IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

# Perbandingan Prediksi Heart Failure Berdasarkan Gejala dengan Algoritma Logistic Regression, SVM, dan Decision Tree (Mei 2024)

**Christopher Kenneth David[1], Bintang Muhammad Ramdhan[2], Valen Claudia Chuardi[3], Nabila Az Zahra Paramita[4], Kenny Budiarso Lawson[5]**
[1-5] Multimedia Nusantara University 15810, Faculty of Engineering and Informatics, Indonesia
Email: christopher.kenneth1@student.umn.ac.id [1], bintang.muhammad@student.umn.ac.id [2], valen.claudia@student.umn.ac.id [3], nabila.az@student.umn.ac.id [4], kenny.budiarso@student.umn.ac.id [5]

***ABSTRACT***-Prediksi penyakit jantung merupakan salah satu tantangan besar dalam bidang kesehatan. Dengan kemajuan teknologi dan ketersediaan data medis yang semakin meningkat, metode machine learning dapat digunakan untuk membantu dalam mendiagnosis penyakit ini secara lebih cepat dan akurat. Penelitian ini bertujuan untuk mengevaluasi performa beberapa algoritma klasifikasi dalam memprediksi penyakit jantung menggunakan dataset dari Kaggle. Algoritma yang diuji meliputi Logistic Regression, Support Vector Machine (SVM), Decision Tree, dan Random Forest. Hasil penelitian menunjukkan bahwa algoritma Random Forest memiliki performa terbaik dengan akurasi sebesar 87%, precision, recall, dan F1-score masing-masing sebesar 0.88. Model Random Forest kemudian di-deploy menggunakan Spyder dari Anaconda untuk memprediksi risiko penyakit jantung berdasarkan input data pasien. Implementasi ini diharapkan dapat membantu dalam mendeteksi dini risiko serangan jantung dan memberikan intervensi medis yang tepat waktu.

***INDEX TERMS*** *Serangan jantung, prediksi, logistic regression, fitur kesehatan pasien, machine learning.*

## BAB 1. Latar Belakang

Serangan jantung adalah kondisi serius yang terjadi ketika jantung tidak dapat memompa darah sebagaimana mestinya. Penyakit ini memiliki banyak gejala yang bisa digunakan sebagai indikator. Oleh karena itu, penting untuk mengidentifikasi fitur-fitur kesehatan yang berkorelasi dengan serangan jantung.

Serangan jantung merupakan masalah kesehatan masyarakat yang serius, terutama pada populasi lansia, yang mengakibatkan tingginya angka kematian, morbiditas, dan biaya pengobatan [1]. Kasus dengan fraksi ejeksi yang dipertahankan yang tidak memerlukan perawatan khusus menjadi lebih umum dalam campuran kasus. Rawat inap terus menjadi hal yang umum dan tingkat rawat inap kembali meningkat meskipun ada upaya untuk menurunkan angka kematian. Untuk mengurangi angka rawat inap, diperlukan definisi yang menyeluruh mengenai faktor penyebab rawat inap ulang dan model perawatan inovatif yang berpusat pada pasien yang memanfaatkan sumber daya masyarakat.

*Machine Learning* adalah sub-bidang kecerdasan buatan (AI) yang berkaitan dengan pembuatan algoritma yang belajar dari data. Algoritma ini dapat meningkatkan kinerjanya pada tugas tertentu tanpa diprogram secara eksplisit. Anggap saja seperti mengajari komputer untuk "belajar" dari contoh-contoh, memungkinkannya membuat prediksi, mengklasifikasikan data, atau bahkan menghasilkan konten baru [2].

Penelitian ini membatasi analisis pada dataset yang terdiri dari berbagai fitur kesehatan pasien, seperti gula darah, kolesterol, dan resting heart rate, untuk memprediksi kemungkinan terkena serangan jantung. Model yang digunakan dalam penelitian ini adalah *Logistic Regression*.

## BAB 2. Telaah Literatur

### 2.1 Logistic Regression

*Logistic Regression* adalah teknik statistik untuk menganalisis dataset di mana terdapat satu atau lebih variabel independen yang menentukan hasil. Hasilnya diukur dengan variabel dependen yang biner. Tidak seperti regresi linier, yang memprediksi hasil yang kontinu, regresi logistik menggunakan fungsi logistik (atau fungsi sigmoid) untuk memadatkan output antara 0 dan 1, yang mewakili probabilitas. Model ini memperkirakan hubungan antara satu atau lebih variabel independen (fitur) dan probabilitas hasil biner. Parameter model regresi logistik biasanya diestimasi menggunakan estimasi kemungkinan maksimum, yang bertujuan untuk menemukan koefisien yang memaksimalkan kemungkinan mengamati data yang diberikan. Dalam praktiknya, regresi logistik banyak digunakan di berbagai bidang seperti kedokteran, keuangan, dan ilmu sosial untuk memprediksi hasil kategorikal

berdasarkan fitur input. Rumus dasar logistic regression adalah:

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_n X_n)}}$$

Di mana:

- $P(Y=1|X)$ adalah probabilitas variabel dependen $Y$ bernilai 1 diberikan variabel independen $X$.
- $\beta 0, \beta 1, ..., \beta n$ adalah koefisien yang diestimasi dari data.
- $X1, X2, ..., Xn$ adalah variabel independen.

Keunggulan dari logistic regression termasuk interpretasi yang mudah dan implementasi yang efisien, namun kelemahannya adalah tidak dapat menangani hubungan non-linear antara variabel independen dan dependen tanpa transformasi atau penambahan variabel baru.

### 2.2 SVM

*Support Vector Machine (SVM)* adalah teknik supervised learning yang digunakan untuk klasifikasi dan regresi. SVM mencari hyperplane optimal yang memisahkan dua kelas dalam fitur space. Dalam teori pembelajaran statistik Vapnikis, *Support Vector Machine (SVM)* pertama kali muncul sebagai pengklasifikasi margin optimal pada tahun 1990-an. SVM telah terbukti lebih efektif daripada metode alternatif ketika digunakan untuk menangani masalah analisis data dunia nyata. SVM mengurangi risiko empiris dengan beroperasi dalam kerangka kerja teori regularisasi. Aktivitas berskala besar seperti pemrosesan teks dan bioinformatika dapat mengambil manfaat dari solusi yang jarang digunakan untuk tantangan klasifikasi dan regresi. SVM ini bekerja dengan metode mencari hyperplane optimal yang memisahkan dua kelas dalam fitur space. Hyperplane optimal adalah yang memiliki margin terluas, yaitu jarak terbesar antara data titik dari kedua kelas yang paling dekat dengan hyperplane. Rumus dasar dari SVM adalah:

$$f(x) = \text{sign}(w \cdot x + b)$$

Di mana:

- $w$ adalah vektor bobot.
- $x$ adalah vektor fitur input.
- $b$ adalah bias.

Untuk kasus linear, SVM berfungsi dengan baik namun dapat ditingkatkan menggunakan kernel trick untuk menangani kasus non-linear dengan memetakan data ke dimensi yang lebih tinggi di mana hyperplane dapat ditemukan. SVM dikenal kuat terhadap overfitting, terutama dalam ruang fitur yang tinggi.

### 2.3 Decision Tree

*Decision Tree* adalah metode predictive modeling yang menggunakan graf atau model *tree-like* dari keputusan dan kemungkinan konsekuensi. Alat ini sangat berguna untuk klasifikasi dan regresi. Untuk menentukan korelasi data dan meramalkan data yang belum terlihat secara akurat, mereka membangun topologi pohon. Setiap simpul keputusan dalam kumpulan data memiliki kemampuan untuk bercabang ke sub kelompok yang lebih jauh. Percabangan dimulai oleh node akar, dan berakhir ketika node daun terakhir berpisah. Model hirarki dari atas ke bawah diwakili oleh *decision tree*. Proses ini menggunakan konsep Information Gain. Contoh formula untuk Information Gain adalah:

$$IG(Y, X) = H(Y) - H(Y|X)$$

Di mana:

- $H(Y)$ adalah entropi awal dari variabel dependen $Y$Y.
- $H(Y|X)$ adalah entropi dari $Y$ setelah memisahkan data berdasarkan variabel independen $X$.

Keunggulan dari decision tree termasuk interpretasi yang mudah dan fleksibilitas dalam menangani data kategorikal dan numerik, namun kelemahannya adalah rentan terhadap overfitting dan performa yang sensitif terhadap perubahan kecil dalam data.

### 2.4 Random Forest

Random Forest adalah ensemble learning method yang digunakan untuk klasifikasi, regresi, dan tugas lainnya. Algoritma ini menggabungkan beberapa pohon keputusan yang dilatih pada sub-sampel yang berbeda dari dataset dan menggunakan rata-rata atau voting untuk meningkatkan akurasi prediksi. Random Forest mengurangi overfitting dengan menggabungkan prediksi dari beberapa pohon yang berbeda, yang masing-masing mungkin saja overfitting ke subset data yang dilatih.

Keunggulan dari Random Forest meliputi:

- **Akurasi Tinggi**: Menggunakan beberapa pohon keputusan meningkatkan akurasi prediksi dan mengurangi risiko overfitting.
- **Robust terhadap Noise**: Mampu menangani data yang memiliki noise dengan baik.
- **Feature Importance**: Dapat digunakan untuk mengidentifikasi fitur yang paling penting dalam dataset.

## BAB 3. Metodologi Penelitian

### 3.1. Eksplorasi Dataset

Dataset yang digunakan yaitu dataset bernama “Heart.csv” yang bersumber dari halaman web “kaggle.com.” Dataset ini berisikan data mengenai variabel indikator yang akan digunakan dalam memprediksi kemungkinan pasien mengalami gagal jantung. Variabel target dijadikan sebagai label Supervised Learning yang menjadi acuan variabel indikator dalam proses Machine

Learning yang menunjukkan apakah pasien mengalami gagal jantung atau tidak.

[14, 15]

Dataset yang digunakan dalam penelitian ini terdiri dari 1025 data pasien dengan 14 kolom variabel, yaitu: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, target.

Flow algoritma yang digunakan dalam penelitian ini meliputi:

1. Pemilihan fitur yang relevan.
2. Pembagian dataset menjadi training set dan testing set.
3. Pembangunan model Logistic Regression, Support Vector Machine (SVM), dan Decision Tree.
4. Evaluasi performa model algoritma menggunakan metrik akurasi, precision, recall, dan f1-score.
5. Validasi model menggunakan K-Fold Cross Validation.

### A. Eksplorasi Data

Eksplorasi data pada penelitian ini menggunakan metode Exploratory Data Analysis (EDA). Pada tahapan awal, dilakukan proses memasukkan import libraries dan membaca dataset ke Jupyter Notebook. Berikut potongan hasil proses memasukkan data dan libraries ke dalam Jupyter Notebook:

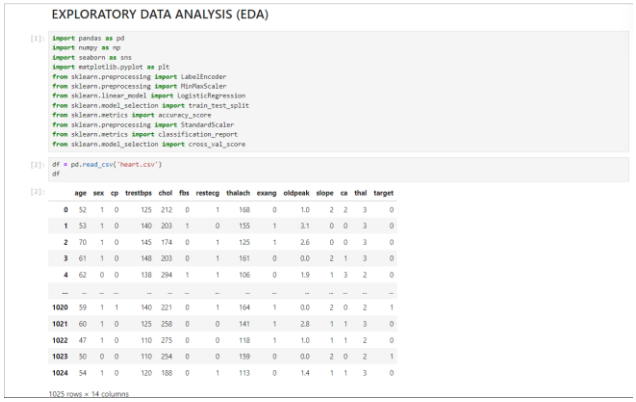


***Fig.1; Import libraries & dataset***

### B. Analisis Informasi Dataset

Kemudian dilakukan proses analisis tipe dan informasi dataset, seperti tipe data dan nilai-nilai statistik dari dataset. Proses ini dilakukan dengan syntax "df.info() dan df.describe()". Berikut figur dari proses analisis statistik dataset:

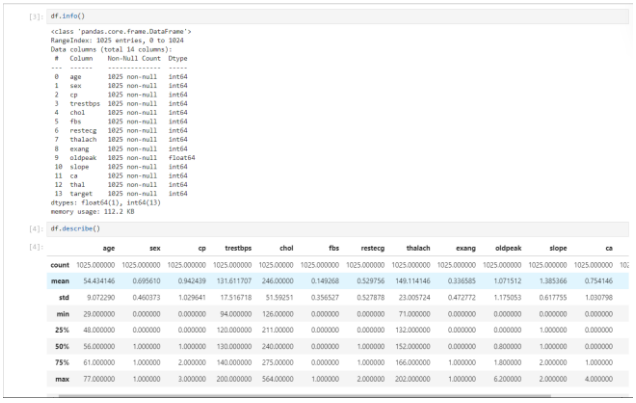


***Fig.2; Analisis statistik dan informasi dataset***

### C. Pengecekkan Missing Value Dataset

Setelah menganalisis statistik dataset, dilakukan proses data cleansing dengan mengecek nilai dataset yang hilang (Missing Value atau data null). Berikut figur yang menunjukkan data missing value/data null pada dataset tersebut:

```
df.isna().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

***Fig.3; Pengecekkan Missing Value dataset***

```
# cek data null
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Data')
plt.show()
```


***Fig.4; Hasil visualisasi pengecekkan Missing Value dataset***

Dari proses pengecekkan missing value, terbukti bahwa dataset yang digunakan pada penelitian ini bersifat bersih, atau tidak memiliki nilai null / missing value data setiap barisan kolomnya. Hal tersebut menunjukkan data pada dataset yang telah dijadikan dataframe bersifat valid sepenuhnya.

### D. Visualisasi Distribusi Nilai Variabel

Setelah data dipastikan valid, kemudian dilakukan proses visualisasi distribusi nilai pada beberapa variabel dataframe. Hal ini guna mengetahui tingkat distribusi nilai pada tiap-tiap variabel yang digunakan dalam proses training model algoritma.

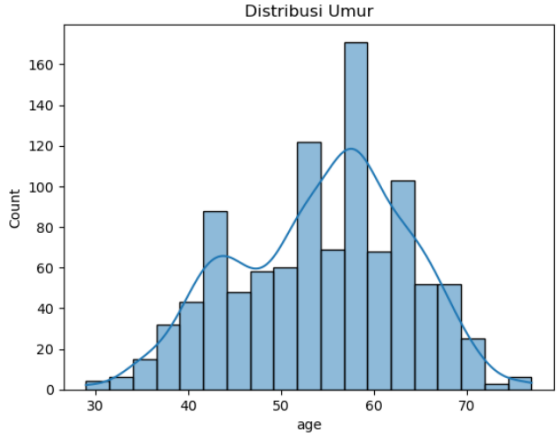


**Fig.5; Distribusi Umur**

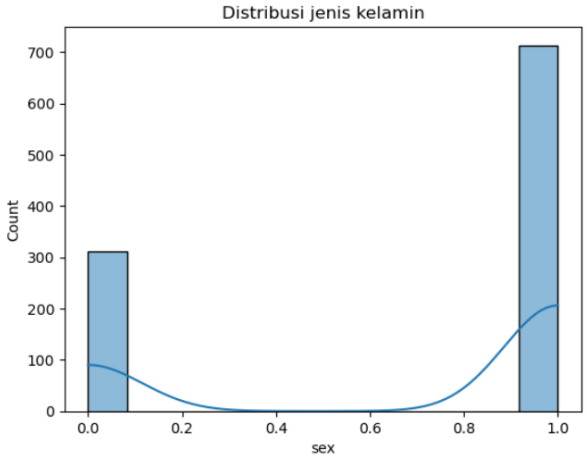


**Fig.6; Distribusi Jenis Kelamin**

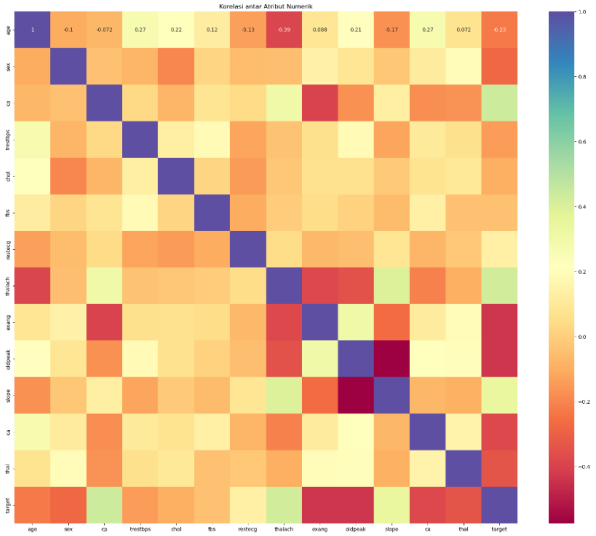


**Fig.7; Heatmap**

```
# Kolom yang tidak memiliki hubungan di drop

cols_to_drop = ['slope', 'ca', 'thal']
df_dropped = df.drop(cols_to_drop, axis=1)

df_dropped
```

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | target |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 0 |
| 1 | 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 |
| 2 | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 |
| 3 | 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 0 |
| 4 | 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1020 | 59 | 1 | 1 | 140 | 221 | 0 | 1 | 164 | 1 | 0.0 | 1 |
| 1021 | 60 | 1 | 0 | 125 | 258 | 0 | 0 | 141 | 1 | 2.8 | 0 |
| 1022 | 47 | 1 | 0 | 110 | 275 | 0 | 0 | 118 | 1 | 1.0 | 0 |
| 1023 | 50 | 0 | 0 | 110 | 254 | 0 | 0 | 159 | 0 | 0.0 | 1 |
| 1024 | 54 | 1 | 0 | 120 | 188 | 0 | 1 | 113 | 0 | 1.4 | 0 |

1025 rows × 11 columns

**Fig.8; Data drop**

```
df_dropped.nunique()
```

```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
target        2
dtype: int64
```

```
df_dropped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  target    1025 non-null   int64
dtypes: float64(1), int64(10)
memory usage: 88.2 KB
```

**Fig.9; Baca setelah drop**

## 3.2. Data Preprocessing

Setelah memasukkan data, kemudian dilakukan proses analisis dataset:

Kemudian, dilakukan proses Data Cleansing, yaitu dengan menggunakan fitur pengecekkan dan penghapusan Missing Value pada dataset:

## 3.3. Implementasi Algoritma

Setelah dataset telah dipastikan bersih dan siap diproses, dilakukan proses persiapan algoritma yang akan diaplikasikan pada dataframe yang sudah disiapkan. Pada penelitian ini, metode penelitian dilakukan dengan menggunakan algoritma Logistic Regression, Support Vector Machine (SVM), Decision Tree, dan Random Forest.

## 3.4. Evaluasi Model

Setelah melakukan modelling pada implementasi algoritma, kemudian dilakukan evaluasi model dengan memanfaatkan evaluasi berdasarkan nilai accuracy, precision, recall, F1-score, dan confusion matrix.

# BAB 4. Implementasi dan Hasil Algoritma

## 4.1 Flow Algoritma

Flow algoritma yang digunakan dalam penelitian ini dapat diuraikan sebagai berikut:

1. **Pengumpulan Data:** Menggunakan dataset heart disease dari Kaggle.
2. **Eksplorasi dan Praproses Data:**
   - Memuat dataset dan melakukan eksplorasi awal.
   - Membersihkan data, seperti menangani nilai yang hilang dan menghapus fitur yang tidak relevan.
3. **Pembagian Dataset:** Membagi dataset menjadi data latih (training set) dan data uji (testing set) dengan rasio 80:20.
4. **Implementasi Model:** Mengimplementasikan tiga algoritma klasifikasi: Logistic Regression, Suppor Vector Machine (SVM), Decision Tree, dan Random Forest.
5. **Pelatihan dan Evaluasi Model:**
   - Melatih model menggunakan data latih.
   - Mengevaluasi model menggunakan data uji.
   - Menggunakan metrik akurasi, precision, recall, dan f1-score untuk mengevaluasi performa model.
6. **Validasi Silang:** Melakukan validasi silang (K-Fold) untuk memastikan model tidak overfitting dan mendapatkan estimasi performa yang lebih akurat.
7. **Analisis Hasil:** Membandingkan hasil dari ketiga model berdasarkan metrik evaluasi yang diperoleh.

## 4.2 Hasil Logistic Regression

```
# Membuat model regresi logistik
model = LogisticRegression(max_iter=500)

# Melatih model
model.fit(X_train, y_train)

# Memprediksi nilai target pada data testing
y_pred_lr = model.predict(X_test)

# Menghitung akurasi model
accuracy = accuracy_score(y_test, y_pred_lr)
print("Accuracy:", accuracy)

report = classification_report(y_test, y_pred_lr)
print("Classification Report:\n", report)
```

**Fig.10; Akurasi Logistic Regression**

Hasil evaluasi Logistic Regression menunjukkan:

1. Akurasi: 74%
2. Precision: 0.75
3. Recall: 0.75
4. F1-Score: 0.75

Dengan MIN MAX Cross-validation score:

1. Cross-validation scores:[0.82926829 0.84146341 0.81707317 0.82926829 0.84146341 0.76829268 0.85365854 0.7804878 0.76829268 0.80487805]
2. MIN cross-validation score: 0.7683
3. MAX cross-validation score: 0.8537
4. AVG cross-validation score: 0.8134

Dan Error rate: 0.19999999999999996

## 4.3 Hasil SVM

```
y_pred_svm = SVMclassifier.predict(X_test)  # Predict using SVC

# Calculate accuracy
accuracy_svm = accuracy_score(y_test, y_pred_svc)
print("SVM Accuracy:", accuracy_svm)

# Classification report
report_svm = classification_report(y_test, y_pred_svm)
print("SVM Classification Report:\n", report_svm)
```

**Fig.11; Akurasi SVM**

Hasil evaluasi SVM menunjukkan:

1. Akurasi: 83.4%
2. Precision: 0.84
3. Recall: 0.83
4. F1-Score: 0.83

Dengan MIN MAX Cross-validation score:

1. Cross-validation scores:[0.90243902 0.86585366 0.81707317 0.84146341 0.87804878 0.8902439 0.90243902 0.91463415 0.8902439 0.84146341]
2. MIN cross-validation score: 0.8171
3. MAX cross-validation score: 0.9146
4. AVG cross-validation score: 0.8744

Dan Error rate: 0.1658536585365854

## 4.4 Hasil Decission Tree

```
y_pred_dt = strokeTree.predict(X_test)  # Predict using decision tree

accuracy_dt = accuracy_score(y_test, y_pred_dt)
print("Decision Tree Accuracy:", accuracy_dt)

report_dt = classification_report(y_test, y_pred_dt)
print("Decision Tree Classification Report:\n", report_dt)
```

**Fig.12; Akurasi Decision Tree**

Hasil evaluasi Decision Tree menunjukkan:

1. Akurasi: 77%
2. Precision: 0.78
3. Recall: 0.78
4. F1-Score: 0.78

Dengan MIN MAX Cross-validation score:

1. Cross-validation scores: [0.84146341 0.81707317 0.85365854 0.84146341 0.79268293 0.84146341 0.80487805 0.85365854 0.90243902 0.82926829]
2. MIN cross-validation score: 0.7927
3. MAX cross-validation score: 0.9024
4. AVG cross-validation score: 0.8378

Dan Error rate: 0.224390243902439

## 4.5 Hasil Random Forest

```
y_pred_rf = strokeForest.predict(X_test)  # Predict using decision tree

accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Random Forest Classifier Accuracy:", accuracy_rf)

report_rf = classification_report(y_test, y_pred_rf)
print("Random Forest Classifier Report:\n", report_rf)
```

**Fig.13; Akurasi SVM**

Hasil evaluasi SVM menunjukkan:

5. Akurasi: 87%
6. Precision: 0.88
7. Recall: 0.88
8. F1-Score: 0.88

Dengan MIN MAX Cross-validation score:

5. Cross-validation scores:[0.91463415 0.8902439 0.86585366 0.8902439 0.91463415 0.82926829 0.92682927 0.96341463 0.8902439 0.84146341]
6. MIN cross-validation score: 0.8293

7. MAX cross-validation score: 0.9634
8. AVG cross-validation score: 0.8927

Dan Error rate: 0.12195121951219512

### 4.6 Hasil Perbandingan algoritma

Berikut adalah ringkasan hasil perbandingan dari ketiga algoritma:

| Algoritma | Akurasi | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Logistic Regression | 74% | 0.75 | 0.75 | 0.75 |
| SVM | 83.4% | 0.84 | 0.83 | 0.83 |
| Decission Tree | 77% | 0.78 | 0.78 | 0.78 |
| Random Forest | 87% | 0.88 | 0.88 | 0.88 |

Untuk perbandingan ketiga algoritma dalam visual bisa dilihat dalam **Fig.14; Precision Comparison, Fig.15; Recall Comparison, Fig.16; F1-score Comparison, Fig.17; Accuracy Comparison**

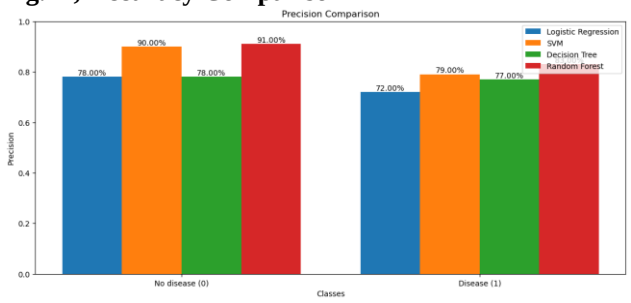


**Fig.14; Precision Comparison**

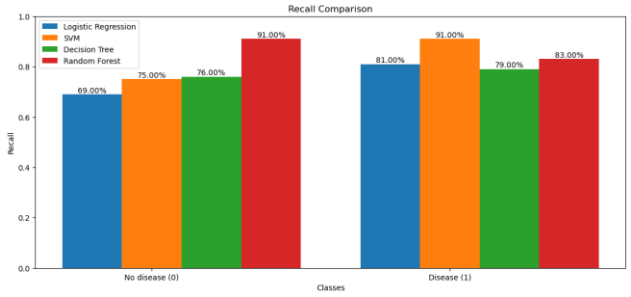


**Fig.15; Recall Comparison**

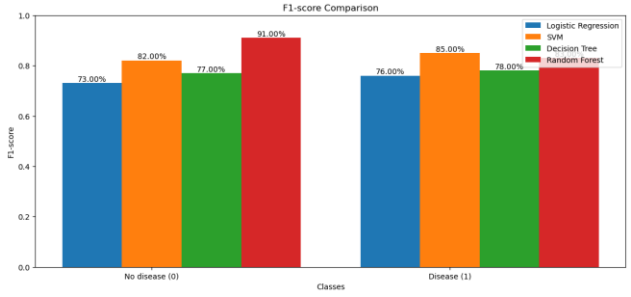


**Fig.16; F1-score Comparison**

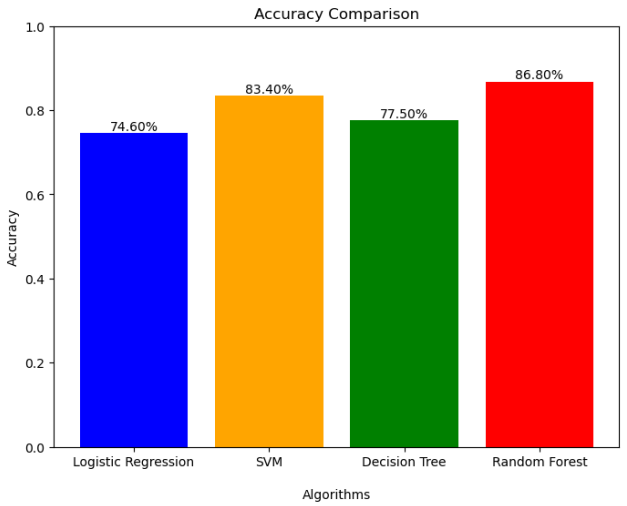


**Fig.17; Accuracy Comparison**

### 4.7 Deployment

Dikarenakan hasil model Random Forest yang menunjukkan performa paling tinggi dibandingkan dengan model lainnya, proses deployment akan menggunakan model Random Forest. Berikut adalah langkah-langkah dan hasil deploy dari model Random Forest.

**Langkah-Langkah Deployment**

1. **Simpan Model**: Model Random Forest yang telah dilatih disimpan menggunakan pickle atau joblib.
2. **Membangun Antarmuka Input**: Menggunakan Spyder dari Anaconda untuk membuat antarmuka yang menerima input dari pasien.
3. **Menerima Input Pasien**: Membuat fungsi untuk menerima input data pasien dan memberikan prediksi berdasarkan model yang telah disimpan.
4. **Tampilkan Prediksi**: Menampilkan hasil prediksi apakah pasien menderita serangan jantung atau tidak.

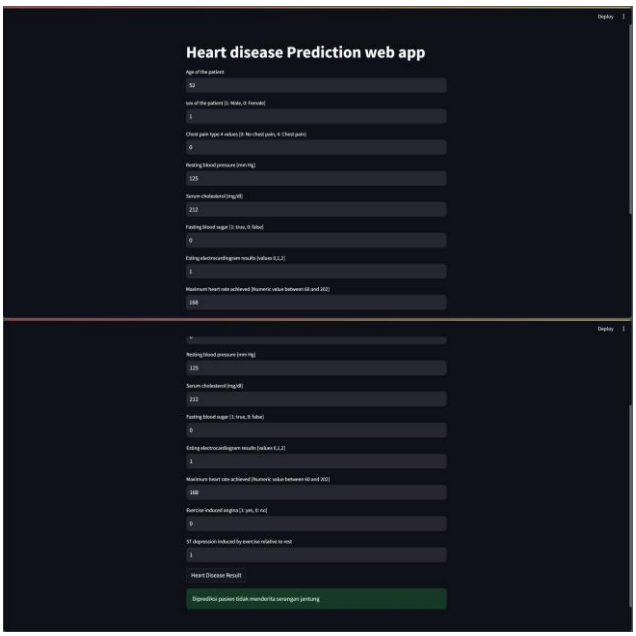


**Fig.18; Deployment**

**Hasil Deployment**

Dengan menggunakan model Algorithm Random Forest, berikut adalah hasil dari proses deployment:

**Antarmuka Web**
Pasien akan menginput data mereka melalui halaman web yang telah disediakan. Berikut adalah contoh data yang diinput oleh pasien:

1. **Age**: 52
2. **Sex**: 1 (Male)
3. **Chest Pain Type**: 0 (No chest pain)
4. **Resting Blood Pressure**: 125
5. **Serum Cholesterol**: 212
6. **Fasting Blood Sugar**: 0 (False)
7. **Resting Electrocardiogram Results**: 1
8. **Maximum Heart Rate Achieved**: 168
9. **Exercise Induced Angina**: 0 (No)
10. **ST Depression Induced by Exercise Relative to Rest**: 1.0

**Prediksi**

Berdasarkan input data pasien tersebut, hasil prediksi dari model Random Forest adalah:

**Pasien tidak menderita serangan jantung**.

# BAB 5. Kesimpulan

Penelitian ini bertujuan untuk mengimplementasikan dan mengevaluasi beberapa algoritma klasifikasi untuk prediksi serangan jantung menggunakan dataset dari Kaggle. Empat algoritma yang diuji dalam penelitian ini adalah Logistic Regression, Support Vector Machine (SVM), Decision Tree, dan Random Forest. Berdasarkan evaluasi performa menggunakan metrik akurasi, precision, recall, dan F1-score, algoritma Random Forest menunjukkan hasil terbaik.

Berikut adalah ringkasan hasil evaluasi dari keempat algoritma yang diuji:

| Algoritma | Akurasi | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Logistic Regression | 74% | 0.75 | 0.75 | 0.75 |
| SVM | 83.4% | 0.84 | 0.83 | 0.83 |
| Decission Tree | 77% | 0.78 | 0.78 | 0.78 |
| Random Forest | 87% | 0.88 | 0.88 | 0.88 |

Algoritma Random Forest memberikan akurasi tertinggi sebesar 87%, dengan precision, recall, dan F1-score masing-masing sebesar 0.88.

**Implementasi Deployment**
Model Random Forest yang menunjukkan performa terbaik telah di-deploy menggunakan Spyder dari Anaconda. Kode deployment menerima input dari pasien, memprosesnya melalui model, dan memberikan hasil prediksi apakah pasien menderita serangan jantung atau tidak.

# REFERENCES


[1] A. P. E. G. K. B. C. G. M. Roubal, "Comparative Methodologic and Practical Considerations for Life Expectancy as a Public Health Mortality Measure," *Public Health Reports,* vol. 137, no. 2, pp. 255-262, 2022.

[2] D. A., "From Life Expectancy to the Meaning of the Life: A New Conception," *Journal of Research on History of Medicine,* 2019.

[3] S. C.-N. W. B. Rick Somers, "Applying Natural Language Processing to Automatically Assess Student Conceptual Understanding from Textual Responses," *Australasian Journal of Educational Technology,* vol. 37, no. 5, pp. 98-115, 2021.

[4] D. K. K. Kalaiselvi, Identifying Diseases and Diagnosis Using Machine Learning, 2020.

[5] V. U. Kumar, A. Krishna, P. Neelakanteswara and C. Z. Basha, "Advanced Prediction of Performance of A Student in An University using Machine Learning Techniques," *Proceedings of the International Conference on Electronics and Sustainable Communication Systems (ICESC 2020),* pp. 121-126, 2020.

[6] M. Mitra, "Algorithms and Machine Learning," *American Research Journal of Electronics and Communication* , vol. 1, no. 1, pp. 1-5, 2020.

[7] A. K. Tiwari, Introduction to Machine Learning, 2017.

[8] D. Meyer, "Logistic Regression - A Review and Comparison of Algorithms," 2023.

[9] J. P. a. M. K. J. Han, "A Comparative Study of Machine Learning Techniques for Binary Classification," 2011.

[10] R. A. I. a. R. C. Battistini, "Logistic Regression Models for Time Series Analysis," 2004.

[11] D. R. A. Y. Z. F. Bin Xu, "Real-time realization of Dynamic Programming using machine learning Methods for IC engine waste heat recovery system power optimization," *Applied Energy 262* , pp. 1-15, 2020.

[12] B. A. V. Govindaraj, "Machine Learning Based Power Estimation for CMOS VLSI Circuits," *APPLIED ARTIFICIAL INTELLIGENCE* , vol. 35, no. 13, pp. 1043-1055, 2021.

[13] B. N. D. R. Nadya Intan Mustika, "Machine Learning Algorithms in Fraud Detection: Case Study on Retail Consumer Financing Company," *Asia Pacific Fraud Journal,* vol. 6, no. 2, pp. 213-221, 2021.

[14] F. K. J. M. G. Christoph Schroer, "A Systematic Literature Review on Applying CRISP-DM Process Model," *Procedia Computer Science 181* , pp. 526-534, 2021.

[15] L. C.-O. C. F. J. H.-O. M. K. N. L. M. J. R.-Q. P. F. Fernando Martinez-Plumed, "CRISP-DM Twenty Years Later: From Data Mining Processes to Data

Science Trajectories," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, pp. 1041-4347, 019.

[16] D. Berrar, "Bayes' Theorem and Naive Bayes Classifier," *Encyclopedia of Bioinformatics and Computational Biology*, 2018.

[17] G. Tzanos, C. Kachris and D. Soudris, "Hardware Acceleration on Gaussian Naive Bayes Machine Learning Algorithm," *International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1-5, 2019.

[18] M. R. Ronak Chavan, "A Literature Review on Hierarchical Naive Bayes," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 7, no. 4, p. 179, 2019.

[19] C. C. Farzad Zafarani, "Differentially Private Naïve Bayes Classifier Using Smooth Sensitivity," *Proceedings on Privacy Enhancing Technologies Symposium*, vol. 2021, no. 4, p. 406–419, 2021.