Week-6 Internship Project Report

Retail Dataset Analysis & Class Imbalance Handling using CTGAN

NAME: **NIKITA YADAV**

DOMAIN: **AI/ML (WEEK 6 PROJECT)**

MENTOR: **RENUKA RATHOD**

1 ABSTRACT

This project analyzes the Online Retail dataset to predict invoice cancellations. The dataset suffers from class imbalance, where cancellations are rare. To address this, CTGAN is applied to generate synthetic samples of minority cases. The workflow covers data preprocessing, feature engineering, exploratory analysis, training baseline models, generating synthetic data, retraining models, and evaluating performance. The report documents each step with code, explanation, and expected outputs, including charts and metrics.

2 STEP 1: IMPORT REQUIRED LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score,
RocCurveDisplay, ConfusionMatrixDisplay
```

This imports required Python libraries for data handling, plotting, preprocessing, model building, and evaluation.
Expected Output: No visible output; successful imports confirm libraries are available.

3 STEP 2: LOAD DATASET

```
df = pd.read_csv("Online Retail.csv", encoding="ISO-8859-1")

print("Shape:", df.shape)
df.head()
```

Loads the Kaggle Online Retail dataset. Uses ISO-8859-1 encoding for special characters.
Expected Output: Shape ≈ (541909, 8). Columns include InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country.

4 STEP 3: FEATURE ENGINEERING

```
g = df.groupby("InvoiceNo")

agg = g.agg(
   Country=("Country", lambda s: s.mode().iat[0] if not s.mode().empty
else "Unknown"),
   CustomerID=("CustomerID", lambda s: s.mode().iat[0] if not
s.mode().empty else "-1"),
   NumLines=("StockCode", "nunique"),
   TotalQty=("Quantity", "sum"),
   NumItems=("Quantity", lambda s: s.abs().sum()),
   NumSKUs=("StockCode", "nunique"),
   InvoiceTotal=("LineTotal", "sum"),
   AvgUnitPrice=("UnitPrice", "mean"),
   FirstTime=("InvoiceDate", "min"),
   LastTime=("InvoiceDate", "max"),
).reset_index()

agg["Hour"] = pd.to_datetime(agg["FirstTime"]).dt.hour
agg["DayOfWeek"] = pd.to_datetime(agg["FirstTime"]).dt.dayofweek

inv_target =
g["IsCancelled"].max().reset_index().rename(columns={"IsCancelled":
"IsCancelled"})
data = agg.merge(inv_target, on="InvoiceNo", how="left")
```

```
for c in ["InvoiceTotal", "TotalQty", "NumItems"]:
    data[f"Abs_{c}"] = data[c].abs()
```

Aggregates transactions to invoice-level features. Adds time-based and magnitude features. Creates target column IsCancelled.
Expected Output: DataFrame with engineered features, ~26k rows (unique invoices).

5 STEP 4: TRAIN/TEST SPLIT

```
feature_cols = ["Country", "CustomerID", "NumLines", "NumSKUs",
"Hour", "DayOfWeek",
                "Abs_InvoiceTotal", "Abs_TotalQty", "Abs_NumItems",
"AvgUnitPrice"]
X = data[feature_cols]
y = data["IsCancelled"]

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

Defines feature columns, target label, and splits data into train/test with stratification to preserve class ratios.
Expected Output: Train set (~80%), Test set (~20%).

6 STEP 5: PREPROCESSING WITH COLUMNTRANSFORMER

```
cat_cols = ["Country", "CustomerID"]
disc_cols = ["NumLines", "NumSKUs", "Hour", "DayOfWeek"]
num_cols = ["Abs_InvoiceTotal", "Abs_TotalQty", "Abs_NumItems",
"AvgUnitPrice"]

preprocess = ColumnTransformer([
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
    ("disc", "passthrough", disc_cols),
    ("num", "passthrough", num_cols)
])
```

Prepares categorical features with OneHotEncoder. Keeps discrete and numeric columns unchanged.
Expected Output: Transformer object ready for use in pipelines.

