Laporan Praktikum Imbalanced Data

Nama : Surya Dwi Satria
Kelas : C7
NIM : 434230148

Langkah Langkah

1. Inisialisasi Library yang digunakan

1) Setup & Instalasi

Penjelasan: Cell ini menginstal dan mengimpor *library* yang diperlukan.

- `pandas`, `numpy`: manipulasi data.
- `matplotlib`: visualisasi (tanpa seaborn sesuai aturan).
- `scikit-learn`: preprocessing (`ColumnTransformer`, `OneHotEncoder`, `StandardScaler`), model (`LogisticRegression`, `RandomForestClassifier`), evaluasi (`StratifiedKFold`, `cross_validate`).
- `imblearn`: *resampler* (ROS, RUS, SMOTE, SMOTE-ENN) dan `Pipeline` khusus yang kompatibel dengan resampling.

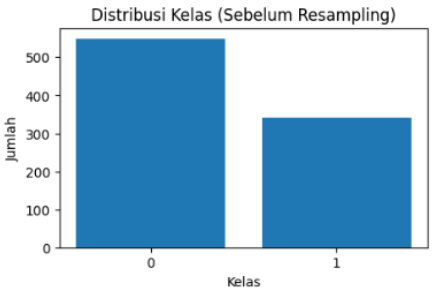
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from pathlib import Path

from sklearn.model_selection import StratifiedKFold,  cross_validate
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline as SkPipeline
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import RandomOverSampler, SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.combine import SMOTEENN
from imblearn.pipeline import Pipeline
RANDOM_STATE = 42
```

2. Memuat Dataset dan menentukan target ouput atau label data

```
# Load data

df = pd.read_csv("train.csv")
target_col = "Survived"
df.columns = df.columns.str.strip()

df.head()
```

3. EDA Singkat & Cek Imbalance

```
print("Ukuran data:", df.shape)
display(df.dtypes)
print("\nMissing values per kolom:\n", df.isna().sum())

assert target_col in df.columns, f"Kolom target '{target_col}' tidak ditemukan!"

# Distribusi kelas
y = df[target_col]
class_counts = y.value_counts().sort_index()
print("\nDistribusi kelas:")
print(class_counts)

# Plot bar sederhana (matplotlib only)
plt.figure(figsize=(5,3))
plt.bar(class_counts.index.astype(str), class_counts.values)
plt.title("Distribusi Kelas (Sebelum Resampling)")
plt.xlabel("Kelas")
plt.ylabel("Jumlah")
plt.show()
```




```
Distribusi kelas:
Survived
0    549
1    342
Name: count, dtype: int64
```

```
Ukuran data: (891, 8)

Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked     object
Survived      int64
dtype: object
```

```
Missing values per kolom:
 Pclass        0
Sex            0
Age          177
SibSp          0
Parch          0
Fare           0
Embarked       2
Survived       0
dtype: int64
```

Penjelasan : Gunakan train.csv (891 baris, 8 kolom: target **“Survived”**), imputasi dulu nilai kosong (`Age`=177 median, `Embarked`=2 modus), lalu encode fitur kategorikal (`Sex`, `Embarked`, `Pclass`) dengan OneHot dan scale numerik seperlunya; karena data imbalanced (0=549 ≈61.6%, 1=342 ≈38.4%), evaluasi pakai F1/Recall/Balanced Accuracy/ROC-AUC bukan akurasi saja, dan pastikan urutan pipeline preprocess → resample (ROS/RUS/SMOTE/SMOTE-ENN) → model.

4. Menentukan Fitur dan Preprocessing

```python
use_cols   = ['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked','Survived']
df = df[use_cols].copy()

X = df.drop(columns=['Survived'])
y = df['Survived']

num_cols = ['Age','SibSp','Parch','Fare']
cat_cols = ['Pclass','Sex','Embarked']  # Pclass diperlakukan kategori (sering lebih bagus)

