

## DAFTAR PUSTAKA

- [1] S. Sandiwarno, “Penerapan Machine Learning Untuk Prediksi Bencana Banjir,” *Jurnal Sistem Informasi Bisnis*, vol. 14, no. 1, hlm. 62–76, Jan 2024, doi: 10.21456/vol14iss1pp62-76.
- [2] F. Dendi Saputra, “Perubahan Iklim Bukan Lagi Isu Global, BMKG Tegaskan Dampaknya Nyata,” BMKG. Diakses: 16 Juli 2025. [Daring]. Tersedia pada: <https://www.bmkg.go.id/berita/perubahan-iklim-bukan-lagi-isu-global-bmkg-tegaskan-dampaknya-nyata>
- [3] CNN Indonesia, “BMKG Ungkap Korelasi Peningkatan Suhu dan Intensitas Cuaca Ekstrem,” CNN Indonesia. Diakses: 16 Juli 2025. [Daring]. Tersedia pada: <https://www.cnnindonesia.com/teknologi/20250324134313-641-1212407/bmkg-ungkap-korelasi-peningkatan-suhu-dan-intensitas-cuaca-ekstrem>
- [4] H. Akbar dan W. K. Sanjaya, “Kajian Performa Metode Class Weight Random Forest pada Klasifikasi Imbalance Data Kelas Curah Hujan,” *Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi*, vol. 3, no. 1, Des 2023, doi: 10.20885/snati.v3i1.30.
- [5] A. Syahreza, N. K. Ningrum, dan M. A. Syahrazy, “Perbandingan Kinerja Model Prediksi Cuaca: Random Forest, Support Vector Regression, dan XGBoost,” *Edumatic: Jurnal Pendidikan Informatika*, vol. 8, no. 2, hlm. 526–534, Des 2024, doi: 10.29408/edumatic.v8i2.27640.
- [6] A. Mosavi, P. Ozturk, dan K. W. Chau, “Flood prediction using machine learning models: Literature review,” 27 Oktober 2018, *MDPI AG*. doi: 10.3390/w10111536.
- [7] BMKG, “Data Online BMKG,” Badan Meteorologi, Klimatologi, dan Geofisika. Diakses: 10 Maret 2025. [Daring]. Tersedia pada: <https://dataonline.bmkg.go.id/dataonline-home>
- [8] Z. A. Dwiyanti dan C. Prianto, “Prediksi Cuaca Kota Jakarta Menggunakan Metode Random Forest,” *Jurnal Tekno Insentif*, vol. 17, no. 2, hlm. 127–137, Okt 2023, doi: 10.36787/jti.v17i2.1136.
- [9] A. M. Siregar, “Klasifikasi Untuk Prediksi Cuaca Menggunakan Esemble Learning,” *PETIR*, vol. 13, no. 2, hlm. 138–147, Sep 2020, doi: 10.33322/petir.v13i2.998.

- [10] S. Putramulyo dan S. Alaa, "Prediksi Curah Hujan Bulanan Di Kota Samarinda Menggunakan Persamaan Regresi Dengan Prediktor Data Suhu dan Kelembapan Udara," vol. 1, no. 2, Des 2018, Diakses: 10 Mei 2025. [Daring]. Tersedia pada: <http://eigen.unram.ac.id>
- [11] G. A. Mursianto, I. M. Falih, M. Irfan, T. Sakinah, dan D. S. Prasvita, "Perbandingan Metode Klasifikasi Random Forest dan XGBoost Serta Implementasi Teknik SMOTE pada Kasus Prediksi Hujan," Sep 2021.
- [12] A. Nugroho dan D. Harini, "Teknik Random Forest untuk Meningkatkan Akurasi Data Tidak Seimbang," *JSITIK*, vol. 2, no. 2, 2024, doi: 10.53624/jsitik.v2i2.XX.
- [13] W. Haryo Sasoko, E. Wahyu Pujiharto, R. Haris, A. Yuza Kania, Kusrini, dan Kusnawi, "Prediksi Banjir Di Dki Jakarta Dengan Menggunakan Algoritma K-Means Dan Random Forest," *Jurnal Informatika dan Teknologi Komputer*, vol. 05, no. 01, hlm. 43–49, 2024, Diakses: 15 Mei 2025. [Daring]. Tersedia pada: <https://ejurnalunsam.id/>
- [14] M. F. Asnawi, H. H. Bisono, M. A. Megantara, dan Kusrini, "Aplikasi Prediksi Banjir Menggunakan Algoritma XGBoost Berbasis Website," vol. 7, Agu 2024, doi: 10.32500/jematech.v7i1.7644.
- [15] A. C. PAMUNGKAS, "PREDIKSI BENCANA DI KOTA SEMARANG MENGGUNAKAN," Skripsi, Universitas Semarang, Semarang, 2025.
- [16] A. Suaif dan E. Sylvianti Rahayu, "ANALISIS FAKTOR DAN POLA KEJADIAN BANJIR DI BANDAR LAMPUNG MENGGUNAKAN ARIMA, RANDOM FOREST, DAN XGBOOST," *Jurnal Teknologi Komputer dan Informatika*, vol. 3, no. 2, Mar 2025.
- [17] I. Fitriyaningsih dan Y. Basani, "Flood Prediction with Ensemble Machine Learning using BP-NN and SVM," *Jurnal Teknologi dan Sistem Komputer*, vol. 7, no. 3, hlm. 93–97, Jul 2019, doi: 10.14710/jtsiskom.7.3.2019.93-97.
- [18] A. Rachmawardani, S. K. Wijaya, dan A. Shopaheluwakan, "SISTEM PERINGATAN DINI BANJIR BERBASIS MACHINE LEARNING: STUDI LITERATUR," *METHOMIKA Jurnal*

