```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score,
roc_curve
from imblearn.over_sampling import RandomOverSampler, SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.utils.class_weight import compute_class_weight
import matplotlib.pyplot as plt

# Step 1: Dataset Import and Exploration
# Load the dataset
file_path = "imbalanced_dataset.csv"
data = pd.read_csv('/content/imbalanced_dataset.csv')

# Display basic dataset details
print("Dataset Overview:")
print(data.head())
print("\nDataset Info:")
print(data.info())
print("\nClass Distribution:")
print(data.iloc[:, -1].value_counts())

# Split features and target
X = data.drop(columns=data.columns[-1])  # Features
y = data[data.columns[-1]]  # Target

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42, stratify=y)

# Step 2: Techniques for Handling Imbalanced Data
# Function to evaluate and display metrics
def evaluate_model(model, X_test, y_test, title="Model Performance"):
    y_pred = model.predict(X_test)
    print(f"\n{title}")
    print(classification_report(y_test, y_pred))
    roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])
    print(f"AUC-ROC: {roc_auc:.4f}")
    return roc_auc

# 1. Original Dataset
print("\n--- Original Dataset ---")
model_original = LogisticRegression(random_state=42)
model_original.fit(X_train, y_train)
auc_original = evaluate_model(model_original, X_test, y_test,
"Original Dataset")

# 2. Random Oversampling
```

```
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X_train, y_train)
model_ros = LogisticRegression(random_state=42)
model_ros.fit(X_resampled, y_resampled)
auc_ros = evaluate_model(model_ros, X_test, y_test, "Random
Oversampling")

# 3. Random Undersampling
rus = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = rus.fit_resample(X_train, y_train)
model_rus = LogisticRegression(random_state=42)
model_rus.fit(X_resampled, y_resampled)
auc_rus = evaluate_model(model_rus, X_test, y_test, "Random
Undersampling")

# 4. SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
model_smote = LogisticRegression(random_state=42)
model_smote.fit(X_resampled, y_resampled)
auc_smote = evaluate_model(model_smote, X_test, y_test, "SMOTE")

# 5. Class Weighting
class_weights = compute_class_weight('balanced', classes=np.array([0,
1]), y=y_train) # Convert classes to numpy array
weights = {i: weight for i, weight in enumerate(class_weights)}
model_weighted = LogisticRegression(class_weight=weights,
random_state=42)
model_weighted.fit(X_train, y_train)
auc_weighted = evaluate_model(model_weighted, X_test, y_test, "Class
Weighting")

# Step 3: Compare Performance
results = pd.DataFrame({
    "Technique": ["Original", "Random Oversampling", "Random
Undersampling", "SMOTE", "Class Weighting"],
    "AUC-ROC": [auc_original, auc_ros, auc_rus, auc_smote,
auc_weighted]
})

print("\n--- Performance Comparison ---")
print(results)

# Plot ROC Curves for visualization
plt.figure(figsize=(10, 6))
for model, label in zip(
    [model_original, model_ros, model_rus, model_smote,
model_weighted],
    ["Original", "Random Oversampling", "Random Undersampling",
"SMOTE", "Class Weighting"]
```

```
):
    fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:, 1])
    plt.plot(fpr, tpr, label=f"{label} (AUC: {roc_auc_score(y_test,
model.predict_proba(X_test)[:, 1]):.2f})")

plt.plot([0, 1], [0, 1], 'k--', label="Random Guess")
plt.title("ROC Curves")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

```
Dataset Overview:
   feature_1  feature_2  feature_3  feature_4  feature_5  feature_6  \
0   2.830036   3.710393  -3.541272  -1.878474   5.259377  -3.542790
1   0.971669   0.288186  -2.124649  -1.419615   0.823406  -0.316680
2   0.331286  -1.591030  -2.092067   0.767135   0.793704  -1.174513
3  -1.478092  -0.343648  -0.038281  -1.361832   1.235385  -0.791223
4   2.579622   0.332168   2.788795  -3.757333   4.051230  -9.283099

   feature_7  feature_8  feature_9  feature_10  target_column
0   2.144518  -3.211325  -0.120127   -2.557971              1
1   0.075697  -1.055648   2.632365   -0.335800              0
2   1.818774  -1.254556   0.281308   -0.987476              0
3  -2.046527   0.071025   1.328145   -0.736580              0
4   2.790275  -4.647829  -1.329250   -1.974946              0

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   feature_1      1000 non-null   float64
 1   feature_2      1000 non-null   float64
 2   feature_3      1000 non-null   float64
 3   feature_4      1000 non-null   float64
 4   feature_5      1000 non-null   float64
 5   feature_6      1000 non-null   float64
 6   feature_7      1000 non-null   float64
 7   feature_8      1000 non-null   float64
 8   feature_9      1000 non-null   float64
 9   feature_10     1000 non-null   float64
 10  target_column  1000 non-null   int64
dtypes: float64(10), int64(1)
memory usage: 86.1 KB
None

Class Distribution:
target_column
```

```
0    900
1    100
Name: count, dtype: int64

--- Original Dataset ---

Original Dataset
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       270
           1       0.00      0.00      0.00        30

    accuracy                           0.89       300
   macro avg       0.45      0.50      0.47       300
weighted avg       0.81      0.89      0.85       300

AUC-ROC: 0.7563

Random Oversampling
              precision    recall  f1-score   support

           0       0.95      0.70      0.80       270
           1       0.20      0.67      0.30        30

    accuracy                           0.69       300
   macro avg       0.57      0.68      0.55       300
weighted avg       0.87      0.69      0.75       300

AUC-ROC: 0.7546

Random Undersampling
              precision    recall  f1-score   support

           0       0.96      0.68      0.80       270
           1       0.21      0.77      0.33        30

    accuracy                           0.69       300
   macro avg       0.59      0.72      0.56       300
weighted avg       0.89      0.69      0.75       300

AUC-ROC: 0.7565

SMOTE
              precision    recall  f1-score   support

           0       0.95      0.71      0.81       270
           1       0.20      0.67      0.31        30

    accuracy                           0.71       300
   macro avg       0.58      0.69      0.56       300
weighted avg       0.88      0.71      0.76       300
```

```
AUC-ROC: 0.7584

Class Weighting
              precision    recall  f1-score   support

           0       0.95      0.69      0.80       270
           1       0.19      0.67      0.30        30

    accuracy                           0.69       300
   macro avg       0.57      0.68      0.55       300
weighted avg       0.87      0.69      0.75       300

AUC-ROC: 0.7572

--- Performance Comparison ---
              Technique   AUC-ROC
0              Original  0.756296
1    Random Oversampling  0.754568
2   Random Undersampling  0.756543
3                  SMOTE  0.758395
4        Class Weighting  0.757160
```