In [1]:
```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

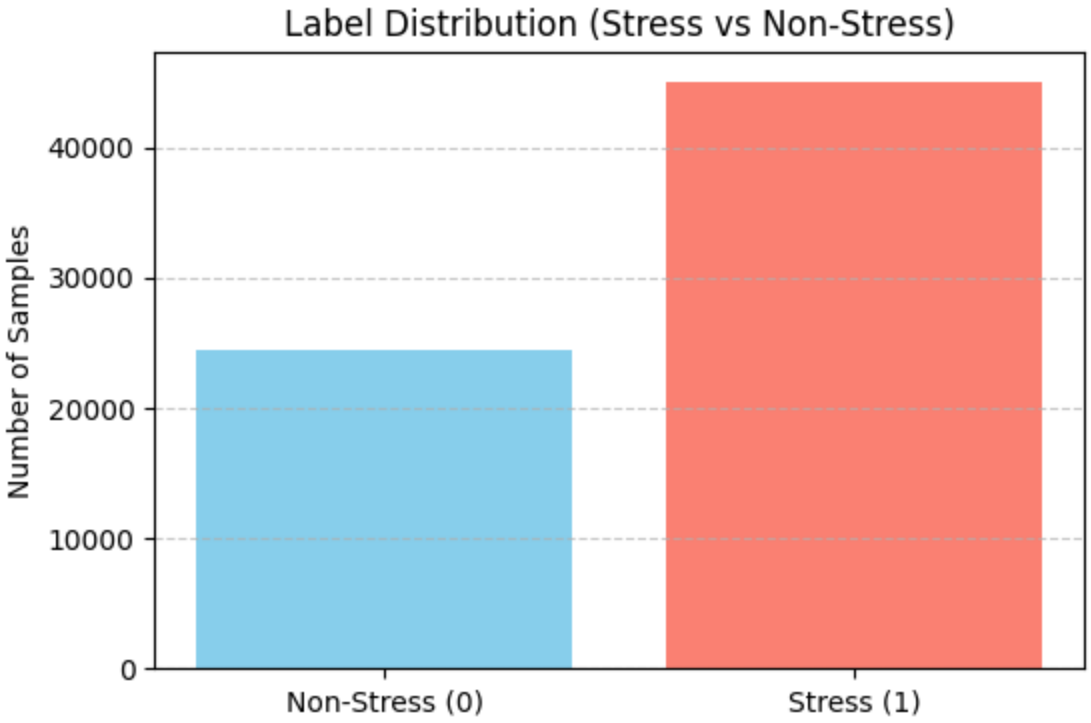
In [34]:
```
# Import libraries
import os
import pickle
import numpy as np
from scipy.signal import resample
import pandas as pd

from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc
from scipy.stats import ttest_rel
import seaborn as sns

import matplotlib.pyplot as plt
```

In [5]:
```
# Load the preprocessed train/test splits
save_dir = "/content/drive/MyDrive/Colab Notebooks/Purdue Coursework/CS573_G

with open(os.path.join(save_dir, 'X_train.pkl'), 'rb') as f:
    X_train = pickle.load(f)

with open(os.path.join(save_dir, 'X_test.pkl'), 'rb') as f:
    X_test = pickle.load(f)

with open(os.path.join(save_dir, 'y_train.pkl'), 'rb') as f:
    y_train = pickle.load(f)

with open(os.path.join(save_dir, 'y_test.pkl'), 'rb') as f:
    y_test = pickle.load(f)

print("Loaded preprocessed data successfully.")
```

```
Loaded preprocessed data successfully.
```

In [24]:
```
# Exploratory Data Analysis(EDA)

# Merge train and test sets for EDA
X_all = np.concatenate((X_train, X_test), axis=0)
y_all = np.concatenate((y_train, y_test), axis=0)

# Stress (1) vs Non-stress (0) counts
unique, counts = np.unique(y_all, return_counts=True)
label_counts = dict(zip(unique, counts))

# Plot
plt.figure(figsize=(6,4))
plt.bar(['Non-Stress (0)', 'Stress (1)'], counts, color=['skyblue', 'salmon'
plt.title('Label Distribution (Stress vs Non-Stress)')
```

```
plt.ylabel('Number of Samples')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



In [27]:
```
# EDA 2: Main feature statistics (Mean, Std)
n_samples, features = X_all.shape

# For each label, calculate mean and std
df = pd.DataFrame(X_all)
df['label'] = y_all

# Group by label (0=Non-stress, 1=Stress)
feature_stats = df.groupby('label').agg(['mean', 'std'])

# Show a few sample statistics
feature_stats.iloc[:, :10]  # First 10 features for display
```