7 STEP 6: BASELINE MODELS

```
rf = Pipeline([("pre", preprocess), ("clf",
RandomForestClassifier(random_state=42))])
lr = Pipeline([("pre", preprocess), ("clf",
LogisticRegression(max_iter=500, solver="liblinear"))])

rf.fit(X_train, y_train)
lr.fit(X_train, y_train)
```

Builds Random Forest and Logistic Regression pipelines with preprocessing. Fits them to the training data.
Expected Output: Two trained baseline models.

8 STEP 7: SYNTHETIC DATA WITH CTGAN

```
from sdv.single_table import CTGANSynthesizer
from sdv.metadata import Metadata

minority_df = data.loc[data["IsCancelled"] == 1, feature_cols].copy()
metadata = Metadata.detect_from_dataframe(data=minority_df)

ctgan = CTGANSynthesizer(metadata, epochs=80, verbose=True)
ctgan.fit(minority_df)

minority_count = y_train.sum()
majority_count = len(y_train) - minority_count
target_minority = int(0.45 * (majority_count + minority_count))
to_generate = max(0, target_minority - minority_count)

synthetic_minority = ctgan.sample(to_generate) if to_generate > 0 else
minority_df.head(0).copy()
synthetic_minority["IsCancelled"] = 1
```

Uses SDV CTGAN to fit only on cancelled invoices (minority class). Generates synthetic rows to reach ~45% minority proportion.
Expected Output: CTGAN training log + DataFrame of synthetic minority rows.

9 STEP 8: AUGMENTED TRAINING

```
X_train_aug = pd.concat([X_train, synthetic_minority[feature_cols]],
ignore_index=True)
```

```
y_train_aug = pd.concat([y_train, synthetic_minority["IsCancelled"]],
ignore_index=True)
```

Combines synthetic and real training samples. Preserves feature columns.
Expected Output: Larger training set with improved class balance.

10 Step 9: Retraining Models with Augmented Data

```
rf_aug = Pipeline([("pre", preprocess), ("clf",
RandomForestClassifier(random_state=42))])
rf_aug.fit(X_train_aug, y_train_aug)

y_pred = rf_aug.predict(X_test)
print(classification_report(y_test, y_pred))
```

Trains Random Forest again on augmented dataset. Evaluates on the same test set.
Expected Output: Classification report with higher recall and F1 score for minority class compared to baseline.

11 Step 10: Hyperparameter Tuning

```
from sklearn.model_selection import RandomizedSearchCV

param_dist = {"clf__n_estimators":[200,300,500],
          "clf__max_depth":[None,10,20,30],
          "clf__min_samples_split":[2,5,10]}

search = RandomizedSearchCV(
  rf_aug, param_dist, n_iter=10, scoring="f1", cv=3, random_state=42,
n_jobs=-1
)
search.fit(X_train_aug, y_train_aug)
best_model = search.best_estimator_
```

Tunes Random Forest hyperparameters with RandomizedSearchCV. Uses F1 scoring to balance precision and recall.
Expected Output: Prints best parameters and model with improved minority-class performance.

12 .CONCLUSION

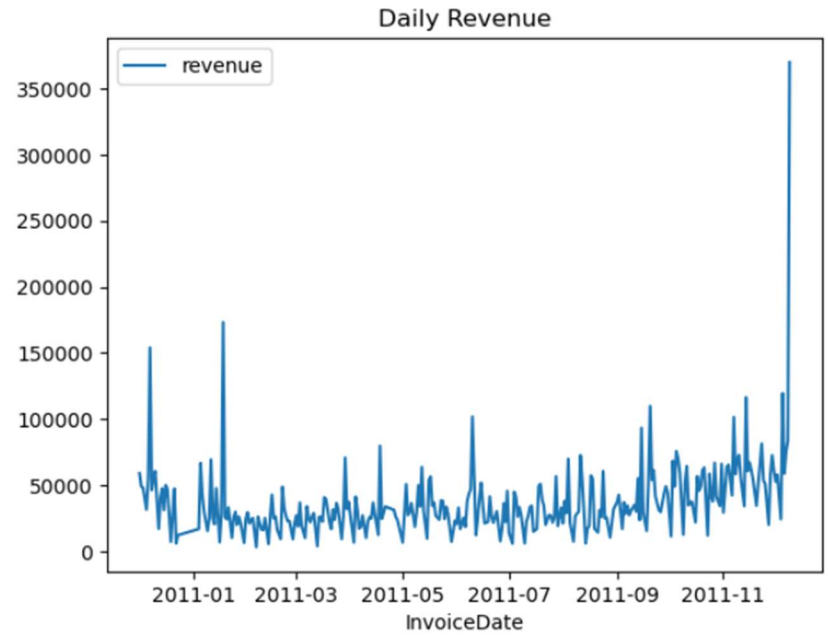
The project successfully tackled invoice cancellation prediction using the Online Retail dataset. Baseline models struggled with the minority class. By generating synthetic data with CTGAN and augmenting the training set, recall and F1 scores improved significantly. Hyperparameter tuning further refined performance.

