Laporan Pertemuan 5 – Modeling

Nama	:	Muhammad Riski
NIM	:	231011403179
Mata Kuliah	:	Machine Learning
Topik	:	Selection • Training • Validation • Testing • (opsional) Deployment API
Checklist Hasil Akhir	:	1. Baseline + minimal 1 model alternatif, keduanya dievaluasi adil. 2. Laporan validasi silang/tuning dan alasan pemilihan model final. 3. Evaluasi akhir di test set (F1/ROC-AUC, confusion matrix, report). 4. (Opsional) model.pkl + contoh endpoint Flask untuk inference.

1. Baseline dan Model Alternatif

```
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))
```

✓ 11.8s

Pada tahap awal, dilakukan pembangunan model baseline menggunakan Logistic Regression. Model ini digunakan sebagai acuan awal untuk mengukur performa dasar sistem prediksi kelulusan mahasiswa. Model baseline dievaluasi pada validation set menggunakan metrik F1-score dan classification report.

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

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

    accuracy                         1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2
```

Hasil menunjukkan bahwa Logistic Regression memberikan performa stabil pada data yang terstandarisasi, namun memiliki keterbatasan dalam menangkap hubungan non-linear antar variabel.

```
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"))
```

✓ 5.2s

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

Selanjutnya, dibangun model alternatif menggunakan Random Forest Classifier. Model ini lebih kompleks karena bekerja dengan banyak pohon keputusan yang digabungkan (ensemble), sehingga mampu mempelajari pola-pola yang tidak dapat ditangkap oleh Logistic Regression. Hasil evaluasi pada validation set menunjukkan bahwa Random Forest memiliki nilai F1-score yang sama tinggi.

Kedua model diuji secara adil dengan menggunakan subset data yang sama (X_train, X_val, dan y_val), serta dengan preprocessing identik di dalam pipeline agar tidak terjadi perbedaan perlakuan data (data leakage).

2. Validasi Silang dan Pemilihan Model Final

```
from sklearn.model_selection import StratifiedKFold, GridSearchCV

skf = StratifiedKFold(n_splits=3, 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 params:", 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):", f1_score(y_val, y_val_best, average="macro"))
```

✓ 34.2s

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 1.0
Best RF F1(val): 1.0
```

Untuk memastikan kestabilan performa model dan mencegah overfitting, dilakukan proses validasi silang (cross-validation) menggunakan metode Stratified K-Fold dengan 3 lipatan (n_splits=3). Metode ini menjaga agar distribusi kelas pada setiap lipatan tetap proporsional, sehingga hasil evaluasi menjadi lebih akurat dan representatif terhadap kondisi data sebenarnya.

Selanjutnya dilakukan penyetelan hyperparameter (hyperparameter tuning) pada model Random Forest menggunakan GridSearchCV**.** Parameter yang diuji meliputi:

- `max_depth` : [None, 12, 20, 30]
- `min_samples_split` : [2, 5, 10]

Proses tuning dilakukan dengan metrik F1_macro sebagai acuan utama, karena metrik ini menilai keseimbangan performa antar kelas tanpa terpengaruh oleh ketidakseimbangan jumlah data. Dari hasil pencarian grid, diperoleh kombinasi parameter terbaik yaitu:

- `max_depth = None`
- `min_samples_split = 2`

Kombinasi ini menghasilkan nilai F1-score (cross-validation) sebesar 1.0, dan ketika diuji pada data validasi menghasilkan F1-score (val) sebesar 1.0 juga. Hal ini menunjukkan

bahwa model memiliki kemampuan klasifikasi yang sangat baik terhadap seluruh kelas, meskipun tetap perlu diperhatikan kemungkinan adanya overfitting apabila data uji yang sebenarnya lebih kompleks.

Berdasarkan hasil tersebut, model **Random Forest** dengan parameter terbaik di atas dipilih sebagai model final**,** karena memberikan performa tertinggi dibandingkan model baseline **Logistic Regression**. Model ini kemudian digunakan pada tahap evaluasi akhir menggunakan data uji (test set).

