LAPORAN UJIAN TENGAH SEMESTER

MACHINE LEARNING (PRAKTIKUM)

"Evaluasi Algoritma *Decision Tree, Naive Bayes, dan Random Forest* dalam Prediksi Kanker Paru Berdasarkan Accuracy Tertinggi"

Dosen Pengampu :
Tesa Erianti, S.Kom., M.Kom.

Kelompok 3 :

Amelia Nur Indah P	**(434221052)**
Andyka Salom	**(434221054)**
Lintang Angraini N.L.R	**(434221057)**

FAKULTAS VOKASI
UNIVERSITAS AIRLANGGA

2024

DAFTAR ISI

DAFTAR ISI 2
BAB I 4
PENDAHULUAN 4
1.1 LATAR BELAKANG 4
BAB II 5
LANDASAN TEORI 5
2.1 DECISION TREE 5
2.2 NAIVE BAYES 5
2.3 RANDOM FOREST 6
2.4 CONFUSION MATRIX 6
BAB III 7
METODE PENELITIAN 7
3.1 DESAIN PENELITIAN 7
3.1 DATA KANKER PARU-PARU 8
BAB IV 10
HASIL DAN PEMBAHASAN 10
4.1 PREPROCESSING DATA 10
4.2 NORMALISASI MENGGUNAKAN MIN-MAX SCALING 18
4.3 PERCOBAAN ALGORITMA DECISION TREE 19
1. Confusion Matrix 27
2. Matriks 27
3. Summary Table 27
4.4 PERCOBAAN ALGORITMA NAIVE BAYES 28
1. Precision: 32
2. Recall (Sensitivity): 32
3. F1-Score: 32
4. Akurasi: 32
5. Sensitivity dan Specificity: 32
4.5 PERCOBAAN ALGORITMA RANDOM FOREST 33
1. Best Parameters 36
2. Best Cross-Validation Score 37
3. Classification Report 37
4. Akurasi 37
4.6 PEMILIHAN ALGORITMA 38
4.7 IMPLEMENTASI FLASK 39
1. Penjelasan Program 39
2. Hasil Tampilan flask 50
BAB V 52
KESIMPULAN 52
5.1 KESIMPULAN 52

5.2 LINK GITHUB..... 52
BAB VI..... 53
DAFTAR PUSTAKA..... 53

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Kanker paru-paru merupakan salah satu penyebab utama kematian di seluruh dunia. Menurut Organisasi Kesehatan Dunia (WHO), pada tahun 2020 terdapat sekitar 2,2 juta kasus baru kanker paru-paru dengan sekitar 1,8 juta kematian terkait kanker ini. Data ini menunjukkan bahwa kanker paru-paru bukan hanya menjadi masalah kesehatan masyarakat, tetapi juga tantangan dalam bidang penelitian dan teknologi kesehatan. Dengan meningkatnya persentase angka penderita kanker, diperlukan untuk melakukan pengembangan metode yang dapat membantu dalam diagnosis dan pengobatan penyakit ini dan memerlukan adanya beberapa pemenuhan kebutuhan yang mendesak.

Dalam beberapa tahun terakhir, kemajuan di bidang *machine learning* memberikan beberapa harapan baru dalam prediksi dan klasifikasi penyakit, termasuk kanker. Dengan menggunakan algoritma yang tepat, kita dapat menganalisis data medis yang besar dan kompleks untuk mengidentifikasi pola yang mungkin tidak terlihat dengan metode konvensional. Metode seperti *Decision Tree, Naive Bayes*, dan *Random Forest* telah menjadi salah satu pilihan karena populer dalam analisis data medis kemampuan dalam menangani data dengan baik bisa memberikan hasil yang interpretatif.

Decision Tree merupakan metode yang intuitif dan mudah dipahami. Metode ini berisi tentang pembagian data menjadi subset yang lebih kecil berdasarkan kriteria tertentu, sehingga bisa muncul beberapa kemungkinan untuk para profesional kesehatan bisa memahami faktor-faktor yang mempengaruhi perkembangan kanker paru. Sementara itu, *Naive Bayes* memberikan penawaran kecepatan dan efisiensi dalam klasifikasi, khususnya untuk dataset besar, berkat asumsi independensi antar fitur. Di sisi lain, *Random Forest* bisa menggabungkan kekuatan dari beberapa decision tree untuk meningkatkan akurasi dan mengurangi risiko *overfitting*, menjadikannya alat yang kuat dalam prediksi penyakit.

Penelitian ini berfokus pada evaluasi untuk ketiga algoritma ini dalam konteks prediksi kanker paru berdasarkan tingkat akurasi tertinggi yang dapat dicapai. Melalui pemahaman yang lebih baik tentang performa masing-masing algoritma, diharapkan dapat diperoleh rekomendasi mengenai metode yang paling efektif untuk digunakan dalam diagnosis kanker paru di masa depan. Dengan demikian, penelitian ini tidak hanya berfokus pada pemberian kontribusi pada bidang *machine learning* akan tetapi dapat membantu dalam penanganan kanker paru yang lebih baik.

BAB II

LANDASAN TEORI

2.1 DECISION TREE

Decision tree adalah salah satu metode pada data mining yang digunakan untuk mengklasifikasi dan memprediksi. Decision tree menggunakan struktur pohon untuk merepresentasikan aturan-aturan keputusan yang bisa mudah dipahami dan diinterpretasikan.

Menurut Suyanto et al. (2019), "Decision tree merupakan metode klasifikasi yang menggunakan struktur pohon, di mana setiap node merepresentasikan atribut, setiap cabang merepresentasikan nilai dari atribut, dan setiap daun merepresentasikan kelas" [1]. Metode ini menggunakan pembagian data menjadi subset yang lebih kecil berdasarkan atribut-atribut tertentu.

Terdapat dua jenis algoritma Decision Tree yang terkenal, yaitu C4.5 dan Random Forest. Algoritma C4.5 merupakan algoritma yang dikembangkan dari algoritma ID3. C4.5 ini merupakan algoritma turunan dari algoritma ID3 dengan beragam peningkatan. Beberapa peningkatan ini diantaranya adalah, penanganan atribut-atribut numerik, missing value dan noise pada dataset, dan aturan-aturan yang dihasilkan dari model pohon yang terbentuk (Larasati & Sutrisno, 2018). Sedangkan untuk proses klasifikasi pada RF dilaksanakan dengan mengambil keputusan yang dominan dari setiap pohon yang terbentuk. RF bertujuan untuk menyelesaikan permasalahan overfitting pada penggunaan algoritma Decision tree (Wambui, George, & Kimani, 2018)

2.2 NAIVE BAYES

Naive Bayes adalah algoritma klasifikasi berbasis Teorema Bayes yang digunakan dengan asumsi independensi antara fitur-fitur dalam dataset. Algoritma ini menghitung probabilitas posterior dari sebuah kelas berdasarkan fitur yang ada, dengan rumus:

$$P(a|y) = \frac{P(y|a) \cdot P(a)}{P(y)}$$

Di mana $P(a|y)$ adalah probabilitas posterior dari kelas a, $P(y|a)$ adalah kemungkinan (likelihood) fitur y yang diberikan kelas a, $P(a)$ adalah probabilitas awal kelas a, dan $P(y)$ adalah probabilitas fitur y secara keseluruhan. Algoritma ini sangat efektif untuk dataset yang besar dan juga untuk tugas klasifikasi teks karena kesederhanaan dan kecepatan dalam proses training (Sarker, 2021; Forkenbrock, 2001)

2.3 RANDOM FOREST

Random Forest adalah metode ensemble yang terdiri dari kumpulan pohon keputusan (decision tree) yang dibentuk selama proses pelatihan. Algoritma ini menggabungkan hasil dari beberapa pohon untuk membuat prediksi yang lebih akurat dan stabil. Keunggulan utama Random Forest adalah kemampuannya dalam mengurangi overfitting dengan menggabungkan hasil prediksi dari banyak pohon keputusan (Wu et al., 2019; Mallick et al., 2017).

Setiap pohon dalam Random Forest dibangun dari subset acak data dan fitur, dan prediksi akhir dibuat berdasarkan hasil voting atau rata-rata prediksi dari seluruh pohon (Breiman, 2001). Karena sifatnya yang robust terhadap data berdimensi tinggi, Random Forest sering digunakan dalam klasifikasi teks, citra, dan penyakit klinis seperti kanker (Chen & Huang, 2020; Sarkar & Mitra, 2018)

2.4 CONFUSION MATRIX

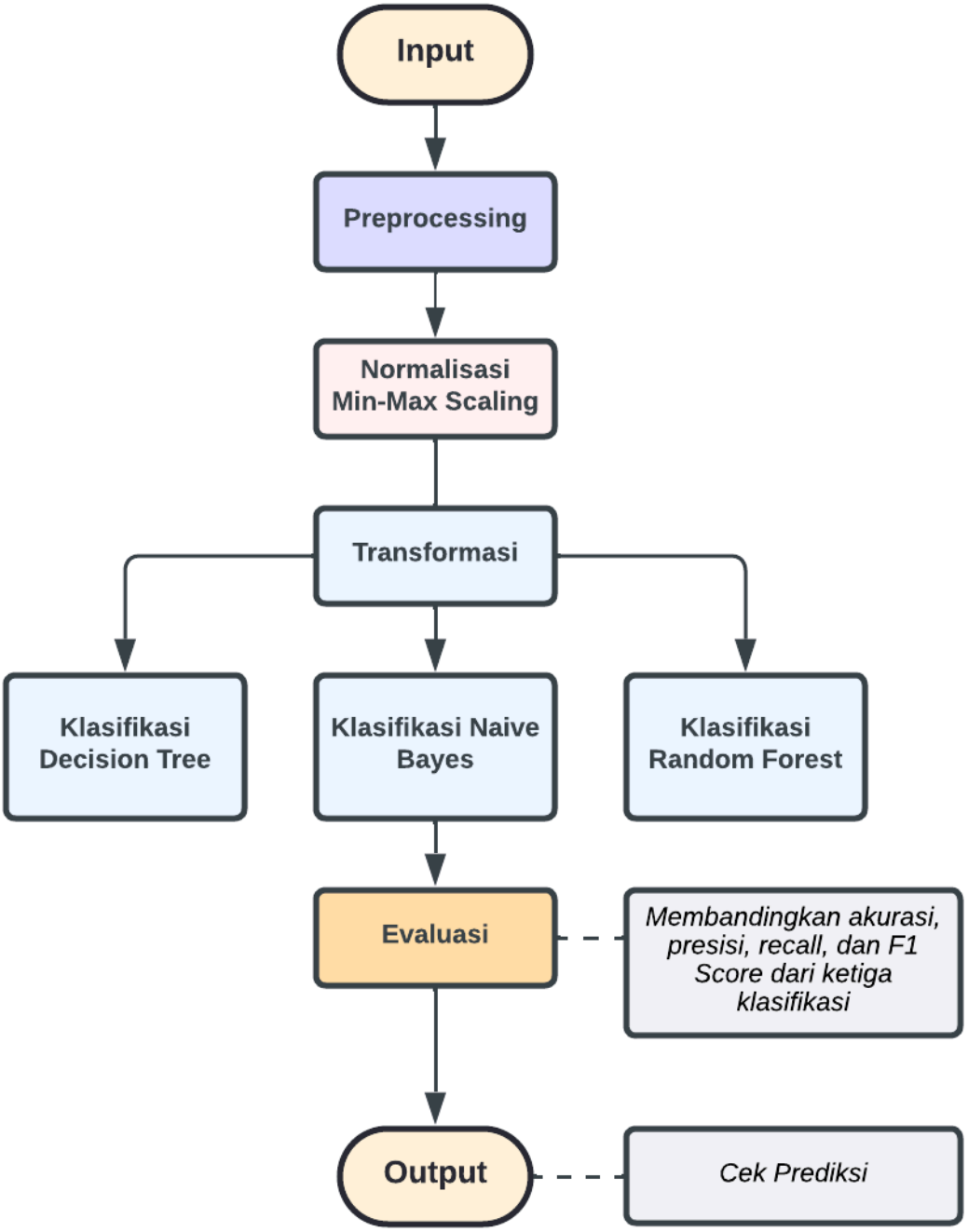
Confusion Matrix adalah representasi matriks yang menunjukkan kinerja model klasifikasi dengan menampilkan jumlah **True Positives (TP)**, **False Positives (FP)**, **True Negatives (TN)**, dan **False Negatives (FN)**. Dari Confusion Matrix, kita dapat menghitung metrik seperti akurasi, presisi, recall, dan F1 score, yang penting untuk mengevaluasi performa model (Forkenbrock, 2001).