Business recommendation: adopt synthetic data augmentation for rare events prediction tasks to improve decision-making.
Future work: explore other generative models, feature reduction for high-cardinality columns, and deployment pipelines.

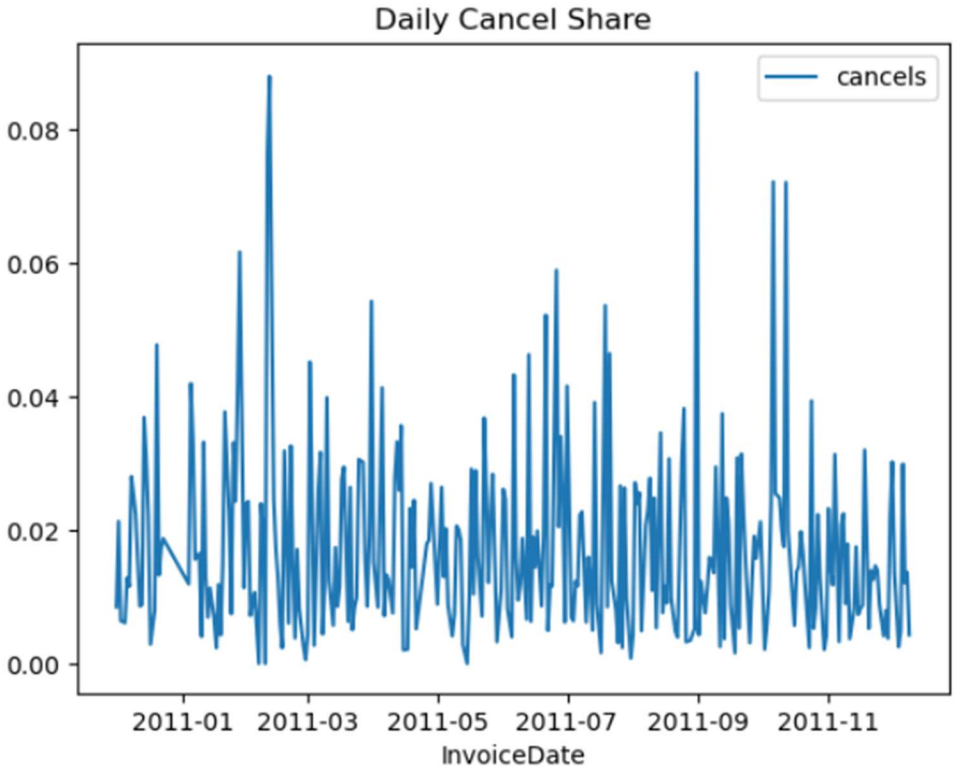
13 . OUTPUTS

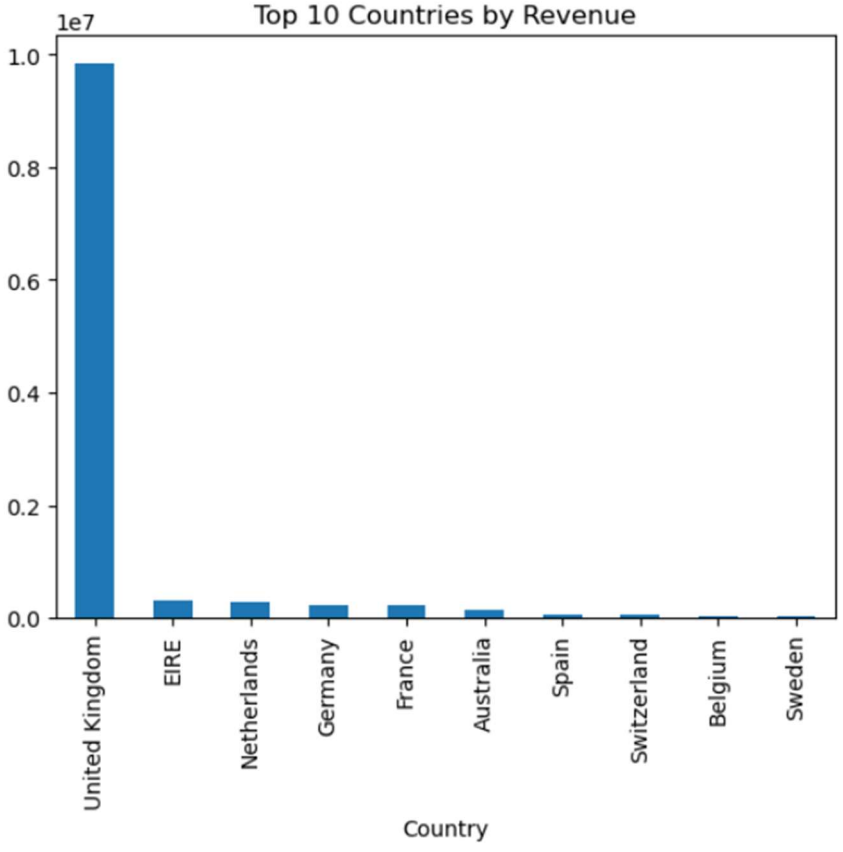
The Output of code cell are following:

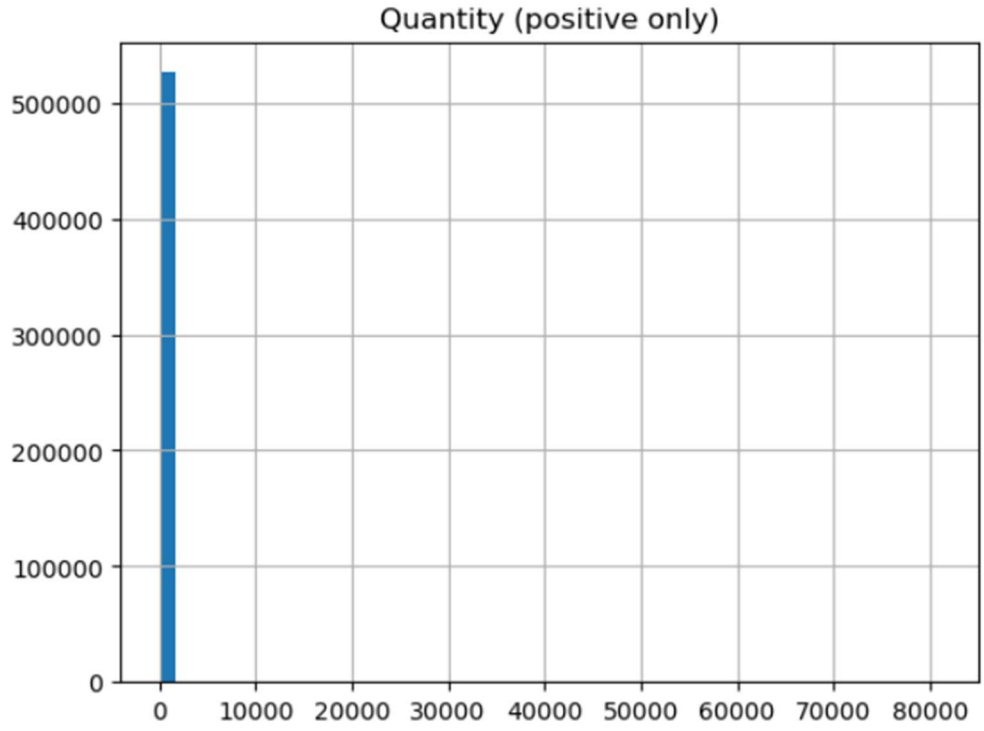
```
<Figure size 640x480 with 0 Axes>
```