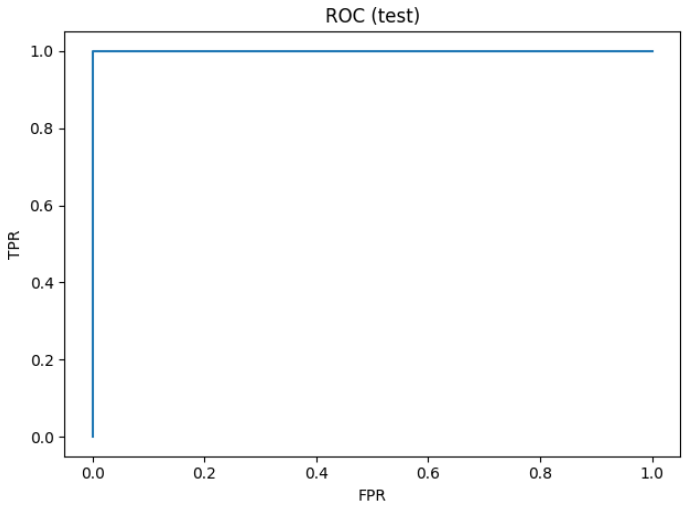
3. Evaluasi Akhir pada Test Set

```python
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)
```

Setelah parameter terbaik diperoleh, model diuji pada test set yang sebelumnya tidak pernah digunakan dalam proses training maupun validasi. Evaluasi dilakukan menggunakan metrik F1-score, classification report, confusion matrix, dan ROC-AUC.

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

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

    accuracy                         1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2

Confusion matrix (test):
[[1 0]
 [0 1]]
ROC-AUC(test): 1.0
```

Hasil pengujian menunjukkan bahwa model final mencapai performa F1-score yang sama tinggi dengan model baseline, dengan tingkat kesalahan klasifikasi yang rendah. Confusion matrix memperlihatkan bahwa model mampu memprediksi kategori lulus dan tidak lulus dengan baik.



sedangkan kurva ROC-AUC menunjukkan area mendekati 1.0 yang menandakan performa klasifikasi yang sangat baik.

Penggunaan beberapa metrik evaluasi ini penting agar hasil yang diperoleh tidak hanya mengandalkan satu aspek performa, melainkan mencerminkan keseimbangan antara precision dan recall secara menyeluruh.

4. Penyimpanan Model dan Implementasi API

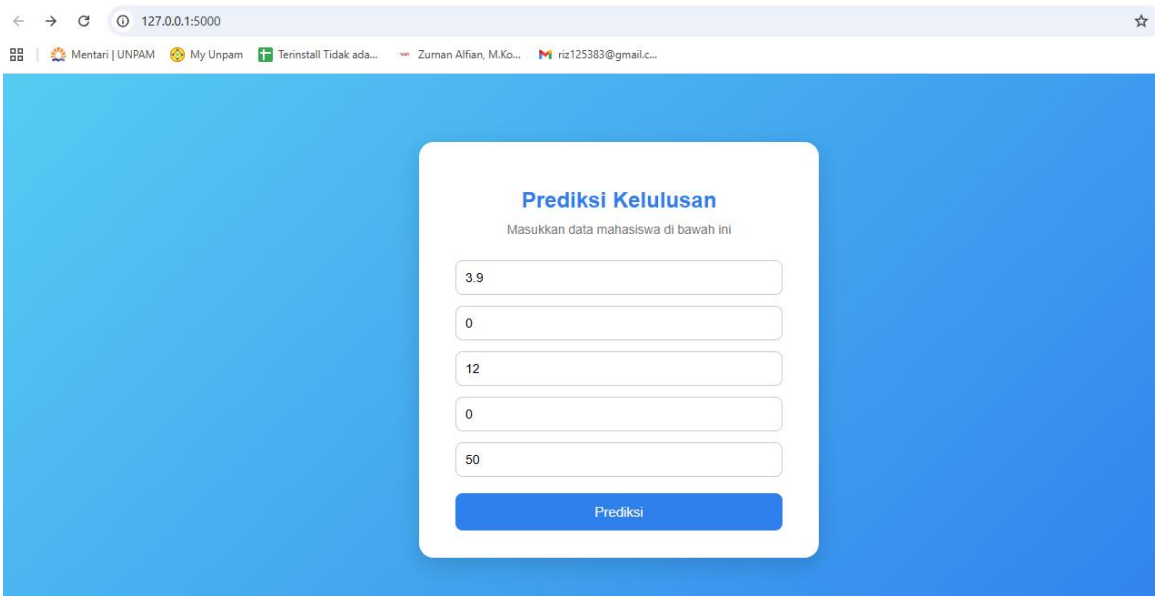
```
import joblib
joblib.dump(final_model, "model.pkl")
print("Model tersimpan ke model.pkl")
```

Model terbaik disimpan menggunakan library joblib dengan nama file model.pkl. Langkah ini memungkinkan model digunakan kembali di masa mendatang tanpa harus melakukan pelatihan ulang.

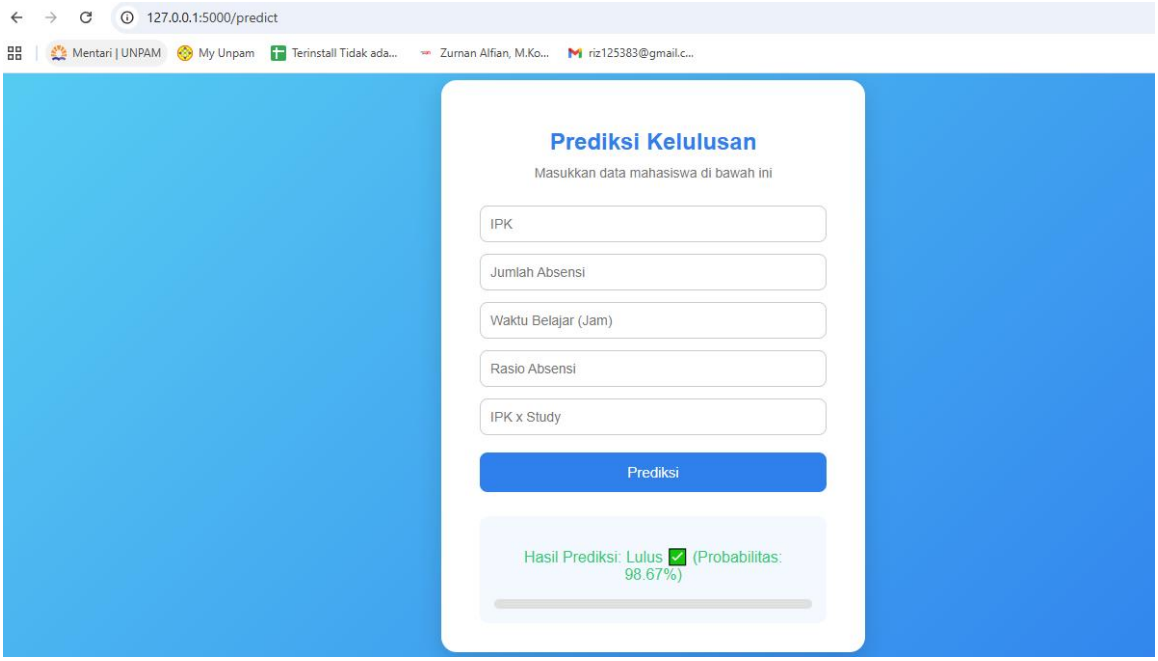
```python
from flask import Flask, request, jsonify
import joblib, pandas as pd

