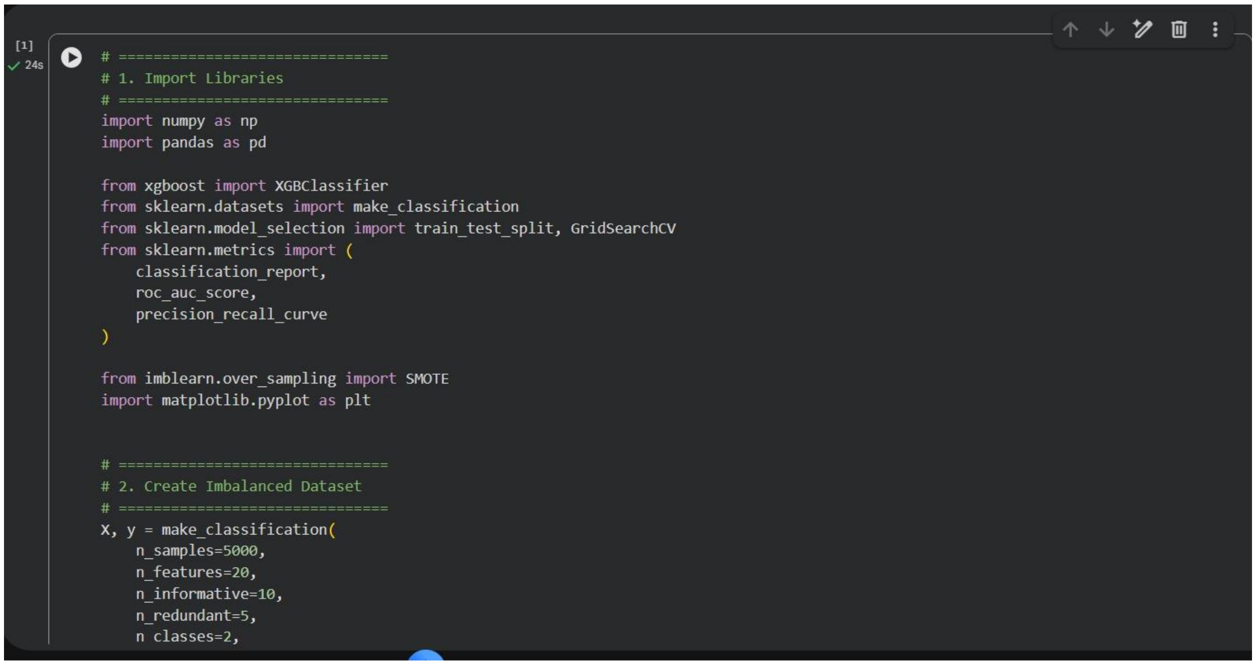
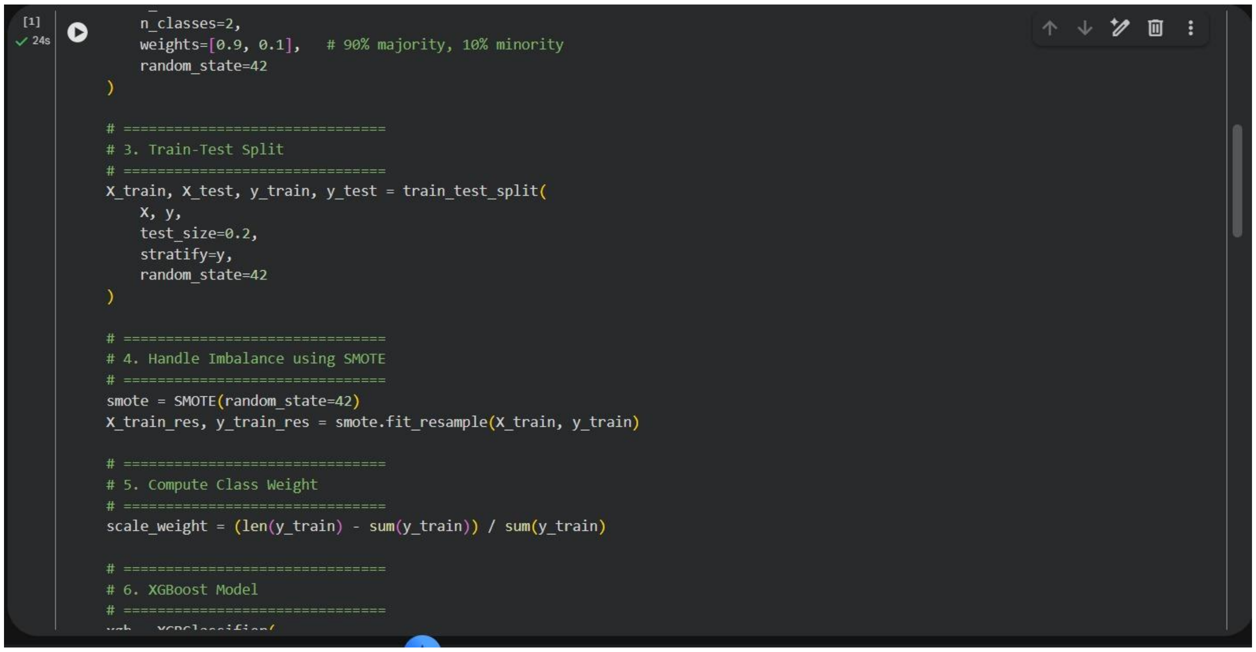
727723EUIT111