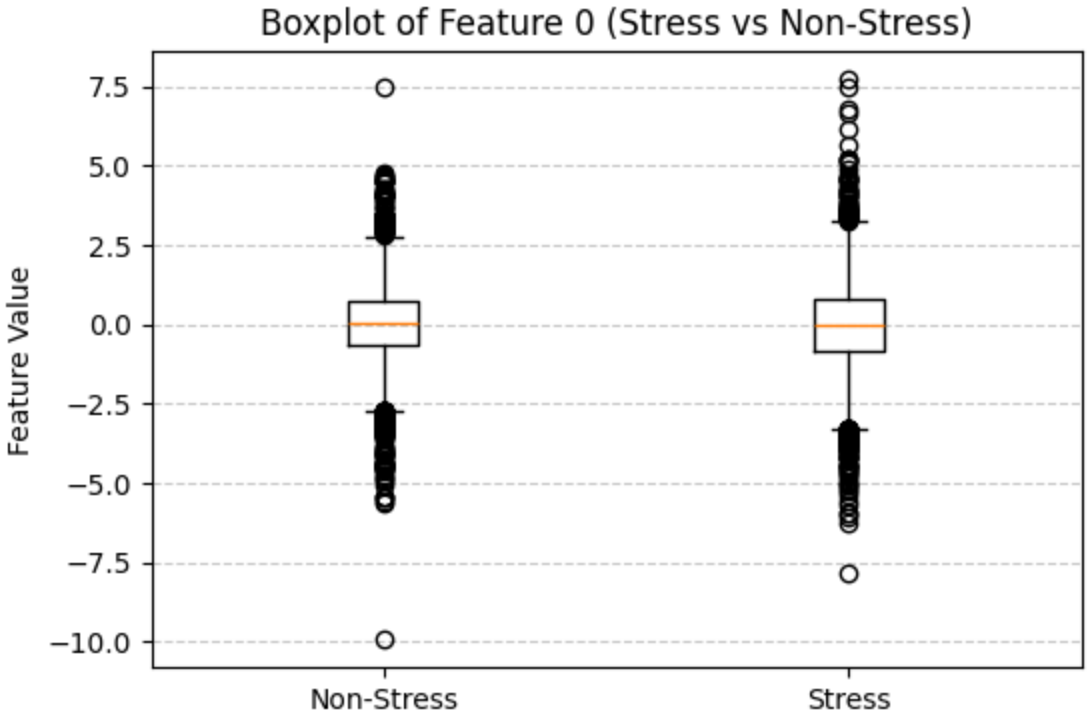
Out[27]:

	0		1		2		
	mean	std	mean	std	mean	std	mean
label							
0	0.006531	0.967959	-0.036712	0.982780	-0.017996	1.031685	-0.087334	1.01
1	-0.000310	1.017912	0.018277	1.008746	0.009898	0.985124	0.048062	0.98

In [26]:
```
# EDA 3: Boxplot for a selected feature
# Example: Pick feature 0 (could be a BVP or EDA-related feature)
feature_idx = 0  # Change this if needed
```

```
plt.figure(figsize=(6,4))
plt.boxplot([df[df['label'] == 0][feature_idx], df[df['label'] == 1][feature
plt.title(f'Boxplot of Feature {feature_idx} (Stress vs Non-Stress)')
plt.ylabel('Feature Value')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

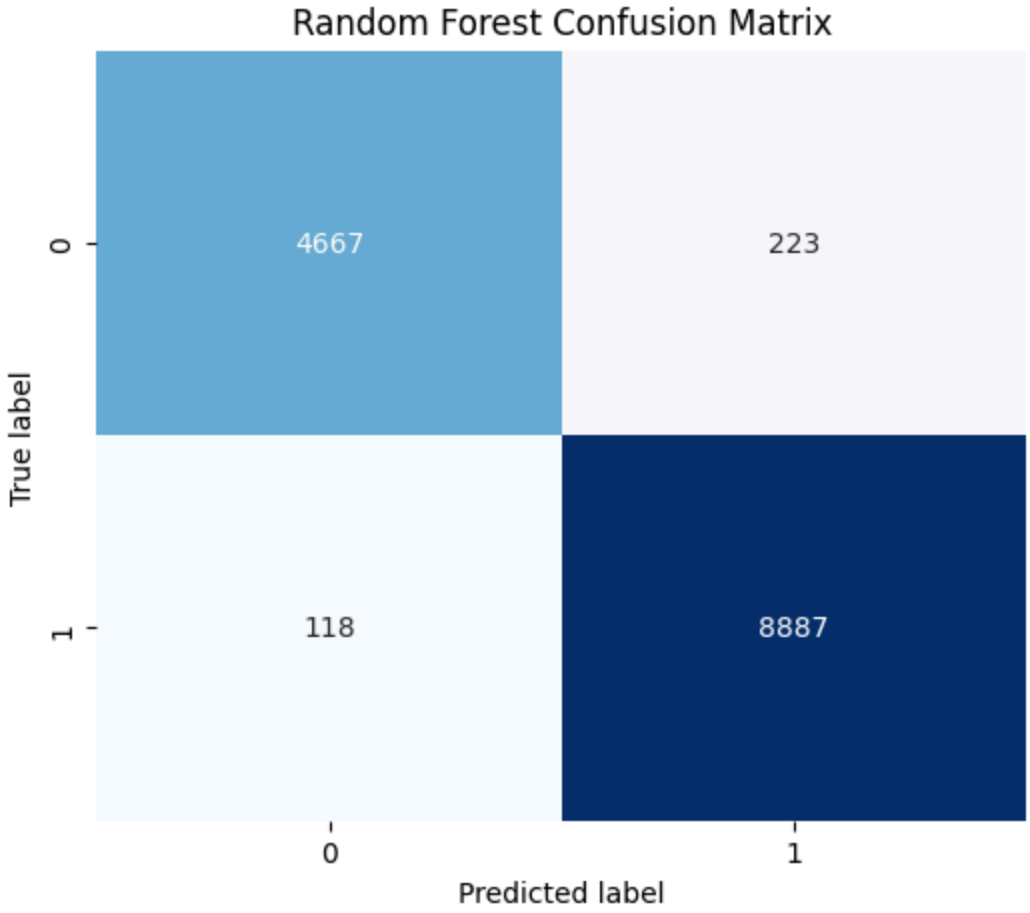
```
<ipython-input-26-5d356bf06b80>:6: MatplotlibDeprecationWarning: The 'label
s' parameter of boxplot() has been renamed 'tick_labels' since Matplotlib 3.
9; support for the old name will be dropped in 3.11.
  plt.boxplot([df[df['label'] == 0][feature_idx], df[df['label'] == 1][featu
re_idx]], labels=['Non-Stress', 'Stress'])
```



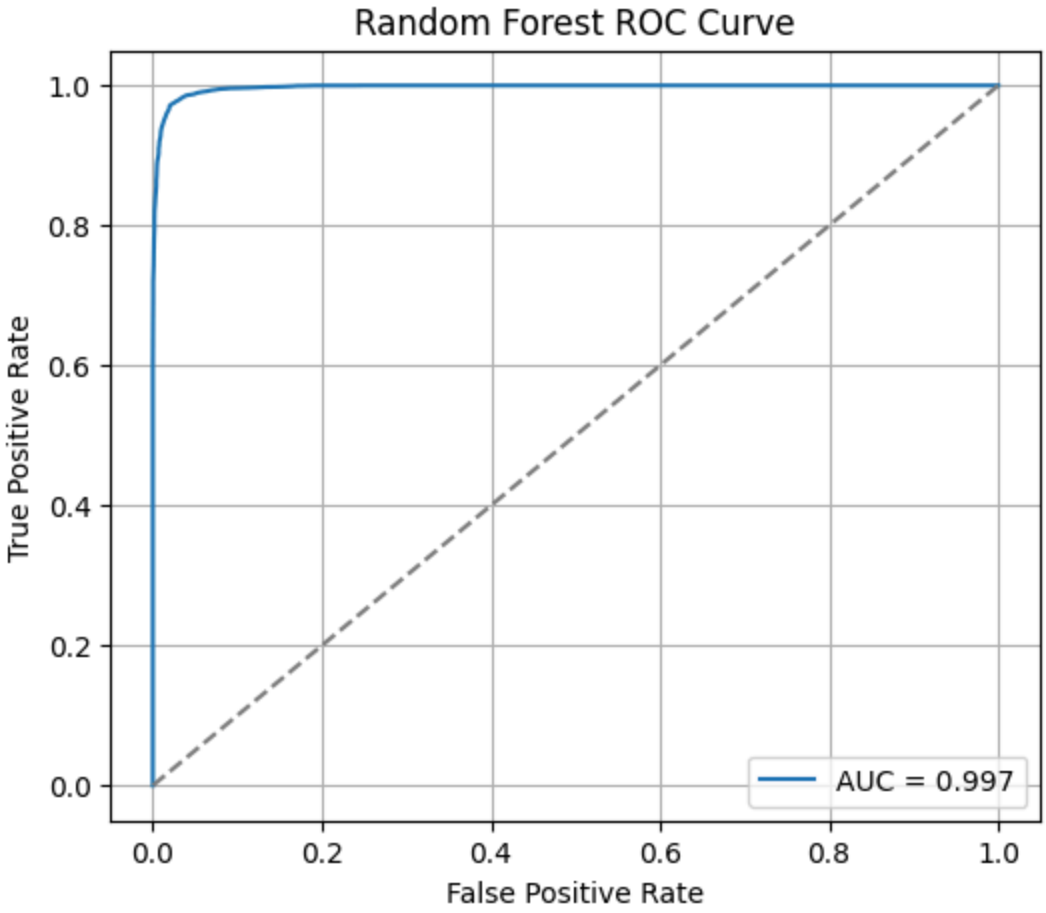
In [6]:
```
# Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)