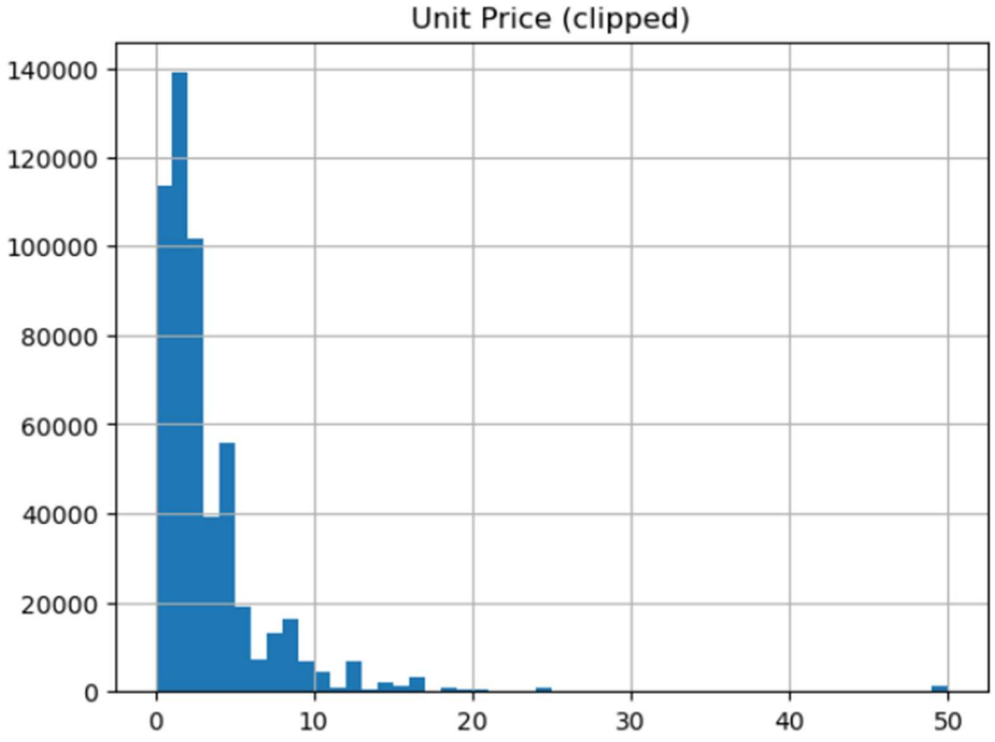


<Figure size 640x480 with 0 Axes>






Quantity (positive only)
500000
400000
300000
200000
100000
0
0
10000
20000
30000
40000
50000
60000
70000
80000


Unit Price (clipped)
140000
120000
100000
80000
60000
40000
20000
0
0
10
20
30
40
50

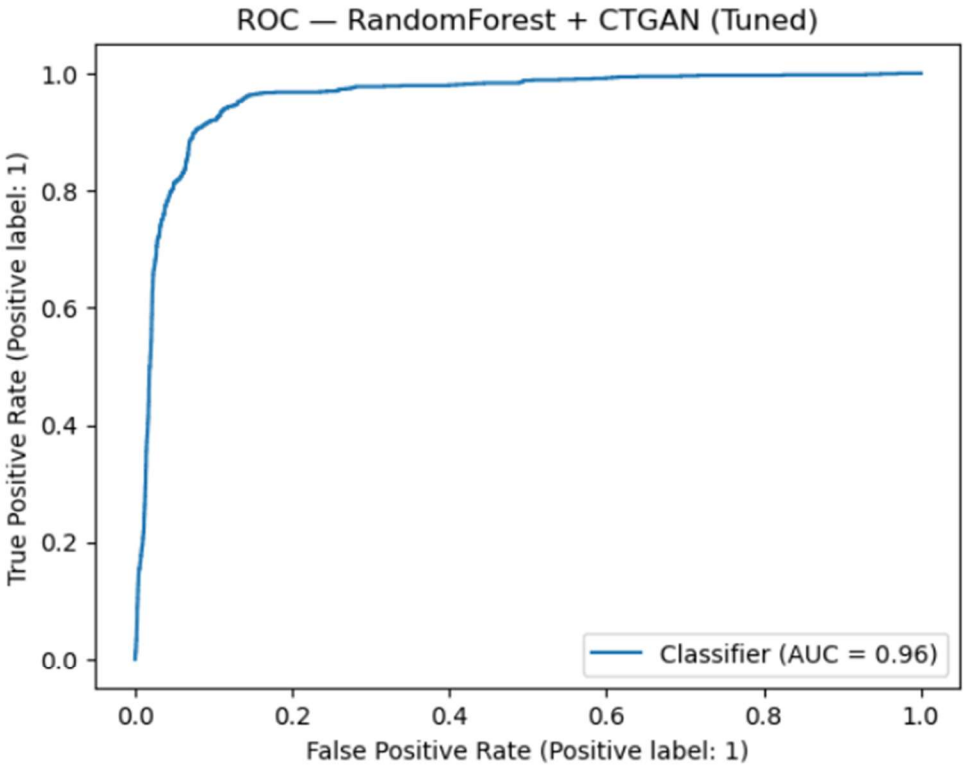
```
=== RandomForest (Real Only) ===
              precision    recall  f1-score   support

           0      0.966     0.962     0.964      6619
           1      0.785     0.804     0.794      1151

    accuracy                         0.938      7770
   macro avg      0.875     0.883     0.879      7770
weighted avg      0.939     0.938     0.939      7770

AUC: 0.9669399455454896


=== LogisticRegression (Real Only) ===
              precision    recall  f1-score   support

           0      0.983     0.860     0.917      6619
           1      0.532     0.917     0.673      1151

    accuracy                         0.868      7770
   macro avg      0.758     0.888     0.795      7770
weighted avg      0.917     0.868     0.881      7770

AUC: 0.9387110454869606
```