*Manajemen Informatika dan Komputerisasi Akuntansi*, vol. 6, no. 6, hlm. 188–198, Okt 2022, doi: 10.46880/jmika.Vol6No2.pp188-198.

- [19] D. Kurniawan, A. Nugroho Jati, dan A. Mulyana, “Perancangan Dan Implementasi Sistem Monitor Cuaca Menggunakan Mikrokontroler Sebagai Pendukung Sistem Peringatan Dini Banjir,” vol. 3, no. 1, Apr 2016, [Daring]. Tersedia pada:  
[http://www.egr.msu.edu/classes/ece480/capstone/fall09/group03/AN\\_hemmanur.pdf](http://www.egr.msu.edu/classes/ece480/capstone/fall09/group03/AN_hemmanur.pdf)
- [20] T. M. . Mitchell, *Machine Learning*. McGraw-Hill, 1997.
- [21] Sebastian. Raschka, *Python machine learning : unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Packt Publishing, 2015.
- [22] J. Han, M. Kamber, dan J. Pei, *Data Mining Third Edition*, 3 ed. 2011.
- [23] T. Chen dan C. Guestrin, “XGBoost: A scalable tree boosting system,” dalam *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Agu 2016, hlm. 785–794. doi: 10.1145/2939672.2939785.
- [24] mavillan, “Calculation of Split Gains.” Diakses: 16 Juli 2025. [Daring]. Tersedia pada:  
[https://github.com/microsoft/LightGBM/issues/5986?utm\\_source=chatgpt.com](https://github.com/microsoft/LightGBM/issues/5986?utm_source=chatgpt.com)
- [25] R. J. Hyndman, “Measuring forecast accuracy,” 2014. [Daring]. Tersedia pada: [www.otexts.org/fpp/2/5](http://www.otexts.org/fpp/2/5),
- [26] D. Chicco, M. J. Warrens, dan G. Jurman, “The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation,” *PeerJ Comput Sci*, vol. 7, hlm. 1–24, 2021, doi: 10.7717/PEERJ-CS.623.
- [27] BNPB, “Data Banjir BNPB,” Badan Nasional Penanggulangan Bencana. Diakses: 10 Mei 2025. [Daring]. Tersedia pada:  
<https://dibi.bnpb.go.id/superset/dashboard/2/>
- [28] V. K. M. Putri, “Pembagian Musim di Indonesia,” Kompas.com Skola. Diakses: 10 Juni 2025. [Daring]. Tersedia pada:

<https://www.kompas.com/skola/read/2022/08/12/070000269/pembagian-musim-di-indonesia>