print("\n=== Random Forest Evaluation ===")
print(classification_report(y_test, rf_pred, digits=4))
print("ROC-AUC:", roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1]))
print("Confusion Matrix:\n", confusion_matrix(y_test, rf_pred))
```

```
=== Random Forest Evaluation ===
              precision    recall  f1-score   support

           0     0.9753    0.9544    0.9648      4890
           1     0.9755    0.9869    0.9812      9005

    accuracy                         0.9755     13895
   macro avg     0.9754    0.9706    0.9730     13895
weighted avg     0.9755    0.9755    0.9754     13895

ROC-AUC: 0.9970772088671483
Confusion Matrix:
 [[4667  223]
 [ 118 8887]]
```

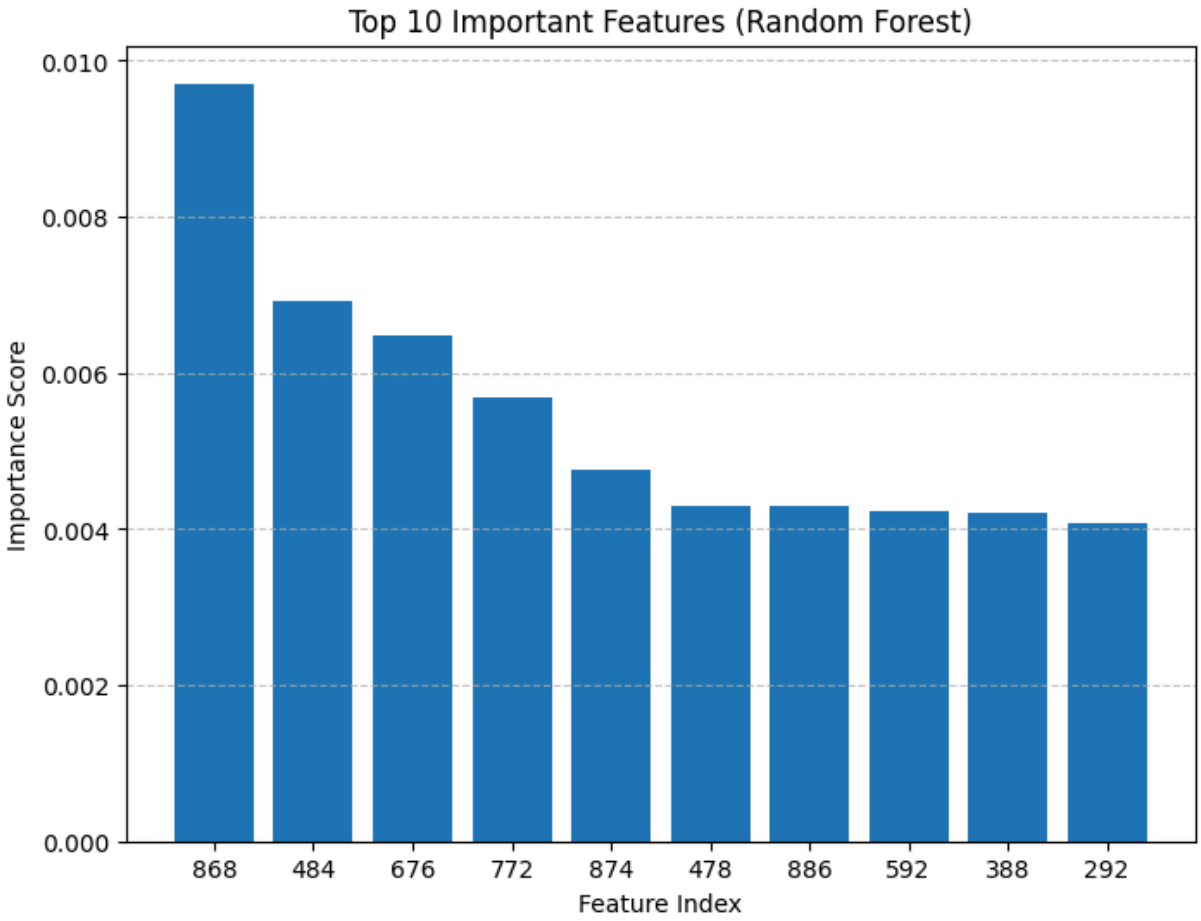
In [9]:
```python
# Predict on test set
rf_pred = rf.predict(X_test)

