✅ Gaussian Naive Bayes (GNB)

```
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_ma
```

```
url = 'https://github.com/manas8173/diabeties/raw/main/heart_data.csv'
heart = pd.read_csv(url, sep=',')  # explicitly use comma separator
display(heart)
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	a
0	0	18393	2	168	62.0	110	80	1	1	0	
1	1	20228	1	156	85.0	140	90	3	1	0	
2	2	18857	1	165	64.0	130	70	3	1	0	
3	3	17623	2	169	82.0	150	100	1	1	0	
4	4	17474	1	156	56.0	100	60	1	1	0	
...	...	...	...	...	...	...	...	...	...	...	
69995	99993	19240	2	168	76.0	120	80	1	1	1	
69996	99995	22601	1	158	126.0	140	90	2	2	0	
69997	99996	19066	2	183	105.0	180	90	3	1	0	
69998	99998	22431	1	163	72.0	135	80	1	2	0	
69999	99999	20540	1	170	72.0	120	80	2	1	0	

70000 rows × 13 columns

📥 Step 1: Load & Preprocess

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Assume `heart` is already loaded
X = heart.drop("cardio", axis=1)
y = heart["cardio"]
```

```
# Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 2: Train GNB Model

```
from sklearn.naive_bayes import GaussianNB

gnb_model = GaussianNB()
gnb_model.fit(X_train_scaled, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

Step 3: Evaluate GNB Accuracy

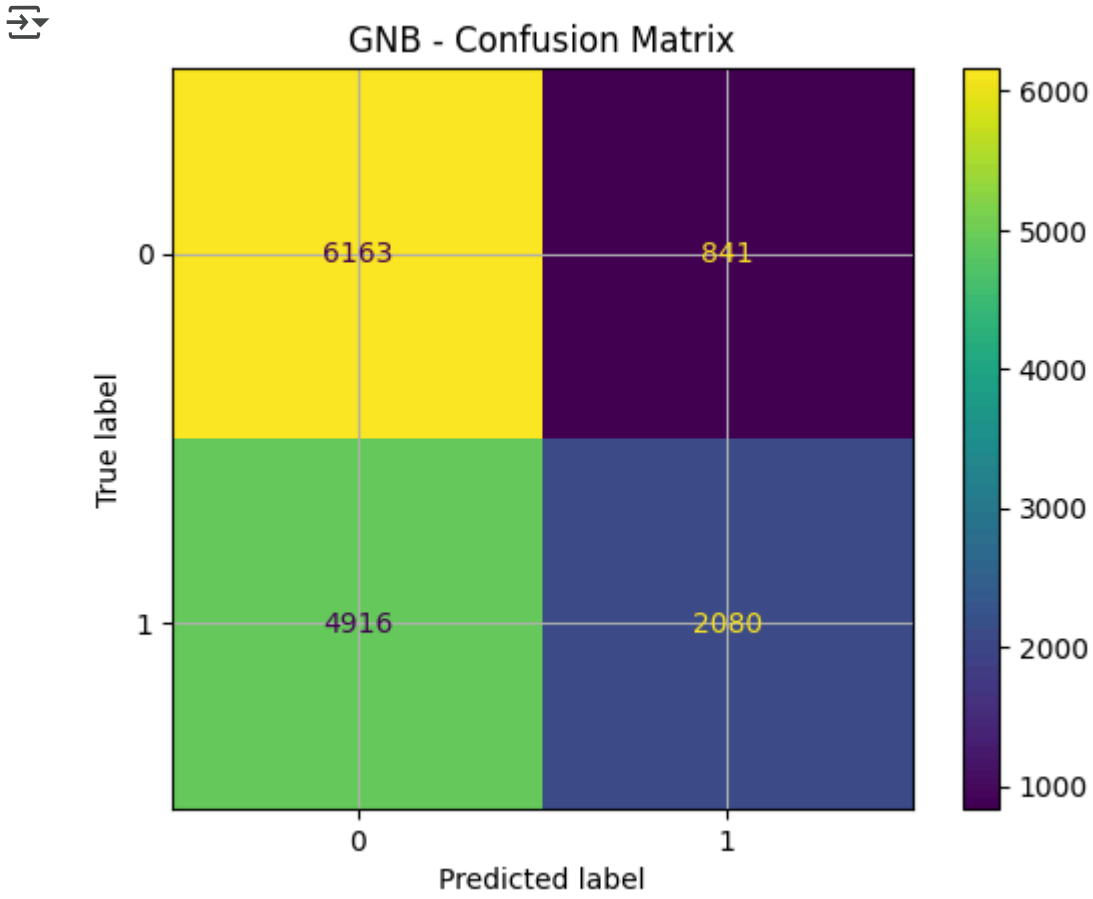
```
from sklearn.metrics import accuracy_score, classification_report

y_pred_gnb = gnb_model.predict(X_test_scaled)

print("Accuracy:", accuracy_score(y_test, y_pred_gnb))
print(classification_report(y_test, y_pred_gnb))
```

```
Accuracy: 0.5887857142857142
              precision    recall  f1-score   support

           0       0.56      0.88      0.68      7004
           1       0.71      0.30      0.42      6996

    accuracy                           0.59     14000
   macro avg       0.63      0.59      0.55     14000
weighted avg       0.63      0.59      0.55     14000
```