# Kompatibilitas OneHotEncoder (sklearn lama pakai 'sparse', baru pakai 'sparse_output')
ohe_kwargs = {}
if tuple(int(x) for x in sklearn.__version__.split('.')[:2]) >= (1, 2):
    ohe_kwargs['sparse_output'] = False
else:
    ohe_kwargs['sparse'] = False

num_pipe = SkPipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

cat_pipe = SkPipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('ohe', OneHotEncoder(handle_unknown='ignore', **ohe_kwargs))
])

preprocess = ColumnTransformer(
    transformers=[
        ('num', num_pipe, num_cols),
        ('cat', cat_pipe, cat_cols),
    ],
    remainder='drop'
)

print("Jumlah NaN setelah seleksi fitur:")
print(X.isna().sum())
```

```
Jumlah NaN setelah seleksi fitur:
Pclass       0
Sex          0
Age        177
SibSp        0
Parch        0
Fare         0
Embarked     2
dtype: int64
```

Penjelasan : menyiapkan alur preprocessing yang benar memilih kolom, isi nilai hilang (median/modus), encode kategori (OneHot), scale numerik, dan menyatukannya dalam ColumnTransformer agar aman dipakai di Pipeline (anti error dan anti data leakage saat CV).

5. Baseline Model (Tanpa Resampling)

```python
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

def eval_model(pipeline, X, y, cv):
    scoring = {
        "roc_auc": "roc_auc",
        "f1": "f1",
        "precision": "precision",
        "recall": "recall",
        "balanced_accuracy": "balanced_accuracy",
    }
    scores = cross_validate(pipeline, X, y, cv=cv, scoring=scoring, n_jobs=-1)
    return {k: float(np.mean(v)) for k, v in scores.items() if k.startswith("test_")}

baseline_clf = LogisticRegression(max_iter=500, class_weight=None)
baseline_pipe = SkPipeline(steps=[
    ("preprocess", preprocess),
    ("clf", baseline_clf),
])

baseline_scores = eval_model(baseline_pipe, X, y, cv)
baseline_scores
```

```
{'test_roc_auc': 0.8513981884591237,
 'test_f1': 0.7257926039978063,
 'test_precision': 0.7547281330780817,
 'test_recall': 0.7015771526001705,
 'test_balanced_accuracy': 0.7788452902283588}
```

Penjelasan : Kode itu menjalankan baseline Logistic Regression dengan StratifiedKFold (5-fold), di mana seluruh preprocessing (imputasi, one-hot, scaling) dimasukkan ke dalam Pipeline agar anti-leakage, hasil rata-rata CV

menunjukkan ROC-AUC ≈ 0.85 (pemisahan kelas cukup baik), F1 ≈ 0.73 (keseimbangan presisi recall sudah oke), Precision ≈ 0.75 (prediksi “selamat” cukup tepat) dan Recall ≈ 0.70 (sekitar 70% kasus “selamat” tertangkap), serta Balanced Accuracy ≈ 0.78 (jauh di atas 0.5 pada data imbalanced). Ini menjadi pembanding awal.

6. Eksperimen Resampling (di dalam Pipeline)

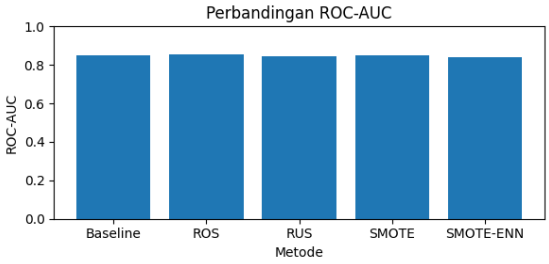
```python
from collections import OrderedDict
from imblearn.over_sampling import RandomOverSampler, SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.combine import SMOTEENN
from imblearn.pipeline import Pipeline  # pastikan dari imblearn

resamplers = OrderedDict([
    ("ROS", RandomOverSampler()),
    ("RUS", RandomUnderSampler()),
    ("SMOTE", SMOTE()),
    ("SMOTE-ENN", SMOTEENN()),
])