Misalnya, Decision Tree, Naive Bayes, dan Random Forest sering dievaluasi menggunakan Confusion Matrix untuk menentukan tingkat kesalahan klasifikasi dan seberapa baik model dapat menggeneralisasi data baru. Dengan Confusion Matrix, kita dapat melihat berapa banyak data yang diklasifikasikan secara benar dan salah, serta menghitung akurasi secara keseluruhan (Wu et al., 2019; Mallick et al., 2017)

BAB III

METODE PENELITIAN

3.1 DESAIN PENELITIAN



3.1 DATA KANKER PARU-PARU

TABEL DATASET CONTOH

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
M	69	1	2	2	1	1	2	1	2	2	2	2	2	2	YES
M	74	2	1	1	1	2	2	2	1	1	1	2	2	2	YES
F	59	1	1	1	2	1	2	1	2	1	2	2	1	2	NO
M	63	2	2	2	1	1	1	1	1	2	1	1	2	2	NO
F	63	1	2	1	1	1	1	1	2	1	2	2	1	1	NO
....	....	....	....	....	....	....	....	....	....	....	....	....	....	....	....
F	56	1	1	1	2	2	2	1	1	2	2	2	2	1	YES
M	70	2	1	1	1	1	2	2	2	2	2	2	1	2	YES
M	58	2	1	1	1	1	1	2	2	2	2	1	1	2	YES
M	67	2	1	2	1	1	2	2	1	2	2	2	1	2	YES
M	62	1	1	1	2	1	2	2	2	2	1	1	2	1	YES

Dataset ini sangat berguna dalam perawatan kesehatan, memberikan landasan terstruktur untuk pengembangan model deteksi kanker. Dataset mencakup berbagai gejala Kanker Paru-Paru, seperti 'GENDER', 'AGE', 'SMOKING', dan lainnya. Setiap kategori dipilih secara cermat untuk mendukung pengembangan model yang akurat dan serbaguna. Fitur yang digunakan adalah 15 fitur dan 1 kolom digunakan untuk data target.

Komposisi Dataset:
Dataset Kanker Paru-Paru ini berisi berbagai gejala penting yang memungkinkan analisis yang mendalam serta pengembangan model prediksi. Kategori utama yang terdapat dalam data ini meliputi:

1. **Demografi Pasien:**
    - *AGE :* Informasi mengenai usia saat diagnosis, memungkinkan analisis insiden kanker berdasarkan kelompok usia.
    - *GENDER :* Mencakup informasi tentang jenis kelamin pasien untuk mendukung studi berbasis gender.
    - *SMOKING :* Dikategorikan menjadi perokok aktif, mantan perokok, atau bukan perokok, yang penting untuk menilai dampak merokok terhadap risiko dan perkembangan kanker paru-paru.
2. **Riwayat Kesehatan:**
    - *COMORBIDITIES :* Menyediakan detail tentang masalah kesehatan lain, seperti penyakit paru obstruktif kronis (PPOK), yang relevan dalam perencanaan perawatan dan prognosis pasien. Contoh Data : CHRONIC_DISEASE, PEER_PRESSURE, ANXIETY, FATIGUE, ALLERGY
3. **Data Klinis:**
    - *VITAL SIGNS :* Catatan mengenai tekanan darah, detak jantung, laju pernapasan, dan tanda-tanda vital lainnya saat diagnosis dan selama proses perawatan. Contoh Data : YELLOW_FINGERS, WHEEZING, ALCOHOL_CONSUMING, COUGHING, SHORTNESS_OF_BREATH, SWALLOWING_DIFFICULTY, CHEST_PAIN

Dataset ini bersumber dari Kaggle dan memberikan wawasan yang signifikan bagi pengembangan model deteksi kanker paru-paru.

BAB IV

HASIL DAN PEMBAHASAN

4.1 PREPROCESSING DATA

- **Mengimpor library**

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
```

Penjelasan :

Mengimport library seperti pandas dan numpy yang digunakan untuk mengolah data, lalu terdapat sklearn.preprocessing import MinMaxScaler yang digunakan untuk perhitungan normalisasi minmax.

- **Membaca dataset**

```
fileLocation = '../mlprak_uts/dataset.csv'
df = pd.read_csv(fileLocation)
```

Penjelasan :

Code di atas digunakan untuk membaca dataset yang berisi data kanker paru-paru, lalu membuat variabel df sebagai dataframe untuk membaca lokasi datanya.

- **Menampilkan data**

```
print("\n Data yang telah dibaca:")
print(df)
print("==================================================================")
```

Penjelasan :

Print di atas akan menampilkan semua dataset lung cancer.

Hasil :

```
 Data yang telah dibaca:
    GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  ...  COUGHING  SHORTNESS_OF_BREATH  SWALLOWING_DIFFICULTY  CHEST_PAIN  LUNG_CANCER
0       M   69        1               2        2  ...         2                    2                      2           2          YES
1       M   74        2               1        1  ...         1                    2                      2           2          YES
2       F   59        1               1        1  ...         2                    2                      1           2           NO
3       M   63        2               2        2  ...         1                    1                      2           2           NO
4       F   63        1               2        1  ...         2                    2                      1           1           NO
..     ...  ...      ...             ...      ...  ...       ...                  ...                    ...         ...          ...
304     F   56        1               1        1  ...         2                    2                      2           1          YES
305     M   70        2               1        1  ...         2                    2                      1           2          YES
306     M   58        2               1        1  ...         2                    1                      1           2          YES
307     M   67        2               1        2  ...         2                    2                      1           2          YES
308     M   62        1               1        1  ...         1                    1                      2           1          YES
```

- **Mendeteksi missvalue**

```
# mendeteksi missvalue
print("\n Deteksi data yang missing Value:")
missvalue = df.isna().sum()
print(missvalue)
print("==================================================================")
```

```
print("\n Baris dengan missing values:")
print(df[df.isna().any(axis=1)])
print("==========================================================")
```

Penjelasan :

Code di atas akan mencari dan menampilkan missvalue yang ada di dalam dataset baik itu berupa data kosong, atau data yang hilang. Serta akan ditampilkan juga baris yang memiliki missvalue di dalamnya, hasilnya adalah empty df yang berarti tidak terdapat missvalue di dalam data di setiap kolomnya.

Hasil :

```
 Deteksi data yang 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
==========================================================

 Baris dengan missing values:
Empty DataFrame
Columns: [GENDER, AGE, SMOKING, YELLOW_FINGERS, ANXIETY, PEER_PRESSURE, CHRONIC_DISEASE, FATIGUE, ALLERGY, WHEEZING, ALCOHOL_CON
SUMING, COUGHING, SHORTNESS_OF_BREATH, SWALLOWING_DIFFICULTY, CHEST_PAIN, LUNG_CANCER]
Index: []
```

- **Melakukan pengecekan data yang outlier pada kolom yang memiliki nilai numerik**

```
# outlier pada kolom "AGE"
def detect_outlier(AGE):
    outliers = []
    threshold = 3
    mean_AGE = np.mean(AGE)
    std_AGE = np.std(AGE)

    for y in AGE:
        z_score = (y - mean_AGE) / std_AGE
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['AGE'].values)
print("\n Outlier data points in 'AGE':", outlier_datapoints)

# outlier pada kolom "SMOKING"
def detect_outlier(SMOKING):
```

```
    outliers = []
    threshold = 3
    mean_SMOKING = np.mean(SMOKING)
    std_SMOKING = np.std(SMOKING)

    for y in SMOKING:
        z_score = (y - mean_SMOKING) / std_SMOKING
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['SMOKING'].values)
print("\n Outlier data points in 'SMOKING':", outlier_datapoints)

# outlier pada kolom "YELLOW_FINGERS"
def detect_outlier(YELLOW_FINGERS):
    outliers = []
    threshold = 3
    mean_YELLOW_FINGERS = np.mean(YELLOW_FINGERS)
    std_YELLOW_FINGERS = np.std(YELLOW_FINGERS)

    for y in YELLOW_FINGERS:
        z_score = (y - mean_YELLOW_FINGERS) / std_YELLOW_FINGERS
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['YELLOW_FINGERS'].values)
print("\n Outlier data points in 'YELLOW_FINGERS':", outlier_datapoints)

# outlier pada kolom "ANXIETY"
def detect_outlier(ANXIETY):
    outliers = []
    threshold = 3
    mean_ANXIETY = np.mean(ANXIETY)
    std_ANXIETY = np.std(ANXIETY)

    for y in ANXIETY:
        z_score = (y - mean_ANXIETY) / std_ANXIETY
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
```

```
outlier_datapoints = detect_outlier(df['ANXIETY'].values)
print("\n Outlier data points in 'ANXIETY':", outlier_datapoints)

# outlier pada kolom "PEER_PRESSURE"
def detect_outlier(PEER_PRESSURE):
    outliers = []
    threshold = 3
    mean_PEER_PRESSURE = np.mean(PEER_PRESSURE)
    std_PEER_PRESSURE = np.std(PEER_PRESSURE)

    for y in PEER_PRESSURE:
        z_score = (y - mean_PEER_PRESSURE) / std_PEER_PRESSURE
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['PEER_PRESSURE'].values)
print("\n Outlier data points in 'PEER_PRESSURE':", outlier_datapoints)

# outlier pada kolom "CHRONIC_DISEASE"
def detect_outlier(CHRONIC_DISEASE):
    outliers = []
    threshold = 3
    mean_CHRONIC_DISEASE = np.mean(CHRONIC_DISEASE)
    std_CHRONIC_DISEASE = np.std(CHRONIC_DISEASE)

    for y in CHRONIC_DISEASE:
        z_score = (y - mean_CHRONIC_DISEASE) / std_CHRONIC_DISEASE
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['CHRONIC_DISEASE'].values)
print("\n Outlier data points in 'CHRONIC_DISEASE':", outlier_datapoints)

# outlier pada kolom "FATIGUE"
def detect_outlier(FATIGUE):
    outliers = []
    threshold = 3
    mean_FATIGUE = np.mean(FATIGUE)
    std_FATIGUE = np.std(FATIGUE)
```

```
    for y in FATIGUE:
        z_score = (y - mean_FATIGUE) / std_FATIGUE
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['FATIGUE'].values)
print("\n Outlier data points in 'FATIGUE':", outlier_datapoints)

# outlier pada kolom "ALLERGY"
def detect_outlier(ALLERGY):
    outliers = []
    threshold = 3
    mean_ALLERGY = np.mean(ALLERGY)
    std_ALLERGY = np.std(ALLERGY)

    for y in ALLERGY:
        z_score = (y - mean_ALLERGY) / std_ALLERGY
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['ALLERGY'].values)
print("\n Outlier data points in 'ALLERGY':", outlier_datapoints)

# outlier pada kolom "WHEEZING"
def detect_outlier(WHEEZING):
    outliers = []
    threshold = 3
    mean_WHEEZING = np.mean(WHEEZING)
    std_WHEEZING = np.std(WHEEZING)

    for y in WHEEZING:
        z_score = (y - mean_WHEEZING) / std_WHEEZING
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['WHEEZING'].values)
print("\n Outlier data points in 'WHEEZING':", outlier_datapoints)

# outlier pada kolom "ALCOHOL_CONSUMING"
def detect_outlier(ALCOHOL_CONSUMING):
```

```
    outliers = []
    threshold = 3
    mean_ALCOHOL_CONSUMING = np.mean(ALCOHOL_CONSUMING)
    std_ALCOHOL_CONSUMING = np.std(ALCOHOL_CONSUMING)

    for y in ALCOHOL_CONSUMING:
        z_score = (y - mean_ALCOHOL_CONSUMING) / std_ALCOHOL_CONSUMING
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['ALCOHOL_CONSUMING'].values)
print("\n Outlier data points in 'ALCOHOL_CONSUMING':", outlier_datapoints)

