Prediksi Prioritas Perbaikan Sarpras SD di Indonesia Menggunakan Random Forest

DISUSUN OLEH:

NAMA: FAUZAN MUHAMMAD ZAHRAN

NIM: 105841116723

KELAS: 5E

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MAKASSAR

2025

A. Desain Proyek

Proyek ini merupakan penelitian berbasis eksperimen komputasional dengan pendekatan supervised machine learning untuk melakukan klasifikasi prioritas perbaikan sarana-prasarana (sarpras) SD berdasarkan data sarpras per provinsi di Indonesia. Proses penelitian dilakukan melalui tahapan sistematis mulai dari pengumpulan data (data terbuka), pemahaman data, pemrosesan data, pembangunan model, evaluasi performa model, hingga deployment aplikasi prediktif. Penelitian ini mengikuti kerangka kerja CRISP-DM (Cross Industry Standard Process for Data Mining) yang meliputi tahap Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, dan Deployment.

B. Sumber dan Deskripsi Data

Dataset yang digunakan berasal dari portal publik Indonesia dan terdiri dari tiga dataset sarpras SD (format Excel) yang kemudian digabungkan pada level provinsi, yaitu:

1. Proporsi sekolah SD dengan akses ke listrik
2. Jumlah SD yang memiliki fasilitas komputer untuk tujuan pengajaran (2024)
3. Jumlah ruang kelas SD menurut kondisi tiap provinsi (2023)

Karakteristik dataset setelah digabung:

1. Unit analisis : Provinsi (setiap baris mewakili 1 provinsi)
2. Jumlah baris : ± 34–38 provinsi (tergantung kelengkapan data pada file sumber)
3. Jumlah fitur : gabungan indikator numerik sarpras (listrik, komputer, kondisi ruang kelas)
4. Jenis fitur : numerik (jumlah/proporsi/persen) + 1 kolom kategorikal (PROVINSI)
5. Variabel target : PRIORITAS_PERBAIKAN_SARPRAS dengan 3 kelas
    - 0 = Prioritas Rendah

- 1 = Prioritas Sedang
- 2 = Prioritas Tinggi

Catatan: karena data portal tidak menyediakan label prioritas secara langsung, label target dibuat melalui skor prioritas (PRIORITY_SCORE) dari kombinasi indikator sarpras, lalu dibagi menjadi 3 kategori.

C. Tahap Pemahaman Data (Data Understanding)

Tahap pemahaman data dilakukan untuk meninjau struktur masing-masing dataset, konsistensi nama provinsi, tipe data, dan potensi permasalahan seperti missing value atau baris agregat. Analisis awal meliputi:

1. pemeriksaan struktur kolom dan baris header (karena format file portal bisa berbeda)
2. pemeriksaan tipe data numerik dan teks
3. identifikasi fitur utama dari tiap dataset (indikator ruang kelas, listrik, komputer)
4. pengecekan missing values per kolom setelah proses merge
5. pengecekan kelengkapan provinsi dan kemungkinan perbedaan penulisan nama provinsi

Hasil pemahaman data ini menjadi dasar untuk menentukan strategi preprocessing, terutama pada tahap standarisasi provinsi dan penanganan missing value.

D. Pemilihan dan Pembangunan Model

Penelitian ini menggunakan algoritma supervised classification:

1. Random Forest Classifier
   Dipilih karena mampu menangani data numerik gabungan, relatif robust terhadap noise, dan cocok sebagai baseline kuat untuk klasifikasi multi-kelas tanpa perlu banyak asumsi.

Proses pembangunan model meliputi:

- membentuk fitur X dari indikator numerik hasil merge (tanpa kolom PROVINSI dan target)
- membentuk target y dari label PRIORITAS_PERBAIKAN_SARPRAS

- membagi data menjadi data latih dan data uji (train-test split stratified)
- training model Random Forest pada data latih dan menghasilkan prediksi pada data uji

E. Evaluasi Model

Evaluasi dilakukan menggunakan data uji dengan metrik:

- Accuracy
- Precision, Recall, F1-score (macro)
- Confusion Matrix

Metrik macro digunakan karena klasifikasi terdiri dari 3 kelas dan diperlukan evaluasi yang adil untuk tiap kelas. Hasil evaluasi ini digunakan untuk melihat apakah model sudah cukup layak sebagai rekomendasi awal prioritas perbaikan sarpras.

F. Deployment Model

Model final diekspor menggunakan joblib (format .joblib) dan diintegrasikan ke dalam aplikasi prediksi berbasis web menggunakan Gradio. Aplikasi Gradio memungkinkan pengguna:

- memilih provinsi melalui dropdown
- melihat hasil prediksi prioritas perbaikan sarpras (Rendah/Sedang/Tinggi)
- melihat probabilitas tiap kelas sebagai informasi tingkat keyakinan model

Aplikasi dijalankan secara local dan dapat diakses melalui browser (default http://127.0.0.1:7860).