Step 4: Confusion Matrix

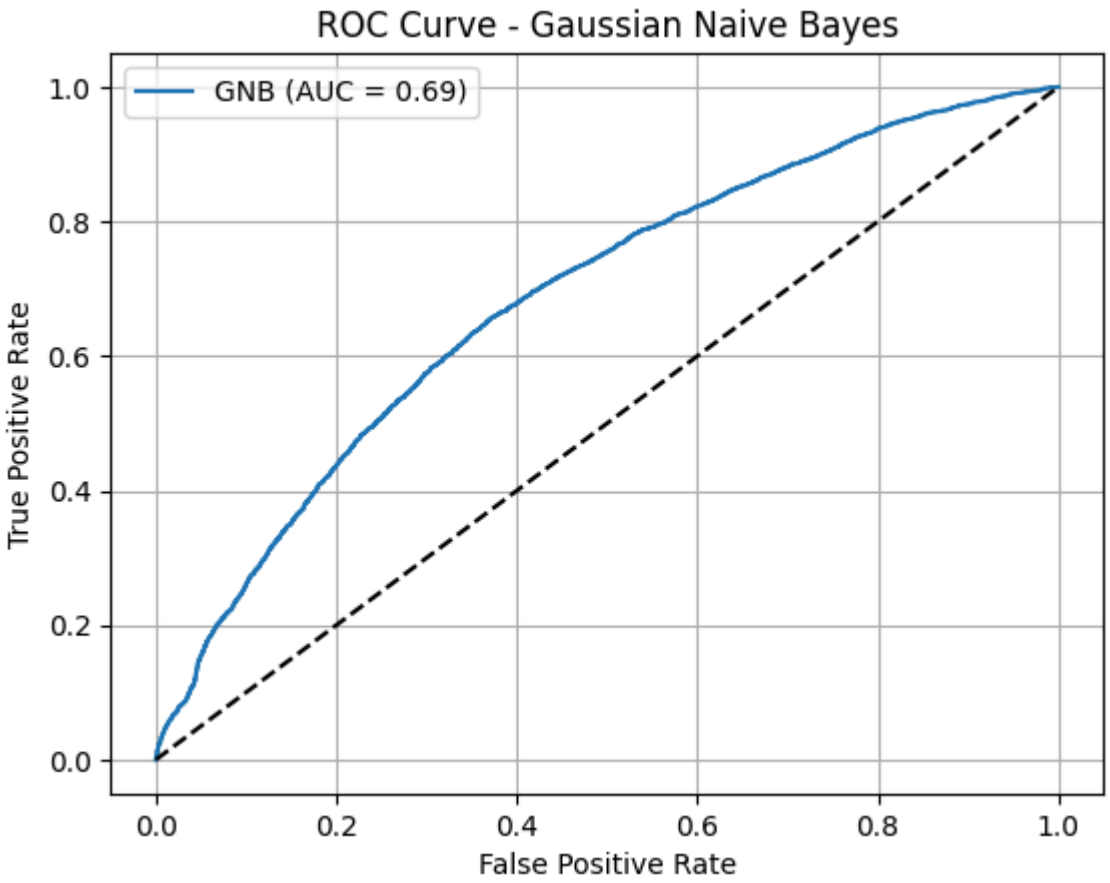
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred_gnb)
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("GNB - Confusion Matrix")
plt.grid()
plt.show()
```



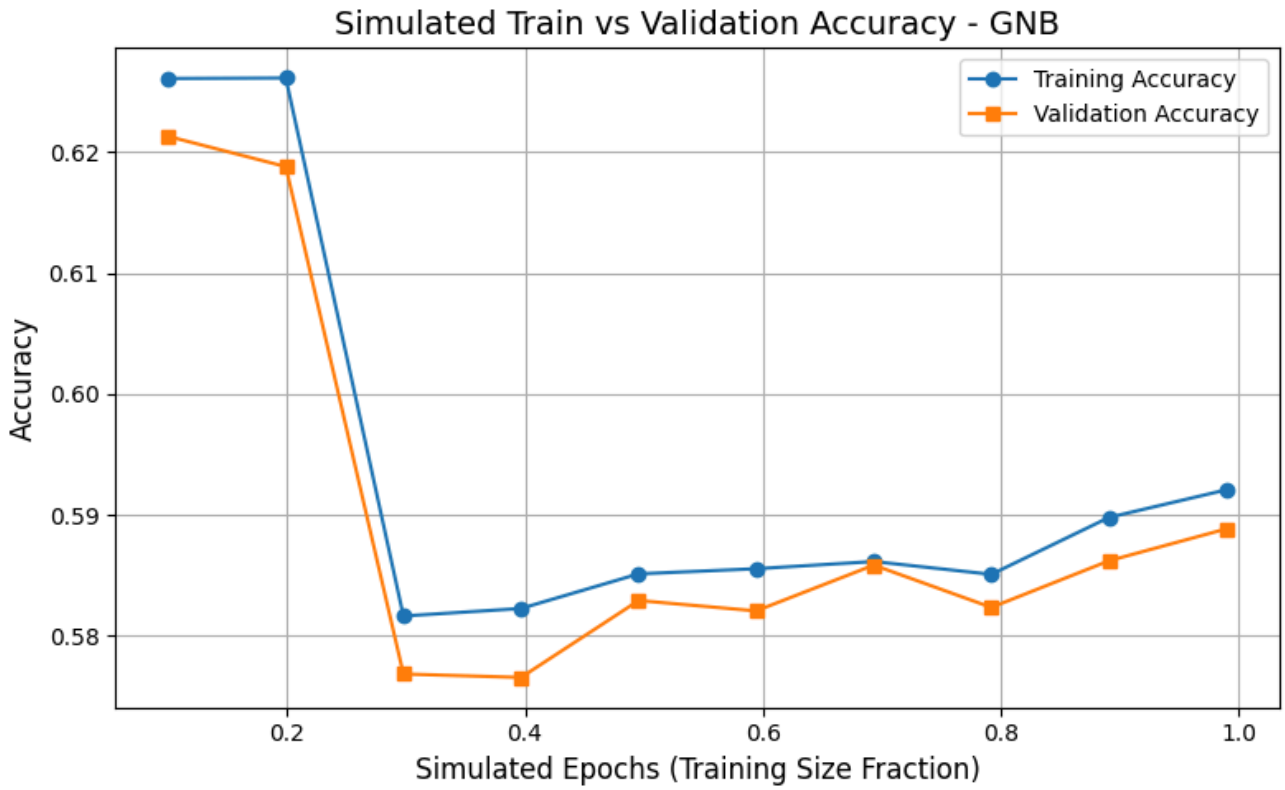
📈 Step 5: ROC Curve

```
from sklearn.metrics import roc_curve, auc

y_prob_gnb = gnb_model.predict_proba(X_test_scaled)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob_gnb)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label="GNB (AUC = %0.2f)" % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Gaussian Naive Bayes")
plt.legend()
plt.grid()
plt.show()
```



📉 Step 6: Train vs Validation Accuracy Curve python Copy Edit

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

train_sizes = np.linspace(0.1, 0.99, 10)
train_scores = []
val_scores = []

for frac in train_sizes:
    X_part, _, y_part, _ = train_test_split(X_train_scaled, y_train, train_size
    model = GaussianNB()
    model.fit(X_part, y_part)
    train_scores.append(model.score(X_part, y_part))
    val_scores.append(model.score(X_test_scaled, y_test))

plt.figure(figsize=(8, 5))
plt.plot(train_sizes, train_scores, label='Training Accuracy', marker='o')
plt.plot(train_sizes, val_scores, label='Validation Accuracy', marker='s')
plt.xlabel("Simulated Epochs (Training Size Fraction)", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.title("Simulated Train vs Validation Accuracy - GNB", fontsize=14)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Step 2: Train XGBoost with Epoch Tracking