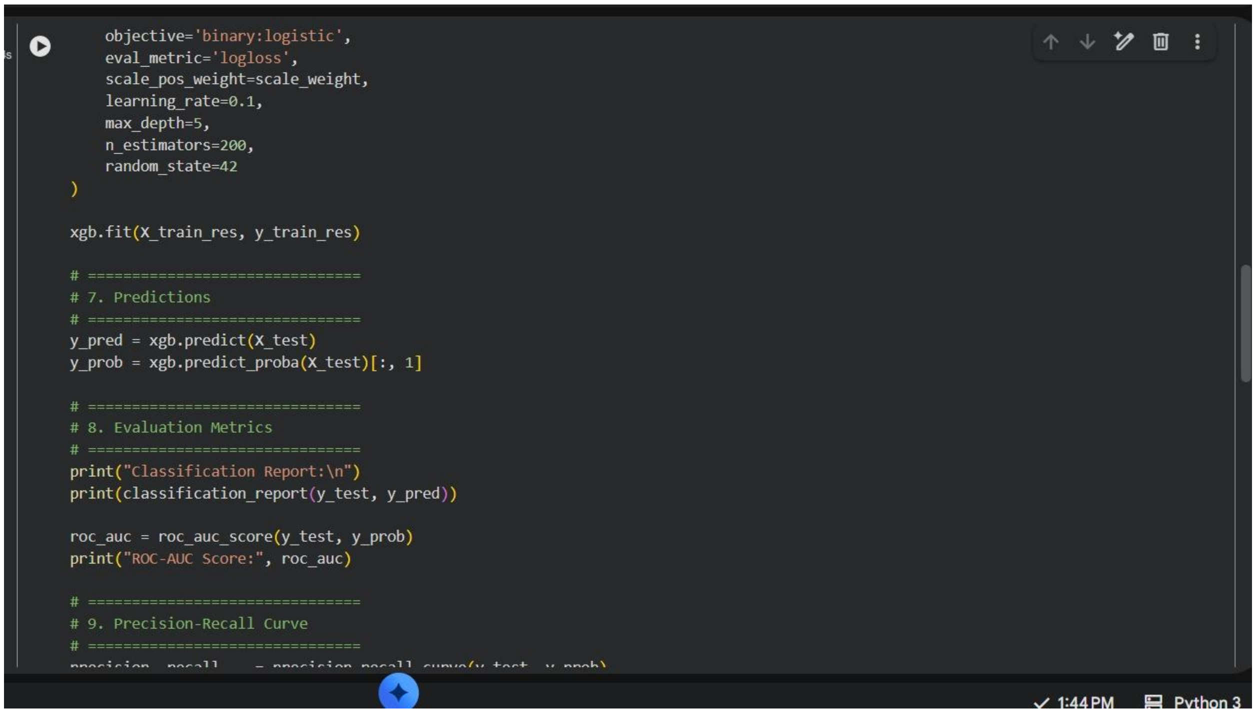
KRITHIKA K



```
[1]
24s
# ==============================
# 1. Import Libraries
# ==============================
import numpy as np
import pandas as pd

from xgboost import XGBClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import (
    classification_report,
    roc_auc_score,
    precision_recall_curve
)

from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt


# ==============================
# 2. Create Imbalanced Dataset
# ==============================
X, y = make_classification(
    n_samples=5000,
    n_features=20,
    n_informative=10,
    n_redundant=5,
    n_classes=2,
```

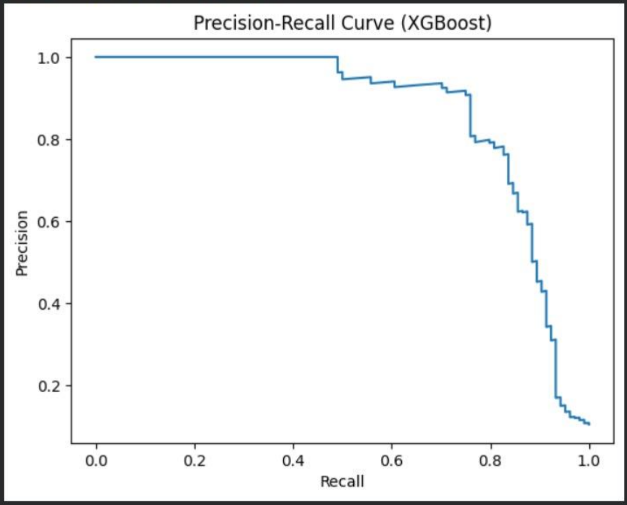


```
[1]
24s
    n_classes=2,
    weights=[0.9, 0.1],   # 90% majority, 10% minority
    random_state=42
)

# ==============================
# 3. Train-Test Split
# ==============================
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42
)

# ==============================
# 4. Handle Imbalance using SMOTE
# ==============================
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

# ==============================
# 5. Compute Class Weight
# ==============================
scale_weight = (len(y_train) - sum(y_train)) / sum(y_train)

# ==============================
# 6. XGBoost Model
# ==============================
```



```python
    objective='binary:logistic',
    eval_metric='logloss',
    scale_pos_weight=scale_weight,
    learning_rate=0.1,
    max_depth=5,
    n_estimators=200,
    random_state=42
)

xgb.fit(X_train_res, y_train_res)

# ==============================
# 7. Predictions
# ==============================
y_pred = xgb.predict(X_test)
y_prob = xgb.predict_proba(X_test)[:, 1]

# ==============================
# 8. Evaluation Metrics
# ==============================
print("Classification Report:\n")
print(classification_report(y_test, y_pred))

roc_auc = roc_auc_score(y_test, y_prob)
print("ROC-AUC Score:", roc_auc)

# ==============================
# 9. Precision-Recall Curve
# ==============================
precision, recall, _ = precision_recall_curve(y_test, y_prob)
```

```
   macro avg       0.84      0.96      0.87      1000
weighted avg       0.95      0.94      0.95      1000

ROC-AUC Score: 0.9346668956043955
```


Precision-Recall Curve (XGBoost)
Precision
Recall

✓ 1:44 PM
Python 3