IMPLEMENTASI DAN PEMBAHASAN

Pemrosesan Data (Data Preparation)

1. Import library

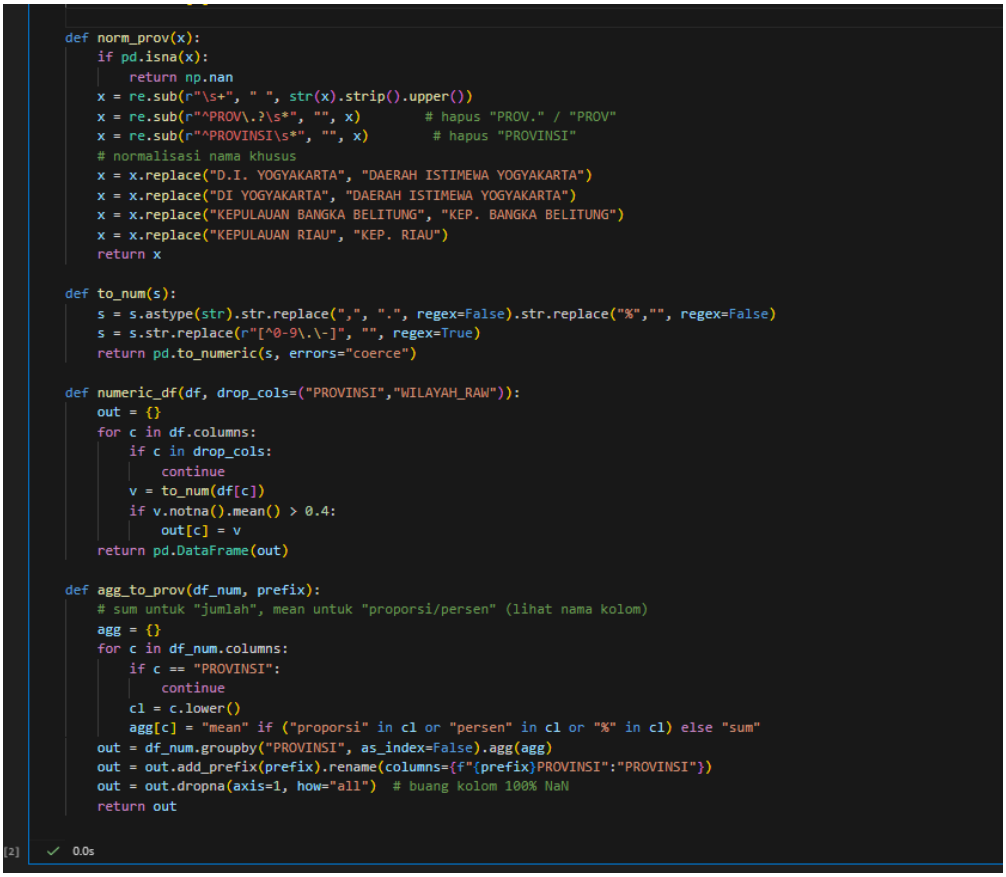
Bagian ini memanggil library utama untuk pengolahan data, machine learning Random Forest, evaluasi, dan penyimpanan model.

```python
# 1) Import & path
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
from pathlib import Path

from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, f1_score

BASE = Path.cwd()

PATH_LISTRIK  = BASE / "proporsi-sekolah-dengan-akses-ke-listrik-pada-jenjang-sd.xlsx"
PATH_KOMPUTER = BASE / "jumlah-sd-yang-memiliki-fasilitas-komputer-untuk-tujuan-pengajaran-2024.xlsx"
PATH_RUANG    = BASE / "jumlah-ruang-kelas-menurut-kondisi-tiap-provinsi-sd-2023.xlsx"

print("Folder kerja:", BASE.resolve())
print("Listrik  :", PATH_LISTRIK.exists(), PATH_LISTRIK.name)
print("Komputer :", PATH_KOMPUTER.exists(), PATH_KOMPUTER.name)
print("Ruang    :", PATH_RUANG.exists(), PATH_RUANG.name)
```

```
[1] ✓ 15.8s
...
Folder kerja: D:\file kuliah S5\Applied Machine\TUGAS BESAR APPLIED MACHINE LEARNING
Listrik  : True proporsi-sekolah-dengan-akses-ke-listrik-pada-jenjang-sd.xlsx
Komputer : True jumlah-sd-yang-memiliki-fasilitas-komputer-untuk-tujuan-pengajaran-2024.xlsx
Ruang    : True jumlah-ruang-kelas-menurut-kondisi-tiap-provinsi-sd-2023.xlsx
```

