Nama : Firman Ramadhan Saputra

NIM : 231011400891

Kelas : 05TPLE015

Mata Kuliah : Machine Learning

LAPORAN LEMBAR KERJA PERTEMUAN 5 — MODELING

1. **Muat Data**

   Melanjutkan hasil pada pertemuan 4, Dimana dataset telah diproses menjadi **processed_kelulusan.csv**, analisis dilanjutkan dengan memuat file tersebut dan melakukan split ulang menjadi tiga subset data, yaitu training, validation, dan testing. Setelah pembagian data berhasil dilakukan, selanjutnya adalah membangun **baseline model** menggunakan **Logistic Regression** dengan pipeline preprocessing. Berikut dengan contoh codenya:

```
# Opsi B (Pakai processed_kelulusan.csv lalu split ulang)

import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("processed_kelulusan.csv")
X = df.drop("Lulus", axis=1)
y = df["Lulus"]

X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)

print(f"Berikut adalah hasil SPlit ulang: \n", X_train.shape, X_val.shape, X_test.shape)
```

   Outputnya:

```
Berikut adalah hasil SPlit ulang:
 (53, 5) (12, 5) (12, 5)
```

   Output (53, 5) (12, 5) (12, 5) menunjukkan bahwa dataset berhasil dibagi menjadi train, validation, dan test set sesuai proporsi (sekitar 70% : 15% : 15%), dengan masing-masing subset memiliki 5 fitur.

2. **Baseline Model dan Pipeline**

Bangun baseline terstandar menggunakan Logistic Regression dan Pipeline Preprocessing, dengan cara berikut:

```python
# Langkah 2 - Baseline model dan Pipeline
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, classification_report

num_cols = X_train.select_dtypes(include="number").columns

pre = ColumnTransformer([
    ("num", Pipeline([("imp", SimpleImputer(strategy="median")),
                      ("sc", StandardScaler())]), num_cols),
], remainder="drop")

logreg = LogisticRegression(max_iter=1000, class_weight="balanced", random_state=42)
pipe_lr = Pipeline([("pre", pre), ("clf", logreg)])

pipe_lr.fit(x_train, y_train)
y_val_pred = pipe_lr.predict(x_val)
print("Berikut hasil dari Baseline model dan Pipeline \nBaseline (logReg) F1(val): ", f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

Berikut sedikit penjelasan dari code:

Penjelasan Code

- **Pipeline dibuat** untuk preprocessing data:
    - SimpleImputer(strategy="median") → mengisi nilai kosong dengan median.
    - StandardScaler() → menstandarkan fitur numerik agar memiliki distribusi rata-rata 0 dan standar deviasi 1.
- **Logistic Regression** digunakan sebagai model baseline dengan parameter:
    - max_iter =1000 → jumlah iterasi maksimum.
    - class_weight = "balanced" → menyeimbangkan kelas agar tidak bias.
- **Training** dilakukan dengan pipe_lr.fit(x_train, y_train) menggunakan data training.
- **Evaluasi** dilakukan pada **validation set** (x_val, y_val) menggunakan:
    - **F1-score** (macro average).
    - **classification_report** (precision, recall, f1-score, dan support).

Dengan outputnya berikut:

```
Berikut hasil dari Baseline model dan Pipeline
Baseline (LogReg) F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         6
           1      1.000     1.000     1.000         6

    accuracy                         1.000        12
   macro avg      1.000     1.000     1.000        12
weighted avg      1.000     1.000     1.000        12
```

Berikut Penjelasanya:

- F1(val) = 1.0 → artinya model sempurna dalam memprediksi data pada validation set.
- precision = 1.0 → semua prediksi kelas (0 maupun 1) benar, tidak ada false positive.
- recall = 1.0 → semua data aktual kelas (0 maupun 1) berhasil diprediksi, tidak ada false negative.
- support = 6 per kelas (total 12) → ada 6 sampel untuk kelas 0 dan 6 sampel untuk kelas 1 di validation set.
- accuracy = 1.0 → model memprediksi semua sampel dengan benar, tanpa kesalahan sama sekali.

3. **Model Alternatif (Random Forest)**

Setelah membangun baseline dengan Logistic Regression pada langkah sebelumnya, selanjutnya pada Langkah 3 dilakukan pengujian menggunakan model alternatif Random Forest untuk melihat apakah performanya dapat memberikan hasil yang lebih baik dibanding baseline. Berikut dengan codenya:

```python
# Langkah 3 - Model Alternatif (Random Forest)
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=300, max_features="sqrt", class_weight="balanced", random_state=42
)
pipe_rf = Pipeline([("pre", pre), ("clf", rf)])

pipe_rf.fit(X_train, y_train)
y_val_rf = pipe_rf.predict(X_val)
print("RandomForest F1(val):", f1_score(y_val, y_val_rf, average="macro"))
```

Dengan outputnya sebagai berikut:

```
RandomForest F1(val): 1.0
```

Hasil dari **Random Forest** menunjukkan **F1-score = 1.0**, sama persis dengan **baseline Logistic Regression** sebelumnya. Artinya, pada dataset ini kedua model berhasil memprediksi data validasi dengan sempurna.

4. Validasi Silang & Tuning Ringkas

Setelah memperoleh hasil awal yang sama antara baseline Logistic Regression dan model alternatif Random Forest, pada Langkah 4 dilakukan proses validasi silang dan tuning hyperparameter untuk mengoptimalkan performa model Random Forest sehingga dapat memberikan hasil yang lebih stabil dan dapat diandalkan.

```python
# Langkah 4 - Validasi Silang dan Tuning Ringkas
from sklearn.model_selection import StratifiedKFold, GridSearchCV

skf = StratifiedKFold(n_splits=2, shuffle=True, random_state=42)
param = {
    "clf__max_depth": [None, 12, 20, 30],
    "clf__min_samples_split": [2, 5, 10]
}
gs = GridSearchCV(pipe_rf, param_grid=param, cv=skf,
                  scoring="f1_macro", n_jobs=1, verbose=1)
gs.fit(x_train, y_train)
print("Best param: ", gs.best_params_)
print("Best CV F1: ", gs.best_score_)

best_rf = gs.best_estimator_
y_val_best = best_rf.predict(x_val)
print("Best RF F1(val): \n", f1_score(y_val, y_val_best, average="macro"))
```

Dengan output:

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 1.0
Best RF F1(val): 1.0
```

Hasil validasi silang dengan **n_splits=2** menghasilkan nilai **F1-score sempurna (1.0)** baik pada proses *cross-validation* maupun data validasi. Hal ini memang wajar mengingat ukuran dataset yang sangat kecil, dengan distribusi kelas yang seimbang dan pola pemisahan fitur yang relatif sederhana, sehingga model mampu memprediksi dengan benar tanpa kesalahan. Namun demikian, nilai sempurna ini tidak serta-merta

menjamin performa yang sama pada data baru dengan jumlah lebih besar dan variasi lebih kompleks. Oleh karena itu, hasil ini lebih mencerminkan keterbatasan dataset ketimbang kemampuan model secara umum.

5. **Evaluasi Akhir (Test Set)**

Setelah proses validasi silang dan penyetelan hyperparameter menghasilkan model terbaik, tahap selanjutnya adalah **evaluasi akhir menggunakan data uji**. Langkah ini bertujuan untuk menilai sejauh mana model dapat melakukan prediksi pada data baru yang belum pernah dilihat sebelumnya. Evaluasi dilakukan dengan berbagai metrik seperti **F1-score, classification report, confusion matrix**, serta **ROC-AUC** jika model mendukung probabilitas prediksi, guna memperoleh gambaran performa model secara menyeluruh.

```python
# Langkah 5 — Evaluasi Akhir (Test Set)
from sklearn.metrics import confusion_matrix, roc_auc_score, precision_recall_curve, roc_curve
import matplotlib.pyplot as plt

final_model = best_rf  # atau pipe_lr jika baseline lebih baik
y_test_pred = final_model.predict(X_test)

print("F1(test):", f1_score(y_test, y_test_pred, average="macro"))
print(classification_report(y_test, y_test_pred, digits=3))
print("Confusion matrix (test):")
print(confusion_matrix(y_test, y_test_pred))

# ROC-AUC (jika ada predict_proba)
if hasattr(final_model, "predict_proba"):
    y_test_proba = final_model.predict_proba(X_test)[:,1]
    try:
        print("ROC-AUC(test):", roc_auc_score(y_test, y_test_proba))
    except:
        pass
    fpr, tpr, _ = roc_curve(y_test, y_test_proba)
    plt.figure(); plt.plot(fpr, tpr); plt.xlabel("FPR"); plt.ylabel("TPR"); plt.title("ROC (test)")
    plt.tight_layout(); plt.savefig("roc_test.png", dpi=120)
```