- [29] D. R. Legates dan G. J. McCabe, "Evaluating the use of 'goodness-of-fit' measures in hydrologic and hydroclimatic model validation," *Water Resour Res*, vol. 35, no. 1, hlm. 233–241, 1999, doi: 10.1029/1998WR900018.
- [30] A. Géron, "Hands-On Machine Learning With Scikit-Learn, Keras, And Tensorflow PDF Scan to Download."
- [31] S. J. Taylor dan B. Letham, "Forecasting at scale," 27 September 2017. doi: 10.7287/peerj.preprints.3190v2.
- [32] Supari, *PETA RATA-RATA CURAH HUJAN DAN HARI HUJAN PERIODE 1991 – 2020 INDONESIA*. Pusat Informasi Perubahan Iklim BMKG, 2021.
- [33] A. Triady Ujung, A. Laila Nugraha, dan H. Sugiastu Firdaus, "KAJIAN PEMETAAN RISIKO BENCANA BANJIR KOTA SEMARANG DENGAN MENGGUNAKAN SISTEM INFORMASI GEOGRAFIS," 2019.
- [34] C. Bergmeir, R. J. Hyndman, dan B. Koo, "A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction," 2015. [Daring]. Tersedia pada: <http://www.buseco.monash.edu.au/depts/ebs/pubs/wpapers/>
- [35] T. Saito dan M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS One*, vol. 10, no. 3, Mar 2015, doi: 10.1371/journal.pone.0118432.

# LAMPIRAN

## A. LAMPIRAN A

### Kode Program Utama

#### A.1 Pembersihan Data dan Visualisasi Cuaca

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

kolom_numerik = ["tn", "tx", "tavg", "rh_avg", "ff_avg",
                 "rr", "elevasi"]

df = pd.read_csv("/content/Dataset_Final_with_Elevasi.csv",
                 parse_dates=["tanggal"], dtype={k: str for k in
                 kolom_numerik})

# Ganti nilai kosong atau salah input ke NaN
df[kolom_numerik] = df[kolom_numerik].replace(["-", "na",
        "NaN", ""], np.nan)

for kolom in kolom_numerik:
    df[kolom] = pd.to_numeric(df[kolom], errors='coerce')

# Cek dan bersihkan nilai suhu dan kelembapan yang tidak
logis
for kol in ["tavg", "tn", "tx"]:
    df.loc[df[kol] == 0, kol] = np.nan
    df.loc[df[kol] > 50, kol] = np.nan
for kode in [9999, 999, 8888, 888]:
    for kol in kolom_numerik:
        df.loc[df[kol] == kode, kol] = np.nan
df.loc[df["rh_avg"] == 0, "rh_avg"] = np.nan
df.loc[df["rh_avg"] > 100, "rh_avg"] = np.nan
```

```

# Hapus data yang mengandung nilai kosong
df.dropna(subset=kolom_numerik + ["tanggal"], inplace=True)

# Tampilkan statistik deskriptif hasil pembersihan
print(df[["tavg", "rh_avg", "rr", "ff_avg"]].describe())

# Tambahkan kolom bulan untuk analisis musiman
df["bulan"] = df["tanggal"].dt.month

# Visualisasi rata-rata curah hujan bulanan
hujan_bulanan = df.groupby("bulan")["rr"].mean()
plt.figure(figsize=(8, 4))
plt.plot(hujan_bulanan.index, hujan_bulanan.values,
marker="o")
plt.title("Rata-rata Curah Hujan Bulanan")
plt.xlabel("Bulan")
plt.ylabel("Curah Hujan (mm)")
plt.grid()
plt.show()

# Visualisasi distribusi suhu rata-rata harian
plt.figure(figsize=(8, 4))
plt.hist(df["tavg"], bins=20, alpha=0.7)
plt.title("Distribusi Suhu Rata-rata Harian")
plt.xlabel("Suhu Rata-rata (°C)")
plt.ylabel("Frekuensi")
plt.show()

```

```
# Visualisasi distribusi kelembapan udara
plt.figure(figsize=(8, 4))
plt.hist(df["rh_avg"], bins=20, alpha=0.7)
plt.title("Distribusi Kelembapan Udara")
plt.xlabel("Kelembapan (%)")
plt.ylabel("Frekuensi")
plt.show()
```

## A.2 Kode Visualisasi dan Analisis Data Prediksi Banjir



```

# Load data

df_diy = pd.read_csv("data/dataset_diy_final.csv")

df_jabar = pd.read_csv("data/dataset_jabar_final.csv")

df_diy["wilayah"] = "DIY"

df_jabar["wilayah"] = "Jabar"

df_all = pd.concat([df_diy, df_jabar], ignore_index=True)


