LAMPIRAN

Lampiran 1. *Souce Code* Pengujian Model

Source Code Utama (Pengujian 3 Algortima)

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# 1. Load Data
df = pd.read_csv('dataset.csv')
X = df.drop('Status Kelulusan', axis=1)
y = df['Status Kelulusan']

#
==========================================================
=============
# 2. HASIL HYPERPARAM
#
==========================================================
=============
model_c45 = DecisionTreeClassifier(
    criterion='entropy',
    max_depth=3,            #
    min_samples_split=2,    #
    random_state=42
)

model_rf = RandomForestClassifier(
    n_estimators=100,       #
    max_depth=5,            #
    min_samples_split=2,    #
    random_state=42
)

model_xgb = XGBClassifier(
    learning_rate=0.01,     #
    n_estimators=100,       #
```

```
    max_depth=3,            #
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

models = {
    'C4.5 (Decision Tree)': model_c45,
    'Random Forest': model_rf,
    'XGBoost': model_xgb
}

#
==============================================================
==============
# 3. EVALUASI PER LIPATAN (FOLD)
#
==============================================================
==============
print("=== MEMULAI EVALUASI AKHIR (10-FOLD CV) ===")
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
final_results = {}

for model_name, model in models.items():
    print(f"\n{'-'*65}")
    print(f"Evaluasi Algoritma: {model_name}")
    print(f"{'-'*65}")

    # Tetap gunakan pipeline agar SMOTE hanya bekerja di Data Latih
    pipeline = ImbPipeline([
        ('smote', SMOTE(random_state=42)),
        ('model', model)
    ])

    metrics = {'accuracy': [], 'precision': [], 'recall': [], 'f1': [], 'train_f1': []}

    fold_no = 1
    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]

        # FIT MODEL
        pipeline.fit(X_train, y_train)

        # Prediksi Data Latih (Untuk Overfitting Gap)
        y_train_pred = pipeline.predict(X_train)
        train_f1 = f1_score(y_train, y_train_pred)
        metrics['train_f1'].append(train_f1)
```

```
        # Prediksi Data Uji Murni
        y_test_pred = pipeline.predict(X_test)
        acc = accuracy_score(y_test, y_test_pred)
        prec = precision_score(y_test, y_test_pred)
        rec = recall_score(y_test, y_test_pred)
        f1 = f1_score(y_test, y_test_pred)

        # Simpan metrik
        metrics['accuracy'].append(acc)
        metrics['precision'].append(prec)
        metrics['recall'].append(rec)
        metrics['f1'].append(f1)

        # TAMPILKAN HASIL PER LIPATAN
        print(f"  [Fold {fold_no:>2}] Recall: {rec:.4f} | F1-Score: {f1:.4f} |
Akurasi: {acc:.4f}")
        fold_no += 1

    # Agregasi Rata-rata dan Standar Deviasi
    final_results[model_name] = {
        'Recall (Mean)': round(np.mean(metrics['recall']), 4),
        'Precision (Mean)': round(np.mean(metrics['precision']), 4),
        'F1-Score (Mean)': round(np.mean(metrics['f1']), 4),
        'F1-Score (Std / Stabilitas)': round(np.std(metrics['f1']), 4),
        'Accuracy (Mean)': round(np.mean(metrics['accuracy']), 4),
        'Gen. Gap (Overfitting)': round(np.mean(metrics['train_f1']) -
np.mean(metrics['f1']), 4)
    }