# Compute confusion matrix
rf_cm = confusion_matrix(y_test, rf_pred)

# Plot confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(rf_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```



In [20]:
```
# Predict probabilities
rf_probs = rf.predict_proba(X_test)[:, 1]

# Compute ROC curve and AUC
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs)
rf_auc = roc_auc_score(y_test, rf_probs)

# Plot ROC curve
plt.figure(figsize=(6, 5))
plt.plot(rf_fpr, rf_tpr, label=f'AUC = {rf_auc:.3f}')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.title('Random Forest ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



In [29]:
```
# Feature Importance Analysis
importances_rf = rf.feature_importances_
# Top 10 important features (Random Forest)
indices_rf = np.argsort(importances_rf)[-10:][::-1]

plt.figure(figsize=(8,6))
plt.bar(range(len(indices_rf)), importances_rf[indices_rf], align='center')
plt.xticks(range(len(indices_rf)), indices_rf)
plt.xlabel('Feature Index')
plt.ylabel('Importance Score')
plt.title('Top 10 Important Features (Random Forest)')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



In [14]:
```
# 5-fold Cross-validation (Random Forest)
rf_scores = cross_val_score(rf, X_train, y_train, cv=5, scoring='accuracy')
print("=== Random Forest 5-Fold Cross-Validation ===")
print(f"Fold Accuracies: {np.round(rf_scores, 4)}")
print(f"Mean Accuracy: {np.round(np.mean(rf_scores), 4)}")
print(f"Std Dev: {np.round(np.std(rf_scores), 4)}\n")
```

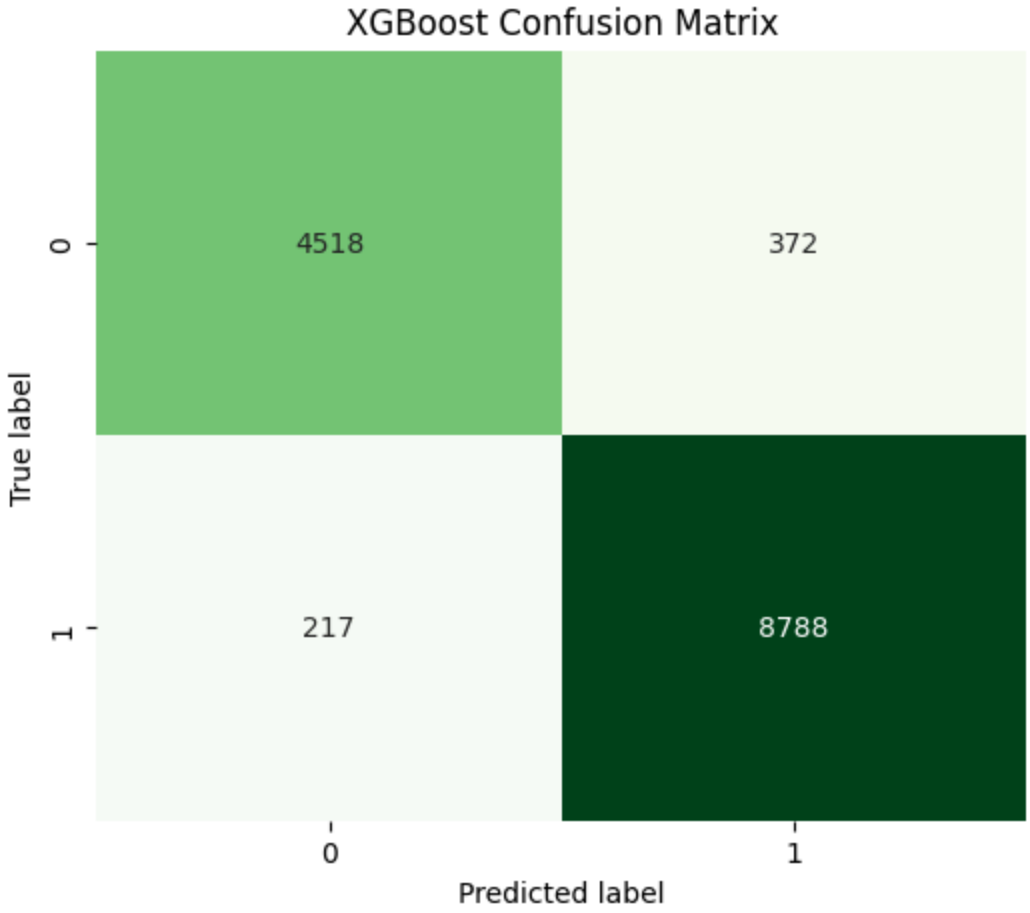
```
=== Random Forest 5-Fold Cross-Validation ===
Fold Accuracies: [0.9711 0.9711 0.9735 0.9735 0.9727]
Mean Accuracy: 0.9724
Std Dev: 0.0011
```