# Visualisasi Distribusi Kelembapan Udara

plt.figure(figsize=(12, 6))

sns.histplot(data=df_all, x="rh_avg", hue="wilayah", bins=30,
kde=True,

               element="step", stat="density",
common_norm=False, alpha=0.4)

plt.title("Distribusi Kelembapan Udara (rh_avg) per Wilayah")

plt.xlabel("Kelembapan Udara (%)")

plt.ylabel("Kepadatan")

plt.xticks(range(50, 105, 5))

plt.grid(True)

plt.legend(title="Wilayah")

plt.tight_layout()

plt.show()

```

```

# Visualisasi Distribusi Kecepatan Angin

plt.figure(figsize=(10, 5))

sns.histplot(data=df_all, x="ff_avg", hue="wilayah", bins=30,
kde=True,

                element="bars",                stat="percent",
common_norm=False, alpha=0.5)

plt.title("Perbandingan Distribusi Kecepatan Angin Rata-rata
(ff_avg) per Wilayah")

plt.xlabel("Kecepatan Angin (m/s)")

plt.ylabel("Persentase (%)")

plt.grid(True)

plt.tight_layout()

plt.show()


# Visualisasi Curah Hujan vs Jumlah Banjir Bulanan

df_diy['tanggal'] = pd.to_datetime(df_diy['tanggal'])
df_jabar['tanggal'] = pd.to_datetime(df_jabar['tanggal'])

df_diy['bulan'] = df_diy['tanggal'].dt.month
df_jabar['bulan'] = df_jabar['tanggal'].dt.month


# DIY

plt.figure(figsize=(10, 5))

sns.barplot(x='bulan', y='rr', data=df_diy, color='skyblue',
label='Curah Hujan')

ax2 = plt.twinx()

sns.lineplot(x='bulan', y='banjir', data=df_diy, color='red',
label='Jumlah Banjir', ax=ax2)

```

```

plt.title('Curah Hujan vs Jumlah Banjir Bulanan - DIY')

plt.tight_layout()

plt.show()

# Jabar

plt.figure(figsize=(10, 5))

sns.barplot(x='bulan', y='rr', data=df_jabar,
color='lightgreen', label='Curah Hujan')

ax2 = plt.twinx()

sns.lineplot(x='bulan', y='banjir', data=df_jabar,
color='orange', label='Jumlah Banjir', ax=ax2)

plt.title('Curah Hujan vs Jumlah Banjir Bulanan - Jawa Barat')

plt.tight_layout()

plt.show()

```

### A.3 Kode Prediksi Mingguan

```

df['week'] =
df['tanggal'].dt.isocalendar().week.astype(int)

df['year'] = df['tanggal'].dt.year.astype(int)

fitur_mingguan = ["month", "tavg", "rh_avg",
"ff_avg", "rr", "elevasi"]

aggr = df.groupby(["year", "week"],
as_index=False)[fitur_mingguan].mean()

aggr["month"] = pd.to_datetime(
    aggr["year"].astype(str) + '-W' +
    aggr["week"].astype(str) + '-1',
    format='%Y-W%W-%w'
).dt.month

```

#### A.4 Kode Logika *Outlier* Berdasarkan Domain

```
# Logika outlier berdasarkan domain
if 'rh_avg' in df.columns:
    df.loc[(df['rh_avg'] < 10) | (df['rh_avg'] > 100),
            'rh_avg'] = np.nan
if 'ff_avg' in df.columns:
    df.loc[(df['ff_avg'] < 0) | (df['ff_avg'] > 25),
            'ff_avg'] = np.nan
if 'rr' in df.columns:
    df.loc[(df['rr'] < 0) | (df['rr'] > 500), 'rr'] =
np.nan
if 'elevasi' in df.columns:
    df.loc[(df['elevasi'] < 0) | (df['elevasi'] >
4000), 'elevasi'] = np.nan
```

#### A.5 Kode Fungsi Klasifikasi Musim

```
def klasifikasi_musim(bulan):
    if bulan in [12, 1, 2]:
        return 'musim_hujan'
    elif bulan in [3, 4, 5]:
        return 'musim_peralihan_1'
    elif bulan in [6, 7, 8]:
        return 'musim_kemarau'
    else:
        return 'musim_peralihan_2'

df['musim'] =
df['bulan'].apply(klasifikasi_musim)
df = pd.get_dummies(df, columns=['musim'])
```

## A.6 Kode Visualisasi Distribusi Data Berdasarkan Kategori Musim

### a. Kode untuk DIY:

```
df = pd.read_csv("data/dataset_diy_final.csv")
pelatih = PelatihModelBanjir()
df = pelatih.tambah_fitur_musiman(df, 'tanggal')
plt.figure(figsize=(6, 4))
sns.countplot(x='musim_monsoon', data=df,
order=['musim_hujan', 'transisi_kemarau', 'musim_kemarau',
'transisi_hujan'])
plt.title('Distribusi Data Berdasarkan Musim (DIY)')
plt.xlabel('Musim')
plt.ylabel('Jumlah Data')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```