results = {"Baseline": baseline_scores}

for name, sampler in resamplers.items():
    pipe = Pipeline(steps=[
        ("preprocess", preprocess),                 # <-- PREPROCESS DULU
        ("resample", sampler),                      # <-- BARU RESAMPLING
        ("clf", LogisticRegression(max_iter=1000)),  # <-- LALU MODEL
    ])
    scores = eval_model(pipe, X, y, cv)
    results[name] = scores

pd.DataFrame(results).T
```

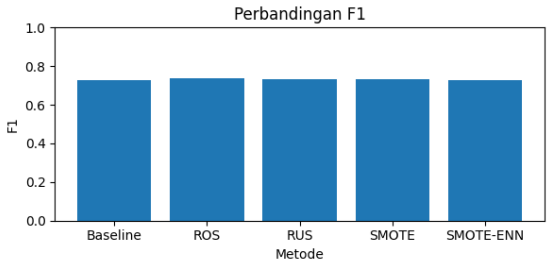
	test_roc_auc	test_f1	test_precision	test_recall	test_balanced_accuracy
Baseline	0.851398	0.725793	0.754728	0.701577	0.778845
ROS	0.852483	0.738450	0.714884	0.765814	0.787269
RUS	0.846675	0.733941	0.708463	0.762958	0.783114
SMOTE	0.850399	0.731900	0.714643	0.751279	0.781828
SMOTE-ENN	0.839899	0.729093	0.697963	0.765772	0.779049

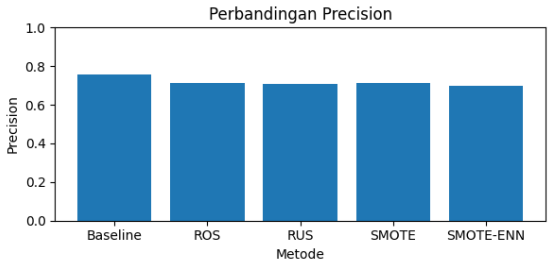
Penjelasan : Resampling membantu dibanding baseline, **ROS** memberi **F1 tertinggi** (≈0.738), **Recall tertinggi** (≈0.766), **Balanced Acc** tertinggi (≈0.787), dan **ROC-AUC** juga terbaik (≈0.852); trade-off-nya **Precision turun** (≈0.715 vs baseline 0.755). **RUS** dan **SMOTE** mendekati ROS tapi sedikit di bawah di AUC/BA. **SMOTE-ENN** menaikkan recall namun paling menurunkan precision dan AUC.

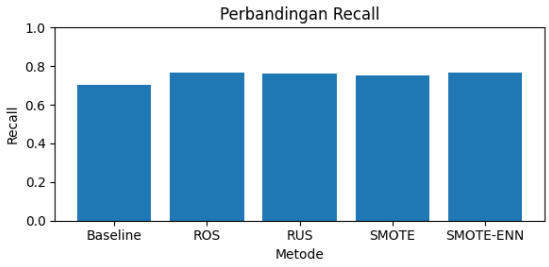
7. Visualisasi Perbandingan Skor

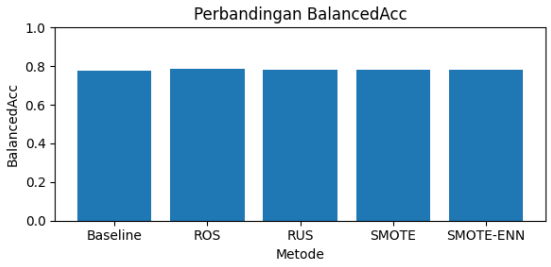
```python
res_df = pd.DataFrame(results).T
metrics = ["test_roc_auc", "test_f1", "test_precision", "test_recall", "test_balanced_accuracy"]
res_df = res_df[metrics]
res_df = res_df.rename(columns={
    "test_roc_auc": "ROC-AUC",
    "test_f1": "F1",
    "test_precision": "Precision",
    "test_recall": "Recall",
    "test_balanced_accuracy": "BalancedAcc",
})
for col in res_df.columns:
    plt.figure(figsize=(6,3))
    plt.bar(res_df.index.astype(str), res_df[col].values)
    plt.title(f"Perbandingan {col}")
    plt.xlabel("Metode")
    plt.ylabel(col)
    plt.ylim(0, 1)
    plt.xticks(rotation=0)
    plt.tight_layout()
    plt.show()
```







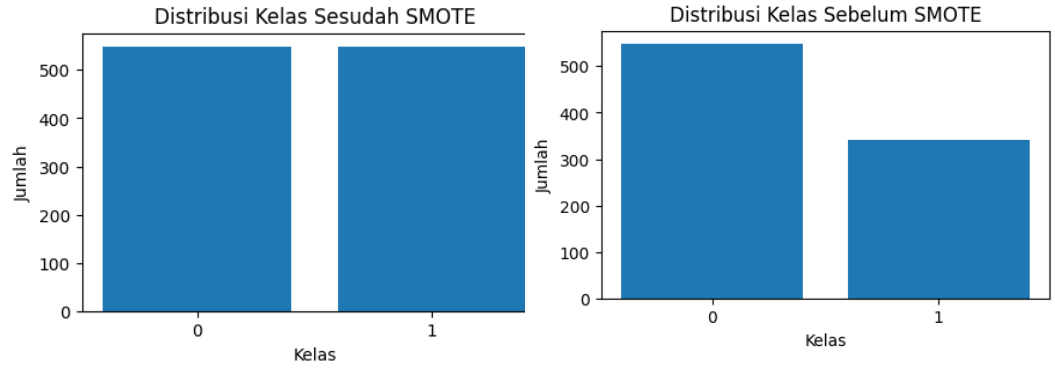




Penjelasan : Grafik membandingkan kinerja tiap metode (Baseline, ROS, RUS, SMOTE, SMOTE-ENN) pada lima metrik. Hasilnya: **ROS** paling konsisten unggul **F1**, **Recall**, dan **Balanced Accuracy** tertinggi, serta **ROC-AUC** setara/teratas; trade-off-nya **Precision** turun dibanding **Baseline**. **RUS** dan **SMOTE** berada di tengah (sedikit di bawah ROS). **SMOTE-ENN** menaikkan recall tetapi **precision** dan **AUC** paling rendah.

8. Distribusi Kelas: Sebelum vs Sesudah (Contoh Satu Metode)