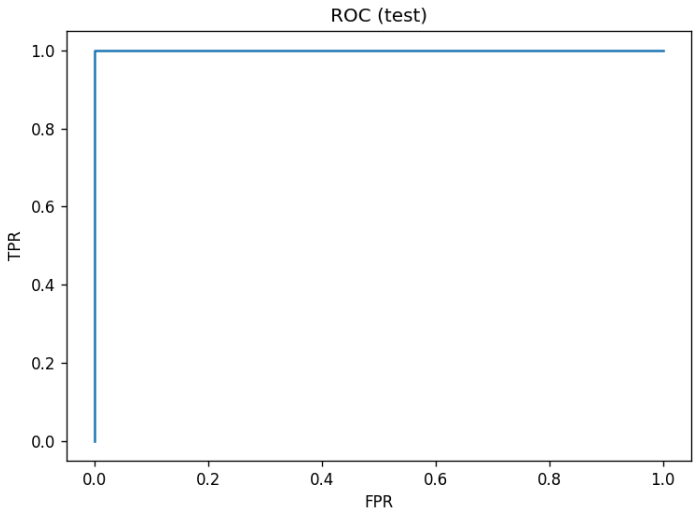
Dengan outputnya sebagai berikut:

```
F1(test): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         6
           1      1.000     1.000     1.000         6

    accuracy                          1.000        12
   macro avg      1.000     1.000     1.000        12
weighted avg      1.000     1.000     1.000        12

Confusion matrix (test):
[[6 0]
 [0 6]]
ROC-AUC(test): 1.0
```

Dari hasil pengujian atau pengetesan pada data, dapat dilihat bahwa:

- F1(test) = 1.0 → artinya model mampu melakukan prediksi dengan sempurna pada data uji.
- Pada classification report, terdapat kelas 0 dan 1 masing-masing dengan support = 6 (total 12 data) → menunjukkan data uji seimbang antara kedua kelas.
- Precision, Recall, dan F1-score untuk kelas 0 dan 1 semuanya = 1.0 → berarti model tidak melakukan kesalahan klasifikasi sama sekali pada kedua kelas.
- Confusion matrix:
    - Baris pertama [6 0] → ada 6 data kelas 0, semuanya diprediksi benar sebagai kelas 0.
    - Baris kedua [0 6] → ada 6 data kelas 1, semuanya diprediksi benar sebagai kelas 1.
- ROC-AUC (test) = 1.0 → menunjukkan kemampuan model dalam membedakan kelas sangat sempurna (tidak ada overlap antara prediksi kelas 0 dan 1).

Berikut dengan output dari Grafik ROC (test):



- Garis ROC pada grafik hampir menempel pada sisi kiri atas (melewati titik (0,1)), yang menunjukkan performa model sangat baik dalam membedakan kelas positif dan negatif.
- TPR (True Positive Rate) = 1.0 → artinya semua data positif terdeteksi dengan benar oleh model.

- FPR (False Positive Rate) = 0.0 → tidak ada data negatif yang salah diprediksi menjadi positif.
- Area Under Curve (AUC) = 1.0 → menggambarkan model memiliki kemampuan klasifikasi yang sempurna pada data uji ini.