```
from xgboost import XGBClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

# Define model with eval_metric inside the constructor
xgb_model = XGBClassifier(
    n_estimators=100,
    use_label_encoder=False,
    eval_metric='logloss'  # ✅ move here
)

# Train and capture evaluation result
xgb_model.fit(
    X_train_scaled, y_train,
    eval_set=[(X_train_scaled, y_train), (X_test_scaled, y_test)],
    verbose=False
)

# Get training results
results = xgb_model.evals_result()
epochs = range(1, len(results['validation_0']['logloss']) + 1)
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:09]
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

📈 Step 3: Plot Epoch-wise Accuracy (Simulated)

```
train_acc = []
val_acc = []

for i in range(1, 101):  # from 1 to 100 estimators
    model = XGBClassifier(
        n_estimators=i,
        use_label_encoder=False,
        eval_metric='logloss'
    )
    model.fit(X_train_scaled, y_train)
    train_acc.append(accuracy_score(y_train, model.predict(X_train_scaled)))
    val_acc.append(accuracy_score(y_test, model.predict(X_test_scaled)))

plt.plot(range(1, 101), train_acc, label='Training Accuracy', color='blue')
plt.plot(range(1, 101), val_acc, label='Validation Accuracy', color='orange')
plt.xlabel("Epochs (Boosting Rounds)")
plt.ylabel("Accuracy")
plt.title("Epoch-wise Accuracy - XGBoost")
plt.legend()
plt.grid(True)
plt.show()
```

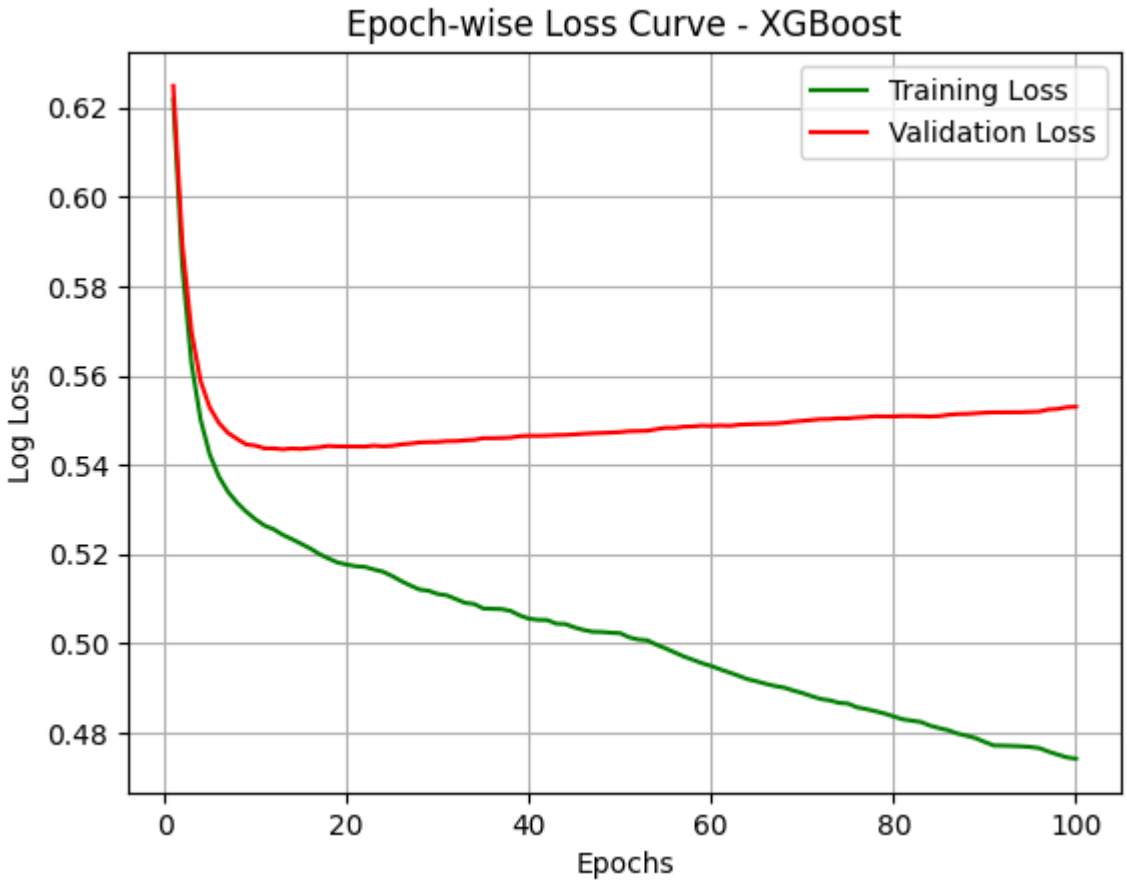
```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:4
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:4
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:4
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.
```

