LAPORAN AKHIR

MATA KULIAH STATISTIKA MULTIVARIAT

KELAS B

"PENGARUH REDUKSI DIMENSI PCA TERHADAP KINERJA ALGORITMA KLASIFIKASI PADA DATASET HEPATITIS"

DISUSUN OLEH KELOMPOK "XII":

1. MUHAMAD HARIS HARTANTO (21083010045)
2. AMANDA AULIA (21083010048)
3. ALYSSA AMORITA AZZAH (21083010057)

DOSEN PENGAMPU:

PRISMAHARDI AJI RIYANTOKO, S.Si., M.Si.

TRIMONO, S.Si., M.Si.

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2023

DAFTAR ISI

DAFTAR ISI 1
DAFTAR GAMBAR 3
DAFTAR TABEL 4
BAB I PENDAHULUAN 5
1.1. Latar Belakang 5
1.2. Permasalahan 5
1.3. Tujuan 5
1.4. Manfaat 6
BAB II TINJAUAN PUSTAKA 7
2.1. Teori Penunjang 7
2.1.1. Reduksi Dimensi 7
2.1.2. *Principal Component Analysis* (PCA) 7
2.1.3. *Machine Learning* 8
2.1.4. Klasifikasi 8
2.1.5. Logistic Regression 9
2.1.6. Decision Tree 9
2.1.7. K-Nearest Neighbour (KNN) 9
2.2. Penelitian Terkait 10
2.2.1. Klasifikasi pada Dataset Penyakit Hati Menggunakan Algoritma Support Vector Machine, K-NN, dan Naive Bayes 10
2.2.2. Analisis Pengaruh PCA Pada Klasifikasi Kualitas Air Menggunakan Algoritma K-Nearest Neighbor dan Logistic Regression 11
2.2.3. Analisis Performa Algoritma K-Nearest Neighbor dan Reduksi Dimensi Menggunakan Principal Component Analysis 11
BAB III METODOLOGI PENELITIAN 13
3.1. *Dataset* 13
3.2. Prosedur Analisis 15
BAB IV HASIL DAN PEMBAHASAN 16
4.1. Prapemrosesan Data 16
4.1.1. Pembersihan Data 16
4.1.2. Transformasi Data 17
4.2. Analisis Data Eksploratif 17
4.2.1. Statistik Deskriptif 17
4.2.2. Jumlah Data 19
4.2.3. Korelasi 19
4.3. Principal Component Analysis (PCA) 20
4.3.1. Uji Asumsi 20
4.3.2. Nilai Eigen dan Vektor Eigen 21
4.3.3. Transformasi Data 22
4.4. Pengujian Model Klasifikasi 23
4.5. Evaluasi Model Klasifikasi 24

BAB V KESIMPULAN 26
DAFTAR PUSTAKA 27
LAMPIRAN 28
Lampiran 1. Kode Program *Import Library* 28
Lampiran 2. Kode Program Pembersihan Data 28
Lampiran 3. Imputasi *Missing Value* 29
Lampiran 4. Kode Program Kode Program Transformasi Data (Normalisasi Data) 30
Lampiran 5. Kode Program Analisis Data Eksploratif 31
Lampiran 6. Kode Program Principal Component Analysis (PCA) 31
Lampiran 7. Pembentukan Model Klasifikasi 33
Lampiran 8. Kode Program Pengujian Model Klasifikasi 34

DAFTAR GAMBAR

Gambar 1. Flowchart Analisis 15
Gambar 2. Grafik Jumlah Nilai Atribut class (class).....19
Gambar 3. Grafik Jumlah Nilai Atribut sex (jenis kelamin).....19
Gambar 4. Visualisasi matriks korelasi 20
Gambar 5. Hasil Uji KMO dan Bartlett.....20
Gambar 6. *Scree Plot* 22

DAFTAR TABEL

Tabel 1. Deskripsi Dataset Hepatitis....13
Tabel 2. Potongan Dataset setelah Pembersihan Data....16
Tabel 3. Potongan Dataset Setelah Transformasi Data....17
Tabel 4. Statistik Deskriptif Atribut Tipe Numerik....18
Tabel 5. Statistik Deskriptif Atribut Tipe Kategorik....18
Tabel 6. Hasil Nilai Eigen dan Vektor Eigen....21
Tabel 7. Potongan Dataset Setelah Transformasi Data PCA....22
Tabel 8. Hasil Pengujian Menggunakan Dataset Sebelum Reduksi Dimensi PCA....23
Tabel 9. Hasil Pengujian Menggunakan Dataset Sesudah Reduksi Dimensi PCA....23
Tabel 10. Perbandingan Performa Model Klasifikasi....24

BAB I
PENDAHULUAN

1.1. Latar Belakang

Hepatitis merupakan penyakit peradangan hati karena infeksi virus yang menyerang dan menyebabkan kerusakan pada sel-sel dan fungsi organ hati. Penyakit Hepatitis dapat merusak fungsi organ hati sebagai penetral racun dan sistem pencernaan makanan dalam tubuh yang mengurai sari-sari makanan untuk kemudian disebarkan ke seluruh organ tubuh yang sangat penting bagi manusia [1]. Prediksi hasil pasien hepatitis yang akurat dan pengambilan keputusan yang tepat sangat penting dalam memberikan perawatan yang efektif dan meningkatkan kesempatan hidup pasien. Dalam konteks ini, penggunaan algoritma klasifikasi berbasis pembelajaran mesin dapat menjadi alat yang berguna dalam memprediksi hasil dan dapat membantu pengambilan keputusan medis.