print("\n\n" + "="*80)
print("HASIL KOMPARASI KINERJA ALGORITMA (RATA-RATA DARI 10
FOLD)")
print("="*80)
print(pd.DataFrame(final_results).T.sort_values(by='F1-Score (Mean)',
ascending=False).to_string())
print("="*80)
```

Source Code Hyperparamter Tuning

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import GridSearchCV
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# 1. Load Data
df = pd.read_csv('dataset.csv')
X = df.drop('Status Kelulusan', axis=1)
y = df['Status Kelulusan']

# 2. Definisi Model dan Parameter
models = {
    'C4.5 (Decision Tree)': (DecisionTreeClassifier(random_state=42), {
        'model__criterion': ['entropy'],
        'model__max_depth': [3, 5, 10, None],
        'model__min_samples_split': [2, 5, 10]
    }),
    'Random Forest': (RandomForestClassifier(random_state=42), {
        'model__n_estimators': [50, 100, 200],
        'model__max_depth': [5, 10, 20, None],
        'model__min_samples_split': [2, 5, 10]
    }),
    'XGBoost': (XGBClassifier(use_label_encoder=False, eval_metric='logloss',
random_state=42), {
        'model__learning_rate': [0.01, 0.1, 0.2],
        'model__n_estimators': [50, 100, 200],
        'model__max_depth': [3, 5, 7]
    })
}

print("=== MEMULAI PENCARIAN PARAMETER TERBAIK ===")
best_parameters = {}

for name, (model, params) in models.items():
    print(f"\nMencari untuk {name} (Mohon tunggu)...")

    # SMOTE tetap di dalam pipeline agar tidak Data Leakage saat pencarian!
    pipeline = ImbPipeline([
        ('smote', SMOTE(random_state=42)),
        ('model', model)
    ])

    # Pencarian dengan 5-Fold CV
    # verbose=1 agar Anda bisa melihat loading-nya
    grid = GridSearchCV(pipeline, params, cv=5, scoring='f1', n_jobs=1,
verbose=2)
    grid.fit(X, y)

    # Simpan dan cetak hasil
    best_parameters[name] = grid.best_params_
```

```
    print(f"SELESAI! Parameter Terbaik {name}:")
    print(grid.best_params_)

print("\nSilakan catat parameter di atas untuk dimasukkan ke Skrip 2.")
```

Source Code Feature Importance

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import StratifiedKFold
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier

#
==========================================================
==========
# 1. PERSIAPAN DATA DAN MODEL
#
==========================================================
==========
df = pd.read_csv('dataset.csv')
X = df.drop('Status Kelulusan', axis=1)
y = df['Status Kelulusan']
nama_fitur = X.columns

# Parameter XGBoost Terbaik Anda
xgb_model = XGBClassifier(
    learning_rate=0.01,       #
    n_estimators=100,         #
    max_depth=3,              #
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Array penampung bobot
importances_akumulasi = np.zeros(len(nama_fitur))

print("Sedang mengeksekusi Ekstraksi Feature Importance (mohon tunggu)...")
```

```
#
==========================================================
==========
# 2. PROSES 10-FOLD CV UNTUK MENDAPATKAN RATA-RATA BOBOT
#
==========================================================
==========
for train_idx, test_idx in skf.split(X, y):
    X_train, y_train = X.iloc[train_idx], y.iloc[train_idx]

    # Inisialisasi SMOTE secara manual untuk lipatan ini
    smote = SMOTE(random_state=42)
    X_train_sm, y_train_sm = smote.fit_resample(X_train, y_train)

    # Latih model
    xgb_model.fit(X_train_sm, y_train_sm)

    # Tambahkan bobot fiturnya (feature_importances_)
    importances_akumulasi += xgb_model.feature_importances_

# Hitung rata-rata bobot dari 10 lipatan
rata_rata_importances = importances_akumulasi / 10.0

#
==========================================================
==========
# 3. CETAK BOBOT FITUR
#
==========================================================
==========
# Buat DataFrame agar mudah diurutkan
df_importances = pd.DataFrame({'Fitur': nama_fitur, 'Bobot':
rata_rata_importances})
df_importances = df_importances.sort_values(by='Bobot', ascending=True) #
Ascending untuk grafik barh

print("\n" + "="*50)
print("RATA-RATA BOBOT FITUR (Dari yang Terbesar)")
print("="*50)
# Buat salinan sementara untuk dicetak secara menurun (Descending)
df_cetak = df_importances.sort_values(by='Bobot', ascending=False)
for index, row in df_cetak.iterrows():
    print(f"{row['Fitur']:<30}: {row['Bobot']:.4f}")
print("="*50 + "\n")