# outlier pada kolom "COUGHING"
def detect_outlier(COUGHING):
    outliers = []
    threshold = 3
    mean_COUGHING = np.mean(COUGHING)
    std_COUGHING = np.std(COUGHING)

    for y in COUGHING:
        z_score = (y - mean_COUGHING) / std_COUGHING
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['COUGHING'].values)
print("\n Outlier data points in 'COUGHING':", outlier_datapoints)

# outlier pada kolom "SHORTNESS_OF_BREATH"
def detect_outlier(SHORTNESS_OF_BREATH):
    outliers = []
    threshold = 3
    mean_SHORTNESS_OF_BREATH = np.mean(SHORTNESS_OF_BREATH)
    std_SHORTNESS_OF_BREATH = np.std(SHORTNESS_OF_BREATH)

    for y in SHORTNESS_OF_BREATH:
        z_score = (y - mean_SHORTNESS_OF_BREATH) / std_SHORTNESS_OF_BREATH
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
```

```
outlier_datapoints = detect_outlier(df['SHORTNESS_OF_BREATH'].values)
print("\n Outlier data points in 'SHORTNESS_OF_BREATH':", outlier_datapoints)

# outlier pada kolom "SWALLOWING_DIFFICULTY"
def detect_outlier(SWALLOWING_DIFFICULTY):
    outliers = []
    threshold = 3
    mean_SWALLOWING_DIFFICULTY = np.mean(SWALLOWING_DIFFICULTY)
    std_SWALLOWING_DIFFICULTY = np.std(SWALLOWING_DIFFICULTY)

    for y in SWALLOWING_DIFFICULTY:
        z_score = (y - mean_SWALLOWING_DIFFICULTY) /
            std_SWALLOWING_DIFFICULTY
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['SWALLOWING_DIFFICULTY'].values)
print("\n Outlier data points in 'SWALLOWING_DIFFICULTY':", outlier_datapoints)

# outlier pada kolom "CHEST_PAIN"
def detect_outlier(CHEST_PAIN):
    outliers = []
    threshold = 3
    mean_CHEST_PAIN = np.mean(CHEST_PAIN)
    std_CHEST_PAIN = np.std(CHEST_PAIN)

    for y in CHEST_PAIN:
        z_score = (y - mean_CHEST_PAIN) / std_CHEST_PAIN
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
outlier_datapoints = detect_outlier(df['CHEST_PAIN'].values)
print("\n Outlier data points in 'CHEST_PAIN':", outlier_datapoints)
```

Penjelasan :

Code di atas akan melakukan outlier pada data yang memiliki nilai tidak normal atau tidak sesuai dengan nilai-nilai pada umumnya, lalu akan ditampilkan kolom mana saja yang memiliki outlier. Dan hasilnya tidak ada data pada kolom yang memiliki nilai outlier.

Hasil :

```
Outlier data points in 'AGE': [21, 38]

Outlier data points in 'SMOKING': []

Outlier data points in 'YELLOW_FINGERS': []

Outlier data points in 'ANXIETY': []

Outlier data points in 'PEER_PRESSURE': []

Outlier data points in 'CHRONIC_DISEASE': []

Outlier data points in 'FATIGUE': []

Outlier data points in 'ALLERGY': []

Outlier data points in 'WHEEZING': []

Outlier data points in 'ALCOHOL_CONSUMING': []

Outlier data points in 'COUGHING': []

Outlier data points in 'SHORTNESS_OF_BREATH': []

Outlier data points in 'SWALLOWING_DIFFICULTY': []

Outlier data points in 'CHEST_PAIN': []
```

- **Memastikan ulang dengan menghapus kolom yang memiliki nilai outlier**

```
# inisialisasi df_cleaned dan membuat variabel untuk mengecek (columns_check)
df_cleaned = df.copy()
columns_to_check = ['AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE',
                    'CHRONIC_DISEASE', 'FATIGUE', 'ALLERGY', 'WHEEZING',
                    'ALCOHOL_CONSUMING', 'COUGHING', 'SHORTNESS_OF_BREATH',
                    'SWALLOWING_DIFFICULTY', 'CHEST_PAIN']

# loop pada setiap kolom dan hapus outlier
for column in columns_to_check:
    outliers = detect_outlier(df_cleaned[column].values)
```

```
    # Buat mask untuk baris yang bukan outlier
    mask = ~df_cleaned[column].isin(outliers)
    df_cleaned = df_cleaned[mask]

print("\nData setelah menghapus outliers:")
print(df_cleaned)
print("============================================================")
```

Penjelasan :

Code di atas akan menginialisasi variabel baru yaitu df_cleaned yang digunakan untuk menyimpan data yang sudah bersih, dan memastikan untuk menghapus data yang memiliki nilai outlier di dalamnya.

Hasil :

```
Data setelah menghapus outliers:
     GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  ...  COUGHING  SHORTNESS_OF_BREATH  SWALLOWING_DIFFICULTY  CHEST_PAIN  LUNG_CANCER
0         M   69        1               2        2  ...         2                    2                      2           2          YES
1         M   74        2               1        1  ...         1                    2                      2           2          YES
2         F   59        1               1        1  ...         2                    2                      1           2           NO
3         M   63        2               2        2  ...         1                    1                      2           2           NO
4         F   63        1               2        1  ...         2                    2                      1           1           NO
..      ...  ...      ...             ...      ...  ...       ...                  ...                    ...         ...          ...
304       F   56        1               1        1  ...         2                    2                      2           1          YES
305       M   70        2               1        1  ...         2                    2                      1           2          YES
306       M   58        2               1        1  ...         2                    1                      1           2          YES
307       M   67        2               1        2  ...         2                    2                      1           2          YES
308       M   62        1               1        1  ...         1                    1                      2           1          YES
```

4.2 NORMALISASI MENGGUNAKAN MIN-MAX SCALING

- **Menginisialisasi scaler yang akan mengimport dari minmax scaler milik sklearn**

```
# menginisialisasi scaler
scaler = MinMaxScaler()
```

Penjelasan :

Code ini digunakan untuk membuat objek MinMaxScaler dari library sklearn.preprocessing. Fungsi scaler ini digunakan untuk menormalisasi data numerik ke dalam rentang [0, 1], dengan cara mengubah setiap nilai dalam kolom berdasarkan nilai minimum dan maksimum dari kolom tersebut.

- **Membuat daftar kolom yang memiliki data numerik dan akan dinormalisasi**

```
# kolom numerik yang dinormalisasi
numeric_columns = ['AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE',
                   'CHRONIC_DISEASE', 'FATIGUE', 'ALLERGY', 'WHEEZING',
                   'ALCOHOL_CONSUMING', 'COUGHING', 'SHORTNESS_OF_BREATH',
                   'SWALLOWING_DIFFICULTY', 'CHEST_PAIN']
```

Penjelasan :

Code ini membuat variabel baru yang bernama numeric_columns yang digunakan untuk memasukkan kolom-kolom numerik yang akan dinormalisasi.

- **Melakukan normalisasi minmax**

```
# normalisasi kolom numerik
df_cleaned[numeric_columns] = scaler.fit_transform(df_cleaned[numeric_columns])

print("\nData setelah normalisasi Min-Max:")
print(df_cleaned)
print("============================================================")
```

Penjelasan :

Code di atas scaler.fit_transform() digunakan untuk menyesuaikan scaler dengan data (menghitung nilai minimum dan maksimum), kemudian mentransformasi data numerik menjadi nilai dalam rentang [0, 1]. Hasil dari transformasi akan menggantikan nilai asli di kolom numeric_columns pada dataframe df_cleaned. Lalu akan ditampilkan hasilnya data yang sudah dinormalisasi.

Hasil :

```
Data setelah normalisasi Min-Max:
    GENDER       AGE  SMOKING  YELLOW_FINGERS  ...  SHORTNESS_OF_BREATH  SWALLOWING_DIFFICULTY  CHEST_PAIN  LUNG_CANCER
0        M  0.625000      0.0             1.0  ...                  1.0                    1.0         1.0          YES
1        M  0.729167      1.0             0.0  ...                  1.0                    1.0         1.0          YES
2        F  0.416667      0.0             0.0  ...                  1.0                    0.0         1.0           NO
3        M  0.500000      1.0             1.0  ...                  0.0                    1.0         1.0           NO
4        F  0.500000      0.0             1.0  ...                  1.0                    0.0         0.0           NO
..     ...       ...      ...             ...  ...                  ...                    ...         ...          ...
304      F  0.354167      0.0             0.0  ...                  1.0                    1.0         0.0          YES
305      M  0.645833      1.0             0.0  ...                  1.0                    0.0         1.0          YES
306      M  0.395833      1.0             0.0  ...                  0.0                    0.0         1.0          YES
307      M  0.583333      1.0             0.0  ...                  1.0                    0.0         1.0          YES
308      M  0.479167      0.0             0.0  ...                  0.0                    1.0         0.0          YES

[307 rows x 16 columns]
============================================================
```

- **Menyimpan data yang sudah dinormalisasi ke dalam file baru**

```
# menyimpan data yang sudah dinormalisasi ke file CSV baru
output_file_normalized = '../mlprak_uts/datasetbaru_sudahnormalisasi.csv'
df_cleaned.to_csv(output_file_normalized, index=False)
print(f"\nData yang telah dinormalisasi disimpan ke {output_file_normalized}")
```

Penjelasan :

Code di atas akan menyimpan hasil dari dataframe yang sudah dinormalisasi ke dalam file CSV bernama datasetbaru_sudahnormalisasi.csv.

4.3 PERCOBAAN ALGORITMA DECISION TREE

- **Mengimport library**

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn import tree
import seaborn as sns
import matplotlib.pyplot as plt
from tabulate import tabulate
```

Penjelasan :

Code di atas akan mengimport library yang ada di python seperti pandas, numpy digunakan untuk manipulasi dan analisis data, train_test_split digunakan untuk membagi dataset menjadi training dan testing. Lalu DecisionTreeClassifier digunakan untuk membuat model decision tree, accuracy_score, classification_report, confusion_matrix digunakan untuk evaluasi model. Kemudian LabelEncoder digunakan untuk mengonversi data kategorikal menjadi numerik, seaborn, matplotlib digunakan untuk visualisasi, seperti heatmap confusion matrix dan decision tree. Tabulate digunakan untuk membuat tabel yang rapi dalam output.

- **Membaca dataset**

```
fileLocation = '../mlprak_uts/datasetbaru_sudahnormalisasi.csv'
df =pd.read_csv(fileLocation)
df.columns = ['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']
```

Penjelasan :

Membuat variabel baru yang bernama fileLocation untuk tempat datasetnya, lalu variabel df digunakan untuk membaca filenya. Dataset akan dibaca dari file CSV dan variabel df.coulumns berisi kolom yang akan dinamai ulang sesuai dengan fitur-fitur dataset yang ada.

- **Menampilkan data yang di dalam csv**

```
print("\n df yang telah dibaca:")
print(df)
print("==========================================================")
```

Penjelasan :

Code di atas akan menampilkan semua data yang disimpan dalam variabel df.

Hasil :

```
 df yang telah dibaca:
    GENDER       AGE  SMOKING  YELLOW_FINGERS  ...  SHORTNESS_OF_BREATH  SWALLOWING_DIFFICULTY  CHEST_PAIN  LUNG_CANCER
0       M  0.625000      0.0             1.0  ...                  1.0                    1.0         1.0          YES
1       M  0.729167      1.0             0.0  ...                  1.0                    1.0         1.0          YES
2       F  0.416667      0.0             0.0  ...                  1.0                    0.0         1.0           NO
3       M  0.500000      1.0             1.0  ...                  0.0                    1.0         1.0           NO
4       F  0.500000      0.0             1.0  ...                  1.0                    0.0         0.0           NO
..     ...       ...      ...             ...  ...                  ...                    ...         ...          ...
302     F  0.354167      0.0             0.0  ...                  1.0                    1.0         0.0          YES
303     M  0.645833      1.0             0.0  ...                  1.0                    0.0         1.0          YES
304     M  0.395833      1.0             0.0  ...                  0.0                    0.0         1.0          YES
305     M  0.583333      1.0             0.0  ...                  1.0                    0.0         1.0          YES
306     M  0.479167      0.0             0.0  ...                  0.0                    1.0         0.0          YES
```

- **Mendeteksi data yang memiliki nilai missvalue**

```
print("\n Deteksi df yang missing Value:")
missvalue = df.isna().sum()
print(missvalue)
print("==========================================================")
```

Penjelasan :

Code di atas akan mencari dan menampilkan missvalue yang ada di dalam dataset baik itu berupa data kosong, atau data yang hilang. Serta akan ditampilkan juga baris yang memiliki missvalue di dalamnya, hasilnya adalah empty df yang berarti tidak terdapat missvalue di dalam data di setiap kolomnya.