In [7]:
```
# XGBoost
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb.fit(X_train, y_train)
xgb_pred = xgb.predict(X_test)

print("\n=== XGBoost Evaluation ===")
print(classification_report(y_test, xgb_pred, digits=4))
print("ROC-AUC:", roc_auc_score(y_test, xgb.predict_proba(X_test)[:, 1]))
print("Confusion Matrix:\n", confusion_matrix(y_test, xgb_pred))
```

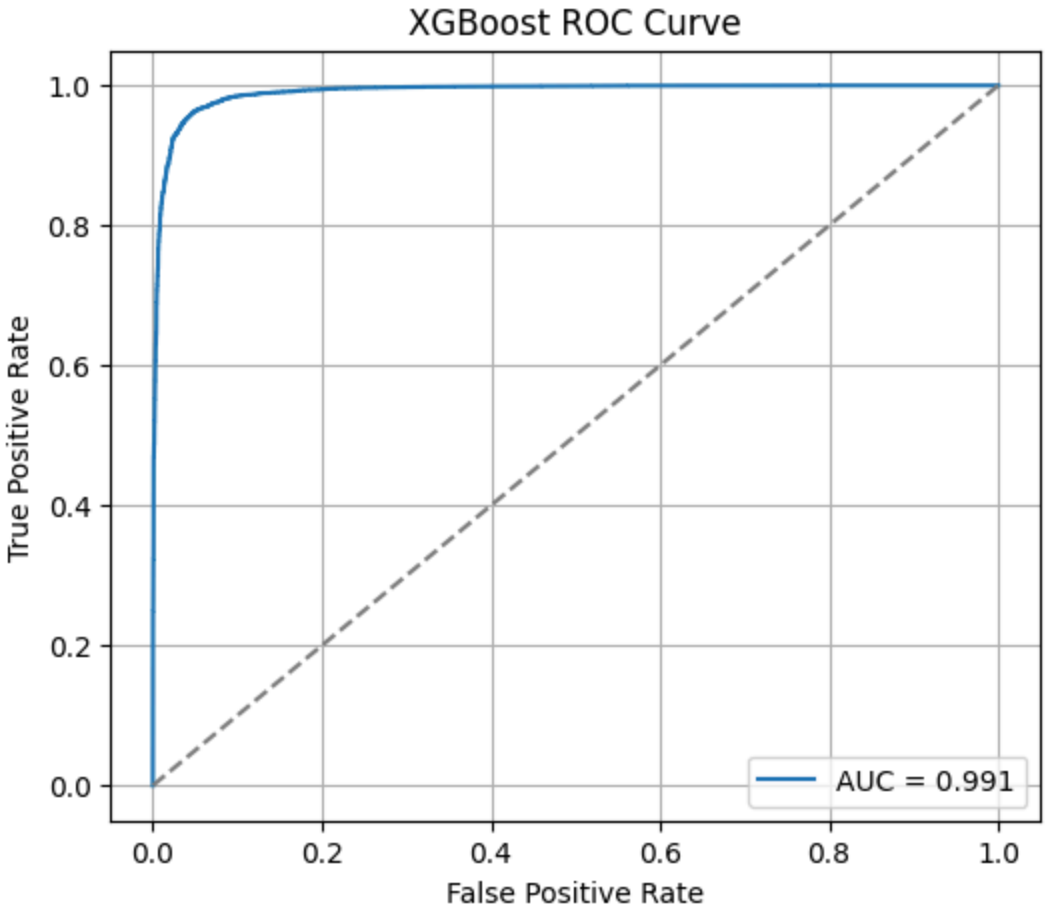
```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
5:10:46] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

```
=== XGBoost Evaluation ===
              precision    recall  f1-score   support

           0     0.9542    0.9239    0.9388      4890
           1     0.9594    0.9759    0.9676      9005

    accuracy                         0.9576     13895
   macro avg     0.9568    0.9499    0.9532     13895
weighted avg     0.9576    0.9576    0.9575     13895