### b. Kode untuk Jawa Barat:

```

df = pd.read_csv("data/dataset_jabar_final.csv")
pelatih = PelatihModelBanjir()
df = pelatih.tambah_fitur_musiman(df, 'tanggal')
plt.figure(figsize=(6, 4))

sns.countplot(x='musim_monsoon', data=df,
order=['musim_hujan', 'transisi_kemarau', 'musim_kemarau',
'transisi_hujan'])

plt.title('Distribusi Data Berdasarkan Musim (JAWA BARAT)')
plt.xlabel('Musim')
plt.ylabel('Jumlah Data')
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()

```

## A.7 Kode

### a. kode implementasi untuk DIY:

```

df_diy['interaksi_rr_elevasi'] = df_diy['rr'] *
df_diy['elevasi']

df_diy['jumlah_kejadian'] =
df_diy['jumlah_kejadian'].astype(int)

selected_features = df_diy[["rr", "elevasi",
"interaksi_rr_elevasi", "jumlah_kejadian"]]

plt.figure(figsize=(8, 6))

sns.heatmap(selected_features.corr(), annot=True,
cmap="coolwarm")

plt.title("Matriks Korelasi Fitur Cuaca dan Target (DIY)")
plt.tight_layout()
plt.show()

```

b. Dan untuk Jawa Barat:

```
df_jabar['interaksi_rr_elevasi'] = df_jabar['rr'] *  
df_jabar['elevasi']  
  
df_jabar['jumlah_kejadian'] =  
df_jabar['jumlah_kejadian'].astype(int)  
  
selected_features = df_jabar[["rr", "elevasi",  
"interaksi_rr_elevasi", "jumlah_kejadian"]]  
  
plt.figure(figsize=(8, 6))  
  
sns.heatmap(selected_features.corr(), annot=True,  
cmap="coolwarm")  
  
plt.title("Matriks Korelasi Fitur Cuaca dan Target  
(Jawa Barat)")  
  
plt.tight_layout()  
  
plt.show()
```

#### A.8 Kode Pelatihan Model Regresi

```

models = {
    "rf": RandomForestRegressor(n_estimators=100,
                                random_state=42),
    "xgb": XGBRegressor(n_estimators=100, random_state=42),
    "lgbm": LGBMRegressor(n_estimators=100, random_state=42)
}

fitur = ["month", "day", "tn", "tx", "rh_avg", "ff_avg",
         "rr", "elevasi", "is_rainy", "is_windy", "rr_roll3",
         "tavg_roll3"]

targets = ["tavg", "rr", "rh_avg", "ff_avg"]

for target in targets:
    X = df[fitur].values
    y = df[target].values

    for name, model in models.items():
        model.fit(X, y)

        print(f"Model {name.upper()}, target {target} telah
        dilatih.")

```

## A.9 Kode Evaluasi Regresi (Cuaca)

```

# Train-test split: 2018-2022 untuk training, 2023 untuk
testing

train = df[df['year'] <= 2022]

test = df[df['year'] == 2023]


# Inisialisasi model

models = {

    "RF": RandomForestRegressor(n_estimators=100,
random_state=42),

    "XGB": XGBRegressor(n_estimators=100, random_state=42),

    "LGBM": LGBMRegressor(n_estimators=100,
random_state=42)

}

# Hitung metrik

    r2 = r2_score(y_true, y_pred)

    mae = mean_absolute_error(y_true, y_pred)

    rmse = np.sqrt(mean_squared_error(y_true, y_pred))

```

#### A.10 Kode Fungsi Prediksi Cuaca Saat ini

```

def predict_current_weather(city, api_key, base_url,
trained_models):

    cuaca = fetch_weather_from_city(city, api_key, base_url)

    now = datetime.now()

fitur_input = {

    "month": now.month,

    "day": now.day,

    "tn": cuaca["tmin"],

    "tx": cuaca["tmax"],

    "rh_avg": cuaca["rh"],

    "ff_avg": cuaca["wind"],

    "rr": cuaca["rain"],

    "elevasi": cuaca["elevasi"],

    "is_rainy": int(cuaca["rain"] > 0),

    "is_windy": int(cuaca["wind"] > 10),

    "rr_roll3": cuaca["rain"],

    "tavg_roll3": (cuaca["tmax"] + cuaca["tmin"]) / 2

}

df_input = pd.DataFrame([fitur_input])

hasil_prediksi = {}

for target in ["tavg", "rr", "rh_avg", "ff_avg"]:

    prediksi_semua_model = []

    for algo in ["rf", "xgb", "lgbm"]:

        key = f"{algo}_{target}"

if key in trained_models:

    prediksi = trained_models[key].predict(df_input)[0]

    prediksi_semua_model.append(prediksi)

    hasil_prediksi[target] = np.mean(prediksi_semua_model)

print(f"Prediksi Suhu: {hasil_prediksi['tavg']:.2f}°C, Curah
Hujan: {hasil_prediksi['rr']:.2f} mm, " f"Kelembapan:
{hasil_prediksi['rh_avg']:.2f}%, Angin:
{hasil_prediksi['ff_avg']:.2f} m/s")