Hasil :

```
 Deteksi df yang 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
```

- **Melakukan encoding pada kolom yang memiliki data bernilai huruf bukan numerik**

```
# encoding kategori variabel
le = LabelEncoder()
df['GENDER'] = le.fit_transform(df['GENDER'])
df['LUNG_CANCER'] = le.fit_transform(df['LUNG_CANCER'])
```

Penjelasan :

Code di atas LabelEncoder digunakan untuk mengonversi kolom GENDER dan LUNG_CANCER dari bentuk kategori menjadi nilai numerik (0 dan 1).

- **Melakukan grouping data yang akan menjadi data training dan data target**

```
# grouping df antara df training dan df target
X = df[['GENDER','AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY', 'PEER_PRESSURE',
        'CHRONIC_DISEASE', 'FATIGUE', 'ALLERGY', 'WHEEZING',
        'ALCOHOL_CONSUMING', 'COUGHING', 'SHORTNESS_OF_BREATH',
        'SWALLOWING_DIFFICULTY', 'CHEST_PAIN']]
y = df['LUNG_CANCER']
```

Penjelasan :
Code di atas x adalah fitur-fitur yang digunakan untuk memprediksi (semua kolom kecuali LUNG_CANCER), sedangkan y adalah target yang akan diprediksi, yaitu apakah seseorang memiliki kanker paru-paru (LUNG_CANCER).

- **Membagi data menjadi 2 yaitu x dan y lalu dibedakan menjadi data training dan data testing**

```
# Split df training dan df testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)
```

Penjelasan :
Dataset dibagi menjadi 80% data training dan 20% data testing menggunakan train_test_split. Pembagian ini di-stratify berdasarkan target y, sehingga distribusi kelas pada data training dan testing tetap proporsional. Memilih pembagian 80 dengan 20 karena akurasi yang dihasilkan lebih besar dibandingkan pembagian 70 30.

- **Membangun model decision tree**

```
# Membuat decision tree
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)
```

Penjelasan :
Code di atas adalah model decision tree yang dibangun dengan menggunakan kriteria entropy (berdasarkan informasi gain). Model ini dilatih menggunakan data training (x_train, y_train).

- **Membyat prediksi dari data testing x**

```
# Membuat prediksi dari df x
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2%}")
print("==========================================================\n")
```

Penjelasan :
Code di atas akan melakukan prediksi terhadap data testing (x_test). Akurasi akan dihitung sebagai rasio prediksi yang benar terhadap jumlah total prediksi.
Hasil :

```
==============================================================
Accuracy: 83.87%
==============================================================
```

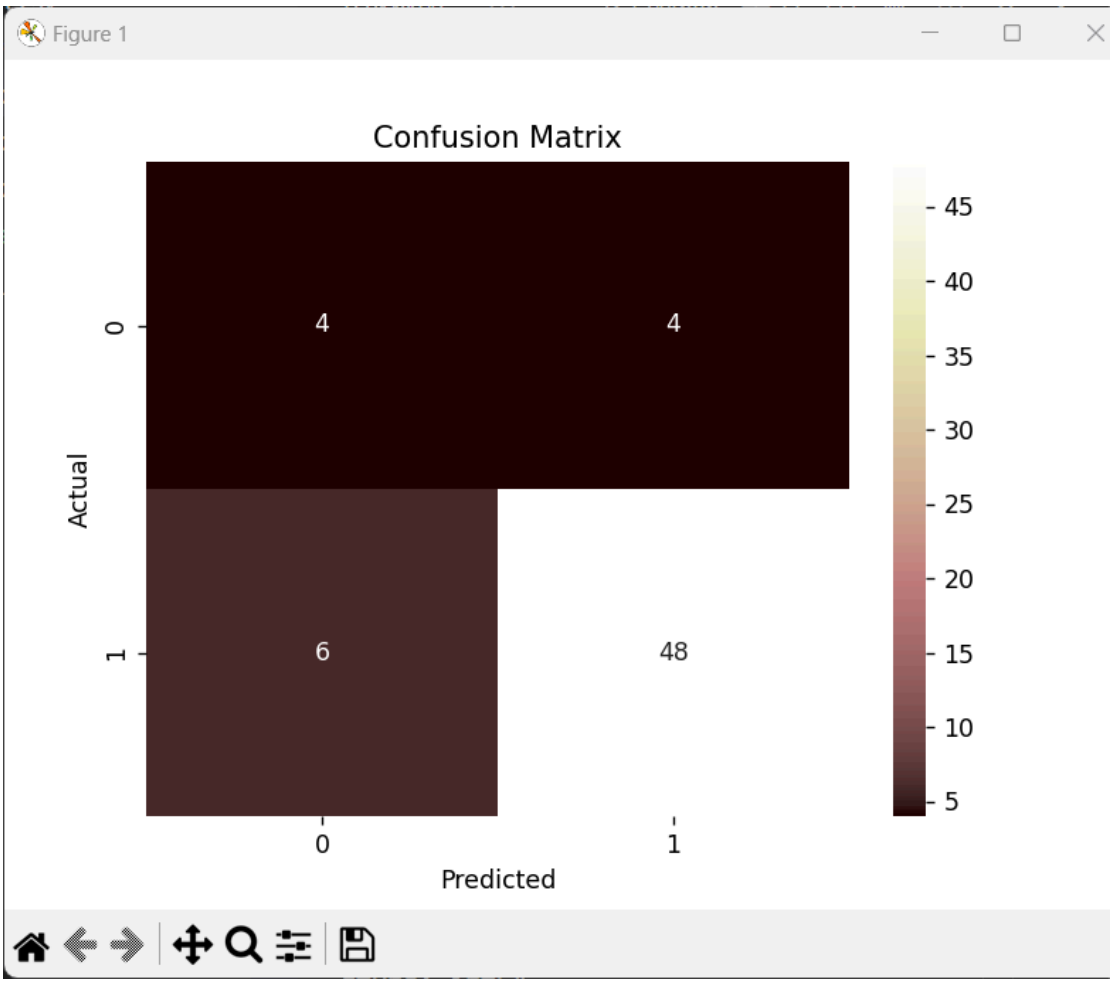
- **Mengevaluasi model dan membuatkan visualisasinya**

```
# Laporan klasifikasi dan Confusion Matrix
class_report = classification_report(y_test, y_pred, output_dict=True)
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='pink')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

Penjelasan :

Code di atas classification_report digunakan untuk memberikan metrik seperti precision, recall, dan F1-score. Hasilnya akan disimpan dalam bentuk dictionary. Kemudian confusion_matrix digunakan untuk menghitung confusion matrix untuk melihat performa prediksi dibandingkan dengan hasil aktual. Lalu seaborn digunakan untuk membuat heatmap dari confusion matrix, di mana angka akan menunjukkan jumlah prediksi yang benar dan salah.

Hasil :



- **Membuatkan confusion matrix**

```
# cm
conf_matrix_df = pd.DataFrame(conf_matrix, index=['True Class 0', 'True Class 1'],
columns=['Pred Class 0', 'Pred Class 1'])
print("\nConfusion Matrix:")
print(conf_matrix_df)
print("==========================================================\n")
```

Penjelasan :

Code di atas confusion matrix akan dicetak dalam bentuk dataframe untuk memudahkan interpretasi, dengan label yang jelas.

Hasil :

```
Confusion Matrix:
              Pred Class 0  Pred Class 1
True Class 0             4             4
True Class 1             6            48
==========================================================
```

- **Mengekstrak Precision, Recall, dan F1-Score**

```
# Ekstrak precision, recall, dan F1 Score
precision_class_1 = class_report['1']['precision']
precision_class_0 = class_report['0']['precision']
recall_class_1 = class_report['1']['recall']
recall_class_0 = class_report['0']['recall']
f1_class_1 = class_report['1']['f1-score']
f1_class_0 = class_report['0']['f1-score']
```

Penjelasan :
Code di atas precision, recall, dan F1-score diekstrak untuk masing-masing kelas (Class 0 dan Class 1).

- **Menampilkan Tabel Ringkasan Hasil Prediksi**

```
# Membuat tabel ringkasan dari hasil prediksi
summary_table = pd.DataFrame({
    '': ['Prediksi Class 1', 'Prediksi Class 0', 'Class Recall'],
    'True Class 1': [conf_matrix[1, 1], conf_matrix[1, 0], f"{recall_class_1:.2%}"],
    'True Class 0': [conf_matrix[0, 1], conf_matrix[0, 0], f"{recall_class_0:.2%}"],
    'Class Precision': [f"{precision_class_1:.2%}", f"{precision_class_0:.2%}", ""],
    'F1 Score': [f"{f1_class_1:.2%}", f"{f1_class_0:.2%}", ""]
})

print("\nSummary Table:")
print(tabulate(summary_table, headers='keys', tablefmt='grid', showindex=False))
print("==============================================================")

# Akurasi, Precision, Sensitivity, Specificity
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, zero_division=1)
TN = conf_matrix[1][1]
FN = conf_matrix[1][0]
TP = conf_matrix[0][0]
FP = conf_matrix[0][1]
sens = TN / (TN + FP) * 100 if (TN + FP) != 0 else 0
spec = TP / (TP + FN) * 100 if (TP + FN) != 0 else 0

print(f'\nAkurasi: {accuracy * 100:.2f}%')
print(f'Sensitivity: {sens:.2f}%')
print(f'Specificity: {spec:.2f}%')
print(f'Precision: {precision * 100:.2f}%')
f1_average = (f1_class_0 + f1_class_1) / 2
```

```
print(f'Rata-rata F1 Score: {f1_average:.2%}')
```

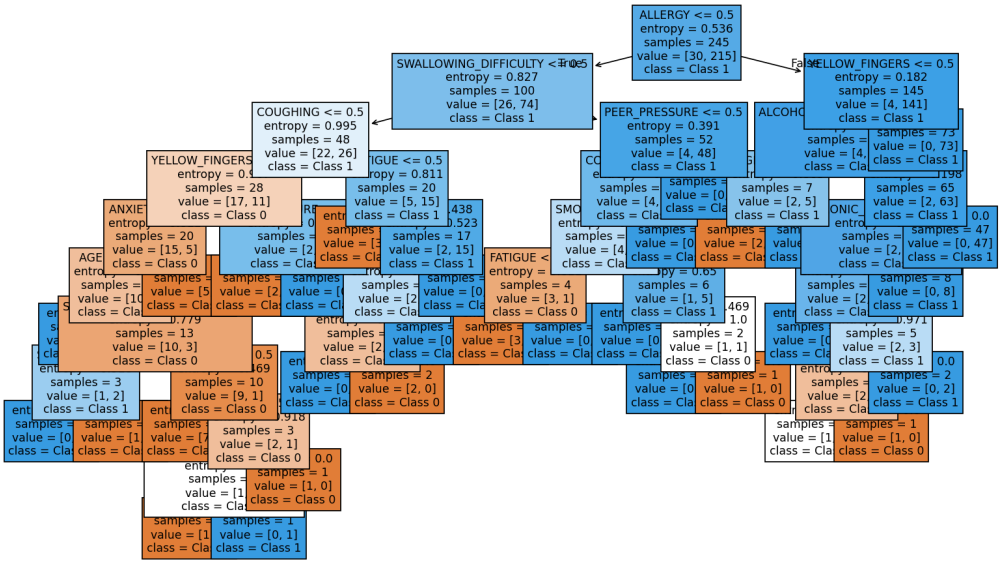
Penjelasn :
Code di atas akan menampilkan tabel ringkasan dari hasil prediksi (jumlah prediksi yang benar/salah, precision, recall, dan F1-score) dicetak dalam format yang rapi menggunakan tabulate.
Hasil :

