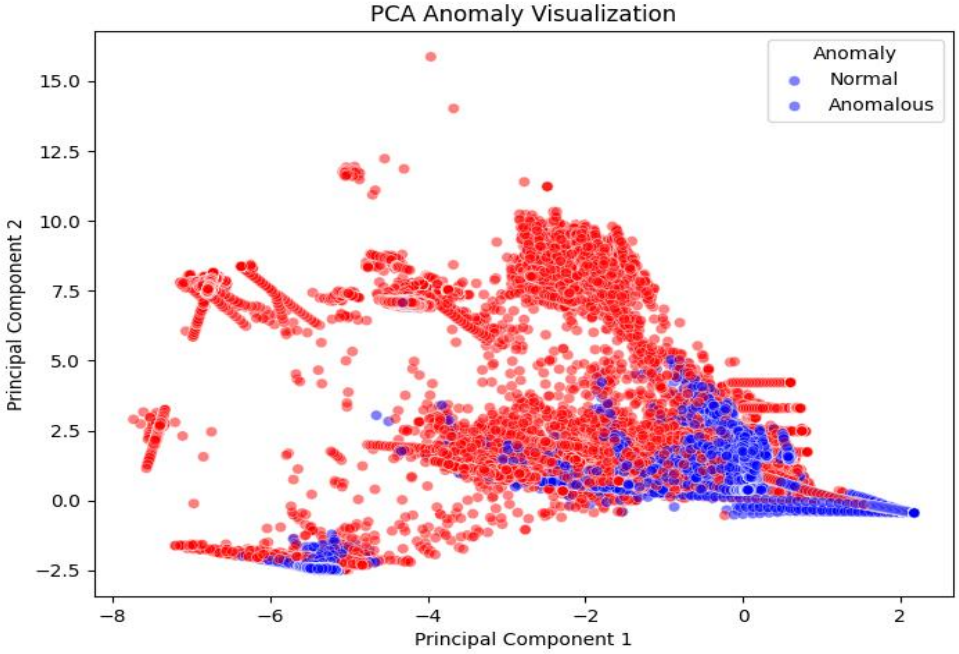
Classification Report:

	precision	recall	f1-score	support
0	0.50	0.90	0.64	247637
1	0.50	0.10	0.17	246384
accuracy			0.50	494021
macro avg	0.50	0.50	0.41	494021
weighted avg	0.50	0.50	0.41	494021

Confusion Matrix:


PCA Visualization of Network Traffic
Principal Component 2
Principal Component 1


Confusion Matrix
223108
24529
221522
24862
Actual
Predicted
0
1
200000
175000
150000
125000
100000
75000
50000
25000


PCA Anomaly Visualization
Anomaly
Normal
Anomalous
Principal Component 2
Principal Component 1