ROC-AUC: 0.9905294945207672
Confusion Matrix:
 [[4518  372]
 [ 217 8788]]
```

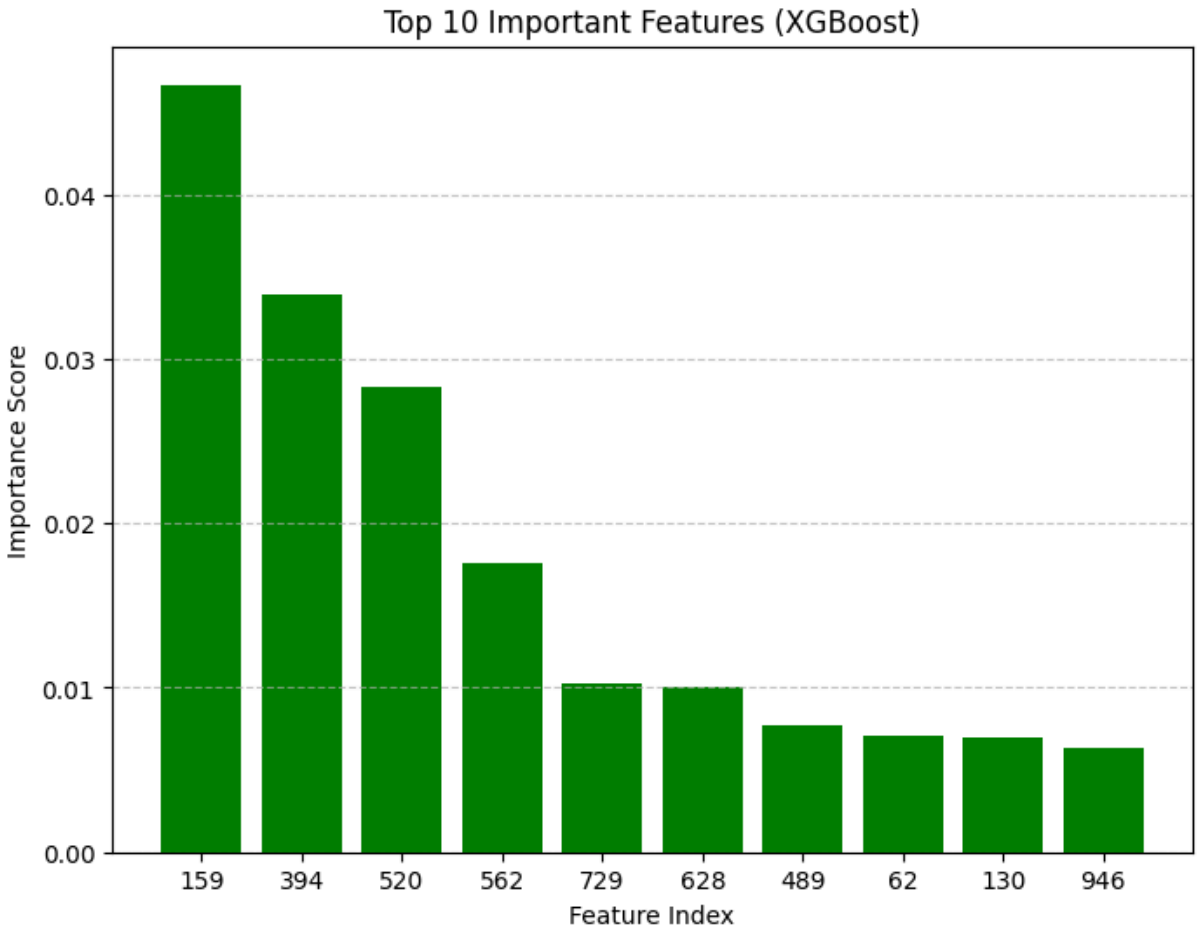
In [10]:
```
# Predict on test set
xgb_pred = xgb.predict(X_test)

# Compute confusion matrix
xgb_cm = confusion_matrix(y_test, xgb_pred)

# Plot confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(xgb_cm, annot=True, fmt='d', cmap='Greens', cbar=False)
plt.title('XGBoost Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```



In [21]:
```
# Predict probabilities
xgb_probs = xgb.predict_proba(X_test)[:, 1]

# Compute ROC curve and AUC
xgb_fpr, xgb_tpr, _ = roc_curve(y_test, xgb_probs)
xgb_auc = roc_auc_score(y_test, xgb_probs)

# Plot ROC curve
plt.figure(figsize=(6, 5))
plt.plot(xgb_fpr, xgb_tpr, label=f'AUC = {xgb_auc:.3f}')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.title('XGBoost ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



In [30]:
```
# Feature Importance Analysis
importances_xgb = xgb.feature_importances_
# Top 10 important features (XGBoost)
indices_xgb = np.argsort(importances_xgb)[-10:][::-1]

plt.figure(figsize=(8,6))
plt.bar(range(len(indices_xgb)), importances_xgb[indices_xgb], align='center
plt.xticks(range(len(indices_xgb)), indices_xgb)
plt.xlabel('Feature Index')
plt.ylabel('Importance Score')
plt.title('Top 10 Important Features (XGBoost)')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



In [15]:
```
xgb_scores = cross_val_score(xgb, X_train, y_train, cv=5, scoring='accuracy'
print("=== XGBoost 5-Fold Cross-Validation ===")
print(f"Fold Accuracies: {np.round(xgb_scores, 4)}")
print(f"Mean Accuracy: {np.round(np.mean(xgb_scores), 4)}")
print(f"Std Dev: {np.round(np.std(xgb_scores), 4)}")
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
6:07:44] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
6:08:39] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
6:09:33] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
6:10:27] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
6:11:21] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
=== XGBoost 5-Fold Cross-Validation ===
Fold Accuracies: [0.9534 0.9528 0.9566 0.9539 0.9544]
Mean Accuracy: 0.9542
Std Dev: 0.0013
```

In [ ]:

In [39]:
```python
# Random Forest 5-Fold Cross-validation
#rf_cv_scores = cross_val_score(rf, X_flat, y_all, cv=5, scoring='accuracy')

# XGBoost 5-Fold Cross-validation
#xgb_cv_scores = cross_val_score(xgb, X_flat, y_all, cv=5, scoring='accuracy

# 5-Fold Cross-validation results
rf_cv_scores = np.array([0.9711, 0.9711, 0.9735, 0.9735, 0.9727])
xgb_cv_scores = np.array([0.9534, 0.9528, 0.9566, 0.9539, 0.9544])

# Paired t-test
t_stat, p_val = ttest_rel(rf_cv_scores, xgb_cv_scores)

print("=== 5-Fold CV Accuracy ===")
print(f"Random Forest mean accuracy: {np.mean(rf_cv_scores):.4f}")
print(f"XGBoost mean accuracy: {np.mean(xgb_cv_scores):.4f}\n")

print("=== Paired t-test ===")
print(f"t-statistic: {t_stat:.4f}")
print(f"p-value: {p_val:.4e}")
```

```
=== 5-Fold CV Accuracy ===
Random Forest mean accuracy: 0.9724
XGBoost mean accuracy: 0.9542

=== Paired t-test ===
t-statistic: 41.0612
p-value: 2.1024e-06
```

In [40]:
```python
summary_table = pd.DataFrame({
    "Model": ["Random Forest", "XGBoost"],
    "Mean CV Accuracy": [np.mean(rf_cv_scores), np.mean(xgb_cv_scores)],
    "Std Dev": [np.std(rf_cv_scores), np.std(xgb_cv_scores)]
})

print("\n=== Summary Table ===")
print(summary_table)
```

```
=== Summary Table ===
           Model  Mean CV Accuracy   Std Dev
0  Random Forest           0.97238  0.001085
1        XGBoost           0.95422  0.001303
```