Salah satu tantangan dalam membuat model klasifikasi dengan menggunakan data di bidang medis, seperti *dataset* hepatitis adalah tingginya dimensi data akibat banyaknya fitur yang tersedia. Permasalahan tersebut menyebabkan pemodelan menjadi kompleks dan memakan waktu yang lama. Salah satu metode yang dapat digunakan untuk mengatasi permasalahan tersebut adalah dengan melakukan reduksi dimensi. Reduksi dimensi adalah proses mengurangi jumlah variabel atau fitur dalam dataset dengan tujuan mempertahankan sebanyak mungkin informasi yang relevan [2]. Tujuan utama dari reduksi dimensi adalah untuk mengatasi masalah dimensi tinggi dalam dataset, di mana terdapat banyak fitur yang dapat mempengaruhi kompleksitas pemodelan dan kinerja algoritma.

Metode reduksi dimensi yang paling umum digunakan adalah *Principal Component Analysis* (PCA). PCA merupakan suatu teknik yang digunakan untuk menyederhanakan suatu data yang dilakukan dengan cara transformasi data secara linier sehingga akan terbentuk suatu sistem koordinat baru dengan varians maksimum. Analisis komponen utama digunakan untuk mereduksi dimensi dari suatu data tanpa mengurangi karakteristik dari data tersebut secara signifikan [3]. Dengan melakukan reduksi dimensi diharapkan dapat mengurangi kompleksitas pemodelan dan meningkatkan efisiensi dalam pemodelan algoritma klasifikasi pada *dataset* hepatitis.

1.2. Permasalahan

Tantangan dalam pemodelan klasifikasi pada *dataset* hepatitis adalah dimensi data yang tinggi akibat banyaknya fitur yang ada. Hal ini dapat menyebabkan kompleksitas pemodelan dan mempengaruhi efisiensi algoritma klasifikasi.

1.3. Tujuan

- Menerapkan reduksi dimensi menggunakan metode *Principal Component Analysis (PCA)* pada *dataset* Hepatitis.
- Membandingkan kinerja algoritma klasifikasi antara *dataset* yang direduksi dengan metode Principal Component Analysis (PCA) dan *dataset* asli tanpa reduksi.

1.4. Manfaat

- Mengetahui hasil reduksi dimensi menggunakan metode Principal Component Analysis (PCA) pada dataset hepatitis.
- Mengetahui pengaruh reduksi dimensi menggunakan metode Principal Component Analysis (PCA) terhadap kinerja algoritma klasifikasi pada dataset hepatitis dengan dan tanpa reduksi.

BAB II
TINJAUAN PUSTAKA

2.1. Teori Penunjang

2.1.1. Reduksi Dimensi

Reduksi Dimensi merupakan suatu proses pengurangan jumlah fitur atau dimensi tanpa menghapus informasi penting yang terdapat pada suatu data. Reduksi Dimensi umumnya dilakukan sebagai metode *pre-processing* untuk membuat model klasifikasi yang baik. Reduksi dimensi dapat membuat model menjadi lebih mudah dipahami dan mempermudah dalam melakukan visualisasi data. Selain itu, jumlah waktu dan memori yang dibutuhkan oleh algoritma data mining berkurang dengan adanya reduksi dimensi [2].

2.1.2. *Principal Component Analysis* (PCA)

Principal Component Analysis atau analisis komponen utama merupakan suatu teknik yang digunakan untuk menyederhanakan suatu data yang dilakukan dengan cara transformasi data secara linier sehingga akan terbentuk suatu sistem koordinat baru dengan varians maksimum. Analisis komponen utama digunakan untuk mereduksi dimensi dari suatu data tanpa mengurangi karakteristik dari data tersebut secara signifikan. Tujuan utama analisis komponen utama adalah untuk mengurangi dimensi peubah yang saling berhubungan dan cukup banyak variabelnya sehingga lebih mudah untuk menginterpretasikan data-data tersebut [3]. Langkah dari algoritma PCA sebagai berikut [4]:

1. Menghitung *mean* ($\underline{\boldsymbol{X}}$) dari data pada tiap dimensi menggunakan persamaan:

$$\overline{\boldsymbol{X}} = \frac{\boldsymbol{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{X_i}$$