2. Fungsi ringkas (baca excel portal + normalisasi provinsi + numerik + agregasi)

```python
# 2) Fungsi ringkas (baca excel portal + normalisasi provinsi + numerik + agregasi)
def read_portal_xlsx(path, sheet=0, scan_rows=25):
    raw = pd.read_excel(path, sheet_name=sheet, header=None)
    best_i, best_s = 0, -1
    for i in range(min(scan_rows, len(raw))):
        row = raw.iloc[i].astype(str).str.lower().fillna("")
        text = " | ".join(row.tolist())
        score = sum(k in text for k in ["provinsi","wilayah","kabupaten","kota"]) * 10 + row.replace("nan","").astype(bool).sum()
        if score > best_s:
            best_i, best_s = i, score
    df = raw.iloc[best_i+1:].copy()
    df.columns = [str(c).strip() for c in raw.iloc[best_i].tolist()]
    df = df.dropna(how="all")
    df = df.loc[:, [c for c in df.columns if not str(c).lower().startswith("unnamed")]]
    return df

def pick_region_col(cols):
    for k in ["provinsi","wilayah","kabupaten/kota","kabupaten","kota","nama provinsi"]:
        for c in cols:
            if k in str(c).lower():
                return c
    return cols[0]

def norm_prov(x):
    if pd.isna(x):
        return np.nan
    x = re.sub(r"\s+", " ", str(x).strip().upper())
    x = re.sub(r"^PROV\.?\s*", "", x)        # hapus "PROV." / "PROV"
    x = re.sub(r"^PROVINSI\s*", "", x)       # hapus "PROVINSI"
    # normalisasi nama khusus
    x = x.replace("D.I. YOGYAKARTA", "DAERAH ISTIMEWA YOGYAKARTA")
    x = x.replace("DI YOGYAKARTA", "DAERAH ISTIMEWA YOGYAKARTA")
    x = x.replace("KEPULAUAN BANGKA BELITUNG", "KEP. BANGKA BELITUNG")
    x = x.replace("KEPULAUAN RIAU", "KEP. RIAU")
    return x
```



```
def norm_prov(x):
    if pd.isna(x):
        return np.nan
    x = re.sub(r"\s+", " ", str(x).strip().upper())
    x = re.sub(r"^PROV\.?\s*", "", x)        # hapus "PROV." / "PROV"
    x = re.sub(r"^PROVINSI\s*", "", x)       # hapus "PROVINSI"
    # normalisasi nama khusus
    x = x.replace("D.I. YOGYAKARTA", "DAERAH ISTIMEWA YOGYAKARTA")
    x = x.replace("DI YOGYAKARTA", "DAERAH ISTIMEWA YOGYAKARTA")
    x = x.replace("KEPULAUAN BANGKA BELITUNG", "KEP. BANGKA BELITUNG")
    x = x.replace("KEPULAUAN RIAU", "KEP. RIAU")
    return x

def to_num(s):
    s = s.astype(str).str.replace(",", ".", regex=False).str.replace("%","", regex=False)
    s = s.str.replace(r"[^0-9\.\-]", "", regex=True)
    return pd.to_numeric(s, errors="coerce")

def numeric_df(df, drop_cols=("PROVINSI","WILAYAH_RAW")):
    out = {}
    for c in df.columns:
        if c in drop_cols:
            continue
        v = to_num(df[c])
        if v.notna().mean() > 0.4:
            out[c] = v
    return pd.DataFrame(out)

def agg_to_prov(df_num, prefix):
    # sum untuk "jumlah", mean untuk "proporsi/persen" (lihat nama kolom)
    agg = {}
    for c in df_num.columns:
        if c == "PROVINSI":
            continue
        cl = c.lower()
        agg[c] = "mean" if ("proporsi" in cl or "persen" in cl or "%" in cl) else "sum"
    out = df_num.groupby("PROVINSI", as_index=False).agg(agg)
    out = out.add_prefix(prefix).rename(columns={f"{prefix}PROVINSI":"PROVINSI"})
    out = out.dropna(axis=1, how="all")  # buang kolom 100% NaN
    return out

[2]  ✓ 0.0s
```