```python
# Ubah X jadi numerik pakai preprocess
Xp = preprocess.fit_transform(X)   # Hanya untuk demo plot

from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
Xp_res, y_res = smote.fit_resample(Xp, y)

before_counts = y.value_counts().sort_index()
after_counts  = pd.Series(y_res).value_counts().sort_index()

import matplotlib.pyplot as plt

plt.figure(figsize=(5,3))
plt.bar(before_counts.index.astype(str), before_counts.values)
plt.title("Distribusi Kelas Sebelum SMOTE"); plt.xlabel("Kelas"); plt.ylabel("Jumlah")
plt.show()

plt.figure(figsize=(5,3))
plt.bar(after_counts.index.astype(str), after_counts.values)
plt.title("Distribusi Kelas Sesudah SMOTE"); plt.xlabel("Kelas"); plt.ylabel("Jumlah")
plt.show()
```



Penjelasan : Grafik itu menunjukkan efek **SMOTE** pada kelas target terlebih dulu mengubah fitur menjadi numerik dengan Xp = preprocess.fit_transform(X) (SMOTE butuh numerik), lalu SMOTE(random_state=42).fit_resample(Xp, y) menambah sampel sintetis untuk **kelas minoritas (1)** hingga seimbang dengan **kelas mayoritas (0)**. Hasilnya, chart **Sebelum** tampak timpang (kelas 0 > 1), sedangkan **Sesudah SMOTE** kedua kelas ~setara. Catatan: langkah ini dibuat untuk **visualisasi**; untuk evaluasi model yang benar, lakukan di **Pipeline: preprocess → resample → model** dalam **cross-validation** agar tidak terjadi data leakage.s