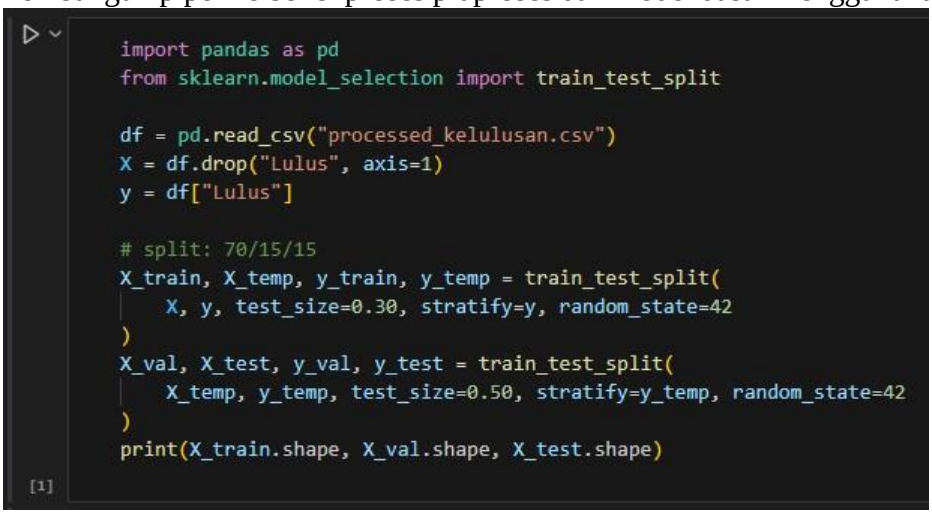
Nama : Zulyan Widyaka K
NIM : 231011403446

Perteman-6

Muat Data
Dataset hasil pembersihan pada pertemuan sebelumnya (processed_kelulusan.csv) dimuat kembali dan dibagi ulang menjadi data latih dan data uji. Tujuan pembagian ulang ini untuk menjaga keseimbangan data dan menghindari kebocoran informasi agar hasil evaluasi model lebih akurat dan objektif.

Pipeline s Baseline Random Forest

Tahapan ini membangun pipeline berisi proses praproses dan model dasar menggunakan algoritma



Random Forest. Pipeline memastikan seluruh tahapan pelatihan dan pengujian berlangsung konsisten serta mencegah terjadinya *data leakage*. Model baseline berfungsi sebagai acuan awal sebelum dilakukan proses optimasi.

```
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))
```
[2]

Validasi Silang

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(pipe, X_train, y_train, cv=skf, scoring="f1_macro", n_jobs=-1)
print("CV F1-macro (train):", scores.mean(), "±", scores.std())
```
[3]

Dilakukan validasi silang untuk menilai performa model secara menyeluruh. Teknik ini melatih dan menguji model pada beberapa bagian data agar hasil penilaian lebih stabil dan tidak bergantung pada satu pembagian data saja.

Tuning Ringkas

```
from sklearn.model_selection import GridSearchCV

param = {
  "clf__max_depth": [None, 12, 20, 30],
  "clf__min_samples_split": [2, 5, 10]
}

gs = GridSearchCV(pipe, param_grid=param, cv=skf,
                  scoring="f1_macro", n_jobs=-1, verbose=1)
gs.fit(X_train, y_train)
print("Best params:", gs.best_params_)
best_model = gs.best_estimator_
y_val_best = best_model.predict(X_val)
print("Best RF — F1(val):", f1_score(y_val, y_val_best, average="macro"))
```
[4]

Tahap tuning dilakukan untuk mencari kombinasi parameter terbaik agar model bekerja lebih optimal. Beberapa parameter penting seperti jumlah pohon, kedalaman maksimum, dan jumlah fitur yang digunakan diuji agar performa model meningkat tanpa menyebabkan *overfitting*.

Evaluasi Akhir (Test Set)

```python
from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve, precision_recall_curve
import matplotlib.pyplot as plt

final_model = best_model  # pilih terbaik; jika baseline lebih baik, gunakan pipe

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 (bila 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)

    prec, rec, _ = precision_recall_curve(y_test, y_test_proba)
    plt.figure(); plt.plot(rec, prec); plt.xlabel("Recall"); plt.ylabel("Precision"); plt.title("PR Curve (test)")
    plt.tight_layout(); plt.savefig("pr_test.png", dpi=120)
```

[5]

Model yang sudah dituning diuji pada data yang belum pernah digunakan selama pelatihan. Evaluasi dilakukan menggunakan metrik seperti *accuracy*, *precision*, *recall*, dan *F1-score* untuk melihat seberapa baik model memprediksi data baru secara nyata.

Simpan Model

```python
import joblib
joblib.dump(final_model, "rf_model.pkl")
print("Model disimpan sebagai rf_model.pkl")
```

[7]

Model yang telah dilatih dan diuji disimpan dalam format file seperti .pkl agar dapat digunakan kembali tanpa perlu melatih ulang. Hal ini juga memudahkan proses implementasi ke dalam sistem prediksi selanjutnya.

Cek Inference Lokal

Untuk mengecek apakah model yang sudah dilatih dan disimpan bisa melakukan prediksi dengan benar di komputer lokal, menggunakan data input contoh (fiktif).