SVM_PCA_2d

July 11, 2024

[1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪ConfusionMatrixDisplay, accuracy_score, precision_score, recall_score,␣
 ↪f1_score, classification_report, roc_curve, auc, RocCurveDisplay
import matplotlib.pyplot as plt
from sklearn.utils import resample
import matplotlib.colors
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.decomposition import PCA
```

[2]:
```
%%time
df_main = pd.read_csv('../../Dataset/IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)/
 ↪MainDataset/dataset_main.csv')
```

```
CPU times: total: 1min 8s
Wall time: 1min 53s
```

[3]:
```
len(df_main)
```

[3]: 11916113

[4]:
```
len(df_main[df_main['Label'] == 0])
```

[4]: 10564771

[5]:
```
len(df_main[df_main['Label'] == 1])
```

[5]: 1351342

[6]:
```
df_normal = df_main[df_main['Label'] == 0]
df_attack = df_main[df_main['Label'] == 1]
```

[7]:
```
df_normal_downsampled = resample(df_normal, replace=False, n_samples=25000,␣
 ↪random_state=42)
len(df_normal_downsampled)
```

[7]: 25000

[8]:
```
df_attack_downsampled = resample(df_attack, replace=False, n_samples=25000,␣
↪random_state=42)
len(df_attack_downsampled)
```

[8]: 25000

[9]:
```
df_downsample = pd.concat([df_normal_downsampled, df_attack_downsampled])
len(df_downsample)
```

[9]: 50000

Split data

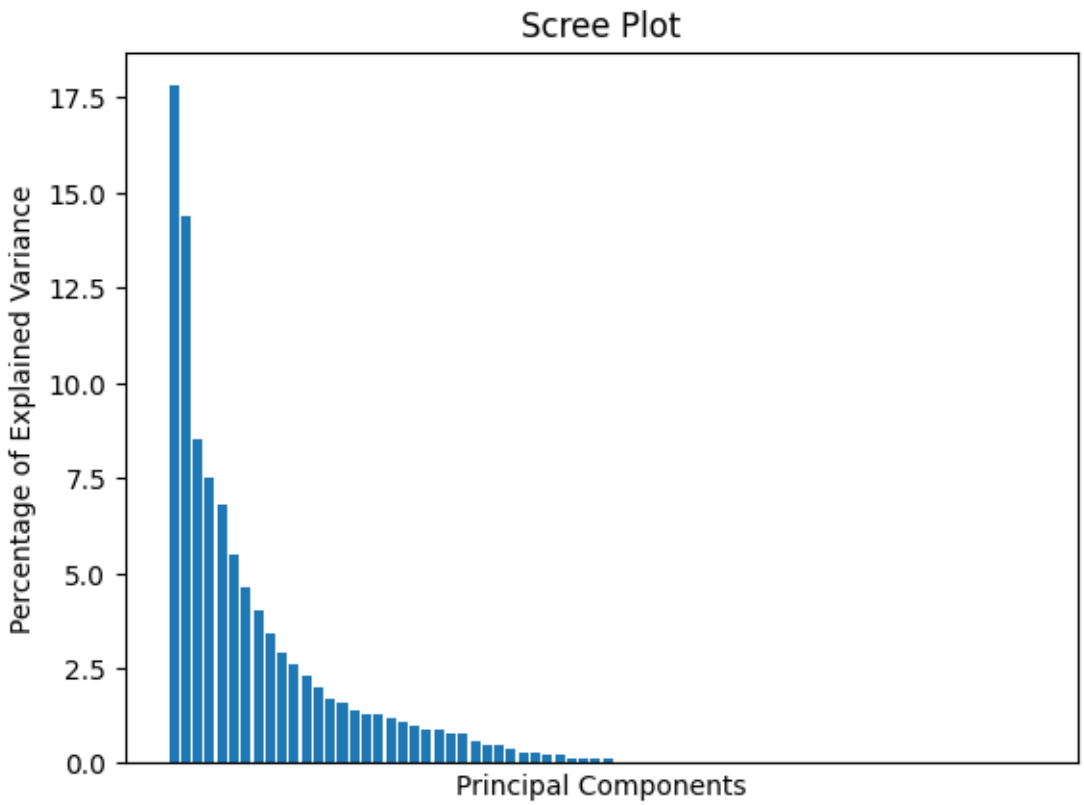
[10]:
```
X = df_downsample.drop(columns='Label')
y = df_downsample['Label']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

