“BIMASAKTI”
Jurnal Riset Mahasiswa Bidang Teknologi Informasi

Volume 8 Nomor 1
P ISSN : 2355-4401–EISSN 3109-1652
Available online at https://ejournal.unikama.ac.id/index.php/JFTI

PERBANDINGAN ALGORITMA XTREME GRADIEN BOOSTING DAN ALGORITMA DECISIEN TREE DALAM KLASIFIKASI HIV/AIDS

Bonefasius Erifan Saru[1)], **Danang Adidya Nugaraha** [2)], **Amak Yunus** [3)]

Teknik Informatika Universitas PGRI Kanjuruhan Malang[1,2,3)]
bonefasius.saru21@gmail.com

Abstrak

Penelitian ini bertujuan membandingkan kinerja algoritma Decision Tree dan Extreme Gradien Boosting (XGBoost) dalam klasifikasi status infeksi HIV/AIDS. Desain penelitian berbentuk kuantitatif dengan pendekatan eksperimental menggunakan dataset sekunder berjumlah 2.139 baris dan 23 atribut dari platform terbuka. Data melalui tahap praproses berupa pemeriksaan nilai hilang, penghapusan duplikat, deteksi dan penanganan outlier menggunakan metode Interquartile Range (IQR), serta normalisasi skala. Model dilatih dan diuji dengan tiga variasi rasio data latih dan uji (70:30. 80:20, dan 90:10). Evaluasi dilakukan menggunakan metric akurasi, presisi, recall, dan F1-score yang diukur melalui confusion matrix. Hasil menunjukan bahwa XGBoost memberikan kinerja tertinggi dengan akurasi 99,16%, presisi 98,17%, recall 99,16%, dan F1-score 99,16% pada rasio data 90:10. Sebaliknya, Decision Tree mencapai akurasi maksimum 95% dengan F1-score sekitar 95% pada rasio data yang sama. Temuan ini menegaskan bahwa XGBoost memiliki akurasi dan kemampuan generalisasi lebih tinggi dibandingkan Decision Tree pada seluruh scenario pembagian data. Penelitian ini menyimpulkan bahwa XGBoost lebih direkomendasikan untuk pengembangan sistem pendukung keputusan berbasis data dalam mendeteksi status infeksi HIV/AIDS.

Kata kunci: *HIV/AIDS; Decision Tree; XGBoost; Machine Learning; Confusion Matrix.*

Abstract

This study compares the performance of the Decision Tree and Extreme Gradient Boosting (XGBoost) algorithms in classifying HIV/AIDS infection status. A quantitative experimental design was employed using a secondary dataset of 2,139 records and 23 attributes obtained from an open-source platform. Data preprocessing included checking for missing values, removing duplicates, detecting and handling outliers with the Interquartile Range *(IQR) method, and applying feature scaling. The models were trained and tested with three data split ratios (70:30, 80:20, and 90:10). Evaluation metrics comprised accuracy, precision, recall, and* F1-score *derived from the* confusion matrix. *The results show that XGBoost achieved the highest performance, reaching 99.16 % accuracy**,** 98.17 % precision**,** 99.16 % recall**,** and 99.16 % F1-score with a 90:10 data split. In comparison, the Decision Tree achieved a maximum accuracy of* ***95 %*** *with an* F1-score *of approximately 95 % under the same conditions. These findings confirm that XGBoost consistently outperforms the Decision Tree in accuracy and generalization across all data-split scenarios. This research concludes that XGBoost is more suitable for developing data-driven decision support systems for HIV/AIDS infection detection.*

Keywords: *HIV/AIDS; Decision Tree; XGBoost; Machine Learning; Confusion Matrix.*

1. PENDAHULUAN

HIV/AIDS merupakan masalah kesehatan global yang menyerang sistem kekebalan tubuh, khususnya sel CD4, dan dapat berkembang menjadi AIDS bila tidak ditangani (Gumarianto et al., 2022). WHO melaporkan pada 2022 terdapat 1,0–1,7 juta kasus baru HIV dengan ratusan ribu kematian, sementara di Indonesia lebih dari 500 ribu orang hidup dengan HIV pada 2023 (Rahma et al., 2024). Tantangan utama ialah deteksi dan klasifikasi status infeksi secara akurat, sehingga pemanfaatan machine learning menjadi solusi potensial dalam mendukung diagnosis dan pengambilan keputusan klinis.

Algoritma Decision Tree dan Extreme Gradient Boosting (XGBoost) merupakan metode klasifikasi dengan akurasi tinggi dalam diagnosis HIV/AIDS. Decision Tree, yang diperkenalkan melalui konsep CART oleh Breiman dkk. pada 1986, unggul dalam interpretabilitas dan visualisasi, namun rentan terhadap overfitting pada data kompleks. Sementara itu, XGBoost yang dikembangkan oleh Tianqi Chen pada 2014, menggunakan teknik boosting dan regulasi untuk mengatasi kelemahan tersebut, sehingga mampu meningkatkan akurasi dan generalisasi. Keduanya menjadi pilihan potensial tergantung pada kebutuhan interpretasi atau akurasi.

Berbagai penelitian telah mengevaluasi performa algoritma klasifikasi di bidang medis, perhotelan, hingga keuangan. Aish et al. (2025) menunjukkan bahwa XGBoost mencapai akurasi 82% dalam prediksi kekambuhan kanker tiroid, sebanding dengan Random Forest dan lebih unggul dalam mendukung keputusan klinis. Sementara itu, Munir et al. (2024) melaporkan Decision Tree memperoleh akurasi 92,04% pada klasifikasi kanker payudara, melampaui Naïve Bayes dengan 91,15%. Temuan ini menegaskan bahwa Decision Tree dan XGBoost sama-sama memiliki keunggulan akurasi di berbagai domain.

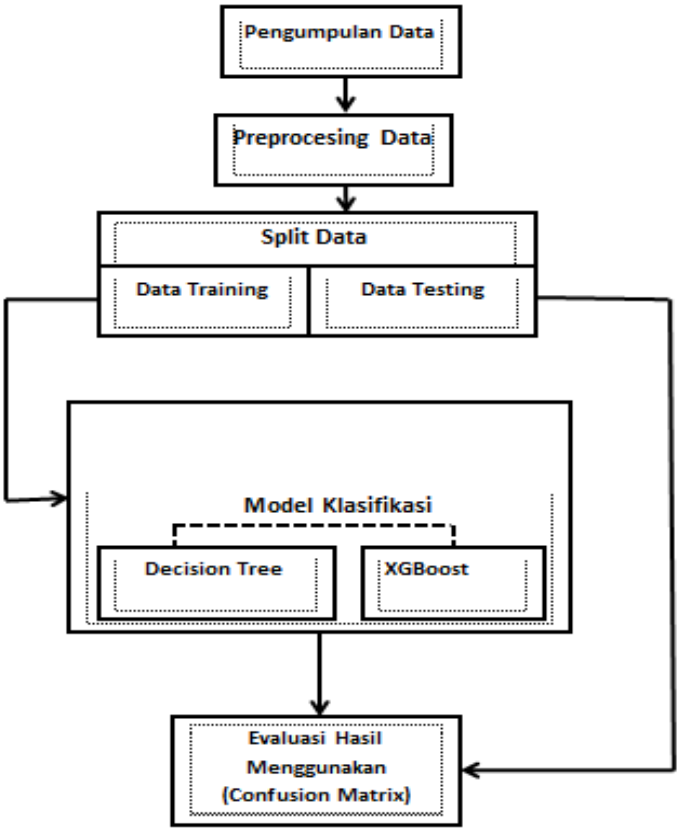
Masih sedikit penelitian yang membandingkan Decision Tree dan XGBoost dalam klasifikasi status infeksi HIV/AIDS. Karena itu, penelitian ini bertujuan mengisi celah tersebut dengan mengembangkan model klasifikasi yang lebih optimal untuk mendukung deteksi dan penanganan HIV/AIDS.

Penelitian ini diharapkan memberi pemahaman lebih dalam mengenai performa kedua algoritma serta berkontribusi pada pengembangan sistem pendukung keputusan berbasis data dalam klasifikasi penyakit.

2. METODE / ALGORITMA

2.1 Rancangan Penelitian

Dalam penelitian ini penerapan data mining menggunakan Algoritma Decision Tree dan Xtreme Gradien Boosting untuk mengklasifiksi status infeksi HIV/AIDS dilakukan beberapa tahapan sebagai berikut.



Gambar 1. Rancangan Penelitian

2.2 Pengumpulan Data

Data penelitian ini berasal dari database Kaggle dengan total 2.140 data HIV/AIDS. Dataset tersebut berisi berbagai atribut fisik dan medis, seperti waktu, pengobatan, usia, berat badan, hemofilia, homoseksual, penggunaan obat, skor karnofsky, riwayat operasi, riwayat antiretroviral, ras, jenis kelamin, gejala, status terapi, serta jumlah sel CD4 dan CD8 pada kondisi awal dan setelah 20 minggu. Data ini digunakan untuk membangun model klasifikasi status infeksi.

2.3 Preprocessing Data

Pengolahan data di Google Colab meliputi pemeriksaan missing values dengan data.isnull().sum() dan visualisasi menggunakan heatmap, serta pengecekan duplikat dengan data.duplicated().sum() yang menunjukkan hasil 0. Outlier dianalisis dengan metode IQR, lalu nilai ekstrim diganti median menggunakan apply(). Setelah pembersihan data, diperoleh dataset akhir berjumlah 2.139 baris dan 23 kolom yang siap untuk analisis lebih lanjut.

2.4 Split Data

Data dibagi menjadi training untuk membangun model dan testing untuk mengukur performa. Pada penelitian ini, pembagian data HIV/AIDS dilakukan secara acak dalam tiga percobaan dengan rasio 70:30, 80:20, dan 90:10, guna mengevaluasi pengaruh variasi proporsi data terhadap hasil klasifikasi XGBoost.

2.5 Model Klasifikasi

Dalam tahap ini dilakukan klasifikasi dataset menggunakan algoritma Decision Tree. Proses klasifikasi dengan Decision Tree kemudian dibandingkan dan disempurnakan melalui

metode Extreme Gradient Boosting (XGBoost) untuk memperoleh nilai akurasi yang lebih optimal.

2.6 Evaluasi Hasil Menggunakan Confusion Matrix

Pada tahap ini, Confusion Matrix digunakan sebagai parameter evaluasi untuk algoritma Decision Tree dan XGBoost dalam menentukan nilai akurasi. Melalui Confusion Matrix, dapat dihitung metrik evaluasi seperti akurasi, presisi, True Positive (TP), False Positive (FP), False Negative (FN), dan True Negative (TN) guna menilai kinerja model klasifikasi.

3. HASIL DAN PEMBAHASAN

3.1 Pengumpulan Data

Dalam penelitian ini data yang digunakan adalah data pasien yang terinfeksi HIV/AIDS. Data yang diperoleh dari penelitian sebanyak 2139 data terinveksi HIV/AIDS. Atribut yang digunakan sebanyak 23 atribut diantaranya Waktu, pengobatan, usia, berat badan, hemofilia, homoseksual, penggunaan obat, skor karnofsky, riwayat operasi, pengobatan 30 hari, riwayat antiretroviral, ras, jenis kelamin, stratifikasi 2, stratifikasi, gejala, sedang diterapi, berhenti terapi, jumlah sel cd4 awal, jumlah sel cd4 20 minggu, jumlah sel cd8 awal, jumlah sel cd8 20 minggu, terinfeksi. Langkah-langkah pengumpulan data yang pertama yaitu mengumpulkan data yang siap diolah dalam bentuk CSV, pada proses ini merupakan langkah awal dalam memulai proses pada python.

time	trt	age	wtkg	hemo	homo	drugs	karnof	oprior	z30	preanti	race	gender	str2	strat	symptom	treat	offtrt	cd40	cd420	cd80	cd820	infected
948	2	48	89.8128	0	0	0	100	0	0	0	0	0	0	1	0	1	0	422	477	566	324	0
1002	3	61	49.4424	0	0	0	90	0	1	895	0	0	1	3	0	1	0	162	218	392	564	1
961	3	45	88.452	0	1	1	90	0	1	707	0	1	1	3	0	1	1	326	274	2063	1893	0
1166	3	47	85.2768	0	1	0	100	0	1	1399	0	1	1	3	0	1	0	287	394	1590	966	0
1090	0	43	66.6792	0	1	0	100	0	1	1352	0	1	1	3	0	0	0	504	353	870	782	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1091	3	21	53.298	1	0	0	100	0	1	842	0	1	1	3	0	1	1	152	109	561	720	0
395	0	17	102.9672	1	0	0	100	0	1	417	1	1	1	3	0	0	1	373	218	1759	1030	0
1104	2	53	69.8544	1	1	0	90	0	1	753	1	1	1	3	0	1	0	419	364	1391	1041	0
465	0	14	60	1	0	0	100	0	0	0	0	1	0	1	0	0	0	166	169	999	1838	1
1045	3	45	77.3	1	0	0	100	0	0	0	0	1	0	1	0	1	0	911	930	885	526	0

Tabel 1. Data Penelitian

3.2 Preprocessing Data

Preprocessing data adalah tahap penting dalam proses untuk menyiapkan data mentah agar siap digunakan dalam pemodelan machine learning.

Pengecekan Outliers Setiap Kolom Numerik

```
# 1. List kolom numerik untuk pengecekan outlier
numerical_columns = ['time','age', 'wtkg','karnof','preanti', 'cd40', 'cd420', 'cd80', 'cd820']
```

```
# Menggunakan IQR untuk mendeteksi outliers
def detect_outliers_iqr(data, numerical_columns):
    outliers = {}

    for col in numerical_columns:
        Q1 = data[col].quantile(0.25)
        Q3 = data[col].quantile(0.75)
        IQR = Q3 - Q1

        # Menentukan batas bawah dan atas untuk outlier
```

```
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Menemukan outliers
                outliers[col] = data[(data[col] < lower_bound) | (data[col] > upper_bound)].shape[0]

    return outliers

# Mendeteksi outliers untuk kolom numerik
outliers_iqr = detect_outliers_iqr(data, numerical_columns)
print("Jumlah outliers per kolom menggunakan IQR:")
print(outliers_iqr)
```

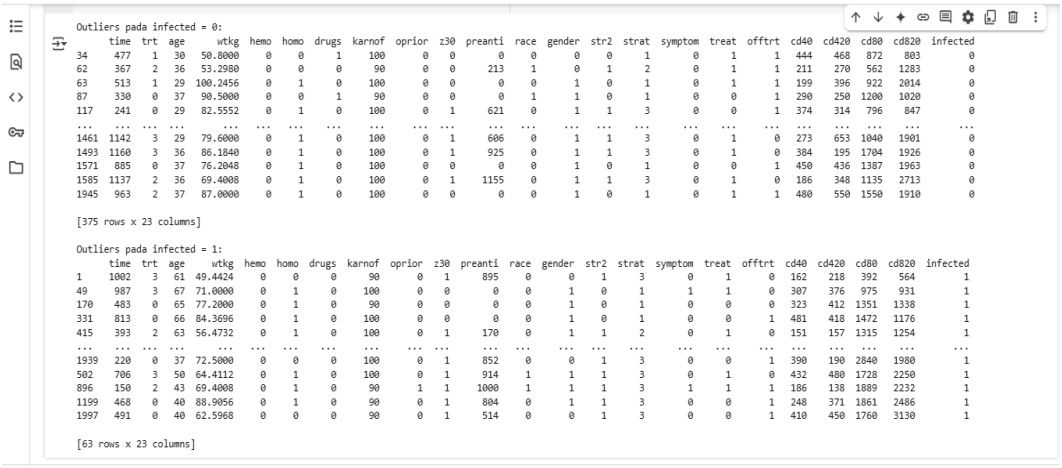
```
Jumlah outliers per kolom menggunakan IQR:
{'time': 49, 'age': 52, 'wtkg': 54, 'karnof': 9, 'preanti': 12, 'cd40': 31, 'cd420': 30, 'cd80': 71, 'cd820': 59}
```

Gambar 2. Hasil pengecekan Outlier

Memisahkan Outlier Berdasarkan Status Infected 0 dan 1

```
# Menggunakan IQR untuk memisahkan outliers berdasarkan status infected
def separate_outliers_by_infected(data, numerical_columns):
    outliers_infected_0 = pd.DataFrame()  # DataFrame untuk outliers pada infected = 0
    outliers_infected_1 = pd.DataFrame()  # DataFrame untuk outliers pada infected = 1

    # Pisahkan data berdasarkan status infected (0 atau 1)
    data_infected_0 = data[data['infected'] == 0]
    data_infected_1 = data[data['infected'] == 1]

    # Fungsi untuk mendeteksi outliers dengan IQR
    def detect_outliers(data_subset):
        outliers_data = pd.DataFrame()

        for col in numerical_columns:
            Q1 = data_subset[col].quantile(0.25)
            Q3 = data_subset[col].quantile(0.75)
            IQR = Q3 - Q1

            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            # Menemukan baris yang mengandung outliers pada kolom tersebut
                    outliers_col = data_subset[(data_subset[col] < lower_bound) | (data_subset[col] > upper_bound)]
```

```
        # Menambahkan baris yang mengandung outliers pada kolom tersebut ke DataFrame outliers_data
        outliers_data = pd.concat([outliers_data, outliers_col])

    # Menghapus duplikat baris
    return outliers_data.drop_duplicates()

# Mendeteksi outliers untuk data infected == 0 dan infected == 1
outliers_infected_0 = detect_outliers(data_infected_0)
outliers_infected_1 = detect_outliers(data_infected_1)

return outliers_infected_0, outliers_infected_1

# Memisahkan outliers berdasarkan status infected
outliers_infected_0, outliers_infected_1 = separate_outliers_by_infected(data, numerical_columns)

# Menampilkan data outliers untuk masing-masing grup infected
print("Outliers pada infected = 0:")
print(outliers_infected_0)
print("\nOutliers pada infected = 1:")
print(outliers_infected_1)
```



```
Outliers pada infected = 0:
      time  trt  age    wtkg  hemo  homo  drugs  karnof  oprior  z30  preanti  race  gender  str2  strat  symptom  treat  offtrt  cd40  cd420  cd80  cd820  infected
34     477    1   30  50.8000     0     0      1     100       0    0        0     0       0     1      1        0      1       1   444    468   872    803         0
62     367    2   36  53.2980     0     0      0      90       0    0      213     1       0     1      2        0      1       1   211    270   562   1283         0
63     513    1   29 100.2456     0     1      0     100       0    0        0     0       1     0      1        0      1       1   199    396   922   2014         0
87     330    0   37  90.5000     0     0      1      90       0    0        0     1       1     0      1        0      0       1   290    250  1200   1020         0
117    241    0   29  82.5552     0     1      0     100       0    1      621     0       1     1      3        0      0       1   374    314   796    847         0
...    ...  ...  ...      ...   ...   ...    ...     ...     ...  ...      ...   ...     ...   ...    ...      ...    ...     ...   ...    ...   ...    ...       ...
1461  1142    3   29  79.6000     0     1      0     100       0    1      606     0       1     1      3        0      1       0   273    653  1040   1901         0
1493  1160    3   36  86.1840     0     1      0     100       0    1      925     0       1     1      3        0      1       0   384    195  1704   1926         0
1571   885    0   37  76.2048     0     1      0     100       0    0        0     0       1     0      1        0      0       1   450    436  1387   1963         0
1585  1137    2   36  69.4008     0     1      0     100       0    1     1155     0       1     1      3        0      1       0   186    348  1135   2713         0
1945   963    2   37  87.0000     0     1      0     100       0    0        0     0       1     0      1        0      1       1   480    550  1550   1910         0

[375 rows x 23 columns]

Outliers pada infected = 1:
      time  trt  age    wtkg  hemo  homo  drugs  karnof  oprior  z30  preanti  race  gender  str2  strat  symptom  treat  offtrt  cd40  cd420  cd80  cd820  infected
1     1002    3   61 49.4424     0     0      0      90       0    1      895     0       0     1      3        0      1       0   162    218   392    564         1
49     987    3   67 71.0000     0     1      0     100       0    0        0     0       1     0      1        1      1       0   307    376   975    931         1
170    483    0   65 77.2000     0     1      0      90       0    0        0     0       1     0      1        0      0       0   323    412  1351   1338         1
331    813    0   66 84.3696     0     1      0     100       0    0        0     0       1     0      1        0      0       1   481    418  1472   1176         1
415    393    2   63 56.4732     0     1      0     100       0    1      170     0       1     1      2        0      1       0   151    157  1315   1254         1
...    ...  ...  ...      ...   ...   ...    ...     ...     ...  ...      ...   ...     ...   ...    ...      ...    ...     ...   ...    ...   ...    ...       ...
1939   220    0   37 72.5000     0     0      0     100       0    1      852     0       0     1      3        0      0       1   390    190  2840   1980         1
502    706    3   50 64.4112     0     1      0     100       0    1      914     1       1     1      3        0      1       0   432    480  1728   2250         1
896    150    2   43 69.4008     0     1      0      90       1    1     1000     1       1     1      3        1      1       1   186    138  1889   2232         1
1199   468    0   40 88.9056     0     1      0      90       0    1      804     0       1     1      3        0      0       1   248    371  1861   2486         1
1997   491    0   40 62.5968     0     0      0      90       0    1      514     0       0     1      3        0      0       1   410    450  1760   3130         1

[63 rows x 23 columns]
```

Gambar 3. Hasil Memisahkan Outlier Berdasarkan Status Infected 0 dan 1

Mengubah Data Outlier ke Dalam Bentuk Median

```
def replace_outliers_with_median(data, numerical_columns):
    # Copy data untuk menghindari perubahan langsung
    data_cleaned = data.copy()

    # Pisahkan data berdasarkan status infected
    for infected_value in [0, 1]:
        subset = data_cleaned[data_cleaned['infected'] == infected_value]

        for col in numerical_columns:
            Q1 = subset[col].quantile(0.25)
```

```
        Q3 = subset[col].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        median_value = subset[col].median()

        # Ganti nilai outlier dengan median hanya untuk baris dengan infected
= infected_value
        mask = (data_cleaned['infected'] == infected_value) & (
                (data_cleaned[col] < lower_bound) | (data_cleaned[col] >
upper_bound)
        )
        data_cleaned.loc[mask, col] = median_value

    return data_cleaned
data_cleaned = replace_outliers_with_median(data, numerical_columns)
 outliers_after_cleaning = detect_outliers_iqr(data_cleaned, numerical_columns)
 print("Jumlah outliers setelah penggantian dengan median:")
 print(outliers_after_cleaning)
```

```
Jumlah outliers setelah penggantian dengan median:
{'time': 274, 'age': 11, 'wtkg': 12, 'karnof': 0, 'preanti': 4, 'cd40': 5, 'cd420': 8, 'cd80': 18, 'cd820': 25}
<ipython-input-18-69a8f6662ca3>:23: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future version of pandas. Value '88.5' has dtype incompatible with int64,
  data_cleaned.loc[mask, col] = median_value
<ipython-input-18-69a8f6662ca3>:23: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future version of pandas. Value '880.5' has dtype incompatible with int64,
  data_cleaned.loc[mask, col] = median_value
```

Gambar 4. Hasil Mengubah Data Outlier Kedalam Bentuk Median

Normalisasi Data

```
from sklearn.preprocessing import MinMaxScaler

# Inisialisasi scaler
scaler = MinMaxScaler()

# Normalisasi hanya kolom numerik
data_normalized = data_cleaned.copy()
data_normalized[numerical_columns]                                    =
scaler.fit_transform(data_cleaned[numerical_columns])

# Tampilkan contoh hasil
print(data_normalized.head())
```

	time	trt	age	wtkg	hemo	homo	drugs	karnof	oprior	z30	preanti	race	gender	str2	strat	symptom	treat	offtrt	cd40	cd420	cd80	cd820	infected
0	0.767461	2	0.765957	0.744052	0	0	0	1.0	0	0	0.000000	0	0	0	1	0	1	0	0.586667	0.610556	0.254598	0.107759	0
1	0.811832	3	0.489362	0.102379	0	0	0	0.5	0	1	0.432158	0	0	1	3	0	1	0	0.153333	0.241084	0.170378	0.237069	1
2	0.778143	3	0.702128	0.722422	0	1	1	0.5	0	1	0.341381	0	1	1	3	0	1	1	0.426667	0.320970	0.406825	0.953125	0
3	0.946590	3	0.744681	0.671954	0	1	0	1.0	0	1	0.675519	0	1	1	3	0	1	0	0.361667	0.492154	0.750242	0.453664	0
4	0.884141	0	0.659574	0.376352	0	1	0	1.0	0	1	0.652825	0	1	1	3	0	0	0	0.723333	0.433666	0.401742	0.354526	0

Gambar 5. Hasil Normalisasi Data

3.3 Split Data

Langkah-langkah untuk menentukan data yang akan dianalisis pada algoritma Decisien Tree dan Xtreme Gradien Boosting terlebih dahulu menentukan data training dan data testing. Data training digunakan untuk membentuk model data mining sedangkan data testing digunakan untuk mengetahui performa model pada proses testing. Penentuan data training dan data testing pada data ekspor dilakukan sebanyak tiga kali dengan presentasi berbeda yaitu : 70% dan 30%, 80% dan 20%, 90% dan 10%.

3.4 Evaluasi Hasil Menggunakan Confusion Matrix

3.5.1 Confusion Matrix Decision Tree (Split Data 70:30)

```
# 3. Memprediksi data testing (30%)
y_pred_30 = dt_model.predict(X_test_30)

# 4. Evaluasi model
print("Evaluasi Decision Tree dengan parameter tuning dan rasio 70% training - 30% testing:")
print("Akurasi:", accuracy_score(y_test_30, y_pred_30))
print("Confusion Matrix:")
print(confusion_matrix(y_test_30, y_pred_30))
print("Classification Report:")
print(classification_report(y_test_30, y_pred_30))
```

Tabel 2. Confusion Matriks 70:30

	Prediksi: 0	**Prediksi: 1**
Aktual: 0 (Negatif)	449 (TN)	37 (FP)
Aktual: 1 (Posetif)	24 (FN)	132 (TP)

Perhitungan Accuracy: $\frac{132 + 449}{132+449+24} = \frac{581}{642} = 0.905$

3.5.2 Confusion Matrix Decision Tree (Split Data 80:20)

```
# 3. Memprediksi data testing (20%)
y_pred_20 = dt_model.predict(X_test_20)

# 4. Evaluasi model
print("Evaluasi Decision Tree dengan parameter tuning dan rasio 80% training - 20% testing:")
print("Akurasi:", accuracy_score(y_test_20, y_pred_20))
print("Confusion Matrix:")
print(confusion_matrix(y_test_20, y_pred_20))
print("Classification Report:")
print(classification_report(y_test_20, y_pred_20))
```

Tabel 3. Confusion Matriks 80:20

	Prediksi: 0	**Prediksi: 1**
Aktual: 0 (Negatif)	312 (TN)	15 (FP)
Aktual: 1 (Posetif)	15 (FN)	86 (TP)

Perhitungan Accuracy: $\frac{86+312}{86+312+15+15} = \frac{398}{428}$ =0.930

3.5.3 Confusion Matrix Decision Tree (Split Data 90:10)

```
# 3. Memprediksi data testing (10%)
y_pred_10 = dt_model.predict(X_test_10)

# 4. Evaluasi model
print("Evaluasi Decision Tree dengan parameter tuning dan rasio 90% training - 10% testing:")
print("Akurasi:", accuracy_score(y_test_10, y_pred_10))
print("Confusion Matrix:")
print(confusion_matrix(y_test_10, y_pred_10))
print("Classification Report:")
print(classification_report(y_test_10, y_pred_10))
```

Tabel 4. Confusion Matriks 90:10

	Prediksi: 0	**Prediksi: 1**
Aktual: 0 (Negatif)	151 (TN)	10 (FP)
Aktual: 1 (Posetif)	12 (FN)	41 (TP)

Perhitungan Accuracy: $\frac{41 + 151}{41+151+10+12} = \frac{192}{214}$ =0.897 (89.7%)

4.5.4 Confusion Matrix XGBoost (Split Data 70:30)

```
# Prediksi pada data testing
y_pred = xgb_model.predict(X_test_30)

# Evaluasi performa model
accuracy = accuracy_score(y_test_30, y_pred)
classification_report_result = classification_report(y_test_30, y_pred, zero_division=0)
conf_matrix = confusion_matrix(y_test_30, y_pred)

# Menampilkan hasil evaluasi
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report_result)
print("Confusion Matrix:\n", conf_matrix)
```

Tabel 5. Confusion Matriks 70:30

	Prediksi: 0	**Prediksi: 1**
Aktual: 0 (Negatif)	471 (TN)	15 (FP)
Aktual: 1 (Positif)	32 (FN)	124 (TP)

Perhitungan Accuracy: $\frac{124+ 471}{124+471+15+32} = \frac{595}{642}$ =0.927

4.5.5 Confusion Matrix XGBoost (Split Data 80:20)

```
# Prediksi pada data testing
```

```
y_pred = xgb_model.predict(X_test_20)

# Evaluasi performa model
accuracy = accuracy_score(y_test_20, y_pred)
classification_report_result = classification_report(y_test_20, y_pred, zero_division=0)
conf_matrix = confusion_matrix(y_test_20, y_pred)

# Menampilkan hasil evaluasi
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report_result)
print("Confusion Matrix:\n", conf_matrix)
```

Tabel 6. Confusion Matriks 80:20

	Prediksi: 0	**Prediksi: 1**
Aktual: 0 (Negatif)	318 (TN)	9 (FP)
Aktual: 1 (Positif)	20 (FN)	81 (TP)

Perhitungan Accuracy: $\frac{81+318}{81+318+9+20} = \frac{399}{428} = 0.933$

4.5.6 Confusion Matrix XGBoost (Split Data 90:10)

```
# Prediksi pada data testing
y_pred = xgb_model.predict(X_test_10)

# Evaluasi performa model
accuracy = accuracy_score(y_test_10, y_pred)
classification_report_result = classification_report(y_test_10, y_pred, zero_division=0)
conf_matrix = confusion_matrix(y_test_10, y_pred)

# Menampilkan hasil evaluasi
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report_result)
print("Confusion Matrix:\n", conf_matrix)
```

Tabel 7. Matriks 85:15

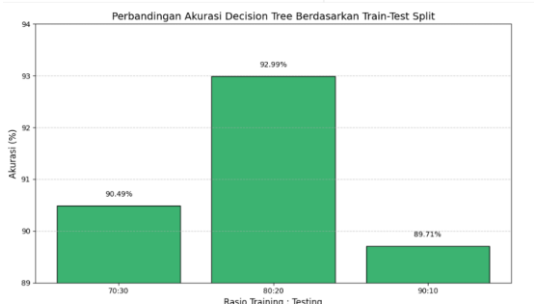
	Prediksi: 0	**Prediksi: 1**
Aktual: 0 (Negatif)	156 (TN)	5 (FP)
Aktual: 1 (Positif)	14 (FN)	39 (TP)

Perhitungan Accuracy: $\frac{39+156}{39+156+5+14} = \frac{195}{214} = 0.912$

3.5 Penerapan Algoritma Decision Tree

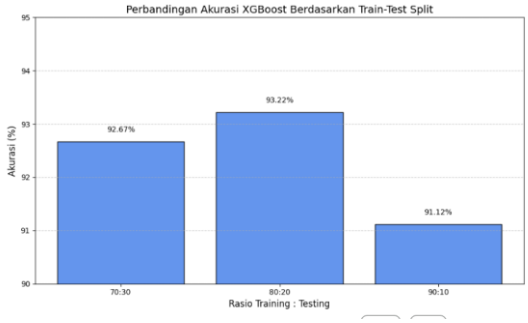
Pada penelitian ini, algoritma Decision Tree diterapkan untuk melakukan klasifikasi terhadap data HIV/AIDS dengan tujuan memprediksi status infeksi pasien berdasarkan sejumlah atribut medis. Proses pelatihan model dilakukan dengan membagi data menjadi data latih dan data uji menggunakan berbagai rasio, mulai dari 70:30 hingga 90:10, guna mengevaluasi pengaruh proporsi data terhadap performa model. Hasil evaluasi menunjukkan

bahwa Decision Tree mampu memberikan akurasi yang cukup tinggi, berkisar antara 89% hingga 92%,



Gambar 6. Perbandingan Akurasi Decision Tree Berdasarkan Train Test Split

Selain Decision Tree, penelitian ini juga menerapkan XGBoost sebagai metode klasifikasi pada dataset HIV/AIDS. Algoritma ensemble berbasis gradient boosting ini mampu mengatasi overfitting dan meningkatkan akurasi prediksi. Pengujian dilakukan dengan rasio data 70:30 hingga 90:10 menggunakan parameter seperti *learning rate*, *n_estimators*, dan *max_depth* untuk memperoleh performa terbaik. Hasilnya, XGBoost menunjukkan akurasi lebih tinggi dibandingkan Decision Tree (92,67%–91,12%) serta memberikan prediksi lebih stabil pada data dengan distribusi kelas tidak seimbang, berdasarkan evaluasi akurasi, precision, recall, dan f1-score.



Gambar 7. Perbandingan Akurasi XGBoost Berdasarkan Train Test Split

4. KESIMPULAN

Hasil pengujian menunjukkan bahwa algoritma XGBoost secara umum lebih unggul dibandingkan Decision Tree dalam klasifikasi HIV/AIDS. XGBoost memberikan akurasi lebih tinggi dan stabil (tertinggi 93,22% pada rasio 80:20) dibandingkan Decision Tree (92,99% pada rasio yang sama), serta konsisten menghasilkan precision di atas 0,85 untuk kelas positif HIV. Sementara itu, Decision Tree lebih sensitif dalam mengenali kasus positif (recall lebih tinggi pada rasio 65:35 hingga 85:15), meski berisiko menimbulkan false positive. Dari sisi f1-score, XGBoost unggul pada kelas negatif dengan nilai di atas 0,95, sedangkan untuk kelas positif kedua algoritma bersaing, meskipun XGBoost lebih stabil secara keseluruhan. Dengan demikian, XGBoost dinilai lebih seimbang dan efektif, sedangkan Decision Tree lebih kuat dalam mendeteksi kasus positif.

5. REFERENSI

[1] Anwar, S., Faujiah, R. L., Hartati, T., & Tohidi, E. (2024). *Jurnal Informatika dan Rekayasa Perangkat Lunak.* Klasifikasi Penentuan Tingkat Penyakit Demam Berdarah dengan menggunakan Algoritma Naïve Bayes *(Studi Kasus Puskesmas Nagreg)*. *6*(1), 205–212.

[2] Dava Maulana, M., Id Hadiana, A., & Rakhmat Umbara, F. (2024). Algoritma Xgboost Untuk Klasifikasi Kualitas Air Minum. JATI (Jurnal Mahasiswa Teknik Informatika), 7(5), 3251–3256.
[3] Gumarianto, R. S., Lardo, S., & Chairani, A. (2022). Hubungan Antara Hitung Jumlah Cd4 Dengan Kejadian Wasting Syndrome Pada Pasien Hiv/Aids Di Rspad Gatot Soebroto Periode Januari-Desember 2020. Jurnal Kedokteran Dan Kesehatan : Publikasi Ilmiah Fakultas Kedokteran Universitas Sriwijaya, 9(2), 133–142.
[4] Jan Melvin Ayu Soraya Dachi, & Pardomuan Sitompul. (2023). Analisis Perbandingan Algoritma XGBoost dan Algoritma Random Forest Ensemble Learning pada Klasifikasi Keputusan Kredit. Jurnal Riset Rumpun Matematika Dan Ilmu Pengetahuan Alam, 2(2), 87–103.
[5] No, V., Ais, S. R., & Sanjaya, U. P. (2025). Edumatic : Jurnal Pendidikan Informatik. Perbandingan Algoritma Random Forest , XGBoost , dan Logistic Regression untuk Prediksi Risiko Kekambuhan Kanker Tiroid. 9(1), 324–332.
[6] Rahma, G., Yulia, Y., & Handiny, F. (2024). Determinan Kejadian HIV AIDS pada Populasi Kunci di Indonesia : Systematic Review. *Jik Jurnal Ilmu Kesehatan*, *8*(1), 158.
[7] Soelistijadi, R., Eniyati, S., Stikubank, U., Studi, P., Rekayasa, T., Grafis, M., Stikubank, U., & Tengah, J. (2024). Pemodelan Prediktif Menggunakan Metode Ensemble Learning XGBoost dalam Peningkatan Akurasi Klasifikasi Penyakit Ginjal. 5(4), 1866–1875.
[8] Wijaya, H. D., & Dwiasnati, S. (2020). Implementasi Data Mining dengan Algoritma Naïve Bayes pada Penjualan Obat. Jurnal Informatika, 7(1), 1–7.
[9] Zulfahmi, I., Syahputra, H., Naibaho, S. I., Maulana, M. A., & Sinaga, E. P. (2023). Perbandingan Algoritma Support Vector Machine (SVM) dan Decision Tree Untuk Deteksi Tingkat Depresi Mahasiswa. *Bina Insani Ict Journal*, *10*(1), 52.