```

#### A.11 Kode Fungsi Prediksi Cuaca Mingguan

```
def predict_weekly_weather_per_day(aggr, trained_models):
    fitur_mingguan = ["month", "tn", "tx", "rh_avg", "ff_avg",
                      "rr", "elevasi"]

    hasil_per_hari = []

    base_input = aggr[fitur_mingguan].iloc[[-1]].copy()

    hari_ini = datetime.now()

    for i in range(7):
        tanggal_prediksi = hari_ini + timedelta(days=i+1)

        base_input["month"] = tanggal_prediksi.month

        hasil_harian = {}

        for target in ["tavg", "rr", "rh_avg", "ff_avg"]:
            prediksi_model = []

            for algo in ["rf", "xgb", "lgbm"]:
                key = f"weekly_{target}_{algo}"

                if key in trained_models:
                    pred =
trained_models[key].predict(base_input)[0]

                    prediksi_model.append(pred)

            hasil_harian[target] = np.mean(prediksi_model)

    hasil_per_hari.append((tanggal_prediksi.strftime("%d %B %Y"),
    hasil_harian))

    for tanggal_str, pred in hasil_per_hari:
        print(f"{tanggal_str} | Curah Hujan: {pred['rr']:.2f}
mm | Suhu: {pred['tavg']:.2f}°C | "

              f"Kelembapan: {pred['rh_avg']:.2f}% | Kecepatan
Angin: {pred['ff_avg']:.2f} m/s")
```

#### A.12 Kode Evaluasi Model dengan *K-Fold Cross Validation* (Banjir)

```

n_splits = 5

skf = StratifiedKFold(n_splits=n_splits, shuffle=True,
random_state=42)

recalls, f1s = [], []

for train_idx, test_idx in skf.split(X, y):

    X_train, X_test = X.iloc[train_idx],
X.iloc[test_idx]

    y_train, y_test = y.iloc[train_idx],
y.iloc[test_idx]

    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    recalls.append(recall_score(y_test, y_pred))

    f1s.append(f1_score(y_test, y_pred))

```

#### A.13 Kode Fungsi *Learning Curve*

```

def plot_learning_curve(model, X, y, title):

    plt.figure(figsize=(8, 5))

    plt.plot(train_sizes, train_scores, label='Train
F1')

    plt.plot(train_sizes, val_scores, label='Validation
F1')

    plt.xlabel("Proporsi Data Latih")

    plt.ylabel("F1 Score")

    plt.title(f"Learning Curve: {title}")

    plt.legend()

    plt.grid(True)

    plt.tight_layout()

    plt.show()

```

#### A.14 Kode Evaluasi Prediksi Banjir (Klasifikasi)

```

from sklearn.metrics import classification_report,
roc_auc_score, confusion_matrix, f1_score, recall_score,
precision_score

# Prediksi pada data uji
y_proba = model.predict_proba(X_test)[:, 1]
y_pred = (y_proba >= best_threshold).astype(int)

# Metrik evaluasi utama
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_proba)
cm = confusion_matrix(y_test, y_pred)

print("Recall:", recall)
print("F1-score:", f1)
print("ROC-AUC:", auc)
print("Confusion Matrix:\n", cm)

print(classification_report(y_test, y_pred, digits=3))

```

#### A.15 Kode Proses *Ensemble Voting* berbobot Berdasarkan *Recall* (Banjir)

```

recalls = [recall_rf, recall_xgb, recall_lgbm]
total_recall = sum(recalls)

if total_recall == 0:
    # Voting tidak dijalankan, gunakan model "paling baik"
    (misal: XGBoost)

    bobot_rf, bobot_xgb, bobot_lgbm = 0, 1, 0

    print("Tidak ada model dengan recall > 0, voting ensemble
    tidak dijalankan. Menggunakan model XGBoost sebagai prediksi
    akhir.")
else:
    bobot_rf = recall_rf / total_recall

    bobot_xgb = recall_xgb / total_recall

    bobot_lgbm = recall_lgbm / total_recall

    print(f"Bobot Voting: RF={bobot_rf:.3f},
    XGB={bobot_xgb:.3f}, LGBM={bobot_lgbm:.3f}")

# Ensemble prediction (bisa langsung digunakan)

final_pred = (bobot_rf * rf_pred) + (bobot_xgb * xgb_pred) +
(bobot_lgbm * lgbm_pred)

final_pred = (final_pred >= 0.5).astype(int) # threshold
0.5, atau threshold lain sesuai hasil tuning