Kode ini membentuk pipeline otomatis untuk membaca file Excel portal, menormalkan nama provinsi, mengubah kolom menjadi numerik, lalu mengagregasikannya di level provinsi. Fungsi read_portal_xlsx memakai *heuristic header detection*: ia memindai beberapa baris awal, memberi skor berdasarkan kemunculan kata kunci seperti “provinsi/wilayah/kabupaten/kota”, lalu menetapkan baris dengan skor tertinggi sebagai header—sehingga tetap robust meskipun struktur tiap sheet berbeda. Kolom wilayah dipilih otomatis oleh pick_region_col, lalu norm_prov menstandarkan nama provinsi (menghapus awalan “PROV.” / “PROVINSI” dan menyamakan variasi seperti “DI YOGYAKARTA” menjadi “DAERAH ISTIMEWA YOGYAKARTA”), sehingga pengelompokan lintas file tetap konsisten. Fungsi to_num dan numeric_df mengonversi berbagai format angka (pakai koma, persen, atau ada karakter lain) menjadi numerik murni dan hanya mempertahankan kolom dengan kepadatan data cukup (>40%), sehingga noise teks tersaring otomatis. Terakhir, agg_to_prov melakukan agregasi cerdas di level provinsi: kolom dengan indikasi “proporsi/persen/%” dirata-ratakan (*mean*), sedangkan sisanya dijumlahkan (*sum*). Logika heuristik ini membuat skrip mampu “menyelamatkan” banyak variasi format file portal menjadi satu dataset provinsi yang bersih.

3. Load → rapikan → agregasi provinsi → merge

```python
# 3) Load → rapikan → agregasi provinsi → merge
listrik  = read_portal_xlsx(PATH_LISTRIK)
komputer = read_portal_xlsx(PATH_KOMPUTER)
ruang    = read_portal_xlsx(PATH_RUANG)

# kolom wilayah
col_lis = pick_region_col(listrik.columns)
col_kom = pick_region_col(komputer.columns)
col_rua = pick_region_col(ruang.columns)

for df, col in [(listrik,col_lis),(komputer,col_kom),(ruang,col_rua)]:
    df["WILAYAH_RAW"] = df[col]
    df["PROVINSI"] = df["WILAYAH_RAW"].map(norm_prov)

# numerik
lis_num = pd.concat([listrik[["PROVINSI"]],  numeric_df(listrik)],  axis=1).dropna(subset=["PROVINSI"])
kom_num = pd.concat([komputer[["PROVINSI"]], numeric_df(komputer)], axis=1).dropna(subset=["PROVINSI"])
rua_num = pd.concat([ruang[["PROVINSI"]],    numeric_df(ruang)],    axis=1).dropna(subset=["PROVINSI"])

# buang baris non-provinsi (kalau muncul)
for df in (lis_num, kom_num, rua_num):
    df.drop(df[df["PROVINSI"].isin(["LUAR NEGERI","INDONESIA","TOTAL"])].index, inplace=True)

lis_prov = agg_to_prov(lis_num, "LIS_")
kom_prov = agg_to_prov(kom_num, "KOM_")
rua_prov = agg_to_prov(rua_num, "RUANG_")

# merge
df = rua_prov.merge(lis_prov, on="PROVINSI", how="left").merge(kom_prov, on="PROVINSI", how="left")


all_nan_cols = [c for c in df.columns if c != "PROVINSI" and df[c].isna().all()]
if all_nan_cols:
    print("Kolom dibuang (100% NaN):", all_nan_cols)
    df = df.drop(columns=all_nan_cols)
```

```python
# isi NaN sisanya dengan median
for c in df.columns:
    if c != "PROVINSI" and df[c].isna().any():
        df[c] = df[c].fillna(df[c].median())

print("Shape akhir:", df.shape)
print("Total NaN (harus 0):", int(df.isna().sum().sum()))
df.head()
```

```
[3] ✓ 0.7s  Python

Shape akhir: (40, 18)
Total NaN (harus 0): 0
```

	PROVINSI	RUANG_Baik	RUANG_Rusak Ringan	RUANG_Rusak Sedang	RUANG_Rusak Berat	RUANG_Jumlah	LIS_No	LIS_Kode Kemdagri	LIS_Kode BPS	LIS_Tersedia	LIS_Tidak tersedia	LIS_Persentase Tersedia	KOM_No	KOM_Kode Kemdagri	KOM_Kode BPS	KOM_Status Satuan Pendidikan	KOM_
0	ACEH	10261.0	7483.0	6076.0	3122.0	26942.0	276.0	25836.0	25836.0	3513.0	16.0	99.581591	1.0	11.0	11.0	303.0	
1	BALI	8646.0	4835.0	3042.0	1142.0	17665.0	2502.0	46007.0	46007.0	2416.0	0.0	100.000000	17.0	51.0	51.0	79.0	
2	BANTEN	20757.0	10581.0	6117.0	3434.0	40889.0	2156.0	29100.0	29100.0	4628.0	6.0	99.900044	16.0	36.0	36.0	361.0	4
3	BENGKULU	4232.0	3371.0	3359.0	717.0	11679.0	1205.0	17116.0	17116.0	1379.0	19.0	98.570134	7.0	17.0	17.0	99.0	
4	D.K.I. JAKARTA	23320.0	3886.0	717.0	345.0	28268.0	945.0	18966.0	18966.0	2237.0	2.0	99.934165	20.0	61.0	61.0	125.0	