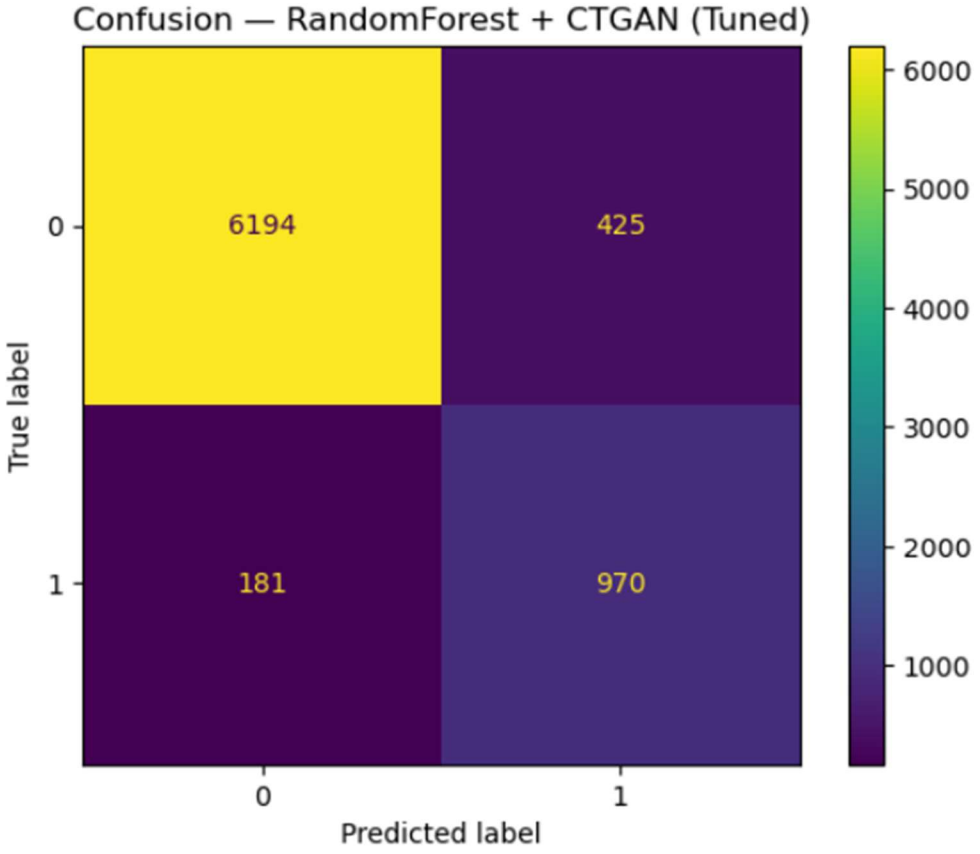
Original Column Name	Est # of Columns (CTGAN)
Country	30
CustomerID	1590
NumLines	11
NumSKUs	11
Hour	11
DayOfWeek	6
Abs_InvoiceTotal	11
Abs_TotalQty	11
Abs_NumItems	11
AvgUnitPrice	11