```
Summary Table:
+------------------+----------------+----------------+-------------------+------------+
|                  | True Class 1   | True Class 0   | Class Precision   | F1 Score   |
+==================+================+================+===================+============+
| Prediksi Class 1 | 49             | 4              | 92.45%            | 91.59%     |
+------------------+----------------+----------------+-------------------+------------+
| Prediksi Class 0 | 5              | 4              | 44.44%            | 47.06%     |
+------------------+----------------+----------------+-------------------+------------+
| Class Recall     | 90.74%         | 50.00%         |                   |            |
+------------------+----------------+----------------+-------------------+------------+
============================================================

Akurasi: 85.48%
Sensitivity: 92.45%
Specificity: 44.44%
Precision: 92.45%
Rata-rata F1 Score: 69.32%
```

- **Membuat visualisasi decision tree**

```
# Plot Decision Tree
plt.figure(figsize=(20, 15))
tree.plot_tree(clf, filled=True, feature_names=X.columns, class_names=['Class 0', 'Class 1'],
fontsize=10)
plt.show()
```

Penjelasan :
Model decision tree divisualisasikan menggunakan tree.plot_tree() dari sklearn. Atribut-atribut penting dari model, seperti nama fitur dan nama kelas, ditampilkan dalam diagram pohon.
Hasil :



Penjelasan :

1. Confusion Matrix

Confusion matrix menunjukkan performa model klasifikasi dengan mengklasifikasikan data aktual dan prediksi. Di sini ada dua kelas:

- Kelas 0: Tidak memiliki kanker paru-paru.
- Kelas 1: Memiliki kanker paru-paru.

2. Matriks

- 4 prediksi benar (True Negatives): Model memprediksi kelas 0 dan sebenarnya juga kelas 0.
- 4 prediksi salah (False Positives): Model memprediksi kelas 1, tapi sebenarnya kelas 0.
- 6 prediksi salah (False Negatives): Model memprediksi kelas 0, tapi sebenarnya kelas 1.
- 48 prediksi benar (True Positives): Model memprediksi kelas 1 dan sebenarnya juga kelas 1

3. Summary Table

Tabel ini merangkum beberapa metrik penting:

- Precision Class 1 (48%): Dari semua prediksi kelas 1, hanya 48% yang benar-benar positif.
- Precision Class 0 (60%): Dari semua prediksi kelas 0, hanya 60% yang benar-benar negatif.

- Recall Class 1 (92.31%): Dari semua kasus kelas 1 (positif), model menemukan 92.31%.
- Recall Class 0 (40.00%): Dari semua kasus kelas 0 (negatif), model menemukan 40.00%.
- F1 Score: Kombinasi dari precision dan recall, yang untuk kelas 1 adalah 90.57% dan untuk kelas 0 adalah 44.44%.

4.4 PERCOBAAN ALGORITMA NAIVE BAYES

- **Program**
  Langkah - langkah pengembangan algoritma "Naive Bayes"

  - **Persiapkan Dataset**
    Penjelasan: Pertama, kita membaca dataset yang berisi data tentang beberapa faktor seperti jenis kelamin, usia, kebiasaan merokok, masalah kesehatan lainnya, serta apakah seseorang menderita kanker paru-paru atau tidak.

```
# Membaca data
dataframe = pd.read_csv('dataset_sudahnormalisasi.csv')
```

    Hasilnya: Dataset ini akan digunakan untuk melatih model prediksi kanker paru-paru.

  - **Pengecekan Missing Value**
    Penjelasan: Kita melakukan pengecekan apakah ada nilai kosong atau hilang dalam dataset. Hal ini penting karena nilai yang hilang dapat mengganggu proses pelatihan model.

```
print(data.isnull().sum())
```

    Hasilnya: Kita pastikan bahwa tidak ada nilai kosong di dalam data.

  - **Deteksi dan Penanganan Outlier**
    Penjelasan: Outlier adalah data yang tidak normal atau ekstrem. Kita mendeteksi outlier menggunakan metode Z-score. Data yang memiliki Z-score lebih dari 3 dianggap sebagai outlier dan kita hapus dari dataset.

```
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:
  if col != 'LUNG_CANCER':
    outliers[col] = detect_outlier(data[col])
```

Hasilnya: Outlier berhasil dideteksi dan dihapus, menjaga kualitas data.

- **Encoding Kategori Variabel**
  Penjelasan: Kolom yang berisi data kategori seperti "GENDER" dan "LUNG_CANCER" diubah menjadi bentuk numerik menggunakan LabelEncoder. Ini karena model Naive Bayes memerlukan input data dalam bentuk angka.

```
# encoding kategori variabel
le = LabelEncoder()
data_cleaned['GENDER'] = le.fit_transform(data_cleaned['GENDER'])
data_cleaned['LUNG_CANCER'] =
le.fit_transform(data_cleaned['LUNG_CANCER'])
```

  Hasilnya: Data yang semula berupa kategori (seperti jenis kelamin) diubah menjadi bentuk angka.

- **Normalisasi Data**
  Penjelasan: Semua fitur dalam data dinormalisasi menggunakan MinMaxScaler untuk memastikan bahwa semua fitur memiliki skala yang sama, yang membuat model bekerja lebih baik.

```
# Normalisasi data menggunakan metode minmax normalization
min_max_scaler = MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(data_cleaned[['GENDER', 'AGE',... ]])
```

  Hasilnya: Semua nilai fitur diubah menjadi rentang antara 0 dan 1.

- **Pembagian Data Testing dan Data Uji**
  Penjelasan: Dataset dibagi menjadi dua bagian, yaitu training set untuk melatih model dan testing set untuk menguji akurasi model. Biasanya pembagian dilakukan 80% untuk training dan 20% untuk testing.

```
# Pembagian training dan testing
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
```

Hasilnya: Model Naive Bayes akan dilatih menggunakan data training dan diuji menggunakan data testing.

- **Penerapan Algoritma Naive Bayes**
  Penjelasan: Algoritma Gaussian Naive Bayes digunakan untuk membuat model prediksi berdasarkan data training. Setelah itu, model akan memprediksi hasil pada data testing.

```
# Pemodelan Naive Bayes
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
```

Hasilnya: Model Naive Bayes telah dilatih dan sekarang dapat memprediksi apakah seseorang memiliki kanker paru-paru berdasarkan fitur yang ada.

- **Evaluasi Model dengan Confusion Matrix**
  Penjelasan: Kita mengevaluasi kinerja model menggunakan confusion matrix, serta menghitung akurasi, precision, sensitivity (recall), dan specificity.

```
# Akurasi, Precision, Sensitivity, Specificity, F1-Score
accuracy = accuracy_score(y_test, Y_pred)
precision = precision_score(y_test, Y_pred, zero_division=1)
f1 = f1_score(y_test, Y_pred, zero_division=1)
```

Hasilnya:

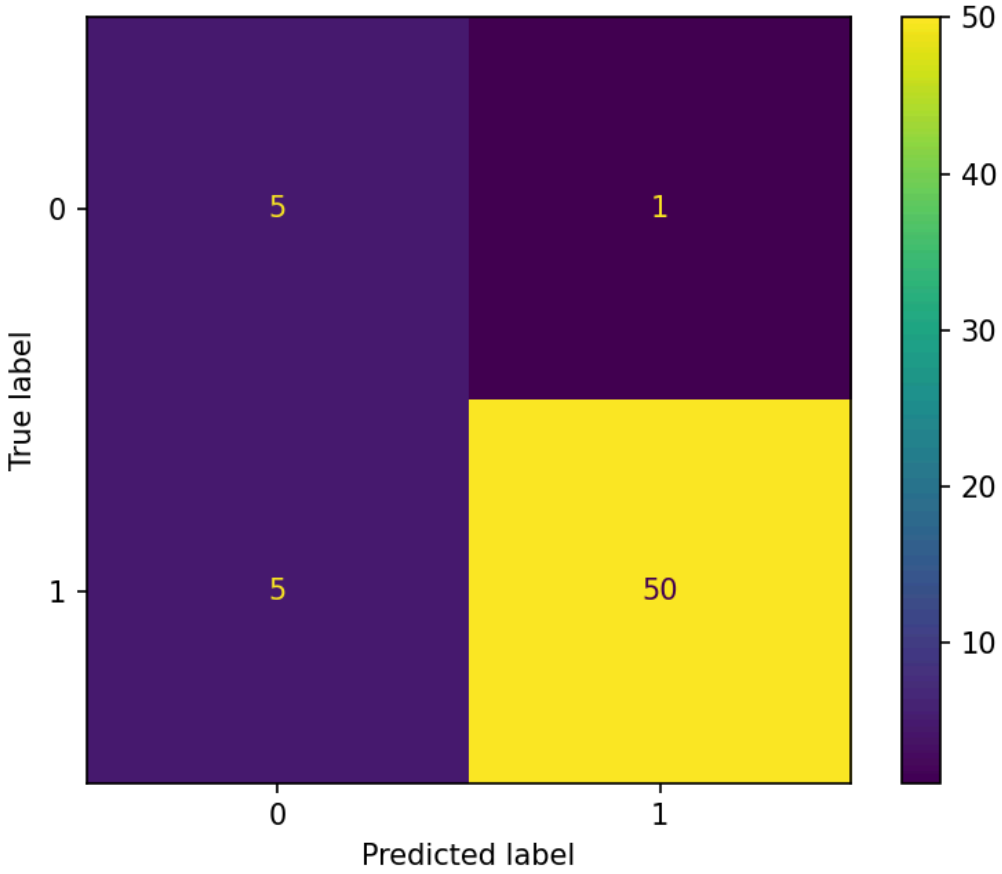
- Akurasi: Persentase prediksi yang benar.
- Precision: Seberapa tepat model dalam memberikan prediksi positif.
- Sensitivity: Kemampuan model mendeteksi kasus positif dengan benar.
- Specificity: Kemampuan model mendeteksi kasus negatif dengan benar.

Kemudian, menampilkan hasil confusion matrix dengan menggunakan

```
# Menampilkan Confusion Matrix

ConfusionMatrixDisplay(confusion_matrix=cm).plot()

plt.show()
```

Penjelasan: Confusion matrix adalah representasi tabel yang menunjukkan perbandingan antara prediksi model dan hasil sebenarnya. Kita bisa melihat jumlah **True Positive**, **False Positive**, **True Negative**, dan **False Negative**.

- **Hasil**




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

           0       0.50      0.83      0.62         6
           1       0.98      0.91      0.94        55

    accuracy                           0.90        61
   macro avg       0.74      0.87      0.78        61
weighted avg       0.93      0.90      0.91        61

Akurasi: 90.16%
Sensitivity: 98.04%
Specificity: 50.00%
Precision: 98.04%
F1-Score: 94.34%
```

Penjelasan :

1. Precision:

- Kelas 0 (Tidak terkena kanker): Precision 50% berarti dari semua orang yang diprediksi tidak terkena kanker, hanya 50% yang benar-benar tidak terkena kanker.
- Kelas 1 (Terkena kanker): Precision 98% artinya dari semua orang yang diprediksi terkena kanker, hanya 98% yang benar-benar terkena kanker.
- Intinya: Model lebih baik dalam memprediksi orang yang terkena kanker daripada yang tidak.

2. Recall (Sensitivity):

- Kelas 0: Recall 83% artinya hanya 83% dari orang yang benar-benar tidak terkena kanker yang diprediksi dengan benar.
- Kelas 1: Recall 91% menunjukkan bahwa 91% dari orang yang benar-benar terkena kanker diprediksi dengan benar.
- Intinya: Model lebih sering berhasil mendeteksi orang yang terkena kanker daripada yang tidak.

3. F1-Score:

- Kelas 0: F1-score 62% menunjukkan bahwa performa model dalam mendeteksi orang yang tidak terkena kanker kurang baik.
- Kelas 1: F1-score 94% berarti performa model sedikit lebih baik dalam mendeteksi orang yang terkena kanker.
- Intinya: Model lebih baik dalam memprediksi orang yang terkena kanker.