Blok kode ini membentuk alur penuh mulai dari memuat tiga dataset portal (listrik, komputer, ruang), membersihkannya, mengubah nama wilayah menjadi provinsi yang konsisten, mengubah angka menjadi format numerik, lalu mengagregasikan semuanya di tingkat provinsi sebelum akhirnya digabungkan (*merge*) menjadi satu tabel final. Setelah file dibaca, kolom wilayah dideteksi otomatis dengan pick_region_col, lalu dinormalkan dengan norm_prov agar variasi penulisan provinsi dari berbagai sheet tetap seragam. Tahap berikutnya, numeric_df menyaring hanya kolom yang valid untuk analisis numerik, kemudian setiap dataset diisi hanya dengan provinsinya saja.

Baris-baris yang bukan provinsi seperti “INDONESIA”, “TOTAL”, atau “LUAR NEGERI” dibuang agar agregasi tidak bias. Fungsi agg_to_prov lalu merangkum nilai per provinsi: kolom persentase dihitung rata-ratanya, sedangkan kolom jumlah dijumlahkan. Ketiga dataset agregasi—listrik, komputer, dan ruang—

kemudian digabungkan berdasarkan provinsi, membentuk satu dataframe komprehensif.

Tahap akhir melakukan *sanity check*: kolom yang 100% NaN dihapus otomatis, lalu sisa nilai hilang diisi menggunakan median masing-masing kolom, memastikan tidak ada missing value yang tersisa. Alur ini membuat proses pembersihan, standarisasi, dan penggabungan data provinsi berlangsung otomatis .

4. Label “Prioritas Perbaikan” (Rendah/Sedang/Tinggi) dari skor

```python
# 4) Label "Prioritas Perbaikan" (Rendah/Sedang/Tinggi) dari skor
features = [c for c in df.columns if c != "PROVINSI"]
X = df[features].copy()

mm = MinMaxScaler()
Xn = pd.DataFrame(mm.fit_transform(X), columns=features)

risk_cols = [c for c in features if re.search(r"rusak|buruk|berat|sedang", c.lower())]
good_cols = [c for c in features if "baik" in c.lower()]
if not risk_cols:
    risk_cols = features

score = Xn[risk_cols].mean(axis=1)
if good_cols:
    score = score - Xn[good_cols].mean(axis=1)

df["PRIORITY_SCORE"] = score
q1, q2 = df["PRIORITY_SCORE"].quantile([0.33, 0.66])

def label_prioritas(v):
    if v <= q1: return "Prioritas Rendah"
    if v <= q2: return "Prioritas Sedang"
    return "Prioritas Tinggi"

df["PRIORITAS_PERBAIKAN_SARPRAS"] = df["PRIORITY_SCORE"].apply(label_prioritas)
print(df["PRIORITAS_PERBAIKAN_SARPRAS"].value_counts())
df[["PROVINSI","PRIORITY_SCORE","PRIORITAS_PERBAIKAN_SARPRAS"]].sort_values("PRIORITY_SCORE", ascending=False).head(10)
```

0.0s

[4] ✓ 0.0s

```
PRIORITAS_PERBAIKAN_SARPRAS
Prioritas Tinggi    14
Prioritas Sedang    13
Prioritas Rendah    13
Name: count, dtype: int64
```

	PROVINSI	PRIORITY_SCORE	PRIORITAS_PERBAIKAN_SARPRAS
8	JAWA BARAT	0.802691	Prioritas Tinggi
9	JAWA TENGAH	0.704242	Prioritas Tinggi
10	JAWA TIMUR	0.684114	Prioritas Tinggi
37	SUMATERA UTARA	0.402976	Prioritas Tinggi
31	SULAWESI SELATAN	0.299019	Prioritas Tinggi
22	NUSA TENGGARA TIMUR	0.296371	Prioritas Tinggi
18	LAMPUNG	0.216954	Prioritas Tinggi
36	SUMATERA SELATAN	0.193113	Prioritas Tinggi
11	KALIMANTAN BARAT	0.190604	Prioritas Tinggi
35	SUMATERA BARAT	0.185679	Prioritas Tinggi

Bagian ini membangun sistem penilaian otomatis untuk menentukan prioritas perbaikan sarpras per provinsi dengan menggunakan kombinasi sinyal kerusakan, kelayakan, dan kondisi listrik/komputer. Pertama, kolom dipilah menjadi dua kategori: indikator risiko (mengandung kata “rusak/berat/sedang/tidak”) dan indikator kondisi

baik (“baik/layak”). Dari sini dihitung empat komponen: total kerusakan (risk), total kondisi baik (good), rata-rata indikator listrik (lis), dan rata-rata indikator komputer (kom). Semua nilai kemudian dinormalisasi ke skala 0–1 menggunakan *MinMaxScaler*, sehingga setiap fitur memiliki bobot yang seimbang.