Chuẩn hóa dữ liệu

[11]:
```
scaler = StandardScaler()
X_train_scaled  = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Vẽ biểu đồ Scree Plot

[12]:
```
pca = PCA()
X_train_pca = pca.fit_transform(X_train_scaled)

per_var = np.round(pca.explained_variance_ratio_*100, decimals=1)
labels = [str(x) for x in range(1, len(per_var)+1)]

plt.bar(x=range(1, len(per_var)+1), height=per_var)
plt.tick_params(
    axis='x',
    which='both',
    bottom=False,
    top=False,
    labelbottom=False)

plt.ylabel('Percentage of Explained Variance')
plt.xlabel('Principal Components')
plt.title('Scree Plot')
plt.show()
```



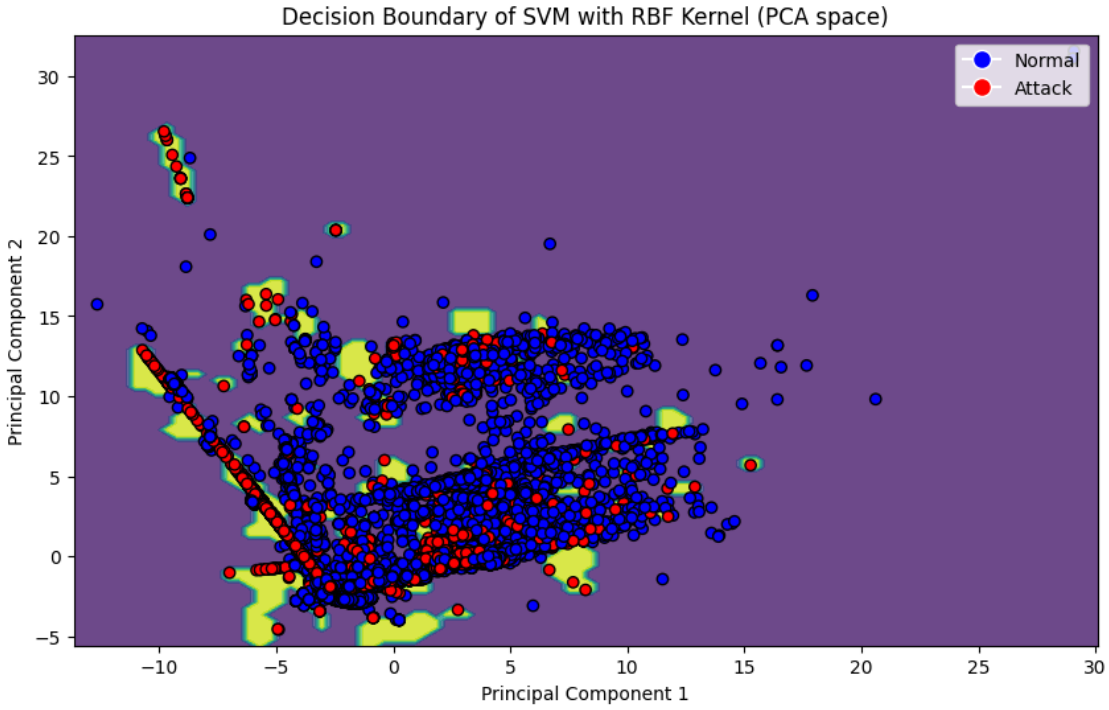
Hyperparameter tuning với K-fold cross-validation

[13]:
```
# %%time
# pca = PCA(n_components=2)
# X_train_pca = pca.fit_transform(X_train_scaled)

# train_pc1_coords = X_train_pca[:, 0]
# train_pc2_coords = X_train_pca[:, 1]
# pca_train_scaled = StandardScaler().fit_transform(np.
 ↪column_stack((train_pc1_coords, train_pc2_coords)))