📉 Step 4: Epoch-wise Loss Curve (Direct from XGBoost)

```
  warnings.warn(smsg, UserWarning)
```

```
plt.plot(epochs, results['validation_0']['logloss'], label='Training Loss', color='green'
plt.plot(epochs, results['validation_1']['logloss'], label='Validation Loss', color='red'
plt.xlabel("Epochs")
plt.ylabel("Log Loss")
plt.title("Epoch-wise Loss Curve - XGBoost")
plt.legend()
plt.grid(True)
plt.show()
```

```
warnings.warn(smsg, UserWarning)
```

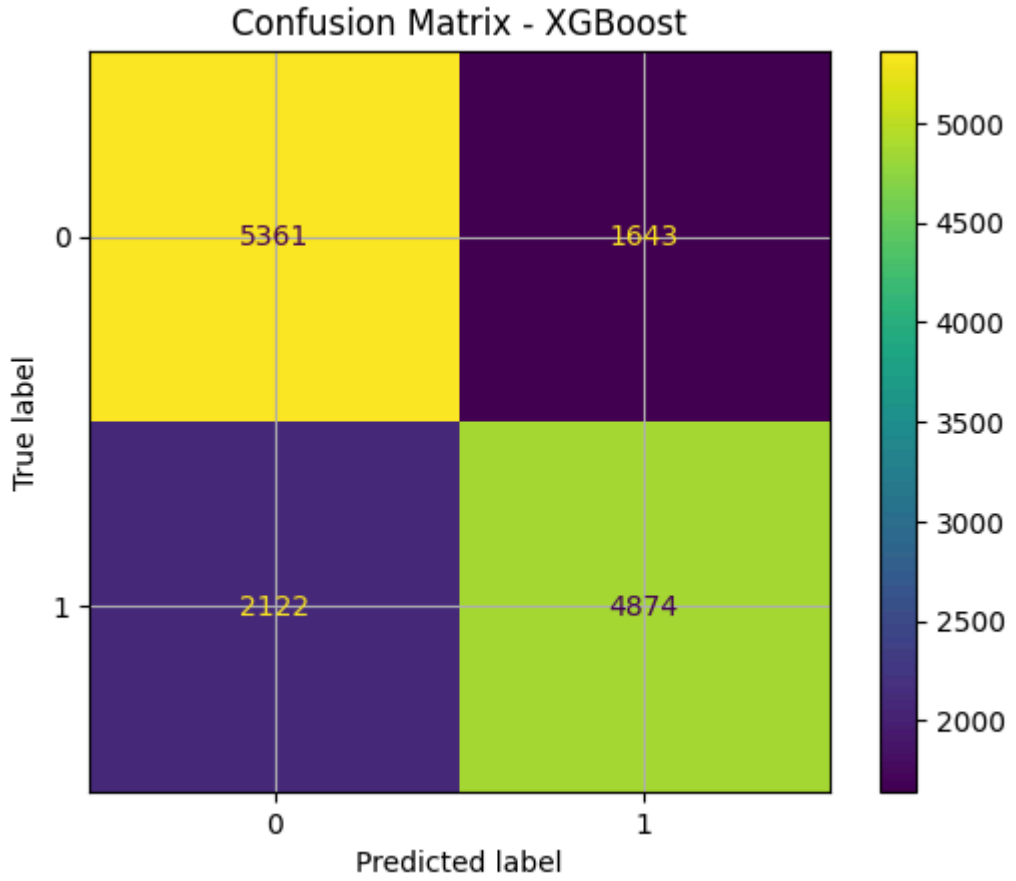


09:43:5
09:43:5
09:43:5
09:43:5
09:43:5
09:43:5
09:43:5

```
warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
Parameters: { "use_label_encoder" } are not used.
```

