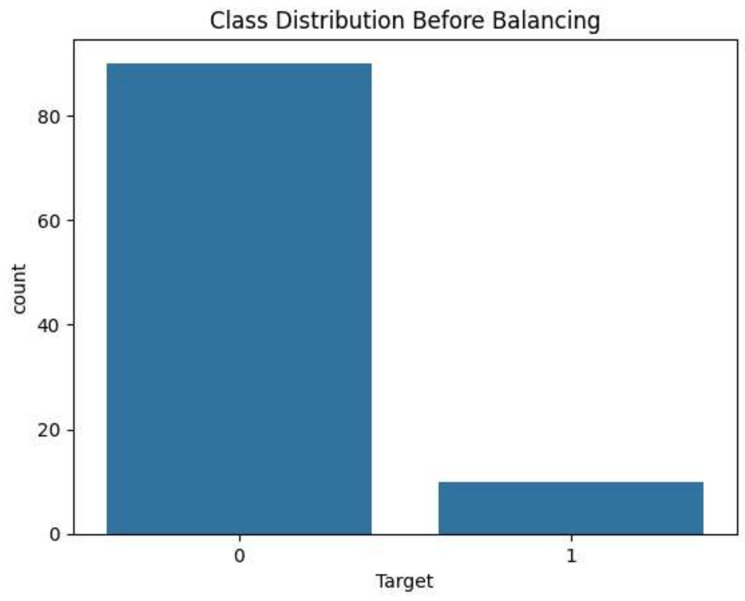
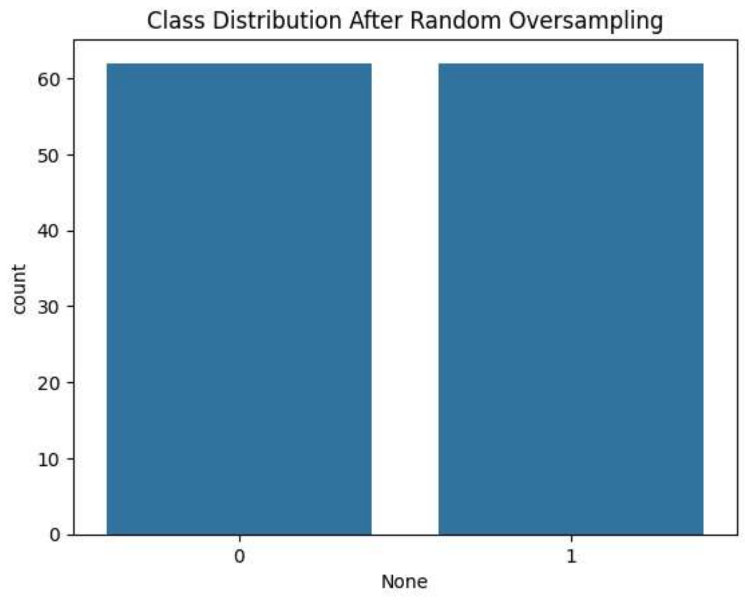
```
# Install required libraries
!pip install imbalanced-learn scikit-learn pandas matplotlib seaborn

# Import libraries
import pandas as pd
from sklearn.datasets import make_classification
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from imblearn.over_sampling import RandomOverSampler, SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.combine import SMOTEENN

# Step 1: Create an imbalanced dataset
X, y = make_classification(
    n_classes=2, class_sep=2, weights=[0.9, 0.1],  # 90:10 imbalance
    n_informative=3, n_redundant=1, flip_y=0,
    n_features=5, n_clusters_per_class=1, random_state=42
)

# Convert to a DataFrame for easy handling
df = pd.DataFrame(X, columns=[f'Feature_{i}' for i in range(1, 6)])
df['Target'] = y

# Visualize class distribution
print("Class distribution before balancing:")
print(df['Target'].value_counts())

sns.countplot(x='Target', data=df)
plt.title("Class Distribution Before Balancing")
plt.show()

# Step 2: Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 3: Random Oversampling
ros = RandomOverSampler(random_state=42)
X_ros, y_ros = ros.fit_resample(X_train, y_train)

print("\nClass distribution after Random Oversampling:")
print(pd.Series(y_ros).value_counts())

sns.countplot(x=pd.Series(y_ros))
plt.title("Class Distribution After Random Oversampling")
plt.show()

# Step 4: Random Undersampling
rus = RandomUnderSampler(random_state=42)
X_rus, y_rus = rus.fit_resample(X_train, y_train)

print("\nClass distribution after Random Undersampling:")
print(pd.Series(y_rus).value_counts())

sns.countplot(x=pd.Series(y_rus))
plt.title("Class Distribution After Random Undersampling")
plt.show()

# Step 5: SMOTE (Synthetic Minority Oversampling)
smote = SMOTE(random_state=42)
X_smote, y_smote = smote.fit_resample(X_train, y_train)

print("\nClass distribution after SMOTE:")
print(pd.Series(y_smote).value_counts())

sns.countplot(x=pd.Series(y_smote))
plt.title("Class Distribution After SMOTE")
plt.show()

# Step 6: SMOTEENN (Combine SMOTE and Undersampling)
smoteenn = SMOTEENN(random_state=42)
X_smoteenn, y_smoteenn = smoteenn.fit_resample(X_train, y_train)

print("\nClass distribution after SMOTEENN:")
print(pd.Series(y_smoteenn).value_counts())
```

```
sns.countplot(x=pd.Series(y_smoteenn))
plt.title("Class Distribution After SMOTEENN")
plt.show()

# Step 7: Train a model with class weights
clf = RandomForestClassifier(class_weight='balanced', random_state=42)
clf.fit(X_train, y_train)

# Predictions and evaluation
y_pred = clf.predict(X_test)
print("\nClassification Report (Using Class Weights):")
print(classification_report(y_test, y_pred))

# Step 8: Visualize resampled data (SMOTE used as an example)
sns.scatterplot(x=X_smote[:, 0], y=X_smote[:, 1], hue=y_smote, palette='coolwarm')
plt.title("Visualization of SMOTE Resampled Data")
plt.show()
```

```
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.12.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.8.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (3.5.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.55.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Class distribution before balancing:
Target
0    90
1    10
Name: count, dtype: int64
```



```
Class distribution after Random Oversampling:
0    62
1    62
Name: count, dtype: int64
```



```
Class distribution after Random Undersampling:
```