

## LAPORAN PERTEMUAN 6 – RANDOM FOREST UNTUK KLASIFIKASI

**Nama** : Farhan Rachmad Rizki  
**NIM** : 231011400893  
**Kelas** : 05TPLE015  
**Mata Kuliah** : Machine Learning

---

### Pendahuluan

Pada pertemuan ini dilakukan analisis menggunakan algoritma Random Forest Classifier untuk melakukan klasifikasi kelulusan mahasiswa berdasarkan beberapa variabel penting seperti IPK, jumlah absensi, dan waktu belajar per minggu.

Dataset yang digunakan adalah `processed_kelulusan.csv`, hasil dari pengolahan dataset sebelumnya.

Tahapan percobaan mencakup:

- Pembagian dataset menjadi train, validation, dan test,
- Pembuatan model Random Forest,
- Validasi silang dan tuning hyperparameter,
- Evaluasi performa model (F1 Score, ROC Curve, Confusion Matrix),
- Menentukan feature importance,
- Menyimpan model ke dalam file `.pkl` dan melakukan uji prediksi sederhana.

Hasil yang diharapkan:

- File `rf_model.pkl` berisi model terbaik,
  - Evaluasi model dengan performa tinggi,
  - Visualisasi ROC dan PR Curve,
  - Informasi fitur yang paling berpengaruh dalam menentukan kelulusan.
- 

### Langkah 1 – Membaca Dataset dan Membagi Data

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

# split: 70/15/15
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=42
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.50, stratify=y_temp, random_state=42
)
print(X_train.shape, X_val.shape, X_test.shape)
```

#### Penjelasan:

Dataset dibagi menjadi tiga bagian untuk memastikan proses pelatihan, validasi, dan pengujian berjalan seimbang tanpa *overfitting*.

---

## Langkah 2 – Preprocessing Data

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

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

### Penjelasan:

Proses ini mengisi nilai kosong (imputasi median) dan menstandarkan data numerik agar memiliki skala yang sama sebelum dimasukkan ke model.

---

## Langkah 3 – Pelatihan Model Awal (Baseline Random Forest)

```
rf = RandomForestClassifier(
    n_estimators=300, max_features="sqrt",
    class_weight="balanced", random_state=42
)

pipe = Pipeline([("pre", pre), ("clf", rf)])
pipe.fit(X_train, y_train)

y_val_pred = pipe.predict(X_val)
print("Baseline RF - F1(val):", f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

### Penjelasan:

Model baseline digunakan untuk mengukur performa awal sebelum dilakukan tuning parameter.

Hasilnya digunakan sebagai pembanding terhadap model yang sudah dioptimalkan.

---

## Langkah 4 – Validasi Silang (Cross Validation)

```
skf = StratifiedKFold(n_splits=3, 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())
```

### Penjelasan:

Validasi silang membantu memastikan model memiliki performa stabil dan tidak hanya cocok untuk data tertentu.

---

## Langkah 5 – Pencarian Parameter Terbaik (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"))
```

### Penjelasan:

GridSearchCV digunakan untuk menemukan kombinasi parameter terbaik agar model lebih optimal dan tidak overfitting.

---

## Langkah 6 – Evaluasi Akhir pada Data Testing

```
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))
```

### Penjelasan:

Tahapan ini menguji kemampuan model pada data yang belum pernah dilihat sebelumnya. Hasilnya menunjukkan seberapa baik model dapat menggeneralisasi data baru.

---

## Langkah 7 – ROC dan PR Curve

```
# 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)
```

### Penjelasan:

ROC-AUC mengukur kemampuan model dalam membedakan antara dua kelas (lulus atau tidak).

Nilai mendekati **1.0** menandakan kinerja yang sangat baik.

---

## Langkah 8 – Feature Importance

```
import numpy as np
importances = final_model.named_steps["clf"].feature_importances_
fn = final_model.named_steps["pre"].get_feature_names_out()
top = sorted(zip(fn, importances), key=lambda x: x[1], reverse=True)
print("Top feature importance:")
for name, val in top[:10]:
    print(f"{name}: {val:.4f}")
```

### Penjelasan:

Tahap ini menampilkan fitur yang paling mempengaruhi hasil klasifikasi. Biasanya, **IPK** dan **Waktu Belajar** menjadi variabel dominan.

---

## Langkah 9 – Penyimpanan Model dan Prediksi

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

# Contoh sekali jalan (input fiktif), sesuaikan nama kolom:
mdl = joblib.load("rf_model.pkl")
sample = pd.DataFrame([
    {"IPK": 3.4,
     "Jumlah_Absensi": 4,
     "Waktu_Belajar_Jam": 7,
     "Rasio_Absensi": 4/14,
     "IPK_x_Study": 3.4*7
    }])
print("Prediksi:", int(mdl.predict(sample)[0]))
```

---

### Penjelasan:

Model terbaik disimpan ke file .pkl agar bisa digunakan kembali tanpa perlu dilatih ulang. Kemudian dilakukan uji coba prediksi pada data fiktif mahasiswa.

---

# Hasil Akhir

Output hasil eksekusi program:

```
(9, 5) (2, 5) (3, 5)
Baseline RF - 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

CV F1-macro (train): 0.7777777777777777 ± 0.15713484026367724
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF - F1(val): 1.0
F1(test): 1.0
      precision    recall   f1-score   support
          0       1.000     1.000     1.000       2
          1       1.000     1.000     1.000       1

accuracy                           1.000       3
macro avg       1.000     1.000     1.000       3
weighted avg    1.000     1.000     1.000       3

Confusion Matrix (test):
[[2 0]
 [0 1]]
ROC-AUC(test): 1.0
Top feature importance:
num_IPK: 0.4207
num_Waktu_Belajar_Jam: 0.2541
num_IPK_x_Study: 0.2394
num_Rasio_Absensi: 0.0515
num_Jumlah_Absensi: 0.0344
Model disimpan sebagai rf_model.pkl
Prediksi: 1
```

---