LAPORAN LEMBAR KERJA PERTEMUAN 5

```
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("Baseline (LogReg) F1(val):", f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

```
Baseline (LogReg) F1(val): 1.0
              precision    recall  f1-score   support

           1      1.000     1.000     1.000         1

    accuracy                         1.000         1
   macro avg      1.000     1.000     1.000         1
weighted avg      1.000     1.000     1.000         1
```

```
# 1️⃣ Import library
import pandas as pd
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report

# 2️⃣ Buat dataset contoh (sesuaikan dengan CSV kamu)
data = {
    "IPK": [3.8, 2.5, 3.4, 2.1, 3.9, 3.2, 2.8, 3.7],
    "Jumlah_Absensi": [3, 8, 4, 12, 2, 5, 7, 3],
    "Waktu_Belajar_Jam": [10, 5, 7, 2, 12, 8, 6, 9],
    "Lulus": [1, 0, 1, 0, 1, 1, 0, 1]
}

df = pd.DataFrame(data)

# 3️⃣ Pisahkan fitur dan target
X = df[['IPK', 'Jumlah_Absensi', 'Waktu_Belajar_Jam']]
y = df['Lulus']

# 4️⃣ Split data (train 70%, test 30%)
# Hilangkan stratify kalau dataset sangat kecil
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
```

```python
# 6️⃣ StratifiedKFold untuk GridSearch
# Gunakan n_splits=2 agar sesuai dataset kecil
skf = StratifiedKFold(n_splits=2, shuffle=True, random_state=42)

# 7️⃣ Parameter Grid
param = {
    "clf__max_depth": [None, 5, 10],
    "clf__min_samples_split": [2, 3]
}

# 8️⃣ GridSearchCV
gs = GridSearchCV(
    pipe_rf, param_grid=param, cv=skf,
    scoring="f1_macro", n_jobs=-1, verbose=1
)

# 9️⃣ Fit model
gs.fit(X_train, y_train)

# 1️⃣0️⃣ Hasil terbaik
print("Best params:", gs.best_params_)
print("Best CV F1:", gs.best_score_)

# 1️⃣1️⃣ Prediksi & evaluasi
y_pred = gs.predict(X_test)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Fitting 2 folds for each of 6 candidates, totalling 12 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```

```python
from sklearn.metrics import f1_score, confusion_matrix

# Pakai model terbaik dari GridSearchCV
final_model = gs.best_estimator_

# Prediksi data test
y_test_pred = final_model.predict(X_test)

# Evaluasi
print("F1 Score (test):", f1_score(y_test, y_test_pred, average="macro"))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
```

```
F1 Score (test): 1.0
Confusion Matrix:
 [[1 0]
 [0 2]]
```

```python
import joblib
joblib.dump(final_model, "model.pkl")
print("Model tersimpan ke model.pkl")
```

```
Model tersimpan ke model.pkl
```