Step 5: Confusion Matrix + Report

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_repo

y_pred_xgb = xgb_model.predict(X_test_scaled)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred_xgb)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix - XGBoost")
plt.grid()
plt.show()

# Report
print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
```

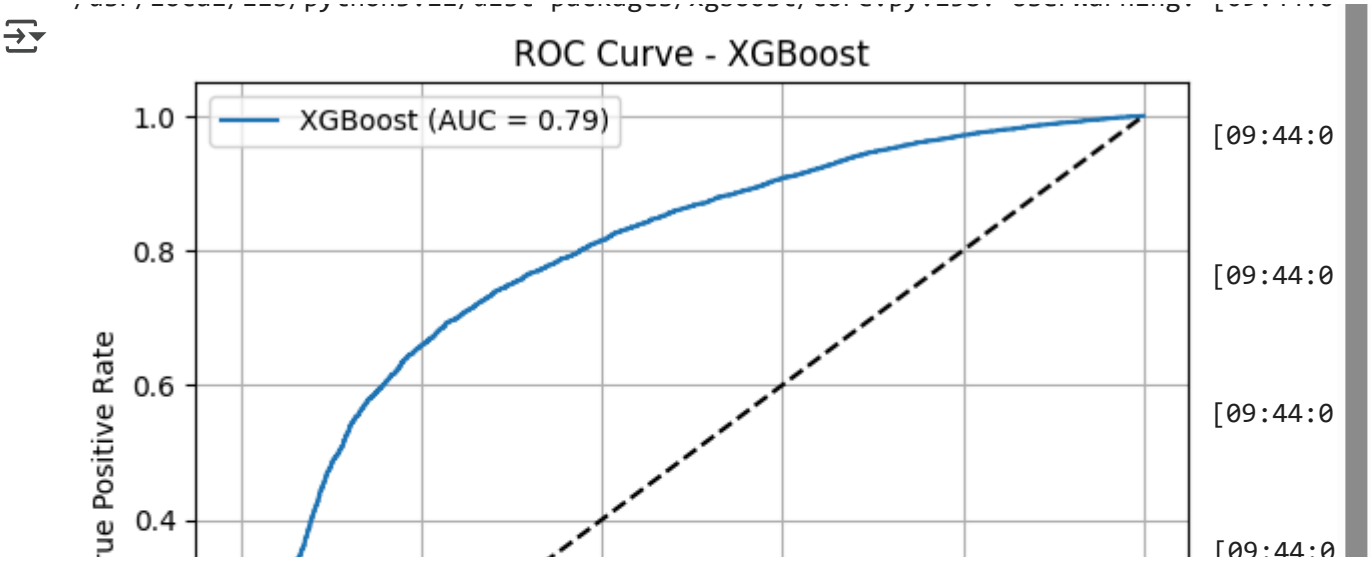



```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:43:5
rning: [09:43:5
rning: [09:43:5
rning: [09:43:5
rning: [09:44:0
rning: [09:44:0
  warnings.warn(smsg, UserWarning)
Classification Report:
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:44:0
Parameters: { "use_label_encoder" } are not used.
              precision    recall  f1-score   support

           0       0.72      0.77      0.74      7004
           1       0.75      0.70      0.72      6996

    accuracy                           0.73     14000
   macro avg       0.73      0.73      0.73     14000
weighted avg       0.73      0.73      0.73     14000

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:44:0
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:44:0
```

Step 6: ROC Curve

```
from sklearn.metrics import roc_curve, auc

y_prob_xgb = xgb_model.predict_proba(X_test_scaled)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob_xgb)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label="XGBoost (AUC = %0.2f)" % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - XGBoost")
plt.legend()
plt.grid()
plt.show()
```

```
  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:44:0
```


ROC Curve - XGBoost
XGBoost (AUC = 0.79)
1.0
0.8
0.6
0.4
ue Positive Rate
[09:44:0
[09:44:0
[09:44:0
[09:44:0