In [45]:
```python
# EDA-only Random Forest Training and Evaluation
# Extract only EDA signal from flattened data
# In flattened data, EDA appears at every 6th feature starting at index 3
X_train_eda = X_train[:, 3::6]
X_test_eda = X_test[:, 3::6]
```

In [46]:
```python
# Random Forest model
rf_eda = RandomForestClassifier(n_estimators=100, random_state=42)
rf_eda.fit(X_train_eda, y_train)
```

Out[46]:

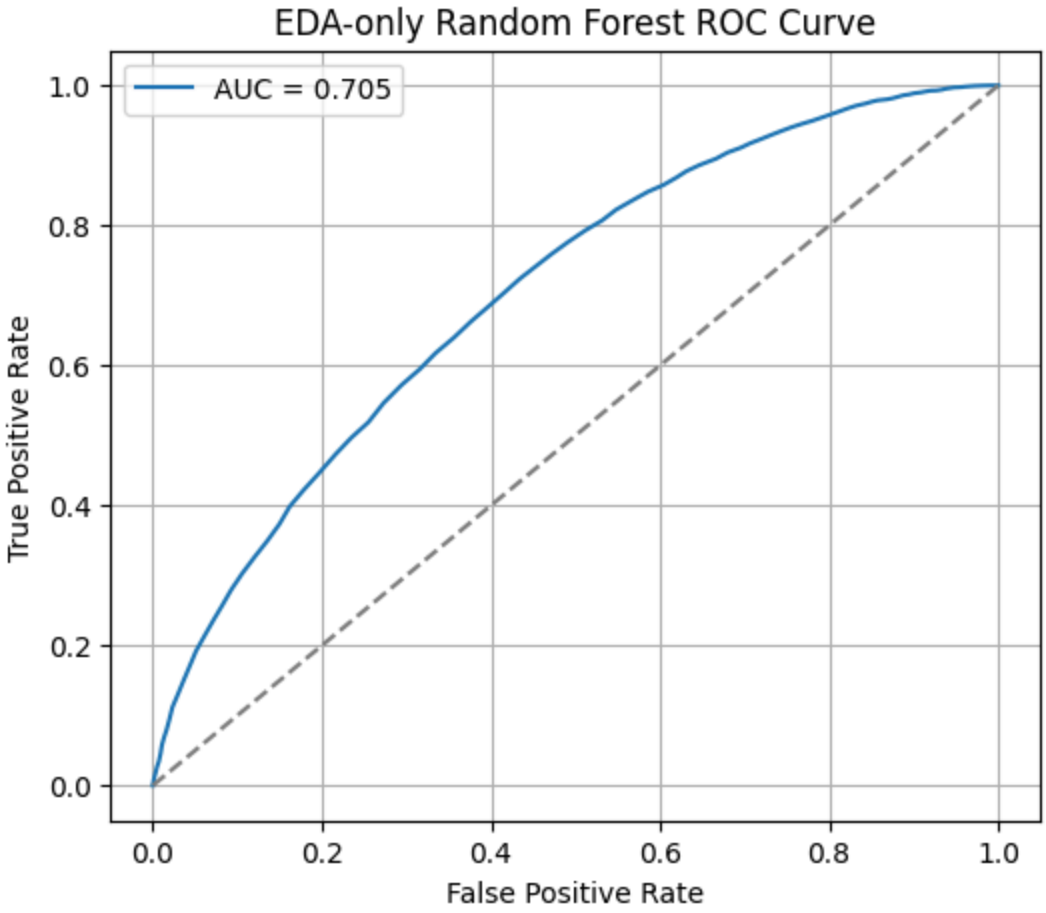
▾ RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

In [47]:
```python
# Evaluation
y_pred_eda = rf_eda.predict(X_test_eda)
probs_eda = rf_eda.predict_proba(X_test_eda)[:, 1]

print("=== EDA-only Random Forest Evaluation ===")
print(classification_report(y_test, y_pred_eda, digits=4))
print("ROC-AUC:", roc_auc_score(y_test, probs_eda))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_eda))
```

```
=== EDA-only Random Forest Evaluation ===
              precision    recall  f1-score   support

           0     0.6019    0.3961    0.4778      4890
           1     0.7234    0.8577    0.7849      9005

    accuracy                         0.6953     13895
   macro avg     0.6627    0.6269    0.6313     13895
weighted avg     0.6807    0.6953    0.6768     13895

ROC-AUC: 0.7052616644468138
Confusion Matrix:
 [[1937 2953]
 [1281 7724]]
```

In [48]:
```
# Plot ROC Curve
fpr_eda, tpr_eda, _ = roc_curve(y_test, probs_eda)
roc_auc_eda = roc_auc_score(y_test, probs_eda)

plt.figure(figsize=(6,5))
plt.plot(fpr_eda, tpr_eda, label=f'AUC = {roc_auc_eda:.3f}')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.title('EDA-only Random Forest ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid(True)
plt.show()
```



In [49]:
```
# Optional: 5-Fold Cross-validation using full dataset
X_all_full = np.concatenate((X_train, X_test), axis=0)
y_all_full = np.concatenate((y_train, y_test), axis=0)
eda_only_full = X_all_full[:, 3::6]

rf_eda_cv_scores = cross_val_score(RandomForestClassifier(n_estimators=100,

print("\n=== 5-Fold CV (EDA-only Random Forest) ===")
print(f"Fold Accuracies: {np.round(rf_eda_cv_scores, 4)}")
print(f"Mean Accuracy: {np.round(np.mean(rf_eda_cv_scores), 4)}")
print(f"Std Dev: {np.round(np.std(rf_eda_cv_scores), 4)}")
```

```
=== 5-Fold CV (EDA-only Random Forest) ===
Fold Accuracies: [0.6979 0.6967 0.6944 0.6926 0.6959]
Mean Accuracy: 0.6955
Std Dev: 0.0019
```