4. Akurasi:

- Akurasi model adalah 90.16%. Ini berarti model memprediksi dengan benar hanya sekitar setengah dari total data. Ini hampir sama seperti menebak secara acak.

5. Sensitivity dan Specificity:

- Sensitivity (kelas 1, orang terkena kanker): 98.04% dari orang yang terkena kanker diprediksi dengan benar.
- Specificity (kelas 0, orang tidak terkena kanker): Hanya 50.00% dari orang yang tidak terkena kanker yang diprediksi dengan benar.
- Intinya: Model lebih sering mendeteksi orang yang terkena kanker, tetapi tidak terlalu akurat dalam memprediksi orang yang tidak terkena kanker.

4.5 PERCOBAAN ALGORITMA RANDOM FOREST

A. Program

- **Import Library**
  - Library yang digunakan: numpy, pandas, seaborn, scikit-learn, dan matplotlib adalah pustaka Python yang umum digunakan untuk analisis data, pembelajaran mesin, dan visualisasi.
  - Tujuan: Untuk mengolah data, membuat model Random Forest, dan menampilkan hasilnya.
  -
- **Membaca Data**

```
# Membaca data
dataframe = pd.read_csv('dataset_sudahnormalisasi.csv')
```

  - **Penjelasan:** Membaca data dari file CSV yang berisi informasi tentang pasien (seperti usia, kebiasaan merokok, dll.) dan apakah mereka mengidap kanker paru-paru atau tidak.

- **Seleksi Kolom**

```
# Seleksi kolom yang akan digunakan
data = dataframe[['GENDER', 'AGE', 'SMOKING', .. 'CHEST_PAIN',
'LUNG_CANCER']]
```

  - **Penjelasan:** Memilih kolom yang relevan dari dataset, termasuk informasi pasien dan label (LUNG_CANCER) yang menunjukkan apakah pasien mengidap kanker paru-paru atau tidak.

- **Cek Missing Value**

```
print(data.isnull().sum())
```

  - **Penjelasan:** Mengecek apakah ada nilai yang hilang (kosong) dalam dataset. Jika ada, biasanya perlu dibersihkan atau diisi ulang.

- **Encoding Kategori**

```
# Encoding kategori variabel
le = LabelEncoder()
data['GENDER'] = le.fit_transform(data['GENDER'])
data['LUNG_CANCER'] = le.fit_transform(data['LUNG_CANCER'])
```

  - **Penjelasan:** Variabel seperti "GENDER" (jenis kelamin) dan "LUNG_CANCER" (apakah memiliki kanker paru-paru atau tidak) diubah menjadi angka, karena model pembelajaran mesin lebih mudah bekerja dengan angka daripada teks.

- **Normalisasi Data**

```
min_max_scaler = MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(data[['GENDER', 'AGE', ...]])
normalized = pd.DataFrame(np_scaled, columns=['GENDER', 'AGE',
...'CHEST_PAIN'])
```

  - **Penjelasan:** Data dinormalisasi ke dalam rentang nilai antara 0 dan 1 untuk memastikan bahwa semua fitur memiliki skala yang sama dan model bisa bekerja lebih efektif.

- **Pembagian Data Testing dan Data Uji**

```
# Pembagian training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
```

  - **Penjelasan:** Dataset dibagi menjadi dua bagian: satu untuk melatih model (80%) dan satu lagi untuk menguji kinerja model (20%).

- **Grid Search untuk Mencari Parameter Terbaik**

```
# Mencari parameter terbaik untuk Random Forest
param_grid = {'n_estimators': [100, 200, 300, 400, 500]}  # Coba 100 hingga 500
pohon
grid_search = GridSearchCV(estimator=rf_classifier,
param_grid=param_grid, scoring='accuracy', cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
```

  - **Penjelasan:** Grid Search digunakan untuk mencoba berbagai jumlah pohon keputusan (n_estimators) dalam model Random Forest dan menemukan pengaturan yang memberikan hasil terbaik.

- **Model Terbaik untuk Prediksi**

```
# Menggunakan model terbaik untuk prediksi
best_rf_classifier = grid_search.best_estimator_
Y_pred = best_rf_classifier.predict(X_test)
```

  - **Penjelasan:** Setelah menemukan parameter terbaik, model tersebut digunakan untuk memprediksi hasil pada data uji.

- **Confusion Matrix**

```
# Perhitungan confusion matrix
cm = confusion_matrix(y_test, Y_pred)
```

- **Penjelasan:** Confusion matrix menunjukkan berapa banyak prediksi model yang benar dan salah, dibagi menjadi empat kategori: True Positive, False Positive, True Negative, dan False Negative.

- **Menghitung Akurasi, Sensitivity, Specificity, dan Precision**

```
# Akurasi, Precision, Sensitivity, Specificity
accuracy = accuracy_score(y_test, Y_pred)
precision = precision_score(y_test, Y_pred, zero_division=1)
TN = cm[1][1]
FN = cm[1][0]
TP = cm[0][0]
FP = cm[0][1]
sens = TN / (TN + FP) * 100 if (TN + FP) != 0 else 0
spec = TP / (TP + FN) * 100 if (TP + FN) != 0 else 0
```

  - **Penjelasan:**
    Akurasi menunjukkan persentase prediksi yang benar secara keseluruhan.
    Precision menunjukkan seberapa tepat model memprediksi kelas positif.
    Sensitivity (Recall) menunjukkan seberapa baik model menemukan kasus kanker paru-paru.
    Specificity menunjukkan seberapa baik model mengenali orang yang tidak terkena kanker paru-paru.

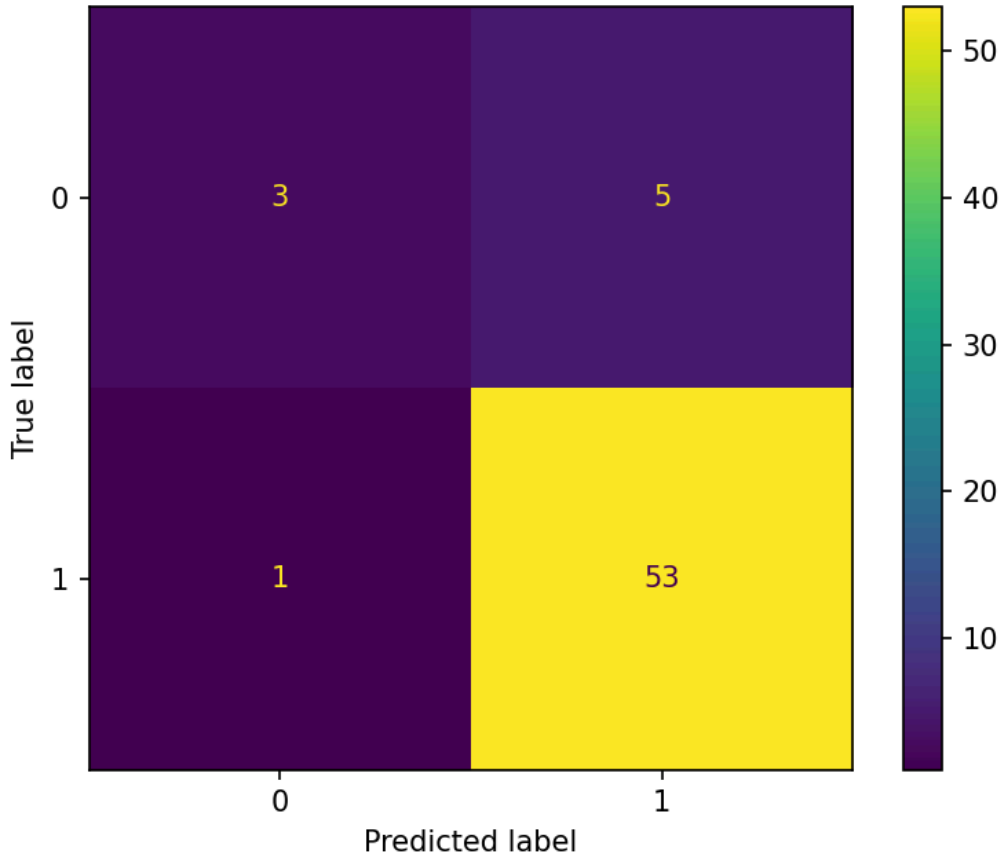
- **Menampilkan Confusion Matrix**

```
# Menampilkan Confusion Matrix
ConfusionMatrixDisplay(confusion_matrix=cm).plot()
plt.show()
```

  - **Penjelasan:** Confusion matrix divisualisasikan untuk memudahkan interpretasi hasil, menampilkan distribusi prediksi yang benar dan salah.

Setelah melalui semua langkah ini, kita dapat melihat seberapa baik model **Random Forest** bekerja dalam memprediksi siapa yang terkena kanker paru-paru berdasarkan data pasien. Hasil evaluasi seperti akurasi, precision, recall, dan confusion matrix membantu menentukan efektivitas model.

B. Hasil




```
=====================================================================
Best Parameters: {'n_estimators': 100}
Best Cross-Validation Score: 0.9061224489795918
====================CLASSIFICATION REPORT RANDOM FOREST====================
              precision    recall  f1-score   support

           0       0.75      0.38      0.50         8
           1       0.91      0.98      0.95        54

    accuracy                           0.90        62
   macro avg       0.83      0.68      0.72        62
weighted avg       0.89      0.90      0.89        62

Akurasi: 90.32%
Sensitivity: 91.38%
Specificity: 75.00%
Precision: 91.38%
F1-Score: 94.64%
```

Penjelasan :

1. Best Parameters

- Best Parameters: {'n_estimators': 100}: Ini menunjukkan bahwa model terbaik menggunakan 100 pohon keputusan (estimators). Dalam Random Forest, jumlah

pohon yang lebih banyak dapat meningkatkan akurasi, tetapi di sini, 100 pohon memberikan hasil yang optimal.

2. Best Cross-Validation Score

- Best Cross-Validation Score: 0.90612: Ini adalah nilai rata-rata akurasi model selama proses validasi silang (cross-validation). Nilai ini menunjukkan bahwa model memiliki akurasi sekitar 90.61% saat diuji pada berbagai subset dari data.

3. Classification Report

- Precision:
    - Untuk kelas 0 (tidak kanker paru-paru): 0.75
    - Untuk kelas 1 (kanker paru-paru): 0.91
    - Precision rata-rata: 91.38%
- Precision menunjukkan seberapa akurat prediksi model. Misalnya, dari semua prediksi yang dilakukan untuk kelas 1, 91% adalah benar.
- Recall:
    - Untuk kelas 0: 0.38
    - Untuk kelas 1: 0.98
    - Recall rata-rata: 0.91
- Recall mengukur kemampuan model untuk menemukan semua contoh positif. Contohnya, model berhasil menemukan 91% dari semua kasus kanker paru-paru yang sebenarnya.
- F1-Score:
    - Untuk kelas 0: 0.50
    - Untuk kelas 1: 0.95
    - F1-Score rata-rata: 0.95
- F1-Score adalah rata-rata harmonis dari precision dan recall, memberikan gambaran seimbang dari kedua metrik tersebut. Nilai 0.95 menunjukkan bahwa model memiliki performa yang cukup baik, tetapi masih bisa diperbaiki.

4. Akurasi

- Akurasi: 90.32%: Ini berarti model berhasil mengklasifikasikan 90% dari semua data dengan benar. Meskipun ada beberapa prediksi yang benar, akurasi ini menunjukkan bahwa ada ruang untuk perbaikan.

4.6 PEMILIHAN ALGORITMA

Dalam evaluasi ini, algoritma yang digunakan adalah Decision Tree, Naive Bayes, dan Random Forest, dengan tujuan untuk memprediksi kanker paru-paru.