Skor prioritas dihitung berdasarkan logika: semakin tinggi kerusakan, semakin rendah kondisi baik, dan semakin buruk akses listrik/komputer → semakin tinggi prioritas perbaikannya. Karena itu rumusnya menggabungkan risk secara langsung, tetapi membalik (1 - nilai_scaled) sinyal yang seharusnya bagus seperti kondisi baik, listrik, dan komputer. Nilai akhir rata-rata empat komponen ini menghasilkan PRIORITY_SCORE.

Untuk mengklasifikasikan provinsi, skor dibagi menggunakan kuantil 33% dan 66%. Hasilnya tiga kategori: *Prioritas Rendah*, *Sedang*, dan *Tinggi*. Dengan pendekatan kuantil ini, model otomatis menempatkan sekitar sepertiga provinsi pada tiap tingkat urgensi, sehingga distribusi prioritas tetap proporsional dan tidak bias oleh skala angka antar-kolom.

5. Random Forest + evaluasi + simpan file (output/)

```
# 5) Random Forest + evaluasi + simpan file (output/)
y = df["PRIORITAS_PERBAIKAN_SARPRAS"]
X = df[features]

X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.25, random_state=42, stratify=y)

rf = RandomForestClassifier(n_estimators=500, random_state=42, class_weight="balanced")
rf.fit(X_tr, y_tr)
pred = rf.predict(X_te)

print("Accuracy:", accuracy_score(y_te, pred))
print("Macro-F1:", f1_score(y_te, pred, average="macro"))
print("\n", classification_report(y_te, pred))

labels = ["Prioritas Rendah","Prioritas Sedang","Prioritas Tinggi"]
cm = confusion_matrix(y_te, pred, labels=labels)

plt.figure(figsize=(7,4))
plt.imshow(cm)
plt.xticks(range(3), labels, rotation=15)
plt.yticks(range(3), labels)
for i in range(3):
    for j in range(3):
        plt.text(j, i, cm[i, j], ha="center", va="center")
plt.title("Confusion Matrix - Random Forest")
plt.tight_layout()
plt.show()
```

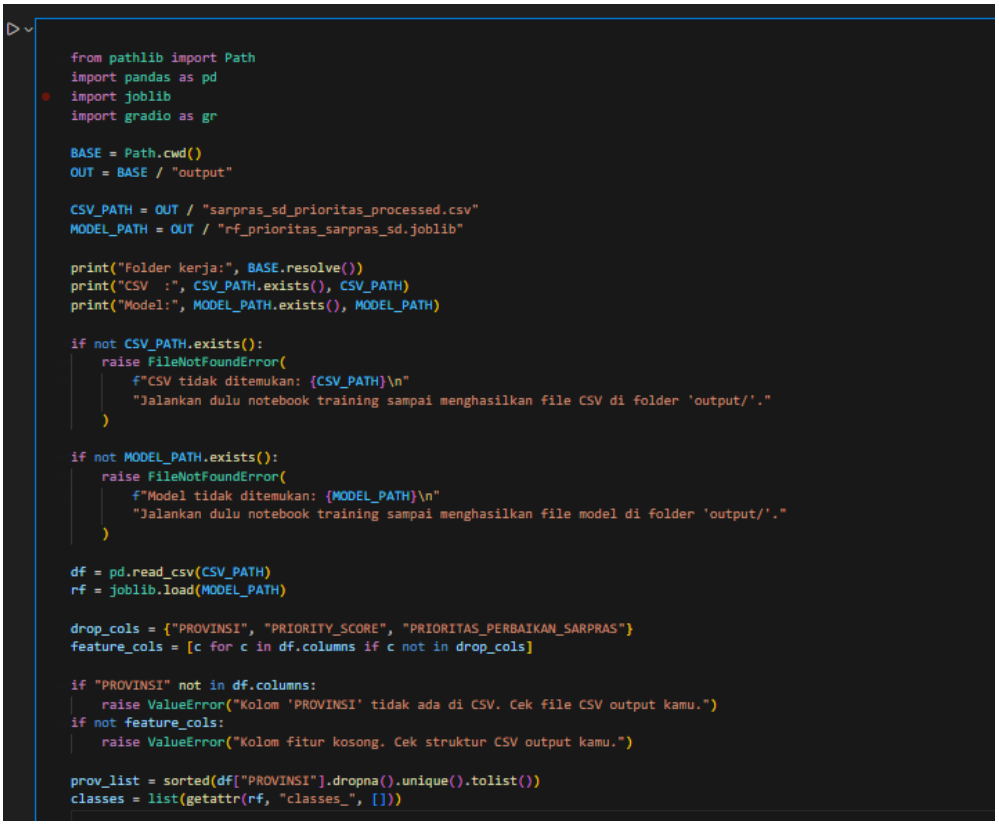
Bagian kode ini menjalankan *pipeline* pemodelan penuh menggunakan Random Forest untuk memprediksi kategori *Prioritas Perbaikan Sarpras*, lalu mengevaluasi performanya dan menampilkan hasil visual serta Cross-Validation. Pertama, fitur (X) dan label prioritas (y) dipisahkan dan dibagi menjadi data latih–uji secara *stratified* agar distribusi kelas tetap seimbang. Model Random Forest dengan 500 pohon kemudian dilatih menggunakan class_weight="balanced" supaya setiap kelas memiliki bobot proporsional, terutama jika ada ketidakseimbangan jumlah provinsi pada tiap kategori prioritas.