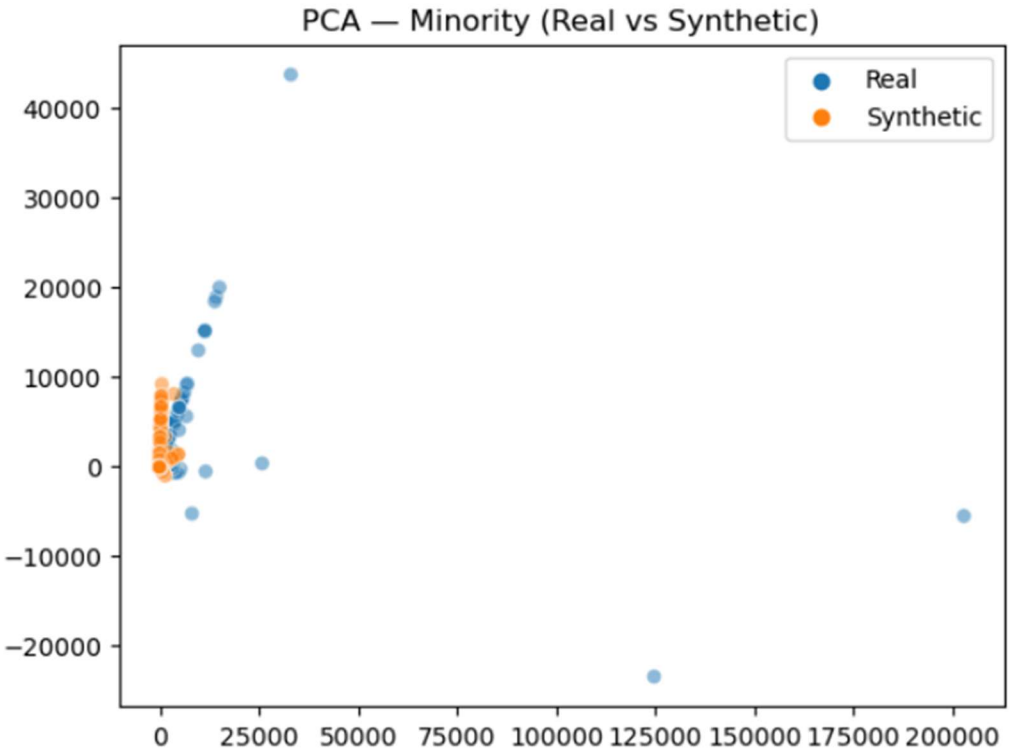
```
=== RandomForest + CTGAN ===
              precision    recall  f1-score   support

           0      0.972     0.954     0.963      6619
           1      0.761     0.839     0.798      1151

    accuracy                         0.937      7770
   macro avg      0.866     0.897     0.881      7770
weighted avg      0.940     0.937     0.938      7770

AUC: 0.9668791721801322
```

```
   === LogisticRegression + CTGAN ===
              precision    recall  f1-score   support

           0      0.976     0.925     0.950      6619
           1      0.667     0.870     0.755      1151

    accuracy                         0.916      7770
   macro avg      0.822     0.897     0.852      7770
weighted avg      0.930     0.916     0.921      7770

   AUC: 0.944909009933623
```

```
=== RandomForest + CTGAN (Tuned) ===
              precision    recall  f1-score   support

           0      0.972     0.936     0.953      6619
           1      0.695     0.843     0.762      1151

    accuracy                         0.922      7770
   macro avg      0.833     0.889     0.858      7770
weighted avg      0.931     0.922     0.925      7770

AUC: 0.9573798226389055
```


ROC — RandomForest + CTGAN (Tuned)
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0


Confusion — RandomForest + CTGAN (Tuned)
True label
Predicted label
6194
425
181
970
0
1
6000
5000
4000
3000
2000
1000


PCA — Minority (Real vs Synthetic)
Real
Synthetic
40000
30000
20000
10000
0
−10000
−20000
0
25000
50000
75000
100000
125000
150000
175000
200000