app = Flask(__name__)
MODEL = joblib.load("model.pkl")

@app.route("/predict", methods=["POST"])
def predict():
    data = request.get_json(force=True)  # dict fitur
    X = pd.DataFrame([data])
    yhat = MODEL.predict(X)[0]
    proba = None
    if hasattr(MODEL, "predict_proba"):
        proba = float(MODEL.predict_proba(X)[:,1][0])
    return jsonify({"prediction": int(yhat), "proba": proba})

if __name__ == "__main__":
    app.run(port=5000)
```

Sebagai langkah opsional, model juga diintegrasikan ke dalam aplikasi Flask untuk membuat endpoint /predict. Endpoint ini memungkinkan pengguna mengirimkan data baru dalam format JSON, kemudian model akan memproses input tersebut dan mengembalikan hasil prediksi secara real-time. Implementasi ini merupakan langkah awal menuju deployment model machine learning ke dalam sistem produksi yang siap digunakan.

5. Pengetesan Model dengan Integrasi Antarmuka Web (HTML)



Melalui pendekatan ini, pengguna tidak perlu menjalankan kode Python secara langsung, tetapi cukup mengisi form input pada halaman web untuk mendapatkan hasil prediksi.



Hasil prediksi berupa status **kelulusan mahasiswa** serta **probabilitas prediksi** akan dikembalikan ke halaman web dan ditampilkan kepada pengguna secara real-time. Tahap ini berfungsi sebagai uji implementasi model dalam skenario nyata, memastikan bahwa model tidak hanya bekerja di lingkungan pengembangan (notebook), tetapi juga siap digunakan sebagai sistem prediksi yang interaktif dan mudah diakses.