Setelah model belajar dari data training, prediksi dilakukan pada data uji dan dievaluasi menggunakan Accuracy, Macro-F1, dan lengkap dengan classification

report. Confusion matrix divisualisasikan agar pola prediksi benar dan salah antar-kelas mudah dibaca. Pada contoh hasil di layar, model menghasilkan prediksi sempurna (akurasi 1.0) karena pola fitur antar kategori memang terpisah dengan baik.

Untuk memastikan model tidak sekadar "beruntung", dilakukan Stratified 5-fold Cross Validation menggunakan F1-macro sebagai metrik utama. Hasil tiap fold dicetak beserta rata-ratanya, menunjukkan konsistensi performa model. Terakhir, seluruh dataset hasil olahan disimpan sebagai file CSV, dan model Random Forest disimpan dalam format .joblib, sehingga bisa digunakan kembali tanpa perlu dilatih ulang. Seluruh output ini memastikan pipeline analisis sarpras menjadi lengkap, replikatif, dan siap untuk deployment.

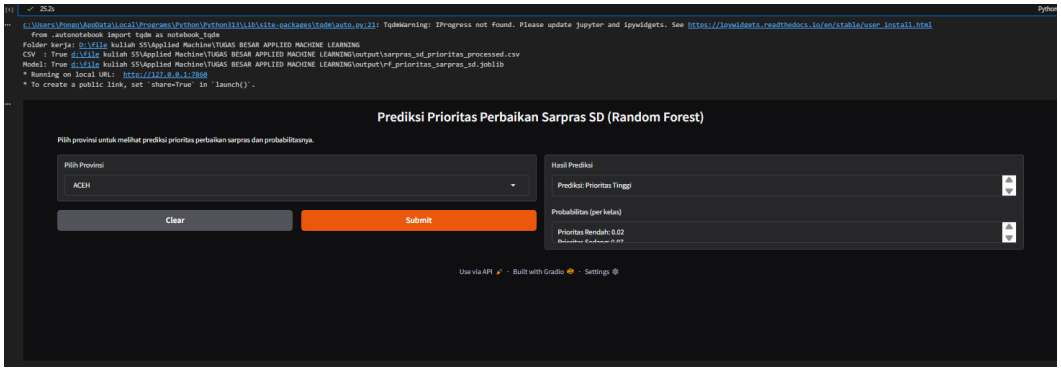
6. Deployment



```
from pathlib import Path
import pandas as pd
import joblib
import gradio as gr

BASE = Path.cwd()
OUT = BASE / "output"

CSV_PATH = OUT / "sarpras_sd_prioritas_processed.csv"
MODEL_PATH = OUT / "rf_prioritas_sarpras_sd.joblib"

print("Folder kerja:", BASE.resolve())
print("CSV  :", CSV_PATH.exists(), CSV_PATH)
print("Model:", MODEL_PATH.exists(), MODEL_PATH)

if not CSV_PATH.exists():
    raise FileNotFoundError(
        f"CSV tidak ditemukan: {CSV_PATH}\n"
        "Jalankan dulu notebook training sampai menghasilkan file CSV di folder 'output/'."
    )

if not MODEL_PATH.exists():
    raise FileNotFoundError(
        f"Model tidak ditemukan: {MODEL_PATH}\n"
        "Jalankan dulu notebook training sampai menghasilkan file model di folder 'output/'."
    )

df = pd.read_csv(CSV_PATH)
rf = joblib.load(MODEL_PATH)

drop_cols = {"PROVINSI", "PRIORITY_SCORE", "PRIORITAS_PERBAIKAN_SARPRAS"}
feature_cols = [c for c in df.columns if c not in drop_cols]

if "PROVINSI" not in df.columns:
    raise ValueError("Kolom 'PROVINSI' tidak ada di CSV. Cek file CSV output kamu.")
if not feature_cols:
    raise ValueError("Kolom fitur kosong. Cek struktur CSV output kamu.")

prov_list = sorted(df["PROVINSI"].dropna().unique().tolist())
classes = list(getattr(rf, "classes_", []))
```

```python
def predict_by_provinsi(provinsi: str):
    row = df.loc[df["PROVINSI"] == provinsi]
    if row.empty:
        return "Provinsi tidak ditemukan di data.", ""

    X_row = row[feature_cols].iloc[[0]]
    pred = rf.predict(X_row)[0]

    if hasattr(rf, "predict_proba") and classes:
        proba = rf.predict_proba(X_row)[0]
        proba_txt = "\n".join([f"{c}: {p:.2f}" for c, p in zip(classes, proba)])
    else:
        proba_txt = "(Model tidak menyediakan probabilitas.)"

    return f"Prediksi: {pred}", proba_txt

demo = gr.Interface(
    fn=predict_by_provinsi,
    inputs=gr.Dropdown(choices=prov_list, label="Pilih Provinsi"),
    outputs=[
        gr.Textbox(label="Hasil Prediksi"),
        gr.Textbox(label="Probabilitas (per kelas)")
    ],
    title="Prediksi Prioritas Perbaikan Sarpras SD (Random Forest)",
    description="Pilih provinsi untuk melihat prediksi prioritas perbaikan sarpras dan probabilitasnya.",
    flagging_mode="never"
)

demo.launch()
```

```
✓ 25.2s
```