#
==========================================================
==========
```

```
# 3. VISUALISASI GRAFIK BAR HORIZONTAL
#
==========
# Buat DataFrame agar mudah diurutkan
df_importances = pd.DataFrame({'Fitur': nama_fitur, 'Bobot':
rata_rata_importances})
df_importances = df_importances.sort_values(by='Bobot', ascending=True)

plt.figure(figsize=(10, 7))
# Membuat bar horizontal dengan warna gradient
colors = plt.cm.Blues(np.linspace(0.4, 1, len(nama_fitur)))
bars = plt.barh(df_importances['Fitur'], df_importances['Bobot'], color=colors,
edgecolor='black', linewidth=0.5)

# Menambahkan teks angka di ujung bar
for bar in bars:
    plt.text(bar.get_width() + 0.005, bar.get_y() + bar.get_height()/2,
             f'{bar.get_width():.4f}',
             va='center', ha='left', fontsize=10)

# Desain Estetika Grafik
plt.xlabel('Tingkat Kontribusi (Mean Feature Importance)', fontsize=12,
fontweight='bold')
plt.ylabel('Atribut Akademik & Demografis', fontsize=12, fontweight='bold')
plt.title('Ekstraksi Bobot Fitur (Feature Importance) - Model XGBoost',
fontsize=14, fontweight='bold')
plt.xlim(0, df_importances['Bobot'].max() + 0.05) # Beri ruang untuk teks
plt.grid(axis='x', linestyle='--', alpha=0.6)

# Simpan dan Tampilkan
file_output = 'Feature_Importance_XGBoost.png'
plt.savefig(file_output, dpi=300, bbox_inches='tight')
plt.show()

print(f"\nSelesai! Gambar berhasil disimpan dengan nama: {file_output}")
```

Source Code Uji Signifikansi Statistik

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import f1_score
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from scipy.stats import friedmanchisquare

#
==========================================================
==========
# 1. PERSIAPAN DATA DAN MODEL
#
==========================================================
==========
df = pd.read_csv('dataset.csv')
X = df.drop('Status Kelulusan', axis=1)
y = df['Status Kelulusan']

models = {
   'C4.5': DecisionTreeClassifier(criterion='entropy', max_depth=3,
min_samples_split=2, random_state=42),
   'Random Forest': RandomForestClassifier(n_estimators=100, max_depth=5,
min_samples_split=2, random_state=42),
   'XGBoost': XGBClassifier(learning_rate=0.01, n_estimators=100,
max_depth=3, use_label_encoder=False, eval_metric='logloss',
random_state=42)
}

skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Penampung skor F1 untuk masing-masing algoritma pada setiap fold
f1_scores = {'C4.5': [], 'Random Forest': [], 'XGBoost': []}

print("Sedang memproses 10-Fold CV untuk Uji Statistik (mohon tunggu)...")

#
==========================================================
==========
# 2. PENGUMPULAN F1-SCORE PER FOLD
#
==========================================================
==========
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]

   for name, model in models.items():
      pipeline = ImbPipeline([('smote', SMOTE(random_state=42)), ('model',
model)])
      pipeline.fit(X_train, y_train)
```

```
        y_pred = pipeline.predict(X_test)
        f1_scores[name].append(f1_score(y_test, y_pred))

#
==========
# 3. UJI STATISTIK FRIEDMAN
#
==========
stat, p_value = friedmanchisquare(f1_scores['C4.5'], f1_scores['Random
Forest'], f1_scores['XGBoost'])

print("\n" + "="*60)
print("HASIL UJI SIGNIFIKANSI STATISTIK (FRIEDMAN TEST)")
print("="*60)
print(f"Nilai Chi-Square (Statistik Friedman) : {stat:.4f}")
print(f"Nilai P-Value                         : {p_value:.6f}")
print("-" * 60)

alpha = 0.05
if p_value < alpha:
    print("KESIMPULAN: P-Value < 0.05. Hipotesis Nol (H0) DITOLAK.")
    print("Terdapat PERBEDAAN YANG SIGNIFIKAN secara statistik antar
model.")
else:
    print("KESIMPULAN: P-Value >= 0.05. Hipotesis Nol (H0) DITERIMA.")
    print("TIDAK ADA perbedaan performa yang signifikan antar model.")
print("="*60)
```

Lampiran 2. Cuplikan Dataset Utama dan Pendamping

a. Dataset Utama (UPMI Bali)