# param_grid = [
#     {'C': [1, 10, 100, 1000],
#      'gamma': ['scale', 1, 0.1, 0.01, 0.001, 0.0001],
#      'kernel': ['rbf']},
# ]

# optimal_params = GridSearchCV(
#     SVC(),
#     param_grid,
#     cv=5,
#     scoring='accuracy',
#     verbose=3
```

```
# )

# optimal_params.fit(pca_train_scaled, y_train)
# print(optimal_params.best_params_)
```

Huấn luyện mô hình và visualization

[14]:
```
%%time

pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
# Huấn luyện mô hình SVM với kernel RBF trên dữ liệu PCA
# clf_svm_pca = SVC(random_state=42, **optimal_params.best_params_)
clf_svm_pca = SVC(random_state=42, C=1000, gamma=1, kernel='rbf')
clf_svm_pca.fit(X_train_pca, y_train)

# Hiển thị ranh giới quyết định
fig, ax = plt.subplots(figsize=(10, 6))

cmap = matplotlib.colors.ListedColormap(['blue', 'red'])

# DecisionBoundaryDisplay từ sklearn.inspection
DecisionBoundaryDisplay.from_estimator(clf_svm_pca, X_train_pca,␣
 ↪response_method="predict", alpha=0.8, cmap='viridis', ax=ax)

# Vẽ scatter plot của dữ liệu PCA
scatter = ax.scatter(X_train_pca[:, 0], X_train_pca[:, 1], c=y_train,␣
 ↪cmap=cmap, edgecolors='k')

# Thêm các thông số cho biểu đồ
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_title('Decision Boundary of SVM with RBF Kernel (PCA space)')

# Tạo chú thích tùy chỉnh
handles = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='blue',␣
 ↪markersize=10, label='Normal'),
           plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='red',␣
 ↪markersize=10, label='Attack')]

ax.legend(handles=handles, loc='upper right')

plt.show()
```




```
CPU times: total: 2min 21s
Wall time: 2min 31s
```

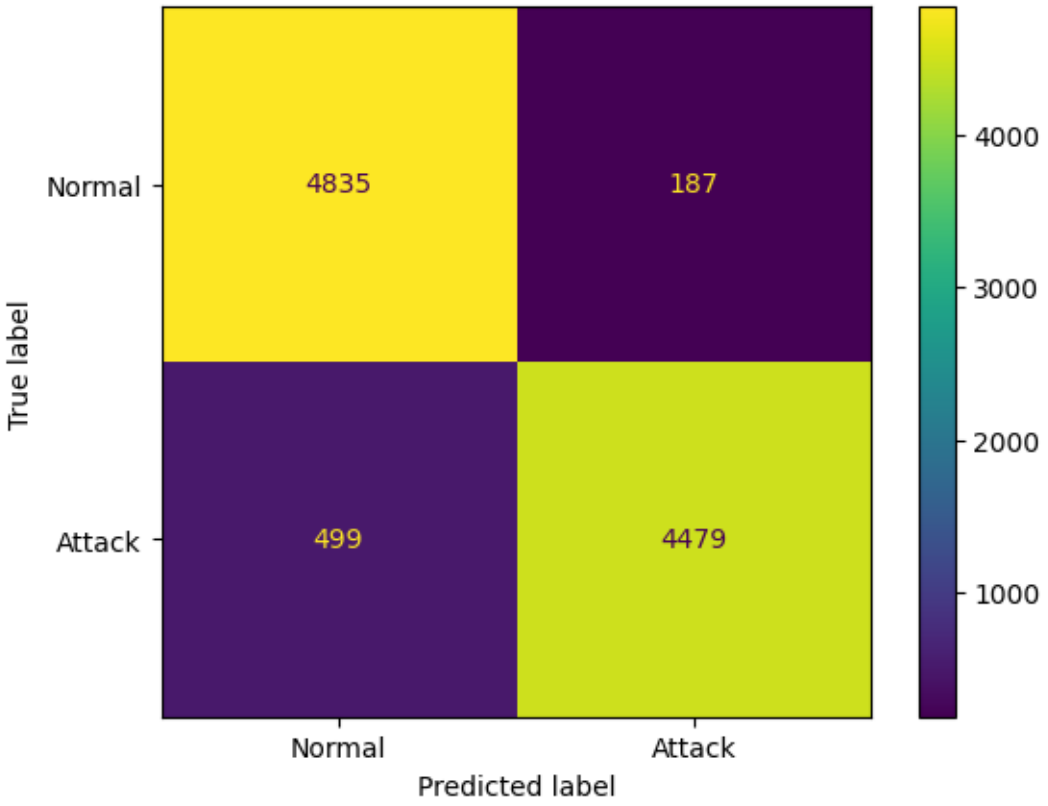
Đánh giá mô hình

[15]:
```
X_test_pca = pca.transform(X_test_scaled)