```

## A.16 Potongan Kode Fitur Input Cuaca

```

fitur_input = {
    "month": datetime.now().month,
    "day": datetime.now().day,
    "tn": cuaca['main']['temp_min'],
    "tx": cuaca['main']['temp_max'],
    "rh_avg": cuaca['main']['humidity'],
    "ff_avg": cuaca['wind']['speed'],
    "rr": cuaca.get("rain", {}).get("1h", 0.0),
    "is_rainy": int(cuaca.get("rain", {}).get("1h", 0.0)
> 0),
    "is_windy": int(cuaca['wind']['speed'] > 10),
    "rr_roll3": cuaca.get("rain", {}).get("1h", 0.0),
    "tavg_roll3": (cuaca['main']['temp_min'] +
cuaca['main']['temp_max']) / 2
}

```

A.17 Kode Fungsi *Load* Model Cuaca

```

def load_weather_models(target):
    models = {}
    for algo, suf in zip(["RandomForest", "XGBoost",
                          "LightGBM"], ["rf", "xgb", "lgbm"]):
        path = f"models/model_{suf}_{target}.pkl"
        models[algo] = joblib.load(path)
    return models

hasil_prediksi = {}
for target in ["tavg", "rr", "rh_avg", "ff_avg"]:
    models = load_weather_models(target)
    X_input = pd.DataFrame([[fitur_input.get(f, 0.0) for
        f in models["RandomForest"].feature_names_in_]],
        columns=models["RandomForest"].feature_names_in_)
    # Rata-rata hasil semua model
    prediksi = np.mean([models[a].predict(X_input)[0]
        for a in models])
    hasil_prediksi[target] = prediksi

```

#### A.18 Kode *Backend* Prediksi Cuaca Mingguan

```

hasil_per_hari = []

for i in range(7):
    tanggal_prediksi = hari_ini + timedelta(days=i+1)
    hari_lengkap =
    format_hari_dan_tanggal(tanggal_prediksi)
    fitur_input["month"] = tanggal_prediksi.month

```

```

hasil_harian = {}
for target in ["tavg", "rr", "rh_avg", "ff_avg"]:
    prediksi_model = []
    for algo in ["rf", "xgb", "lgbm"]:
        key = f"weekly_{target}_{algo}"
        if key in trained_models:
            df_input = pd.DataFrame([fitur_input])
            pred =
trained_models[key].predict(df_input)[0]
            prediksi_model.append(pred)
    prediksi = np.mean(prediksi_model)
    if target == "tavg":
        if prediksi < 10 or prediksi > 45:
            prediksi = (tn + tx) / 2
    hasil_harian[target] = prediksi

fitur_input["tn"] = hasil_harian["tavg"]
fitur_input["tx"] = hasil_harian["tavg"]
fitur_input["rh_avg"] = hasil_harian["rh_avg"]
fitur_input["ff_avg"] = hasil_harian["ff_avg"]
fitur_input["rr"] = hasil_harian["rr"]

rr_pred = hasil_harian["rr"]
if rr_pred < 1.0:
    cuaca = "Tidak Hujan"
    icon = "clear_sky.svg"
elif rr_pred < 5.0:
    cuaca = "Hujan Ringan"
    icon = "light_rain.svg"

```

```

elif rr_pred < 20.0:
    cuaca = "Hujan Sedang"
    icon = "moderate_rain.svg"
elif rr_pred < 50.0:
    cuaca = "Hujan Lebat"
    icon = "heavy_intensity_rain.svg"
else:
    cuaca = "Hujan Sangat Lebat"
    icon = "extreme_rain.svg"
hasil_per_hari.append({
    "tanggal": hari_lengkap,
    **{k: round(v, 2) for k, v in
hasil_harian.items()},
    "desc_id": cuaca,
    "icon": icon
})
return hasil_per_hari