In [50]:
```
# XGBoost model
xgb_eda = XGBClassifier(n_estimators=100, random_state=42, use_label_encoder
xgb_eda.fit(X_train_eda, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
8:38:03] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

Out[50]:

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping
_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=Non
e,
              learning_rate=None, max_bin=None, max_cat_threshol
```

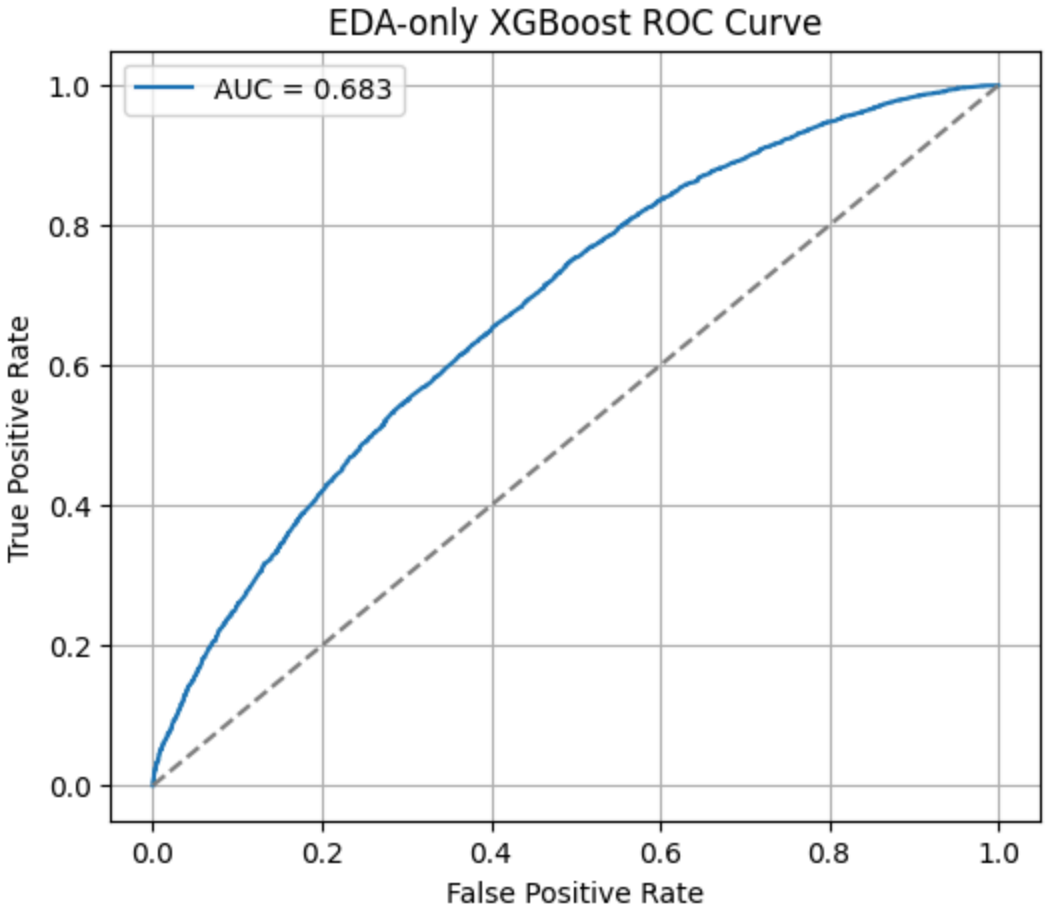
In [51]:

```
# Evaluation
y_pred_eda_xgb = xgb_eda.predict(X_test_eda)
probs_eda_xgb = xgb_eda.predict_proba(X_test_eda)[:, 1]

print("=== EDA-only XGBoost Evaluation ===")
print(classification_report(y_test, y_pred_eda_xgb, digits=4))
print("ROC-AUC:", roc_auc_score(y_test, probs_eda_xgb))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_eda_xgb))
```

```
=== EDA-only XGBoost Evaluation ===
              precision    recall  f1-score   support

           0     0.6118    0.2978    0.4006      4890
           1     0.7018    0.8974    0.7876      9005

    accuracy                         0.6864     13895
   macro avg     0.6568    0.5976    0.5941     13895
weighted avg     0.6701    0.6864    0.6514     13895

ROC-AUC: 0.6827433747895114
Confusion Matrix:
 [[1456 3434]
 [ 924 8081]]
```

In [52]:

```
# Plot ROC Curve
fpr_eda_xgb, tpr_eda_xgb, _ = roc_curve(y_test, probs_eda_xgb)
roc_auc_eda_xgb = roc_auc_score(y_test, probs_eda_xgb)

plt.figure(figsize=(6,5))
plt.plot(fpr_eda_xgb, tpr_eda_xgb, label=f'AUC = {roc_auc_eda_xgb:.3f}')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.title('EDA-only XGBoost ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid(True)
plt.show()
```



In [57]:
```
# 5-Fold Cross-validation
xgb_eda_cv_scores = cross_val_score(xgb_eda, eda_only_full, y_all_full, cv=5

print("\n=== 5-Fold CV (EDA-only XGBoost) ===")
print(f"Fold Accuracies: {np.round(xgb_eda_cv_scores, 4)}")
print(f"Mean Accuracy: {np.round(np.mean(xgb_eda_cv_scores), 4)}")
print(f"Std Dev: {np.round(np.std(xgb_eda_cv_scores), 4)}")
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
8:47:20] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
8:47:29] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
8:47:35] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
8:47:44] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [0
8:47:51] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
=== 5-Fold CV (EDA-only XGBoost) ===
Fold Accuracies: [0.686  0.6879 0.6897 0.6909 0.69  ]
Mean Accuracy: 0.6889
Std Dev: 0.0018
```

In [ ]: