

**LAPORAN UJIAN TENGAH SEMESTER MATA KULIAH  
PEMBELAJARAN MESIN PRAKTIKUM  
“PROGRAM DIAGNOSIS PENYAKIT KANKER PARU - PARU”**



**Dosen Pengampu :**

Indah Werdiningsih, S.Si., M.Kom.

**Disusun Oleh :**

**Kelompok 2 - C3**

Anisa Maharani (434221034)

Salma Aida Yasmi (434221037)

Ratna Firdaus (434221047)

**Program Studi D4 Teknik Informatika**

**Fakultas Vokasi**

**UNIVERSITAS AIRLANGGA**

**07 Oktober 2024**

## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>1</b>
<b>BAB 1 PENDAHULUAN.....</b>	<b>3</b>
1.1 Latar Belakang.....	3
1.2 Rumusan Masalah.....	3
1.3 Tujuan.....	4
1.4 Manfaat.....	4
<b>BAB 2 ALGORITMA MACHINE LEARNING.....</b>	<b>5</b>
2.1 Naive Bayes.....	5
2.2 Decision Tree.....	6
2.3 Random Forest.....	6
<b>BAB 3 PROSES DAN TAHAPAN PROYEK.....</b>	<b>8</b>
<b>BAB 4 HASIL DAN PEMBAHASAN.....</b>	<b>9</b>
4.1 Tahap Pengumpulan Data.....	9
4.2 Tahap Preprocessing Data.....	10
4.3 Tahap Transformasi Data.....	13
4.4 Tahap Evaluasi Algoritma.....	14
4.4.1 Import Library.....	14
4.4.2 Membaca Data.....	15
4.4.3 Seleksi Kolom yang akan Digunakan.....	16
4.4.4 Menampilkan Data Awal.....	16
4.4.5 Grouping Variabel untuk Fitur dan Label.....	16
4.4.6 Pembagian Data Training dan Testing.....	18
4.4.1 Pemodelan Algoritma.....	20
4.4.1.1 Naive Bayes.....	20
4.4.1.2 Decision Tree.....	21
4.4.1.3 Random Forest.....	21
4.4.2 Perhitungan Confusion Matrix.....	22
4.4.2.1 Perhitungan Confusion Matrix.....	22
4.4.2.2 Menampilkan Hasil Confusion Matrix.....	24
4.4.3 Hasil Evaluasi Performa Algoritma.....	27
4.5 Tahap Implementasi Website.....	27
4.5.1 Integrasi Model ke dalam Flask Application.....	28
4.5.1.1 Struktur Proyek.....	28
4.5.1.2 Implementasi Flask Routes.....	28
4.5.1.3 Handling User Input.....	29
4.5.1.4 Prediksi dan Penampilan Hasil.....	30
4.5.2 Tampilan dan User Interface.....	31
4.5.2.1 Halaman Beranda.....	31
4.5.2.2 Halaman about dan risk.....	32
4.5.2.3 Halaman form prediksi gejala.....	33
4.5.2.4 Halaman hasil diagnosa.....	35

4.5.3 Pengujian Sistem.....	35
<b>BAB 5 KESIMPULAN DAN SARAN.....</b>	<b>39</b>
5.1 Kesimpulan.....	39
5.2 Saran.....	39
<b>LAMPIRAN.....</b>	<b>40</b>
<b>DAFTAR PUSTAKA.....</b>	<b>41</b>

# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

*Lung Cancer* atau kanker paru-paru merupakan salah satu penyakit dengan peningkatan jumlah penderita yang signifikan di seluruh dunia. Berdasarkan data terbaru dari IARC (International Agency for Research on Cancer) dari WHO (World Health Organization), jumlah penderita kanker diperkirakan akan mengalami kenaikan hingga 77%, dengan capaian kasus sebanyak 35 juta kasus pada tahun 2050. Pada tahun 2022 diperkirakan terdapat 20 juta kasus kanker yang dilaporkan di 185 negara, dengan 36 jenis kanker yang berbeda. Dari semua jenis tersebut, kanker paru-paru menjadi jenis yang paling umum terjadi, dengan angka kasus mencapai 2,5 juta kasus atau sekitar 12,4% dari total penderita kanker di seluruh dunia.

Tidak sedikit dijumpai kasus kanker paru-paru yang baru terdeteksi pada stadium lanjut, sehingga hal tersebut mengakibatkan penanganan menjadi lebih sulit dan kurang efektif. Proses diagnosa kanker paru-paru umumnya memerlukan sejumlah pemeriksaan fisik, tes laboratorium, serta menggunakan teknologi citra medis yang seringkali memakan waktu dan biaya besar. Oleh karena itu, diperlukan sebuah sistem yang dapat membantu mempercepat dan meningkatkan akurasi dalam diagnosa kanker paru-paru.

Dalam perkembangan teknologi, pemanfaatan teknologi *machine learning* sudah mulai banyak diterapkan di berbagai bidang, termasuk salah satunya yaitu bidang kesehatan. *Machine learning* dirancang untuk mampu mendeteksi pola-pola yang tidak terlihat secara langsung oleh manusia. Pada konteks penyakit kanker paru-paru, teknologi ini memungkinkan pengolahan data medis seperti hasil CT Scan, X-Ray, dan riwayat medis pasien, sehingga dapat memberikan prediksi diagnosa secara lebih cepat dan akurat.

Proyek ini bertujuan untuk menganalisis sejumlah algoritma seperti Naive Bayes, Decision Tree, dan Random Forest yang kemudian akan dipilih untuk dikembangkan dalam sistem diagnosa penyakit paru-paru. Dengan adanya teknologi ini, diharapkan proses diagnosa kanker paru-paru dapat dilakukan lebih sistematis dan menghasilkan hasil yang akurat.

### 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, maka rumusan masalah dalam proyek ini yaitu, Berapa tingkat akurasi algoritma Naive Bayes, Decision Tree, dan Random Forest, serta bagaimana penerapannya dalam mendiagnosa kanker paru-paru melalui data medis?

### **1.3 Tujuan**

Tujuan dari proyek ini adalah mengembangkan sistem diagnosa penyakit kanker paru-paru menggunakan algoritma Naive Bayes, Decision Tree, dan Random Forest, untuk meningkatkan efisiensi dan akurasi diagnosa penyakit kanker paru-paru.

### **1.4 Manfaat**

Manfaat yang diharapkan dari proyek ini adalah peningkatan efisiensi dan akurasi dalam diagnosa penyakit paru-paru, sehingga memungkinkan pasien untuk mendapatkan penanganan yang cepat dan meningkatkan peluang kesembuhan.

## BAB 2

### ALGORITMA MACHINE LEARNING

#### 2.1 Naive Bayes

Naive Bayes adalah salah satu algoritma *machine learning* yang termasuk dalam kategori *supervised learning*. Algoritma ini didasarkan pada Teorema Bayes, yaitu menggunakan probabilitas dalam memprediksi kategori suatu data. Disebut “naive” karena algoritma ini mengasumsikan bahwa semua fitur atau variabel dalam dataset bersifat independen satu sama lain. Persamaan Naive Bayes untuk klasifikasi diberikan pada persamaan yang menggambarkan  $C_i$  sebagai sebuah nilai kelas tertentu,  $C$  sebagai pilihan kelas (himpunan),  $t$  sebagai fitur, dan  $F$  sebagai jumlah fitur. Tujuan dari algoritma Naive Bayes adalah memprediksi kelas berdasarkan probabilitas kemunculan nilai fitur pada kelas tersebut.

Langkah pertama yang dilakukan adalah menghitung *likelihood* dari sebuah *feature vector* dengan mengkalkulasi probabilitas fitur-fiturnya terhadap kelas tertentu. Selanjutnya, normalisasi *likelihood* dilakukan untuk semua kelas agar mendapatkan *class-assignment* menggunakan metode *softmax*, selanjutnya kelas dengan hasil probabilitas tertinggi akan dipilih sebagai hasil prediksi.

Gambar 1. Perhitungan *likelihood*

$$\text{likelihood}(c_i) = P(c_i) \prod_{f=1}^F P(t_f|c_i)$$

Selanjutnya, normalisasi *likelihood* dilakukan untuk semua kelas agar mendapatkan *class-assignment* menggunakan metode *softmax*, selanjutnya kelas dengan hasil probabilitas tertinggi akan dipilih sebagai hasil prediksi.

Gambar 2. Perhitungan *class-assignment*

$$P_{\text{assignment}}(c_i) = \frac{\text{likelihood}(c_i)}{\sum_{c_j \in C} \text{likelihood}(c_j)}$$

$$\hat{c}_i = \arg \max_{c_i \in C} P_{\text{assignment}}(c_i)$$

Kelebihan dari algoritma Naive Bayes adalah dapat digunakan untuk data kuantitatif dan kualitatif, tidak memerlukan jumlah data yang banyak sehingga tidak memerlukan jumlah data training yang banyak, jika terdapat missing value maka akan diabaikan dalam proses perhitungannya, serta proses perhitungan yang cepat dan

efisien, kemudian mudah dalam implementasinya. Namun, algoritma ini juga memiliki beberapa kekurangan yaitu jika probabilitas kondisi bernilai nol (0), maka probabilitas prediksi akan bernilai nol (0). Selain itu, asumsi bahwa masing-masing variabel bersifat independen akan memberikan dampak terhadap hasil akurasi yang kecil, karena tidak terdapat korelasi antar variabelnya.

## 2.2 Decision Tree

*Decision tree* merupakan varian dari *inductive learning* yaitu pendekatan dengan mengembangkan aturan klasifikasi yang dapat menentukan kelas suatu *instance* berdasarkan nilai atributnya (*feature vector*). *Decision tree* dibangun berdasarkan asumsi jika atribut yang ada memberikan informasi yang cukup dan mampu mengklasifikasikan seluruh *instance* di data training. *Decision tree* terdiri dari kumpulan *node* (simpul) yang dihubungkan oleh cabang. Terdapat tiga jenis *node* pada *decision tree* :

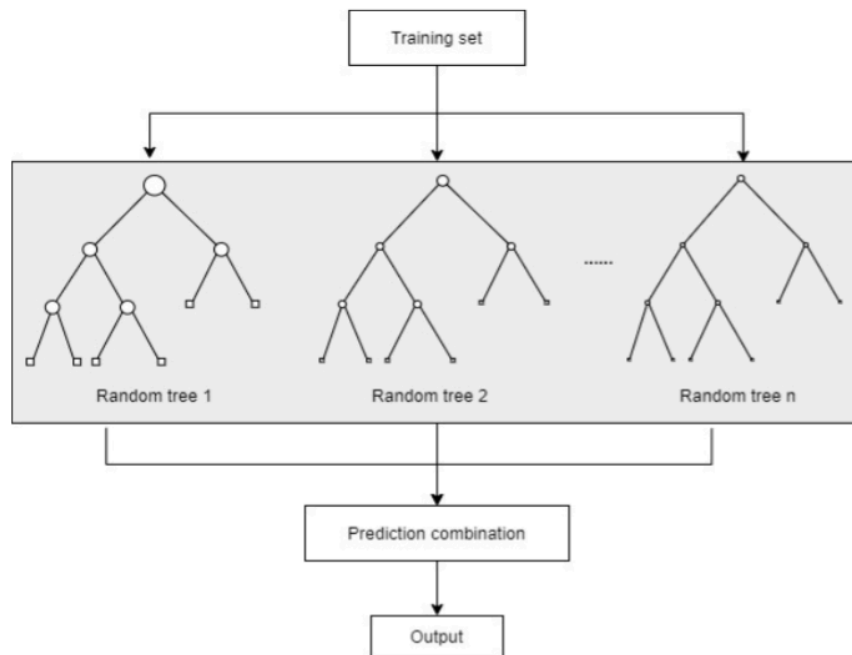
1. *Root node*, merupakan simpul yang tidak memiliki *input* tetapi memiliki *output* lebih dari satu
2. *Internal node*, memiliki satu input dan memiliki output lebih dari dua
3. *Leaf* atau *terminal node*, mempunyai satu *input* dan tidak mempunyai *output*

Pada *decision tree*, setiap *leaf* memiliki sebuah nama kelas. *Root node* dan *internal node* berisi aturan yang digunakan untuk membedakan data yang memiliki karakter berbeda. Beberapa kelebihan algoritma ini yaitu dapat bekerja dengan baik pada fitur yang bertipe kategorikal maupun numerik, serta mampu bekerja dengan data yang memiliki *missing value*. Namun, algoritma *decision tree* secara umum juga memiliki kekurangan diantaranya yaitu mudah mengalami *overfitting*, keterbatasan generalisasi terhadap data baru yang tidak sesuai dengan pola yang telah dipelajari, dan cukup sensitif terhadap perubahan data yang dapat mengakibatkan perubahan signifikan pada struktur *tree*, sehingga mempengaruhi stabilitas hasil prediksi.

## 2.3 Random Forest

*Random Forest* adalah jenis algoritma klasifikasi yang terdiri dari beberapa pohon keputusan. Setiap pohon keputusan dibentuk berdasarkan nilai-nilai vektor sampel acak yang independen dan identik yang kemudian akan didistribusikan untuk semua pohon. *Random Forest* termasuk dalam kelompok algoritma *supervised learning* yang dikembangkan oleh Leo Breiman. Metode ini dikenal memiliki tingkat akurasi yang tinggi dalam melakukan prediksi, bisa menangani inputan variabel dengan jumlah besar tanpa mengalami *overfitting*, dan dapat mengurangi korelasi antar pohon keputusan, seperti yang terjadi pada *ensemble method*. Selain itu, metode ini juga memiliki tingkat kesalahan yang lebih rendah dibandingkan dengan algoritma klasifikasi lainnya.

Gambar 3. Cara kerja *random forest*



Dalam pembuatan pohon keputusan, algoritma *random forest* akan menggunakan *random vector* dengan tahapan sebagai berikut :

1. Memilih secara acak fitur “R” dari total fitur “m” dengan syarat  $R \ll m$
2. Diantara fitur “R”, simpul dihitung menggunakan titik perpecahan terbaik
3. Kemudian node dibagi menjadi simpul anak menggunakan split (hasil pemisahan) terbaik
4. Ulangi langkah 1 hingga 3 hingga jumlah simpul yang diinginkan tercapai
5. Proses ini diulang untuk jumlah “n” kali untuk membuat jumlah pohon sebanyak “n”

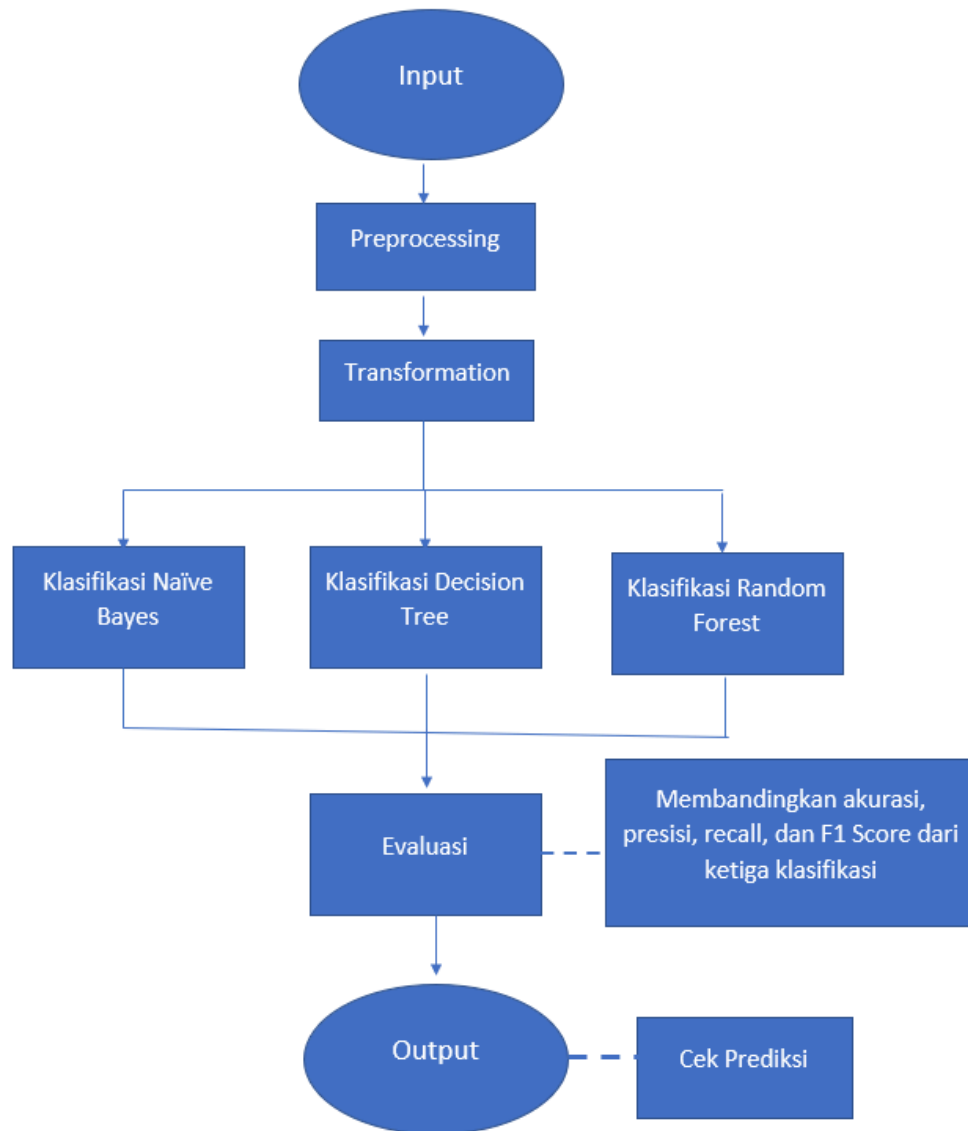
Algoritma *Random Forest* dapat digunakan dengan baik untuk regresi dan klasifikasi, serta memiliki kemampuan untuk mengidentifikasi fitur paling penting dari dataset pelatihan.



### BAB 3

## PROSES DAN TAHAPAN PROYEK

Gambar 4. Alur Sistem



## BAB 4

### HASIL DAN PEMBAHASAN

#### 4.1 Tahap Pengumpulan Data

Data yang digunakan dalam praktikum ini merupakan data sekunder mengenai kanker paru-paru dengan 309 entri. Dataset ini dapat diakses melalui website Kaggle pada tautan <https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer>. Dataset ini terdiri dari 16 atribut, dengan 13 atribut kategorikal yang mencakup gejala fisik terkait penyakit kanker paru-paru. Tiga atribut lainnya adalah *gender* (kategorikal), *age* (numerik), dan *class lung cancer* (label hasil prediksi). Informasi lebih lanjut mengenai atribut dataset dapat dilihat pada Tabel 4.1

Tabel 4.1. Informasi Atribut Dataset

Atribut	Deskripsi	Tipe Data	Nilai
Gender	Jenis kelamin pasien	Kategorikal	M (Male), F (Female)
Age	Usia pasien	Numerikal	Usia 21-87 tahun
Smoking	Status merokok	Kategorikal	YES=2, NO=1
Yellow fingers	Kondisi jari berwarna kuning	Kategorikal	YES=2, NO=1
Anxiety	Kondisi kecemasan	Kategorikal	YES=2, NO=1
Peer pressure	Pengaruh tekanan dari lingkungan	Kategorikal	YES=2, NO=1
Chronic Disease	Riwayat penyakit kronis	Kategorikal	YES=2, NO=1
Fatigue	Kondisi kelelahan	Kategorikal	YES=2, NO=1
Allergy	Riwayat alergi	Kategorikal	YES=2, NO=1
Wheezing	Kondisi mengi	Kategorikal	YES=2, NO=1
Alcohol	Konsumsi alkohol	Kategorikal	YES=2, NO=1

Coughing	Kondisi batuk	Kategorikal	YES=2, NO=1
Shortness of Breath	Kondisi sesak napas	Kategorikal	YES=2, NO=1
Swallowing Difficulty	Kesulitan menelan	Kategorikal	YES=2, NO=1
Chest pain	Nyeri dada	Kategorikal	YES=2, NO=1
Lung Cancer	Status kanker paru-paru	Kategorikal (Label)	YES, NO

## 4.2 Tahap Preprocessing Data

Data preprocessing adalah tahap awal dalam pengolahan data yang bertujuan untuk membersihkan data dari masalah seperti kesalahan (error) atau ketidakkonsistenan, sehingga menghasilkan data yang lebih akurat dan dapat diandalkan untuk digunakan pada tahap analisis atau pemodelan selanjutnya.

Pada tahap preprocessing data, atribut gender dan lung cancer dikonversi dari tipe data teks menjadi tipe data numerik, serta nilai kolom diubah menjadi biner (1 dan 0) dengan mengganti nilai 2 menjadi 1 dan nilai 1 menjadi 0.

Gambar 4. Kode konversi atribut kategorikal menjadi numerikal

```
# Mengubah kolom 'LUNG_CANCER' dan 'GENDER' menjadi numerik
data['LUNG_CANCER'] = data['LUNG_CANCER'].map({'YES': 1, 'NO': 0})
data['GENDER'] = data['GENDER'].map({'M': 0, 'F': 1})

# Ubah nilai kolom menjadi biner (1 dan 0)
binary_columns = ['SMOKING', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE',
                  'CHRONIC_DISEASE', 'FATIGUE ', 'ALLERGY ', 'WHEEZING',
                  'ALCOHOL_CONSUMING', 'COUGHING', 'SHORTNESS_OF_BREATH',
                  'SWALLOWING_DIFFICULTY', 'CHEST_PAIN']

# Ganti nilai 2 menjadi 1 dan nilai 1 menjadi 0
for col in binary_columns:
    data[col] = data[col].apply(lambda x: 1 if x > 1 else 0)
```

Tabel 4.2. Perbedaan Sebelum dan Sesudah Konversi dan Perubahan Menjadi Biner

Sebelum			Sesudah		
GENDER	AGE	SMOKING	GENDER	AGE	SMOKING
M	69	1	0	69	0
M	74	2	0	74	1
F	59	1	1	59	0
M	63	2	0	63	1
F	63	1	1	63	0
F	75	1	1	75	0
M	52	2	0	52	1
F	51	2	1	51	1
F	68	2	1	68	1
M	53	2	0	53	1

Setelah konversi data dan perubahan menjadi biner, dilakukan pengecekan *missing value* dengan menggunakan fungsi `isnull().sum()`.

Gambar 5. Pengecekan missing value

```
print("Pengecekan missing value".center(75, "="))
print(data.isnull().sum())
print("=".center(75, "="))
```

Fungsi `isnull()` mengembalikan **DataFrame** baru dengan nilai boolean, di mana True menunjukkan nilai yang hilang dan False menunjukkan nilai yang tidak hilang. Kemudian, `sum()` menjumlahkan nilai True (dihitung sebagai 1) di setiap kolom untuk memberikan total jumlah nilai yang hilang dalam kolom tersebut.

Setelah melakukan pengecekan, tidak ditemukan *missing value* dalam dataset. Oleh karena itu, penanganan *missing value* tidak diperlukan. Berikut adalah output pengecekan *missing value* di VSCode:

Gambar 6. Output pengecekan missing value

```
=====Pengecekan missing value=====
GENDER      0
AGE         0
SMOKING      0
YELLOW_FINGERS  0
ANXIETY      0
PEER_PRESSURE  0
CHRONIC_DISEASE  0
FATIGUE      0
ALLERGY      0
WHEEZING     0
ALCOHOL_CONSUMING  0
COUGHING     0
SHORTNESS_OF_BREATH  0
SWALLOWING_DIFFICULTY  0
CHEST_PAIN   0
LUNG_CANCER  0
dtype: int64
=====
```

Tahap selanjutnya adalah pengecekan outlier. Outlier adalah nilai yang sangat berbeda dari nilai-nilai lainnya dalam dataset. Berikut adalah kode program untuk melakukan pengecekan outlier:

Gambar 7. Deteksi outlier

```
def detect_outlier(data, threshold=3):
    outliers = []
    mean = np.mean(data)
    std = np.std(data)

    for yy in data:
        z_score = (yy - mean) / std
        if np.abs(z_score) > threshold:
            outliers.append(yy)

    return outliers

outliers = {}
for col in data.select_dtypes(include=['int64', 'float64']).columns:
    outliers[col] = detect_outlier(data[col])

for col, outlier_values in outliers.items():
    if outlier_values:
        print(f"Outlier pada kolom {col}: {outlier_values}")
    else:
        print(f"Tidak ada outlier pada kolom {col}")
```

Fungsi **detect\_outlier** digunakan untuk mengidentifikasi outlier dalam data menggunakan z-score. Fungsi ini menerima parameter data, kumpulan nilai numerik, dan threshold yang default-nya 3. Di dalam fungsi, dihitung nilai rata-rata (mean) dan deviasi standar (std) dari data. Untuk setiap nilai dalam data, z-score dihitung dengan rumus  $(yy - \text{mean}) / \text{std}$ . Jika nilai absolut z-score lebih besar dari threshold, nilai tersebut dianggap outlier dan ditambahkan ke daftar outliers.

Setelah fungsi **detect\_outlier** didefinisikan, kode melakukan pengecekan outlier untuk setiap kolom dalam dataset yang memiliki tipe data numerik (int64 dan float64). Setelah itu, program mencetak hasil pengecekan dengan menampilkan nilai outlier jika ditemukan, atau memberi tahu bahwa tidak ada outlier pada kolom tersebut.

Setelah menjalankan pengecekan outlier, kolom-kolom dalam dataset menunjukkan bahwa kolom *age* memiliki nilai outlier 21 dan 38. Sementara itu, kolom lainnya tidak memiliki outlier yang terdeteksi. Berikut adalah output pengecekan outlier di VSCode :

Gambar 8. Output deteksi outlier

```
=====
Tidak ada outlier pada kolom GENDER
Outlier pada kolom AGE: [21, 38]
Tidak ada outlier pada kolom SMOKING
Tidak ada outlier pada kolom YELLOW_FINGERS
Tidak ada outlier pada kolom ANXIETY
Tidak ada outlier pada kolom PEER_PRESSURE
Tidak ada outlier pada kolom CHRONIC_DISEASE
Tidak ada outlier pada kolom FATIGUE
Tidak ada outlier pada kolom ALLERGY
Tidak ada outlier pada kolom WHEEZING
Tidak ada outlier pada kolom ALCOHOL_CONSUMING
Tidak ada outlier pada kolom COUGHING
Tidak ada outlier pada kolom SHORTNESS_OF_BREATH
Tidak ada outlier pada kolom SWALLOWING_DIFFICULTY
Tidak ada outlier pada kolom CHEST_PAIN
Tidak ada outlier pada kolom LUNG_CANCER
=====
```

Berdasarkan gambar diatas, ditemukan *outlier* pada kolom *age*. Namun, jenis *outlier* ini merupakan variasi data sehingga tidak mempengaruhi hasil akhir diagnosa secara signifikan. Pada rentang umur tersebut juga memiliki kemungkinan terkena kanker paru-paru, meskipun jarang terjadi. Selain itu, dengan melakukan handling pada *outlier* dapat menyebabkan model menjadi bias terhadap populasi umum dalam data, yaitu kasus kanker paru-paru yang dengan usia lebih tua, serta mengabaikan kasus-kasus langka yang masih relevan.

### 4.3 Tahap Transformasi Data

Tahap transformasi data seperti normalisasi sangat penting untuk memastikan hasil prediksi yang akurat dan dapat diandalkan. Dataset yang digunakan sering memiliki skala yang berbeda, terutama pada atribut numerik. Perbedaan skala ini dapat mempengaruhi kinerja algoritma *machine learning*.

Normalisasi data merupakan mengubah rentang antar variabel atau atribut dalam data menjadi lebih menjadi lebih kecil. Adapun metode yang dapat digunakan untuk normalisasi data yaitu Min-Max normalization, Z-score Normalization, dan Feature Scaling. Dalam proyek ini, kami menggunakan salah satu metode normalisasi data yaitu Z-score Normalization.

Z-score normalization merupakan metode normalisasi dengan memanfaatkan nilai mean (rata-rata) dan standard deviation (standar deviasi) dari data. Berikut adalah rumus dari metode Z-Score :

$$\text{nilaibaru} = \frac{\text{nilailama} - \text{mean}}{\text{stdev}}$$

Berikut adalah implementasi rumus tersebut ke dalam kode program sistem. Tahap ini hanya diterapkan pada kolom *age*, karena kolom tersebut memiliki rentang variabel yang cukup besar dibandingkan variabel lainnya.

Gambar 9. Normalisasi menggunakan metode Z-Score

```
# Normalisasi kolom AGE menggunakan metode z-score
standard_scaler = preprocessing.StandardScaler()
data['AGE'] = standard_scaler.fit_transform(data[['AGE']]) # Hanya menormalisasi kolom AGE

print('\nData yang telah dinormalisasi dengan metode z-score standarisasi:')
print(data.head(10)) # Menampilkan 10 baris pertama dari DataFrame terstandarisasi
```

Dengan menggunakan fungsi **StandardScaler** dari library preprocessing, kode ini menghitung rata-rata dan standar deviasi kolom *age*, lalu mengubah nilainya sehingga memiliki distribusi dengan rata-rata 0 dan standar deviasi 1. Hasil normalisasi kemudian ditampilkan dalam 10 baris pertama setelah proses normalisasi yang dapat dilihat pada Gambar 10

Gambar 10. Kolom *Age* Setelah Dilakukan Normalisasi

```
Data yang telah dinormalisasi dengan metode z-score standarisasi:
GENDER    AGE    SMOKING  YELLOW_FINGERS  ...  SHORTNESS_OF_BREATH  SWALLOWING_DIFFICULTY  CHEST_PAIN  LUNG_CANCER
0      0  0.771850      0          1 ...              1              1              1              1
1      0  1.381829      1          0 ...              1              1              1              1
2      1 -0.448107      0          0 ...              1              0              1              0
3      0  0.039876      1          1 ...              0              1              1              0
4      1  0.039876      0          1 ...              1              0              0              0
5      1  1.503825      0          1 ...              1              0              0              1
6      0 -1.302078      1          0 ...              1              0              1              1
7      1 -1.424074      1          1 ...              1              1              0              1
8      1  0.649855      1          0 ...              0              0              0              0
9      0 -1.180082      1          1 ...              0              1              1              1

[10 rows x 16 columns]
```

## 4.4 Tahap Evaluasi Algoritma

### 4.4.1 Import Library

Gambar 11. Library Python

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import matplotlib

matplotlib.use('TkAgg')

pd.options.mode.chained_assignment = None
```

Fungsi import library yaitu untuk memungkinkan penggunaan kode yang telah ditulis sebelumnya dalam modul atau pustaka tertentu. Dengan mengimpor library, kita dapat mengakses fungsi, kelas, dan variabel yang ada tanpa harus menulis ulang kode tersebut. Berikut sejumlah library yang digunakan untuk pengembangan proyek ini :

- **import numpy as np** : Menyediakan array multidimensi dan fungsi matematika yang efisien untuk komputasi numerik.
- **import pandas as pd** : Memungkinkan manipulasi data tabular dengan struktur data seperti DataFrame dan Series.
- **import seaborn as sns** : Memudahkan pembuatan grafik statistik dengan visualisasi yang lebih informatif berbasis matplotlib.
- **from sklearn import preprocessing**: Digunakan untuk pra-pemrosesan data, seperti normalisasi, standarisasi, dan encoding data kategorikal.
- **from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score, precision\_score**: Menyediakan fungsi untuk evaluasi performa model machine learning, termasuk penghitungan metrik klasifikasi.
- **from sklearn.model\_selection import train\_test\_split**: Membagi dataset menjadi subset data latih (training) dan data uji (testing).
- **from sklearn.metrics import ConfusionMatrixDisplay**: Menampilkan confusion matrix dalam bentuk grafik untuk memvisualisasikan hasil klasifikasi model.
- **import matplotlib.pyplot as plt**: Menyediakan antarmuka untuk membuat berbagai jenis grafik dan visualisasi data.
- **import matplotlib**: Library untuk membuat visualisasi data dan mengatur backend grafis.
- **matplotlib.use('TkAgg')**: Mengatur backend grafis matplotlib agar grafik ditampilkan dalam antarmuka GUI menggunakan toolkit Tkinter.

#### 4.4.2 Membaca Data

Gambar 12. Pengkonversian dataset menjadi dataframe

```
data = dataframe = pd.read_csv(r'lungCancer.csv', delimiter=';')
```

Merupakan variabel yang digunakan untuk menyimpan method **read\_csv**. Method tersebut digunakan untuk membaca file CSV lungCancer.csv dan mengonversinya menjadi DataFrame menggunakan library pandas.



#### 4.4.3 Seleksi Kolom yang akan Digunakan

Gambar 13. Seleksi kolom atribut

```
data = dataframe[['GENDER', 'AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE',  
'CHRONIC_DISEASE', 'FATIGUE ', 'ALLERGY ', 'WHEEZING', 'ALCOHOL_CONSUMING', 'COUGHING',  
'SHORTNESS_OF_BREATH', 'SWALLOWING_DIFFICULTY', 'CHEST_PAIN', 'LUNG_CANCER']]
```

Mengambil subset kolom tertentu dari DataFrame, yang berisi fitur-fitur seperti gender, usia, dan sejumlah gejala kanker paru-paru, kemudian disimpan dalam variabel data.

#### 4.4.4 Menampilkan Data Awal

Gambar 14. Menampilkan data

```
print("data awal".center(75, "="))  
print(data)  
print("=".center(75, "="))
```

Mencetak isi dari variabel data, yang merupakan subset DataFrame yang diambil sebelumnya.

Gambar 15. Output print data

```
=====data awal=====  
   GENDER  AGE  SMOKING  YELLOW_FINGERS  ...  SHORTNESS_OF_BREATH  SWALLOWING_DIFFICULTY  CHEST_PAIN  LUNG_CANCER  
0        0   69        0                1  ...                1                1            1            1  
1        0   74        1                0  ...                1                1            1            1  
2        1   59        0                0  ...                1                0            1            0  
3        0   63        1                1  ...                0                1            1            0  
4        1   63        0                1  ...                1                0            0            0  
..      ...  ...      ...              ...  ...              ...              ...            ...            ...  
304       1   56        0                0  ...                1                1            0            1  
305       0   70        1                0  ...                1                0            1            1  
306       0   58        1                0  ...                0                0            1            1  
307       0   67        1                0  ...                1                0            1            1  
308       0   62        0                0  ...                0                1            0            1  
  
[309 rows x 16 columns]  
=====
```

#### 4.4.5 Grouping Variabel untuk Fitur dan Label

Gambar 16. Grouping data menjadi fitur dan label

```
print("GROUPING VARIABEL".center(75, "="))  
X = standardized[['GENDER', 'AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE',  
'CHRONIC_DISEASE', 'FATIGUE ', 'ALLERGY ', 'WHEEZING', 'ALCOHOL_CONSUMING', 'COUGHING',  
'SHORTNESS_OF_BREATH', 'SWALLOWING_DIFFICULTY', 'CHEST_PAIN']].values  
y = data['LUNG_CANCER'].values # Label  
print("data variabel".center(75, "="))  
print(X)  
print("data kelas".center(75, "="))  
print(y)  
print("=====")
```

Grouping variabel bertujuan untuk mengelompokkan data menjadi fitur (variabel yang mempengaruhi hasil) dan label (variabel hasil) untuk diproses dalam perhitungan akurasi dan analisa diagnosa.

- **X = standardized[['GENDER', ...]].values:** Mengambil nilai dari kolom tertentu dalam DataFrame standardized dan menyimpannya dalam variabel X, yang berisi fitur input untuk model.
- **y = data['LUNG\_CANCER'].values:** Mengambil nilai dari kolom LUNG\_CANCER dalam DataFrame data dan menyimpannya dalam variabel y, yang berisi label target untuk klasifikasi.
- **print(X):** Mencetak isi dari variabel X.
- **print(y):** Mencetak isi dari variabel y.

Gambar 17. Output grouping data

```
=====GROUPING VARIABEL=====
=====Data variabel (Fitur)=====
[[ 0.         0.77185028  0.         ...  1.         1.
   1.         ]
 [ 0.         1.38182914  1.         ...  1.         1.
   1.         ]
 [ 1.        -0.44810745  0.         ...  1.         0.
   1.         ]
 ...
 [ 0.        -0.57010322  1.         ...  0.         0.
   1.         ]
 [ 0.         0.52785873  1.         ...  1.         0.
   1.         ]
 [ 0.        -0.08212013  0.         ...  0.         1.
   0.         ]]
=====Data kelas (Label)=====
[1 1 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

#### 4.4.6 Pembagian Data Training dan Testing

Gambar 18. Pembagian data menjadi data training dan data testing

```
print("SPLITTING DATA 20-80".center(75, "="))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
print("instance variabel data training".center(75, "="))
print(X_train)
print("instance kelas data training".center(75, "="))
print(y_train)
print("instance variabel data testing".center(75, "="))
print(X_test)
print("instance kelas data testing".center(75, "="))
print(y_test)
print("=====")
print()
```

Fungsi pembagian data menjadi data training dan data testing sangat penting dalam pengembangan model machine learning. Pembagian ini bertujuan untuk mengevaluasi kinerja model pada data yang belum pernah dilihat sebelumnya, yang membantu mendapatkan gambaran akurat tentang seberapa baik model dapat melakukan generalisasi ke data baru. Dengan memisahkan dataset, kita juga dapat menghindari risiko overfitting, di mana model hanya mengingat data pelatihan dan tidak mampu melakukan prediksi yang baik pada data baru.

- **X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)**: Memisahkan dataset X dan y menjadi data pelatihan (80%) dan data pengujian (20%) menggunakan fungsi `train_test_split`, dengan `random_state=0` untuk memastikan pemisahan yang konsisten.
- **print(X\_train)**: Mencetak isi dari variabel `X_train`, yaitu fitur untuk data pelatihan.
- **print(y\_train)**: Mencetak isi dari variabel `y_train`, yaitu label untuk data pelatihan.
- **print(X\_test)**: Mencetak isi dari variabel `X_test`, yaitu fitur untuk data pengujian.
- **print(y\_test)**: Mencetak isi dari variabel `y_test`, yaitu label untuk data pengujian.

Gambar 19. Hasil pembagian data

```

=====SPLITTING DATA 20-80=====
=====instance variabel data training=====
[[ 1.         -1.42407363  1.         ...  1.         1.
   0.         ]
 [ 0.         1.13783759  1.         ...  1.         1.
   1.         ]
 [ 0.         -0.44810745  1.         ...  1.         0.
   0.         ]
 ...
 [ 1.         -1.42407363  1.         ...  1.         1.
   0.         ]
 [ 1.         0.16187141  1.         ...  0.         1.
   1.         ]
 [ 0.         -0.81409477  0.         ...  0.         0.
   0.         ]]

=====instance kelas data training=====
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 0 1
 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1]

=====instance variabel data testing=====
[[ 1.         0.6498545  0.         0.         1.         0.
   1.         0.         1.         1.         1.         0.
   0.         1.         0.         ]
 [ 0.         0.16187141  1.         1.         1.         1.
   0.         1.         1.         0.         1.         1.
   1.         0.         1.         ]
 [ 0.         -0.08212013  0.         0.         0.         0.
   1.         0.         1.         0.         1.         1.
   1.         1.         1.         ]
 [ 0.         0.6498545  0.         0.         1.         1.
   1.         0.         0.         0.         1.         0.
   0.         0.         0.         ]
 [ 1.         0.52785873  1.         1.         1.         1.
   0.         1.         0.         0.         0.         0.
   0.         0.         0.         ]
 [ 1.         0.28386719  1.         1.         1.         1.
   0.         1.         0.         1.         0.         1.
   1.         1.         0.         ]
 [ 1.         -0.93609054  1.         1.         1.         1.
   1.         1.         0.         1.         0.         1.
   1.         1.         0.         ]
 [ 1.         1.01584182  1.         1.         1.         1.
   1.         1.         0.         1.         0.         1.
   0.         1.         1.         ]
 [ 1.         -0.2041159  1.         1.         1.         0.
   0.         1.         1.         0.         1.         0.
   1.         1.         1.         ]

=====instance kelas data testing=====
[1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1]

```

#### 4.4.1 Pemodelan Algoritma

Penerapan code untuk pemodelan algoritma ini sama penerapannya pada ketiga algoritma yang kami gunakan, yang membedakan hanya library algoritma yang digunakan.

##### 4.4.1.1 Naive Bayes

- Library

Gambar 20. Library Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
```

**from sklearn.naive\_bayes import GaussianNB:** Implementasi Naive Bayes untuk klasifikasi, dengan asumsi distribusi data berjenis Gaussian.

- Pemodelan Naive Bayes

Gambar 21. Pemodelan Naive Bayes

```
print("PEMODELAN DENGAN NAIVE BAYES".center(75, "="))
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
accuracy_nb = round(accuracy_score(y_test, Y_pred) * 100, 2)
acc_gaussian = round(gaussian.score(X_train, y_train) * 100, 2)
print("instance prediksi naive bayes:")
print(Y_pred)
```

- **gaussian = GaussianNB():** Membuat objek model Naive Bayes dengan distribusi Gaussian.
- **gaussian.fit(X\_train, y\_train):** Melatih model Naive Bayes menggunakan data pelatihan (X\_train) dan label target (y\_train).
- **Y\_pred = gaussian.predict(X\_test):** Menggunakan model yang telah dilatih untuk memprediksi label dari data uji (X\_test), hasil prediksi disimpan dalam variabel Y\_pred.
- **accuracy\_nb = round(accuracy\_score(y\_test, Y\_pred) \* 100, 2):** Menghitung akurasi model dengan membandingkan prediksi (Y\_pred) dengan label sebenarnya (y\_test), lalu mengalikan hasilnya dengan 100 dan membulatkannya hingga dua desimal.
- **acc\_gaussian = round(gaussian.score(X\_train, y\_train) \* 100, 2):** Menghitung dan menyimpan akurasi model pada data pelatihan (X\_train dan y\_train), juga dibulatkan hingga dua desimal.
- **print(Y\_pred):** Mencetak hasil prediksi yang dihasilkan oleh model Naive Bayes.

Gambar 22. Output pemodelan Naive Bayes

```
=====PEMODELAN DENGAN NAIVE BAYES=====
```

```
instance prediksi naive bayes:
```

```
[1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1]
```

#### 4.4.1.2 Decision Tree

- **Library**

Gambar 23. Library Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

**from sklearn.tree import DecisionTreeClassifier:** Mengimplementasikan algoritma pohon keputusan untuk klasifikasi, yang membagi data berdasarkan fitur untuk membuat keputusan yang jelas, sehingga memudahkan interpretasi dan visualisasi model.

- **Pemodelan Decision Tree**

Gambar 24. Pemodelan Decision Tree

```
print("PERMODELAN DENGAN DECISION TREE".center(75, "="))
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
Y_pred = decision_tree.predict(X_test)
accuracy_tree = round(accuracy_score(y_test, Y_pred) * 100, 2)
print("instance prediksi decision tree : ")
print(Y_pred)
```

Gambar 25. Output pemodelan Naive Bayes

```
=====PERMODELAN DENGAN DECISION TREE=====
instance prediksi decision tree :
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1]
```

#### 4.4.1.3 Random Forest

- **Library**

Gambar 26. Library Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.ensemble import RandomForestClassifier:
```

Mengimplementasikan algoritma ensemble yang menggabungkan banyak pohon keputusan untuk meningkatkan akurasi prediksi dan mengurangi overfitting.

- **Pemodelan Random Forest**

Gambar 27. Pemodelan Random Forest

```
print("PERMODELAN DENGAN RANDOM FOREST".center(75, "="))
random_forest = RandomForestClassifier()
random_forest.fit(X_train, y_train)
Y_pred = random_forest.predict(X_test)
accuracy_rf = round(accuracy_score(y_test, Y_pred) * 100, 2)
print("instance prediksi random forest : ")
print(Y_pred)
```

Gambar 28. Output pemodelan Random Forest

```
=====PERMODELAN DENGAN RANDOM FOREST=====
instance prediksi random forest :
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

#### 4.4.2 Perhitungan Confusion Matrix

##### 4.4.2.1 Perhitungan Confusion Matrix

Penerapan code confusion matrix dibawah ini dilakukan untuk tiga algoritma yang digunakan untuk ditinjau akurasi.

Gambar 29. Perhitungan confusion matrix

```
# Perhitungan confusion matrix
cm = confusion_matrix(y_test, Y_pred)
print('CLASSIFICATION REPORT NAIVE BAYES'.center(75, '='))

accuracy = accuracy_score(y_test, Y_pred)
precision = precision_score(y_test, Y_pred)
print(classification_report(y_test, Y_pred))

TN = cm[1][1] * 1.0
FN = cm[1][0] * 1.0
TP = cm[0][0] * 1.0
FP = cm[0][1] * 1.0
total = TN + FN + TP + FP
sens = TN / (TN + FP) * 100
spec = TP / (TP + FN) * 100

print('Akurasi : ', accuracy * 100, "%")
print('Sensitivity : ' + str(sens))
print('Specificity : ' + str(spec))
print('Precision : ' + str(precision))
print("=====")
print()
```

- **cm = confusion\_matrix(y\_test, Y\_pred)**: Menghitung confusion matrix untuk menilai kinerja model dengan membandingkan label sebenarnya dan hasil prediksi.
- **accuracy = accuracy\_score(y\_test, Y\_pred)**: Menghitung akurasi model dengan membandingkan prediksi dan label sebenarnya.
- **precision = precision\_score(y\_test, Y\_pred)**: Menghitung precision, rasio true positives terhadap total prediksi positif.

- **print(classification\_report(y\_test, Y\_pred))**: Menampilkan laporan klasifikasi dengan metrik seperti precision, recall, f1-score, dan support.
- **TN = cm[1][1] \* 1.0, FN = cm[1][0] \* 1.0, TP = cm[0][0] \* 1.0, FP = cm[0][1] \* 1.0**: Mengambil true negatives, false negatives, true positives, dan false positives dari confusion matrix.
- **total = TN + FN + TP + FP**: Menghitung total kasus dalam dataset.
- **sens = TN / (TN + FP) \* 100**: Menghitung sensitivitas (recall), rasio true positives terhadap total kasus positif.
- **spec = TP / (TP + FN) \* 100**: Menghitung spesifisitas, rasio true negatives terhadap total kasus negatif.
- **print('Akurasi : ', accuracy \* 100, "%"), print('Sensitivity : ' + str(sens)), print('Specificity : ' + str(spec)), print('Precision : ' + str(precision))**: Menampilkan akurasi, sensitivitas, spesifisitas, dan precision model dalam bentuk persentase.

Gambar 30. Hasil perhitungan confusion matrix Naive Bayes

```

=====CLASSIFICATION REPORT NAIVE BAYES=====
              precision    recall  f1-score   support

     0       0.62         0.50         0.56         10
     1       0.91         0.94         0.92         52

 accuracy          0.87         0.87         0.87         62
 macro avg       0.77         0.72         0.74         62
weighted avg       0.86         0.87         0.87         62

Akurasi : 87.09677419354838 %
Sensitivity : 90.74074074074075
Specificity : 62.5
Precision : 0.9074074074074074
=====

```



Gambar 31. Hasil perhitungan confusion matrix Decision Tree

```
=====CLASSIFICATION REPORT DECISION TREE=====
              precision    recall  f1-score   support

     0       0.80      0.40      0.53        10
     1       0.89      0.98      0.94        52

 accuracy      0.89        62
 macro avg      0.85      0.69      0.73        62
weighted avg      0.88      0.89      0.87        62

Akurasi algoritma : 88.70967741935483 %
Sensitivity algoritma : 89.47368421052632
Specificity algoritma : 80.0
Precision algoritma : 0.8947368421052632
=====
```

Gambar 32. Hasil perhitungan confusion matrix Random Forest

```
=====CLASSIFICATION REPORT RANDOM FOREST=====
              precision    recall  f1-score   support

     0       0.83      0.50      0.62        10
     1       0.91      0.98      0.94        52

 accuracy      0.90        62
 macro avg      0.87      0.74      0.78        62
weighted avg      0.90      0.90      0.89        62

Akurasi algoritma : 90.32258064516128 %
Sensitivity algoritma : 91.07142857142857
Specificity algoritma : 83.33333333333334
Precision algoritma : 0.9107142857142857
=====
```

#### 4.4.2.2 Menampilkan Hasil Confusion Matrix

Penerapan code untuk menampilkan confusion matrix dibawah ini dilakukan untuk tiga algoritma yang digunakan untuk ditinjau akurasi.

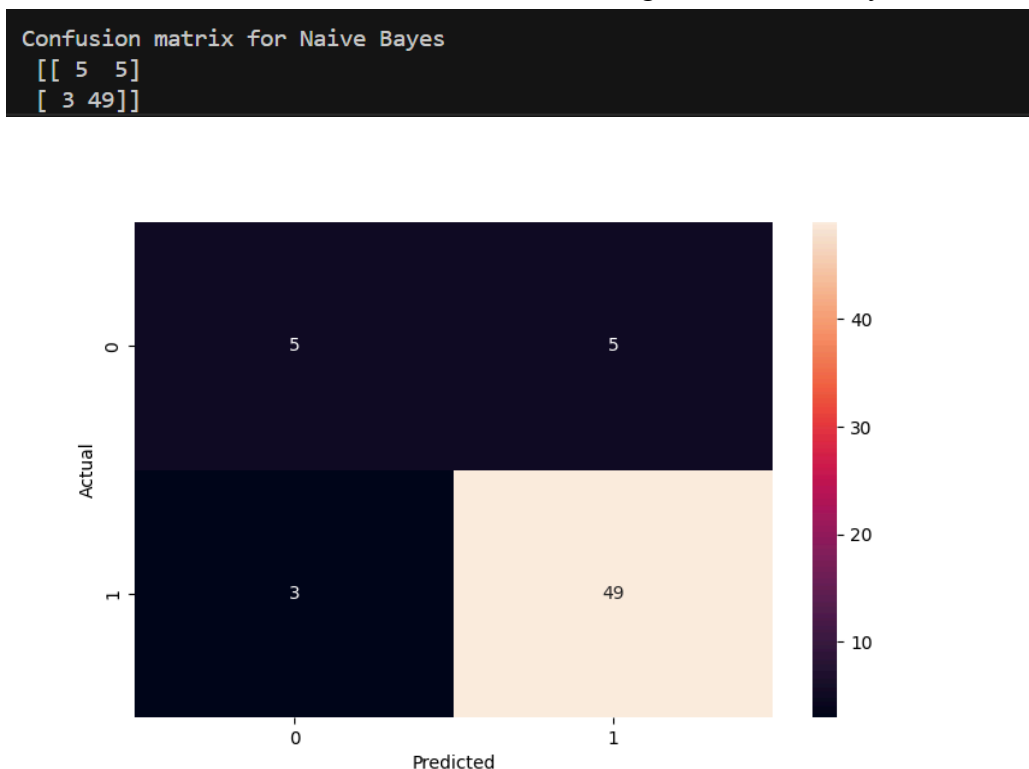
Gambar 33. Kode untuk menampilkan confusion matrix

```
# Menampilkan Confusion Matrix
cm_display = ConfusionMatrixDisplay(confusion_matrix=cm)
print('Confusion matrix for Naive Bayes\n', cm)
f, ax = plt.subplots(figsize=(8, 5))
sns.heatmap(confusion_matrix(y_test, Y_pred), annot=True, fmt=".0f", ax=ax)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

print("=====")
print()
```

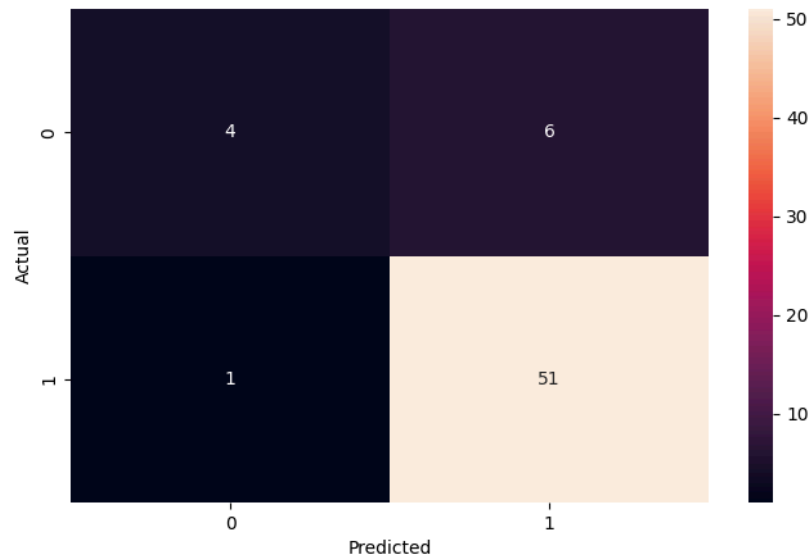
- **cm\_display = ConfusionMatrixDisplay(confusion\_matrix=cm):** Membuat objek ConfusionMatrixDisplay untuk menampilkan confusion matrix yang telah dihitung sebelumnya.
- **print('Confusion matrix for Naive Bayes\n', cm):** Menampilkan confusion matrix di konsol untuk referensi visual.
- **f, ax = plt.subplots(figsize=(8, 5)):** Membuat figure dan axes untuk plot dengan ukuran 8x5 inci.
- **sns.heatmap(confusion\_matrix(y\_test, Y\_pred), annot=True, fmt=".0f", ax=ax):** Menggunakan Seaborn untuk membuat heatmap dari confusion matrix, menambahkan anotasi pada setiap sel, dan menampilkan angka dalam format tanpa desimal.
- **plt.xlabel("Predicted"):** Menambahkan label sumbu x dengan teks "Predicted" untuk menggambarkan prediksi model.
- **plt.ylabel("Actual"):** Menambahkan label sumbu y dengan teks "Actual" untuk menggambarkan label sebenarnya.
- **plt.show():** Menampilkan plot yang telah dibuat di jendela visualisasi.

Gambar 34. Hasil confusion matrix algoritma Naive Bayes



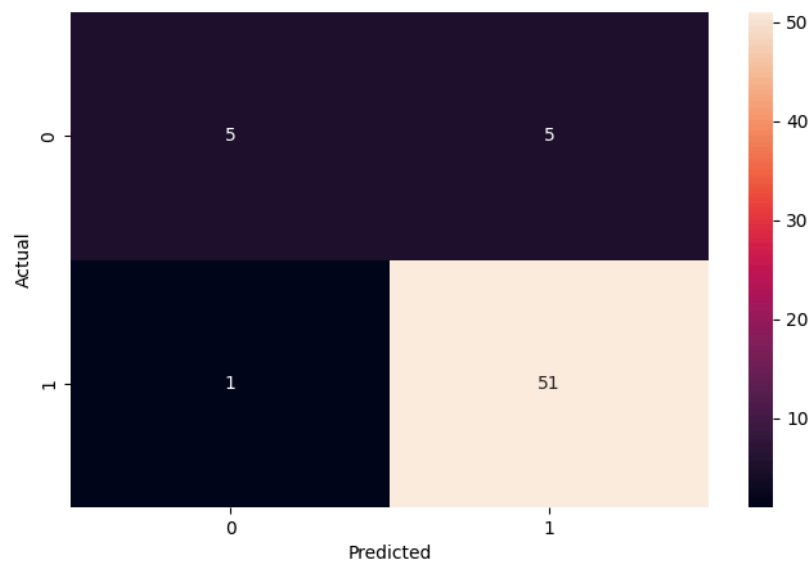
Gambar 35. Hasil confusion matrix algoritma Decision Tree

```
Hasil confusion matrix algoritma Decision Tree  
[[ 4  6]  
 [ 1 51]]
```



Gambar 36. Hasil confusion matrix algoritma Random Forest

```
Hasil confusion matrix algoritma Random Forest  
[[ 5  5]  
 [ 1 51]]
```



#### 4.4.3 Hasil Evaluasi Performa Algoritma

Dalam upaya membangun sistem prediksi kanker paru-paru yang akurat dan handal, kami telah menguji tiga algoritma machine learning, yaitu Random Forest, Naive Bayes, dan Decision Tree. Tujuan dari pengujian ini adalah untuk menentukan algoritma mana yang memberikan performa terbaik berdasarkan metrik evaluasi seperti Akurasi, Presisi, Sensitivitas (Recall), dan Spesifisitas.

Berikut adalah tabel yang menggambarkan hasil evaluasi performa ketiga algoritma berdasarkan metrik yang telah disebutkan:

Tabel 4.4.3 Hasil Evaluasi Algoritma

Algoritma	Akurasi (%)	Presisi	Sensitivitas (Recall)	Spesifisitas
Random Forest	90.32	0.91	90.07	83.33
Naive Bayes	87.09	0.90	90.74	62.50
Decision Tree	88.70	0.89	89.47	80.00

Berdasarkan hasil evaluasi di atas, **Random Forest** dipilih sebagai algoritma yang akan diimplementasikan ke dalam sistem website prediksi kanker paru-paru. Keunggulan Random Forest dalam hal **Akurasi**, **Presisi**, dan **Spesifisitas** menjadikannya pilihan yang paling tepat untuk memberikan prediksi yang akurat dan dapat diandalkan kepada pengguna.

Meskipun **Naive Bayes** menunjukkan sensitivitas yang sedikit lebih tinggi, rendahnya spesifisitasnya membuatnya kurang ideal karena menghasilkan lebih banyak false positives. **Decision Tree** berada di tengah-tengah kedua algoritma tersebut, namun tidak seimbang seperti Random Forest dalam semua metrik evaluasi. Oleh karena itu, Random Forest menawarkan keseimbangan terbaik antara mendeteksi kasus kanker paru-paru dan mengidentifikasi pasien yang sehat dengan akurasi yang tinggi.

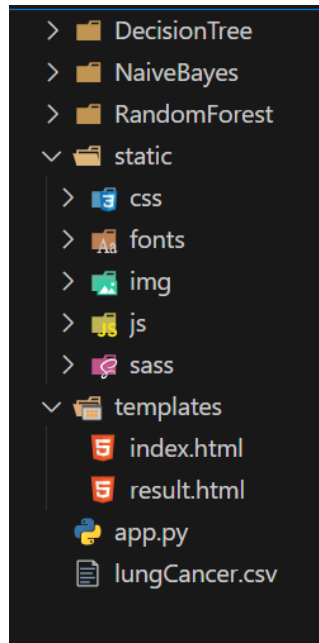
#### 4.5 Tahap Implementasi Website

Untuk membangun sistem prediksi kanker paru-paru yang interaktif dan mudah diakses, kami memilih Flask, sebuah micro framework berbasis Python, sebagai kerangka kerja utama. Flask dipilih karena kesederhanaannya, fleksibilitas, dan kemampuannya untuk dengan mudah mengintegrasikan model machine learning.

## 4.5.1 Integrasi Model ke dalam Flask Application

### 4.5.1.1 Struktur Proyek

Gambar 37. Struktur Direktori Proyek



Gambar diatas merupakan struktur direktori dari pengembangan sistem diagnosa kanker paru-paru. Terdapat sejumlah direktori dan file utama diantaranya yaitu file **app.py** yang merupakan file yang berisi logika aplikasi Flask. Direktori **templates/** yang berisi file HTML untuk tampilan website (index.html dan result.html). Selain itu, terdapat direktori **static/** yang digunakan untuk menyimpan file statis seperti CSS, JavaScript, dan gambar.

### 4.5.1.2 Implementasi Flask Routes

Gambar 38. Implementasi Routes

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/result', methods=['POST'])
def result():
```

Dalam flask, route digunakan untuk menentukan URL atau jalur yang dapat diakses serta menentukan fungsi apa yang harus dijalankan ketika URL

tersebut diakses. Pada file app.py terdapat dua route yaitu **Route / (Index)** digunakan untuk menampilkan formulir input data gejala yang harus diisi oleh pengguna. Selain itu, terdapat **Route /result** yang digunakan untuk mengambil data dari formulir, memproses nya, dan melakukan prediksi menggunakan model Random Forest, serta menampilkan hasil prediksi kepada pengguna.

#### 4.5.1.3 Handling User Input

Gambar 39. Handling user input

```
try:
    # Mengambil data dari form
    name = request.form.get('name')
    age = request.form.get('usia')

    # Mengambil gejala yang dipilih
    gender = request.form.get('gender')
    smoking = request.form.get('smoking')
    yellow_fingers = request.form.get('yellow_fingers')
    anxiety = request.form.get('anxiety')
    peer_pressure = request.form.get('peer_pressure')
    chronic_disease = request.form.get('chronic_disease')
    fatigue = request.form.get('fatigue')
    allergy = request.form.get('allergy')
    wheezing = request.form.get('wheezing')
    alcohol_consuming = request.form.get('alcohol_consuming')
    coughing = request.form.get('coughing')
    shortness_of_breath = request.form.get('shortness_of_breath')
    swallowing_difficulty = request.form.get('swallowing_difficulty')
    chest_pain = request.form.get('chest_pain')
```

Data yang telah diinputkan oleh pengguna melalui formulir di index.html akan diambil menggunakan **request.form**.

Gambar 40. Konversi input menjadi numerik

```
# Konversi input menjadi numerik sesuai preprocessing
gender_num = 0 if gender == 'male' else 1
smoking_num = 1 if smoking == 'yes' else 0
yellow_fingers_num = 1 if yellow_fingers == 'yes' else 0
anxiety_num = 1 if anxiety == 'yes' else 0
peer_pressure_num = 1 if peer_pressure == 'yes' else 0
chronic_disease_num = 1 if chronic_disease == 'yes' else 0
fatigue_num = 1 if fatigue == 'yes' else 0
allergy_num = 1 if allergy == 'yes' else 0
wheezing_num = 1 if wheezing == 'yes' else 0
alcohol_consuming_num = 1 if alcohol_consuming == 'yes' else 0
coughing_num = 1 if coughing == 'yes' else 0
shortness_of_breath_num = 1 if shortness_of_breath == 'yes' else 0
swallowing_difficulty_num = 1 if swallowing_difficulty == 'yes' else 0
chest_pain_num = 1 if chest_pain == 'yes' else 0
```

Setelah data diterima oleh sistem, langkah selanjutnya adalah mengubah menjadi format yang sesuai dengan model. Contohnya mapping nilai "yes" menjadi 1 dan "no" menjadi 0.

#### 4.5.1.4 Prediksi dan Penampilan Hasil

Gambar 41. Prediksi

```
# Melakukan prediksi
prediction = random_forest.predict(input_data)[0]

# Menentukan hasil
diagnosis_result = "Anda kemungkinan mengidap kanker paru-paru." if prediction == 1 else
"Kemungkinan Anda tidak mengidap kanker paru-paru."
```

Setelah data diproses, model Random Forest digunakan untuk melakukan prediksi apakah pengguna mengidap kanker paru-paru atau tidak.

Gambar 42. Mengembalikan hasil prediksi

```
return render_template('result.html',
                       user_name=name,
                       user_age=age,
                       user_gender=gender_str,
                       selected_symptoms=selected_symptoms_str,
                       diagnosis_result=diagnosis_result,
                       suggested_treatment=suggested_treatment)
except Exception as e:
    return render_template('result.html', error=str(e))
```

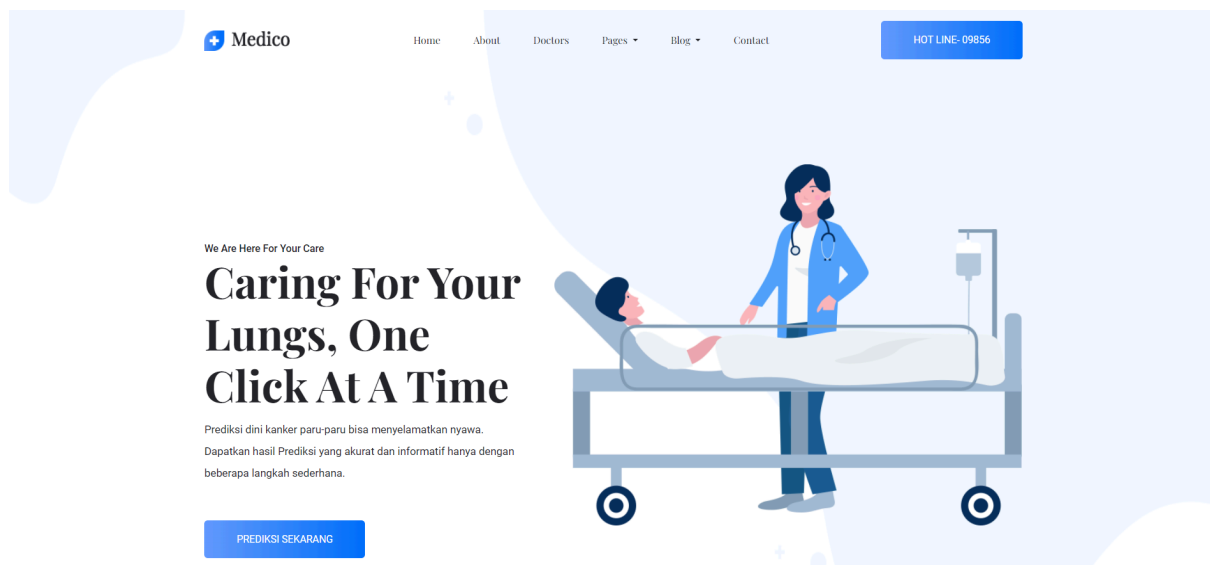
Hasil prediksi, beserta dengan informasi gejala yang dipilih, akan ditampilkan di halaman result.html.

## 4.5.2 Tampilan dan User Interface

### 4.5.2.1 Halaman Beranda

Halaman beranda merupakan tampilan pertama yang muncul ketika pengguna mengakses sistem diagnosa penyakit kanker paru-paru. Tampilan halaman beranda berisi kalimat persuasif yang dapat meyakinkan pengguna dengan jaminan akurasi yang dihasilkan oleh sistem tersebut.

Gambar 43. Halaman Beranda

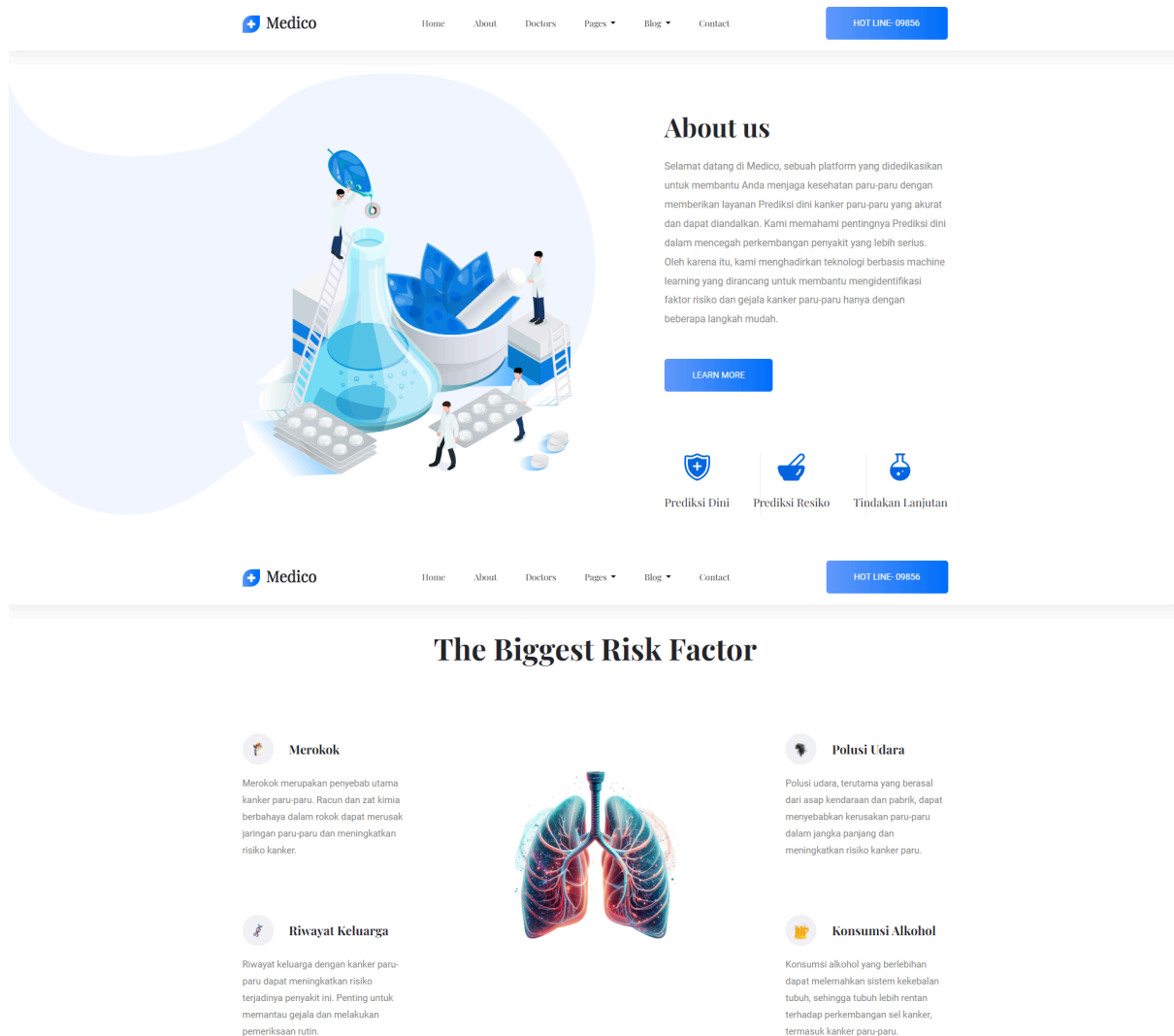




#### 4.5.2.2 Halaman about dan risk

Halaman about digunakan untuk memberikan informasi singkat kepada pengguna mengenai latar belakang dan tujuan sistem. Kemudian halaman risk berisi informasi faktor risiko terbesar yang menjadi penyebab utama seseorang mengidap kanker paru-paru.

Gambar 44. Halaman About dan Risk



#### 4.5.2.3 Halaman form prediksi gejala

Halaman formulir prediksi gejala berisi berbagai gejala yang mungkin dialami pengguna, sesuai dengan fitur utama yang dijelaskan dalam Tabel 4.1. Pengguna harus mengisi formulir tersebut agar sistem dapat menentukan hasil diagnosis penyakit.

Gambar 45. Halaman Form

The screenshot shows a web form titled "Lung Cancer Prediction" on a website called "Medico". The form includes the following fields and options:

- Nama** (Name): A text input field.
- Usia** (Age): A text input field.
- Jenis kelamin:** (Gender) with two radio button options: ☐ **Laki-laki** and ☐ **Perempuan**.
- Apakah Anda seorang perokok?** (Are you a smoker?) with two radio button options: ☐ **Ya** and ☐ **Tidak**.
- Apakah jari-jari Anda berwarna kekuningan?** (Are your fingers yellowish?) with two radio button options: ☐ **Ya** and ☐ **Tidak**.
- Apakah Anda sering mengalami kecemasan?** (Do you often experience anxiety?) with two radio button options: ☐ **Ya** and ☐ **Tidak**.

The website header includes a navigation menu with links: Home, About, Doctors, Pages, Blog, and Contact. A blue button labeled "HOT LINE- 09856" is also visible.

Apakah Anda pernah merasakan tekanan dari lingkungan sekitar untuk merokok atau melakukan kebiasaan yang tidak sehat?

☐ Ya ☐ Tidak

Apakah Anda memiliki riwayat penyakit kronis?

☐ Ya ☐ Tidak

Apakah Anda sering merasa lelah atau keletihan?

☐ Ya ☐ Tidak

Apakah Anda memiliki alergi tertentu?

☐ Ya ☐ Tidak

Apakah Anda sering mengalami mengi (bunyi napas yang menciut)?

☐ Ya ☐ Tidak

Apakah Anda sering mengonsumsi alkohol?

☐ Ya ☐ Tidak

Apakah Anda sering batuk?

☐ Ya ☐ Tidak

Apakah Anda sering mengalami sesak napas?

☐ Ya ☐ Tidak

Apakah Anda mengalami kesulitan saat menelan?

☐ Ya ☐ Tidak

Apakah Anda mengalami nyeri di dada?

☐ Ya ☐ Tidak

PREDICT

#### 4.5.2.4 Halaman hasil diagnosa

Halaman hasil diagnosa merupakan tampilan hasil dari prediksi sistem. Tampilan hasil diagnosa berisi data yang sebelumnya dimasukkan pengguna dalam formulir diagnosa dan terdapat hasil diagnosa serta saran yang diberikan sistem untuk penanganan kedepannya.

Gambar 46. Halaman Hasil Diagnosa

#### Diagnosis Summary

Field	Information
Nama	anisa
Usia	38
Jenis Kelamin	Perempuan
Gejala yang Dipilih	Merokok: Ya Jari Kekuningan: Tidak Kecemasan: Ya Tekanan Lingkungan: Tidak Penyakit Kronis: Ya Kelelahan: Tidak Alergi: Ya Mengi: Tidak Konsumsi Alkohol: Ya Batuk: Ya Sesak Napas: Ya Kesulitan Menelan: Ya Nyeri Dada: Ya
Hasil Diagnosis	Ada kemungkinan Anda mengidap kanker paru-paru.
Saran Perawatan	Disarankan untuk berkonsultasi dengan dokter spesialis paru-paru untuk tindakan lebih lanjut.

Kembali ke Formulir

#### 4.5.3 Pengujian Sistem

Pengujian sistem diagnosa dilakukan untuk mengetahui apakah sistem telah berjalan dengan benar atau tidak. Pada tahap ini kami memasukkan sejumlah data seperti yang dijelaskan pada tabel 4.5.3 dibawah ini :

Tabel 4.5.3 Tabel Pengujian Sistem

Daftar Inputan	Jawaban
Nama	Anisa
Usia	38

Jenis Kelamin	Perempuan
Apakah Anda seorang perokok?	YA
Apakah jari-jari Anda berwarna kekuningan?	TIDAK
Apakah Anda sering mengalami kecemasan?	YA
Apakah Anda pernah merasakan tekanan dari lingkungan sekitar untuk merokok atau melakukan kebiasaan yang tidak sehat?	TIDAK
Apakah Anda memiliki riwayat penyakit kronis?	YA
Apakah Anda sering merasa lelah atau kelelahan?	TIDAK
Apakah Anda memiliki alergi tertentu?	YA
Apakah Anda sering mengalami mengi (bunyi napas yang menciut)?	TIDAK
Apakah Anda sering mengonsumsi alkohol?	YA
Apakah Anda sering batuk?	YA
Apakah Anda sering mengalami sesak napas?	YA
Apakah Anda mengalami kesulitan saat menelan?	YA
Apakah Anda mengalami nyeri di dada?	YA


Hasil diagnosis	Ada kemungkinan anda mengidap kanker paru-paru
-----------------	--

Gambar dibawah ini merupakan implementasi pengisian data yang ada di tampilan sistem. Selanjutnya sistem akan menganalisis data yang diinputkan pengguna untuk kemudian ditampilkan hasil diagnosa dan penanganan lebih lanjut yang dapat dilakukan oleh pengguna, seperti pada gambar dibawah ini,

Gambar 47. Uji Coba Inputan

The screenshot displays the 'Lung Cancer Prediction' web application interface. The header includes the 'Medico' logo, navigation links (Home, About, Doctors, Pages, Blog, Contact), and a 'HOT LINE: 09856' button. The main title 'Lung Cancer Prediction' is centered. The input form contains the following sections:

- Personal Information:**
  - Nama:** Input field containing 'anisa'.
  - Usia:** Input field containing '38'.
  - Jenis kelamin:** Radio buttons for 'Laki-laki' and 'Perempuan' (selected).
- Lifestyle and Medical History Questions:**
  - Apakah Anda seorang perokok?** Radio buttons for 'Ya' (selected) and 'Tidak'.
  - Apakah jari-jari Anda berwarna kekuningan?** Radio buttons for 'Ya' and 'Tidak' (selected).
  - Apakah Anda sering mengalami kecemasan?** Radio buttons for 'Ya' (selected) and 'Tidak'.
  - Apakah Anda pernah merasakan tekanan dari lingkungan sekitar untuk merokok atau melakukan kebiasaan yang tidak sehat?** Radio buttons for 'Ya' and 'Tidak' (selected).
  - Apakah Anda memiliki riwayat penyakit kronis?** Radio buttons for 'Ya' (selected) and 'Tidak'.
  - Apakah Anda sering merasa lelah atau kelelahan?** Radio buttons for 'Ya' and 'Tidak' (selected).
  - Apakah Anda memiliki alergi tertentu?** Radio buttons for 'Ya' (selected) and 'Tidak'.



HomeAboutDoctorsPagesBlogContact

HOT LINE- 09856

Apakah Anda sering mengalami mengi (bunyi napas yang menciut)?

☐ Ya☒ Tidak

Apakah Anda sering mengonsumsi alkohol?

☒ Ya☐ Tidak

Apakah Anda sering batuk?


☒ Ya☐ Tidak

Apakah Anda sering mengalami sesak napas?

☒ Ya☐ Tidak

Apakah Anda mengalami kesulitan saat menelan?

☒ Ya☐ Tidak



HomeAboutDoctorsPagesBlogContact

HOT LINE- 09856

Apakah Anda sering mengalami sesak napas?

☒ Ya☐ Tidak

Apakah Anda mengalami kesulitan saat menelan?

☒ Ya☐ Tidak

Apakah Anda mengalami nyeri di dada?

☒ Ya☐ Tidak

PREDICT

Setelah memasukkan nilai pada halaman form diagnosa, sistem akan mengarahkan ke halaman hasil diagnosa seperti pada gambar 46.

## **BAB 5**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Mode Random Forest menunjukkan kinerja yang lebih baik jika dinilai dari segi akurasi, presisi, recall, F1 Score, dan confusion matrix. Hal ini menunjukkan bahwa Random Forest, sebagai model ensemble, mampu mengatasi masalah overfitting yang sering dialami dalam penggunaan model Decision Tree dan mampu menghasilkan prediksi yang lebih stabil dibandingkan Naive Bayes, yang seringkali terlalu menyederhanakan asumsi independensi antar fitur.

#### **5.2 Saran**

Meskipun Random Forest menunjukkan kinerja yang baik dalam menghasilkan prediksi diagnosa, kita juga perlu untuk mengeksplorasi model lain seperti Gradient Boosting, XGBoost, atau algoritma ensemble lainnya. Hal ini dapat memberikan perspektif baru untuk pengembangan sistem dan meningkatkan akurasi model lebih lanjut.



## **LAMPIRAN**

Link Github :

<https://github.com/Ratna2412/UTS-MLPRAK-KEL2-SMT5.git>

## DAFTAR PUSTAKA

- Gori, T., Sunyoto, A., & Fatta, H. A. (2024, Februari). PREPROCESSING DATA DAN KLASIFIKASI UNTUK PREDIKSI KINERJA AKADEMIK SISWA. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 11(1), 215-224. <https://jtiik.ub.ac.id/index.php/jtiik/article/view/8074/1274>
- Kusnaldi, M. R., Gulo, T., & Aripin, S. (2022, 09 03). Penerapan Normalisasi Data Dalam Mengelompokkan Data Mahasiswa Dengan Menggunakan Metode K-Means Untuk Menentukan Prioritas Bantuan Uang Kuliah Tunggal. *Journal of Computer System and Informatics (JoSYC)*, 3(4), 330-338. <https://doi.org/10.47065/josyc.v3i4.2112>
- Nasution, D. A., Khotimah, H. H., & Chamidah, N. (2019, Januari). PERBANDINGAN NORMALISASI DATA UNTUK KLASIFIKASI WINE MENGGUNAKAN ALGORITMA K-NN. *CESS (Journal of Computer Engineering System and Science)*, 4(1), 78-82. <https://jurnal.unimed.ac.id/2012/index.php/cess/article/download/11458/pdf>
- Putra, J. W. G. (2020). *Pengenalan Konsep Pembelajaran Mesin dan Deep Learning* (1.4 ed.). Self-published.
- Suryanegara, G. A. B., Adiwijaya, & Purbolaksono, M. D. (2021, Februari 20). Peningkatan Hasil Klasifikasi pada Algoritma Random Forest untuk Deteksi Pasien Penderita Diabetes Menggunakan Metode Normalisasi. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 5(1), 114 - 122. <https://doi.org/10.29207/resti.v5i1.2880>