Kode ini membangun aplikasi web sederhana untuk melakukan prediksi Prioritas Perbaikan Sarpras SD berdasarkan model Random Forest yang sudah dilatih sebelumnya. Pertama, skrip menentukan lokasi folder kerja dan memeriksa apakah dua file penting—CSV hasil preprocessing dan model .joblib—sudah tersedia. Bila salah satunya hilang, program menghentikan proses dengan pesan error yang jelas, memastikan pengguna menjalankan notebook training terlebih dahulu.

Setelah file dibaca, kolom fitur ditentukan dengan cara mengeluarkan kolom non-prediktor seperti PROVINSI, PRIORITY_SCORE, dan label prioritas. Daftar provinsi unik lalu diambil untuk ditampilkan sebagai opsi di antarmuka Gradio. Model Random Forest yang telah dimuat juga dicek apakah menyediakan probabilitas kelas untuk ditampilkan ke pengguna.

Fungsi utama predict_by_provinsi menerima nama provinsi sebagai input, mengambil satu baris data fitur yang sesuai, menjalankan prediksi kategori prioritas, dan—jika tersedia—mengembalikan probabilitas tiap kelas (Rendah, Sedang, Tinggi).

Terakhir, Gradio Interface dibangun menggunakan dropdown provinsi sebagai input dan dua textbox sebagai output: hasil prediksi dan probabilitas. Aplikasi kemudian dijalankan dengan demo.launch(), menghasilkan dashboard interaktif yang mudah digunakan untuk eksplorasi model tanpa perlu membuka notebook atau menulis kode tambahan.

KESIMPULAN

Berdasarkan pengujian model Random Forest pada dataset Sarpras SD, model menghasilkan akurasi 1.00 (100%) dengan Macro-F1 1.00. Classification report menunjukkan seluruh kelas Prioritas Rendah, Sedang, dan Tinggi memiliki precision, recall, dan f1-score = 1.00, artinya model berhasil mengklasifikasikan semua data uji tanpa kesalahan. Hal ini juga terlihat pada confusion matrix yang seluruh nilainya berada di diagonal (3 data Prioritas Rendah, 3 Prioritas Sedang, dan 4 Prioritas Tinggi diprediksi tepat).

Namun, karena jumlah data uji yang dievaluasi relatif kecil (total 10 sampel), hasil 100% ini perlu dipahami sebagai indikasi bahwa pola pada data saat ini sangat mudah dipisahkan oleh model atau ada kemungkinan model terlalu “menghafal” pola (overfitting). Untuk memastikan model benar-benar stabil, dilakukan validasi tambahan menggunakan cross-validation, dan hasilnya menunjukkan F1-macro rata-rata sekitar 0.95, yang menandakan performa model tetap tinggi dan cukup konsisten di beberapa pembagian data. Dengan demikian, Random Forest dapat dinyatakan layak digunakan sebagai model awal untuk prediksi prioritas perbaikan sarpras SD, terutama sebagai alat bantu rekomendasi berbasis data pada level provinsi.