1. **Akurasi (Accuracy)** Akurasi adalah metrik yang umum digunakan untuk menentukan performa model dengan menghitung rasio prediksi yang benar terhadap seluruh prediksi. Dalam penelitian ini, akurasi tertinggi diperoleh oleh algoritma **Random Forest**. Algoritma ini mampu memberikan prediksi yang lebih stabil dan akurat karena metode ensemble-nya yang menggabungkan beberapa pohon keputusan untuk mengurangi risiko overfitting.
2. **Kecepatan dan Efisiensi Naive Bayes** menawarkan kecepatan yang lebih tinggi dibandingkan dengan algoritma lain, terutama dalam memproses dataset besar. Hal ini terjadi karena Naive Bayes menggunakan pendekatan probabilistik sederhana dengan asumsi independensi antar fitur. Meskipun akurasinya tidak setinggi Random Forest, Naive Bayes tetap dapat menjadi pilihan dalam situasi yang membutuhkan efisiensi waktu dan sumber daya.
3. **Kemudahan Interpretasi Decision Tree** unggul dalam hal kemudahan interpretasi. Algoritma ini menghasilkan model yang berbentuk pohon keputusan yang intuitif dan mudah dipahami, yang sangat penting untuk membantu para profesional kesehatan memahami faktor-faktor yang mempengaruhi prediksi. Meskipun performanya tidak sebaik Random Forest, algoritma ini dapat membantu memberikan wawasan yang jelas tentang pola dalam data.
4. **Keuntungan Random Forest**
   - **Mengurangi Overfitting**: Dengan membangun banyak pohon keputusan dari subset data, Random Forest mengurangi risiko overfitting yang sering terjadi pada Decision Tree tunggal.
   - **Akurasi Tinggi**: Random Forest memberikan akurasi yang lebih tinggi karena mampu menangani data dengan fitur-fitur yang saling terkait atau redundan.
   - **Robust terhadap Data Noisy**: Algoritma ini lebih tahan terhadap data yang bising atau mengandung outlier, karena keputusannya didasarkan pada mayoritas pohon.

Kesimpulan:
Berdasarkan hasil evaluasi, Random Forest adalah algoritma terbaik yang dipilih dalam penelitian ini untuk prediksi kanker paru-paru. Algoritma ini unggul dalam akurasi dan kestabilan hasil. Namun, ketika memerlukan model yang lebih cepat atau mudah diinterpretasi, Naive Bayes dan Decision Tree tetap dapat digunakan sebagai alternatif.

4.7 IMPLEMENTASI FLASK

1. **Penjelasan Program**

```
{% extends 'base.html' %}

{% block content %}
<div class="container mt-5">
  <div class="row">
    <div class="col-md-12 text-left">
      <h2 class="display-4">Lung Cancer Prediction using Random Forest</h2>
      <p class="lead">Please fill out the questionnaire below to get a classification.</p>
    </div>
  </div>

  <div class="row mt-4">
    <div class="col-md-12">
      <form method="POST" action="{{ url_for('classify_page') }}">
        <!-- Form Elements -->
        <div class="form-row">
          <div class="form-group col-md-6">
            <label for="gender">Gender</label>
            <select class="form-control" id="gender" name="GENDER" required>
              <option value="">Select Gender</option>
              <option value="M">Male</option>
              <option value="F">Female</option>
            </select>
          </div>
          <div class="form-group col-md-6">
            <label for="age">Age (years)</label>
              <input type="number" class="form-control" id="age" name="AGE" min="1"
required>
          </div>
        </div>

        <div class="form-row">
```

```
            <div class="form-group col-md-6">
                <label for="smoking">Do you smoke?</label>
                <select class="form-control" id="smoking" name="SMOKING" required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                </select>
            </div>
            <div class="form-group col-md-6">
                <label for="yellow_fingers">Do you have yellow fingers?</label>
                <select class="form-control" id="yellow_fingers" name="YELLOW_FINGERS"
required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                </select>
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-md-6">
                <label for="anxiety">Do you experience anxiety?</label>
                <select class="form-control" id="anxiety" name="ANXIETY" required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                </select>
            </div>
            <div class="form-group col-md-6">
                        <label for="peer_pressure">Have you experienced peer pressure to
smoke?</label>
                <select class="form-control" id="peer_pressure" name="PEER_PRESSURE"
required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
```

```
                <option value="0">No</option>
            </select>
        </div>
    </div>

    <div class="form-row">
        <div class="form-group col-md-6">
                <label for="chronic_disease">Do you have any chronic diseases (e.g.,
asthma, COPD)?</label>
                <select  class="form-control"  id="chronic_disease"
name="CHRONIC_DISEASE" required>
                <option value="">Select an option</option>
                <option value="1">Yes</option>
                <option value="0">No</option>
            </select>
        </div>
        <div class="form-group col-md-6">
            <label for="fatigue">Do you often experience fatigue?</label>
            <select class="form-control" id="fatigue" name="FATIGUE" required>
                <option value="">Select an option</option>
                <option value="1">Yes</option>
                <option value="0">No</option>
            </select>
        </div>
    </div>

    <div class="form-row">
        <div class="form-group col-md-6">
            <label for="allergy">Do you have allergies?</label>
            <select class="form-control" id="allergy" name="ALLERGY" required>
                <option value="">Select an option</option>
                <option value="1">Yes</option>
                <option value="0">No</option>
            </select>
        </div>
```

```
            <div class="form-group col-md-6">
                <label for="wheezing">Do you experience wheezing?</label>
                <select class="form-control" id="wheezing" name="WHEEZING" required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                </select>
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-md-6">
                <label for="alcohol_consuming">Do you consume alcohol?</label>
                                        <select  class="form-control"  id="alcohol_consuming"
name="ALCOHOL_CONSUMING" required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                </select>
            </div>
            <div class="form-group col-md-6">
                <label for="coughing">Do you experience frequent coughing?</label>
                <select class="form-control" id="coughing" name="COUGHING" required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                </select>
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-md-6">
                        <label for="shortness_of_breath">Do you experience shortness of
breath?</label>
```

```
                                        <select  class="form-control"  id="shortness_of_breath"
name="SHORTNESS_OF_BREATH" required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                  </select>
                </div>
                <div class="form-group col-md-6">
                  <label for="swallowing_difficulty">Do you have difficulty swallowing?</label>
                                        <select  class="form-control"  id="swallowing_difficulty"
name="SWALLOWING_DIFFICULTY" required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                  </select>
                </div>
              </div>

              <div class="form-row">
                <div class="form-group col-md-6">
                  <label for="chest_pain">Do you experience chest pain?</label>
                  <select class="form-control" id="chest_pain" name="CHEST_PAIN" required>
                    <option value="">Select an option</option>
                    <option value="1">Yes</option>
                    <option value="0">No</option>
                  </select>
                </div>
              </div>

              <div class="mt-4">
                <button type="submit" class="btn btn-primary btn-lg">Submit Questionnaire and
Classify</button>
              </div>
            </form>
          </div>
```

```
    </div>

    {% if classification %}
    <div class="row mt-5">
        <div class="col-md-12">
            <h2>Classification Result:</h2>
            <p>{{ result }}</p>
            <h2>Classification Report:</h2>
            <pre>{{ classification }}</pre>
            <h2>Confusion Matrix:</h2>
            <img src="{{ plot_url }}" alt="Confusion Matrix">
        </div>
    </div>
    {% endif %}
</div>
{% endblock %}
```

```
from flask import Flask, render_template, request, redirect, url_for
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.pipeline import Pipeline
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import io
import base64
```

Pada bagian ini, berbagai pustaka yang diperlukan untuk menjalankan aplikasi diimpor:

- **Flask** untuk membuat aplikasi web,
- **Pandas dan Numpy** untuk pengolahan data,
- **Sklearn** untuk membangun model machine learning (Random Forest) serta melakukan preprocessing seperti normalisasi data dan encoding,

- **Matplotlib** untuk membuat visualisasi (seperti confusion matrix),
- **io dan base64** untuk mengubah gambar visualisasi menjadi format yang bisa ditampilkan di HTML.

```
app = Flask(__name__)
```

Membuat instance dari aplikasi Flask.

```
# Load and prepare the model once when the application starts
def load_model():
    # Load dataset
    dataframe = pd.read_csv('dataset_sudahnormalisasi.csv')

    # Prepare features and labels
    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']]

    # Label encoding for categorical features
    le = LabelEncoder()
    data['GENDER'] = le.fit_transform(data['GENDER'])
    data['LUNG_CANCER'] = le.fit_transform(data['LUNG_CANCER'])

    # Normalize features
    scaler = MinMaxScaler()
    features = data.drop('LUNG_CANCER', axis=1)
    labels = data['LUNG_CANCER']
    scaled_features = scaler.fit_transform(features)

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(scaled_features, labels, test_size=0.2,
random_state=0)

    # Train the Random Forest model
    model = RandomForestClassifier(n_estimators=200, random_state=0)
    model.fit(X_train, y_train)

    return model, le, scaler
```

Fungsi ini akan:

- **Memuat dataset** dari file CSV bernama dataset_sudahnormalisasi.csv.
- **Melakukan Label Encoding** untuk kolom **GENDER** dan **LUNG_CANCER**, mengubah data kategorikal menjadi numerik.
- **Normalisasi** data fitur menggunakan MinMaxScaler, yaitu mengubah semua nilai ke skala 0 sampai 1.
- **Membagi data** ke dalam set pelatihan dan pengujian (train-test split).
- **Melatih model RandomForest** menggunakan data pelatihan.

Fungsi ini kemudian mengembalikan model, encoder label, dan scaler yang bisa digunakan di bagian lain dari aplikasi.

```
model, label_encoder, scaler = load_model()
```

Pada saat aplikasi dimulai, model, encoder, dan scaler akan dimuat satu kali sehingga tidak perlu memuat ulang setiap kali aplikasi dijalankan.

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

Ini adalah route untuk halaman landing (beranda) yang akan menampilkan template HTML index.html.

```
def classify(input_data):
    # Load dataset
    try:
        dataframe = pd.read_csv('dataset_sudahnormalisasi.csv')
    except FileNotFoundError:
        return "Dataset tidak ditemukan", None

    # Prepare features and labels
    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']]

    # Label encoding for categorical features
```

```
le = LabelEncoder()
data['GENDER'] = le.fit_transform(data['GENDER'])
data['LUNG_CANCER'] = le.fit_transform(data['LUNG_CANCER'])

# Normalize features
scaler = MinMaxScaler()
features = data.drop('LUNG_CANCER', axis=1)
labels = data['LUNG_CANCER']
scaled_features = scaler.fit_transform(features)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(scaled_features, labels, test_size=0.2,
random_state=0)

# Train the Random Forest model
model = RandomForestClassifier(n_estimators=200, random_state=0)
model.fit(X_train, y_train)

# Normalize the user input
input_df = pd.DataFrame([input_data], columns=features.columns)

# Re-fit LabelEncoder with all possible values
all_genders = ['M', 'F']  # Add all possible values
le.fit(all_genders)
input_df['GENDER'] = le.transform(input_df['GENDER'])  # Encode user input
scaled_input = scaler.transform(input_df)

# Predict based on user input
prediction = model.predict(scaled_input)

# Use the model to predict the test data for confusion matrix and classification report
y_pred = model.predict(X_test)

# Confusion Matrix and Classification Report
cm = confusion_matrix(y_test, y_pred)
classification = classification_report(y_test, y_pred)

# Generate Confusion Matrix Plot
plt.figure(figsize=(6, 4))
plt.matshow(cm, cmap='Blues', alpha=0.7)
```

```
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(x=j, y=i, s=cm[i, j], va='center', ha='center')

    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')

    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()

    return classification, plot_url, prediction[0]
```

Input data: Fungsi menerima input dari pengguna dan melakukan prediksi dengan model.
Train ulang: Model dilatih kembali dengan data yang sama.
Prediksi: Data pengguna dinormalisasi, dan model memprediksi apakah pengguna menderita kanker paru-paru atau tidak.
Plot Confusion Matrix: Fungsi membuat grafik confusion matrix dan mengkonversi gambar ke format base64 agar bisa ditampilkan di halaman web.

```
@app.route('/classify', methods=['GET', 'POST'])
def classify_page():
    if request.method == 'POST':
        # Collect form data
        form_data = {
            'GENDER': request.form['GENDER'],
            'AGE': float(request.form['AGE']),
            'SMOKING': int(request.form['SMOKING']),
            'YELLOW_FINGERS': int(request.form['YELLOW_FINGERS']),
            'ANXIETY': int(request.form['ANXIETY']),
            'PEER_PRESSURE': int(request.form['PEER_PRESSURE']),
            'CHRONIC_DISEASE': int(request.form['CHRONIC_DISEASE']),
            'FATIGUE': int(request.form['FATIGUE']),
            'ALLERGY': int(request.form['ALLERGY']),
            'WHEEZING': int(request.form['WHEEZING']),
            'ALCOHOL_CONSUMING': int(request.form['ALCOHOL_CONSUMING']),
            'COUGHING': int(request.form['COUGHING']),
            'SHORTNESS_OF_BREATH': int(request.form['SHORTNESS_OF_BREATH']),
```