```

#### A.19 Kode Proses Pengambilan Data dan *Preprocessing* Fitur Banjir

```
def fetch_weather_data(lat, lon):
    url =
f"https://api.openweathermap.org/data/2.5/weather?lat={lat}&
lon={lon}&appid={API_KEY}&units=metric"

    res = requests.get(url)

    if res.status_code != 200:
        return None

    data = res.json()

    return {
        "rr": data.get("rain", {}).get("1h", 0.0),
        "rh_avg": data["main"]["humidity"],
        "ff_avg": data["wind"]["speed"],
        "cuaca": data["weather"][0]["description"]
    }
```

## A.20 Implementasi *Voting* Berbobot Berdasarkan Wilayah pada Prediksi Banjir.

```

BOBOT_DIY = {
    "RandomForest": 0.0,
    "XGBoost": 1.0,
    "LightGBM": 0.0
}

BOBOT_JABAR = {
    "RandomForest": 0.306,
    "XGBoost": 0.371,
    "LightGBM": 0.323
}

...

# Dalam fungsi predict_flood(wilayah)
if wilayah == "diy":
    BOBOT = BOBOT_DIY
elif wilayah == "jabar":
    BOBOT = BOBOT_JABAR
else:
    raise ValueError("Wilayah tidak dikenal! Harus 'diy' atau 'jabar'.")

# Proses voting ensemble
skor_voting = 0.0
for algo in ALGOS:
    model, scaler, fitur = load_model_data(wilayah, algo)
    X = pd.DataFrame([[cuaca.get(f, 0.0) for f in fitur]],
                      columns=fitur)
    X_scaled = scaler.transform(X)
    proba = model.predict_proba(X_scaled)[0][1]
    skor_voting += BOBOT[algo] * proba

```

## A.21 Kode Alur *Backend* Untuk Prediksi Cuaca.

```
if request.method == "POST":
    city = request.form.get("city", "").strip().title()

    # 1) real-time
    realtime = predict_current_weather(city, API_KEY,
BASE_URL)

    # 2) weekly
    trained_models = {}
    for target in ["tavg", "rr", "rh_avg",
"ff_avg"]:
        for algo in ["rf", "xgb", "lgbm"]:
            fn =
f"models/model_model_weekly_{target}_{algo}.pkl"
            trained_models[f"model_model_weekly_
{target}_{algo}"] = safe_joblib_load(fn)
        weekly = predict_weekly_weather(city,
trained_models)

    return render_template(
        "index.html",
        realtime_data=realtime,
        weekly_forecast=weekly,
        forecast_city=city,
        local_time=local_time,
        timezone=timezone,
        now=now,
        formatted_now=formatted_now
    )

...
```

## A.22 Kode HTML Prediksi Cuaca

```

<div class="weather-today">

  <div style="font-size: 0.95rem; margin-bottom: 0.5rem;">

    {{ formatted_now }} WIB | {{ realtime_data.city }}

  </div>

  <div class="temp">{{ realtime_data.tavg |
round(1) }}&deg;C</div>

  

  <div style="margin-top:
0.5rem;">{{ realtime_data.weather }}</div>

  <div class="mt-2">Angin: {{ realtime_data.ff_avg |
round(1) }} m/s</div>

  <div>Kelembapan: {{ realtime_data.rh_avg }}%</div>

  <div>Curah Hujan: {{ realtime_data.rr }} mm</div>
</div>

<div class="weather-weekly">

  {% for day in weekly_forecast %}

  <div class="day-card">

    <div>{{ day.tanggal }}</div>

    

```

## A.23 Kode HTML Map

```
function setWeatherMap(lat, lon, city, zoom = 10) {
    if (weatherMap) {
        weatherMap.remove();
    }

    weatherMap = L.map("weatherMap").setView([lat, lon],
zoom);

L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}
.png", {
    maxZoom: 18,
    attribution: "© OpenStreetMap"
}).addTo(weatherMap);

L.marker([lat,
lon]).addTo(weatherMap).bindPopup(city).openPopup();
}
```

#### A.24 Kode *Route AJAX*

```
@app.route("/flood_ajax")
def flood_ajax():
    wilayah = request.args.get("wilayah", "diy")
    predictions, alert = predict_flood(wilayah)
    return jsonify({
        "results": predictions,
        "alert": alert
    })
```

## B. LAMPIRAN B

### Dataset

1. Dataset Cuaca: [data\\_cuaca.xlsx](#)
2. Dataset Prediksi Banjir – DIY: [dataset\\_diy\\_final.xlsx](#)
3. Dataset Prediksi Banjir – Jawa Barat: [dataset\\_jabar\\_final.xlsx](#)