Jenis Kelamin	Usia Masuk	Asal Sekolah	Kategori Wilayah	IPS Semester 1	IPS Semester 2	IPS Semester 3	IPS Semester 4	Delta IPS	Kelompok Keilmuan	Frekuensi Tunggakan	Status Beasiswa	Status Kelulusan
0	22	0	1	3.7	3.55	2.82	3.5	-0.2	0	3	0	1
0	17	0	1	3.4	3.45	3.36	3.5	0.1	0	1	0	0
0	19	0	1	3.4	3.36	3.55	3.5	0.1	0	1	0	0
0	18	0	0	3.5	3.36	3.18	3.63	0.13	0	1	0	0
0	18	0	0	1	0	0	0	-1	0	4	0	1
0	19	0	1	3.6	3.55	3.64	3.63	0.03	0	0	0	0
0	21	0	1	3.7	0	0	0	-3.7	0	4	0	1
1	18	0	0	3.9	3.73	3.82	3.63	-0.27	0	1	1	0
0	18	1	0	3.5	3.73	3.92	4	0.5	0	0	0	0
1	19	0	1	3.4	3.36	3.55	3.38	-0.02	0	3	0	0

b. Dataset Pendamping I (*Haberman's Survival*)

age	operation_year	positive_auxillary_nodes	survival_status
30	64	1	1
30	62	3	1
30	65	0	1
31	59	2	1
31	65	4	1
33	58	10	1

33	60	0	1
34	59	0	2
34	66	9	2
34	58	30	1

c. Dataset Pendamping II (*Breast Cancer Wisconsin*)

Diagnosis	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	radius2	texture2	perimeter2	area2
M	1799	1038	1228	1001	1184	2776	3001	1471	2419	7871	1095	9053	8589	1534
M	2057	1777	1329	1326	8474	7864	869	7017	1812	5667	5435	7339	3398	7408
M	1969	2125	130	1203	1096	1599	1974	1279	2069	5999	7456	7869	4585	9403
M	1142	2038	7758	3861	1425	2839	2414	1052	2597	9744	4956	1156	3445	2723
M	2029	1434	1351	1297	1003	1328	198	1043	1809	5883	7572	7813	5438	9444
M	1245	157	8257	4771	1278	17	1578	8089	2087	7613	3345	8902	2217	2719
M	1825	1998	1196	1040	9463	109	1127	74	1794	5742	4467	7732	318	5391
M	1371	2083	902	5779	1189	1645	9366	5985	2196	7451	5835	1377	3856	5096
M	13	2182	875	5198	1273	1932	1859	9353	235	7389	3063	1002	2406	2432

M	1246	2404	8397	4759	1186	2396	2273	8543	203	8243	2976	1599	2039	2394

smoothness2	compactness2	concavity2	concave_points2	symmetry2	fractal_dimension2	radius3	texture3	perimeter3	area3	smoothness3	compactness3	concavity3	concave_points3	symmetry3	fractal_dimension3
6399	4904	5373	1587	3003	6193	2538	1733	1846	2019	1622	6656	7119	2654	4601	1189
5225	1308	186	134	1389	3532	2499	2341	1588	1956	1238	1866	2416	186	275	8902
615	4006	3832	2058	225	4571	2357	2553	1525	1709	1444	4245	4504	243	3613	8758
911	7458	5661	1867	5963	9208	1491	265	9887	5677	2098	8663	6869	2575	6638	173
1149	2461	5688	1885	1756	5115	2254	1667	1522	1575	1374	205	4	1625	2364	7678
751	3345	3672	1137	2165	5082	1547	2375	1034	7416	1791	5249	5355	1741	3985	1244
4314	1382	2254	1039	1369	2179	2288	2766	1532	1606	1442	2576	3784	1932	3063	8368
8805	3029	2488	1448	1486	5412	1706	2814	1106	897	1654	3682	2678	1556	3196	1151
5731	3502	3553	1226	2143	3749	1549	3073	1062	7393	1703	5401	539	206	4378	1072
7149	7217	7743	1432	1789	1008	1509	4068	9765	7114	1853	1058	1105	221	4366	2075