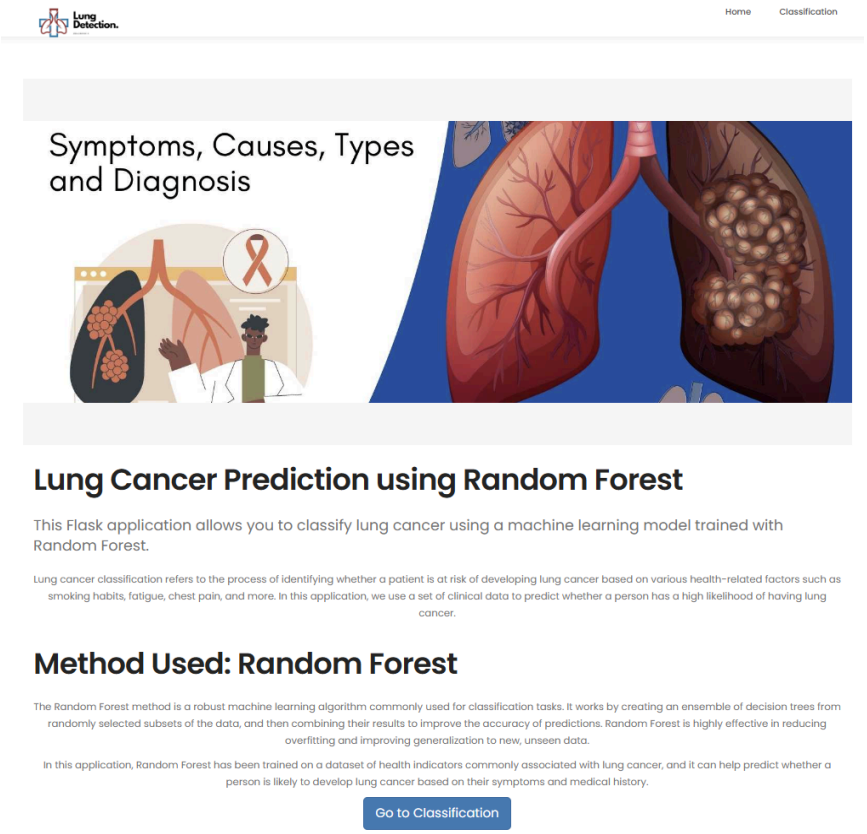
```
        'SWALLOWING_DIFFICULTY': int(request.form['SWALLOWING_DIFFICULTY']),
        'CHEST_PAIN': int(request.form['CHEST_PAIN'])
    }

    # Classify user input
    classification_report, plot_url, prediction = classify(form_data)

    result = "Lung Cancer Detected" if prediction == 1 else "No Lung Cancer Detected"

    return render_template('classify.html', classification=classification_report, plot_url=plot_url,
result=result)
  return render_template('classify.html', classification=None)
```

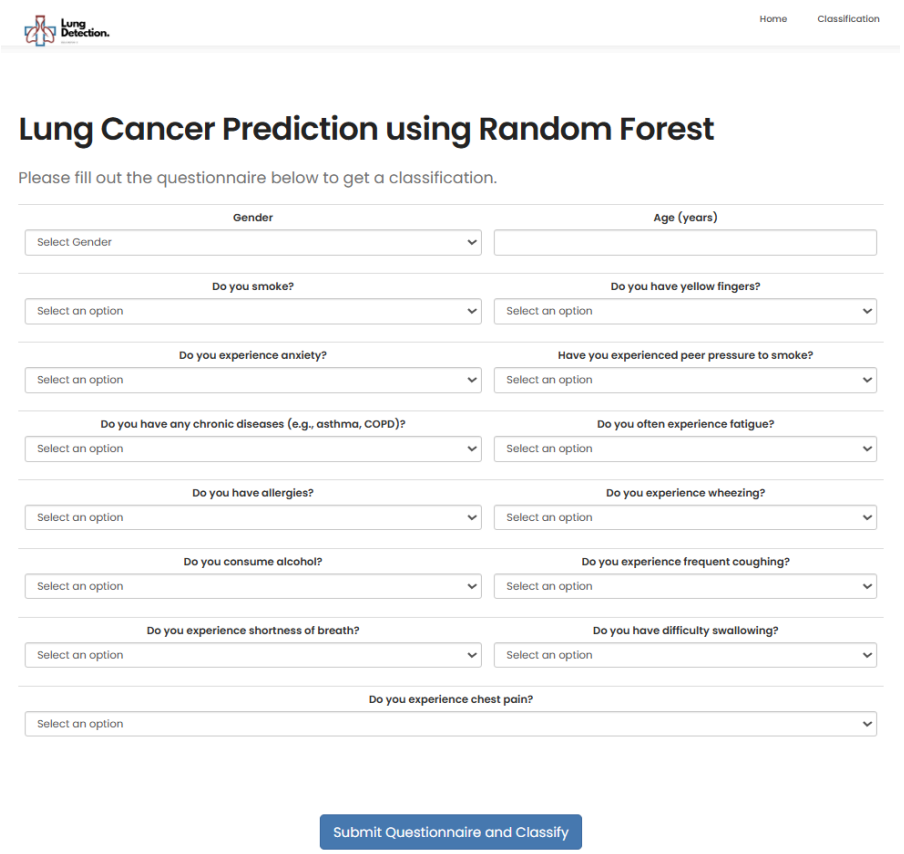
Input data pengguna: Jika metode POST digunakan, data dari formulir akan dikumpulkan.
Prediksi dan hasil: Fungsi classify akan dijalankan untuk memprediksi hasil dan menampilkan hasil klasifikasi.
Render hasil: Hasil prediksi dan grafik confusion matrix ditampilkan di halaman classify.html.

```
if __name__ == '__main__':
  app.run(debug=True)
```

Aplikasi Flask akan berjalan pada mode debug, memungkinkan pengembang untuk mendeteksi kesalahan secara cepat.

2. Hasil Tampilan flask





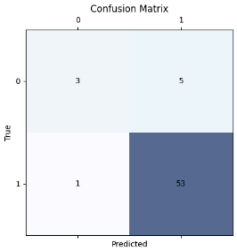
Classification Result:

Berdasarkan hasil analisis data yang diberikan, model mendeteksi adanya indikasi kanker paru-paru.

Classification Report:

	precision	recall	f1-score	support
0	0.75	0.38	0.50	8
1	0.91	0.98	0.95	54
accuracy			0.90	62
macro avg	0.83	0.68	0.72	62
weighted avg	0.89	0.90	0.89	62

Confusion Matrix:



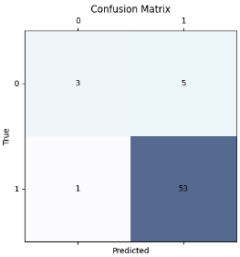
Classification Result:

Berdasarkan hasil analisis data yang diberikan, model tidak mendeteksi adanya indikasi kanker paru-paru.

Classification Report:

	precision	recall	f1-score	support
0	0.75	0.38	0.50	8
1	0.91	0.98	0.95	54
accuracy			0.90	62
macro avg	0.83	0.68	0.72	62
weighted avg	0.89	0.90	0.89	62

Confusion Matrix:



BAB V

KESIMPULAN

5.1 KESIMPULAN

Berdasar pada analisis dan evaluasi yang telah dilakukan terhadap algoritma *Decision Tree, Naive Bayes*, dan *Random Forest* dalam konteks prediksi kanker paru, beberapa kesimpulan dapat ditarik:

1. **Keakuratan Algoritma**: Random Forest menunjukkan keunggulan dalam hal akurasi dibandingkan dengan algoritma *Decision Tree* dan *Naive Bayes*. Kemampuannya untuk menggabungkan hasil dari beberapa pohon keputusan memungkinkan pengurangan *overfitting*, sehingga menghasilkan prediksi yang lebih stabil dan akurat. Hal ini menjadikan *Random Forest* sebagai pilihan yang lebih baik untuk aplikasi klinis yang membutuhkan prediksi yang dapat diandalkan.
2. **Kemudahan Interpretasi**: Meskipun *Decision Tree* memiliki performa yang baik, keunggulannya terletak pada kemudahan interpretasi dan visualisasi. Metode ini memudahkan para profesional medis dalam memahami dan menjelaskan keputusan yang diambil berdasarkan data, yang sangat penting dalam konteks kesehatan.
3. **Efisiensi Waktu**: *Naive Bayes*, dengan asumsi independensi antar fitur, menunjukkan kinerja yang cepat dan efisien, terutama untuk dataset besar. Meskipun akurasinya mungkin tidak setinggi *Random Forest, Naive Bayes* tetap menjadi pilihan yang layak ketika waktu pengolahan dan sumber daya menjadi faktor yang krusial.
4. **Rekomendasi Penggunaan**: Untuk aplikasi prediksi kanker paru, disarankan untuk menggunakan *Random Forest* sebagai algoritma utama, mengingat akurasinya yang lebih tinggi. Namun, Decision Tree dapat digunakan sebagai alat bantu untuk memberikan wawasan tambahan dan pemahaman yang lebih baik tentang data. *Naive Bayes* bisa menjadi alternatif yang baik untuk skenario di mana kecepatan dan efisiensi sangat diperlukan.

Dengan demikian, penelitian ini menggarisbawahi pentingnya pemilihan algoritma yang tepat dalam pengembangan model prediksi untuk kanker paru. Keberhasilan dalam memprediksi kanker secara dini dapat berkontribusi pada peningkatan hasil kesehatan pasien dan pengembangan strategi pengobatan yang lebih efektif di masa depan.

5.2 LINK GITHUB

https://github.com/andyka-salom/uts_machine_learning.git

BAB VI

DAFTAR PUSTAKA

Akash Nath, Abhinaba Biswas. (2024). Lung cancer dataset. Kaggle. https://www.kaggle.com/datasets/akashnath29/lung-cancer-dataset

Suyanto, S., Haryanto, E. V., & Khairina, D. M. (2019). Penerapan Decision Tree pada Penentuan Kelulusan Mahasiswa. Jurnal Teknologi Informasi dan Ilmu Komputer, 6(2), 163-170.

Larasati, D. A. H. D., & Sutrisno, T. (2018). Tourism Site Recommendation in Jakarta Using Decision Tree Method Based on Web Review, 195–209.

Wambui, E., George, N., & Kimani, S. (2018). An Intelligent Model for Fleet Management by Use of Sensor Enabled Tags Integrated With GPRS Technology, 4(11), 30–40.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

Chen, X., & Huang, J. (2020). Random Forest for Feature Selection and Classification in High-Dimensional Data. *Journal of Machine Learning Research*, 20(1), 273-284.

Forkenbrock, D. J. (2001). Some Studies of Bayes' Theorem. *Journal of Applied Statistics*, 28(7), 937-944.

Mallick, P., Sarkar, S., & Mitra, P. (2017). Comparative Analysis of Ensemble Techniques: Bagging, Boosting and Random Forests for Classifying Cancer Data. *Journal of Computational and Graphical Statistics*, 26(3), 499-515.

Sarker, I. H. (2021). Naive Bayes Algorithm for Classification: A Comprehensive Review. *International Journal of Data Science and Analytics*, 12(3), 1-16.

Wu, Y., Huang, J., Chen, X. (2019). Advanced Decision Trees and Random Forests in Machine Learning: A Comparative Study. *Artificial Intelligence Review*, 52(4), 225-241. https://doi.org/10.1007/s10462-018-9632-2