# Dự đoán nhãn của tập kiểm tra
y_pred = clf_svm_pca.predict(X_test_pca)

# Đánh giá độ chính xác
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
```

```
Accuracy: 0.9314
Precision: 0.9599228461208744
Recall: 0.8997589393330655
F1-Score: 0.9288676897552882
```

[16]:
```
# Tính ma trận nhầm lẫn
cm = confusion_matrix(y_test, clf_svm_pca.predict(X_test_pca))

# Hiển thị ma trận nhầm lẫn
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Normal",␣
 ↪"Attack"])
disp.plot(values_format='d')
```

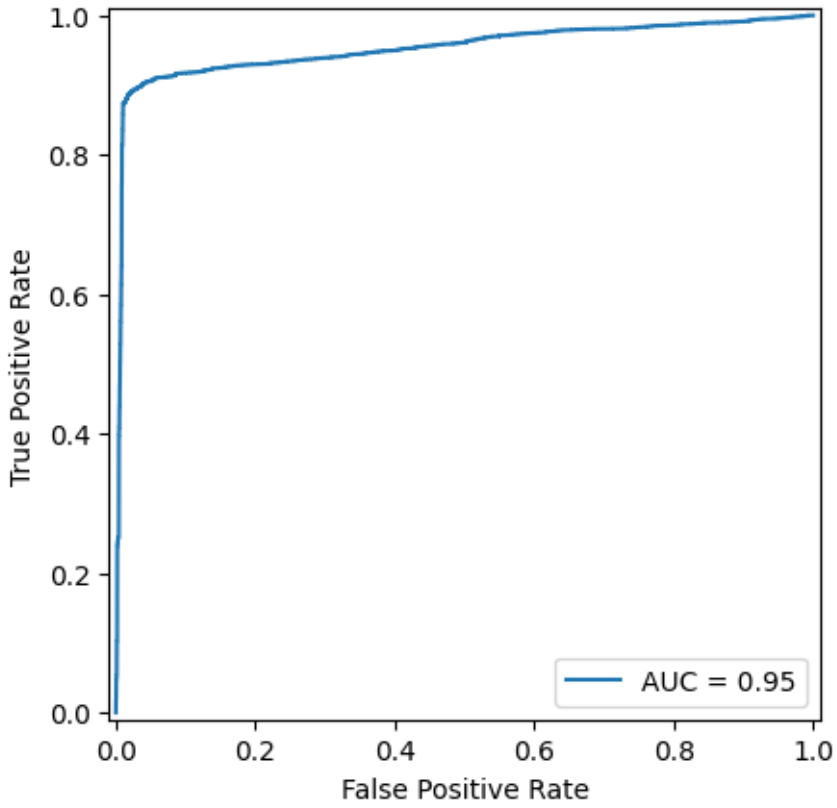
[16]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x236d6243920>



[17]:
```
y_score = clf_svm_pca.decision_function(X_test_pca)
fpr, tpr, _ = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)
print(f'ROC AUC: {roc_auc}')
RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc).plot()
```

```
ROC AUC: 0.9530902518272755
```

[17]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23560001580>



[18]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.96      0.93      5022
           1       0.96      0.90      0.93      4978

    accuracy                           0.93     10000
   macro avg       0.93      0.93      0.93     10000
weighted avg       0.93      0.93      0.93     10000
```