   Dengan:
   $\boldsymbol{n}$ = jumlah data sampel
   $\boldsymbol{X_i}$ = data sampel

2. Menghitung *covariance matrix* ($\boldsymbol{C_x}$) menggunakan persamaan:

$$\boldsymbol{C_x} = \frac{\boldsymbol{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{x_i} - \overline{\boldsymbol{x}})(\mathbf{y_i} - \overline{\boldsymbol{y}})$$

   Dengan:
   x, y = variabel pada data
   $\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}$ = rata-rata (*mean*) dari
   n = jumlah data

3. Menghitung *eigenvalue* menggunakan persamaan:

$$(\boldsymbol{A} - \boldsymbol{\lambda I}) = \mathbf{0}$$

   Dengan:
   A = sebuah matriks
   $\boldsymbol{\lambda}$ = nilai eigen dari A

4. Menghitung *eigenvector* menggunakan persamaan:

$$[\boldsymbol{A} - \boldsymbol{\lambda I}][\boldsymbol{X}] = \boldsymbol{0}$$

Menentukan variabel baru dengan cara mengalikan variabel asli dengan matriks *eigenvector*.

2.1.3. *Machine Learning*

Machine learning merupakan salah satu cabang dari ilmu komputer yang mempelajari bagaimana membuat program komputer yang dapat belajar dari data. Program komputer akan digunakan untuk menemukan pola atau hubungan yang tersembunyi dalam data, sehingga program tersebut dapat melakukan prediksi pada data yang baru [5].

Terdapat dua metode utama *machine learning*, yaitu *supervised learning* dan *unsupervised learning*. Pada *supervised learning*, program komputer akan belajar dari data yang sudah diberi label, sehingga dapat membuat prediksi pada data yang belum diberi label. Contoh penerapan dari metode tersebut adalah klasifikasi *email* spam atau prediksi harga rumah berdasarkan fitur-fitur tertentu. Sedangkan pada *unsupervised learning*, program komputer akan belajar dari data yang tidak memiliki label, sehingga dapat menemukan pola atau struktur yang tersembunyi dalam data tersebut. Contoh penerapan dari *unsupervised learning* adalah *clustering* data atau reduksi dimensi pada data [6].

Selain itu, terdapat metode lain yang dikenal sebagai *reinforcement learning*. Metode ini memanfaatkan proses belajar melalui *trial* and *error* dan memperoleh pembelajaran melalui penguatan dari kritik yang memberikan informasi tentang kualitas solusi yang dihasilkan, tetapi tidak memberikan informasi tentang cara untuk meningkatkannya. Agar mencapai solusi yang lebih baik, algoritma tersebut akan secara berulang-ulang mengeksplorasi ruang solusi [7].

2.1.4. Klasifikasi

Klasifikasi merupakan metode supervised learning yang melakukan proses untuk menemukan model yang dapat membedakan kelas data dengan tujuan agar dapat digunakan untuk memprediksi kelas dari data yang belum diketahui label kelasnya [8]. Model tersebut dapat berupa aturan jika-maka, pohon keputusan, atau rumus matematis. Klasifikasi dalam machine learning memiliki dua tahapan proses, yaitu [6]:

- Pelatihan model
  Proses sekelompok titik data dan label yang bersesuaian yang mencoba untuk mempelajari pola bagaimana titik-titik data tersebut dipetakan ke label yang sesuai.
- Prediksi
  Setelah dilatih, pengklasifikasi (*classifier*) bertindak sebagai fungsi yang mengambil titik data tambahan dan melakukan prediksi kelas atau label. Prediksi tersebut dapat berupa label tertentu atau nilai kontinu yang dapat dilihat sebagai skor kepercayaan untuk label tertentu.

Klasifikasi yang menghasilkan dua label kelas disebut klasifikasi biner, sedangkan klasifikasi yang menghasilkan tiga atau lebih label kelas disebut dengan klasifikasi banyak kelas (*multiple class classification)*.

2.1.5. Logistic Regression

Logistic Regression atau regresi logistik merupakan metode klasifikasi yang sering digunakan untuk melakukan klasifikasi biner dan tidak dapat digunakan untuk melakukan pengklasifikasian dengan banyak kelas, kecuali dilakukan modifikasi tertentu. Regresi logistik memberikan nilai probabilitas untuk setiap titik data berdasarkan seberapa jauh jarak dari *hyperplane*, dibandingkan menggunakan *hyperplane* tersebut sebagai batas pemotongan. Apabila datanya hampir dapat dipisahkan secara linier, maka semua titik yang tidak berdekatan dengan *hyperplane* akan memiliki prediksi yang pasti mendekati 0 atau 1. Namun, jika dua kelas tersebut memiliki tumpang tindih di atas *hyperplane*, maka prediksi akan menjadi lebih tumpul, dan hanya titik-titik yang jauh dari *hyperplane* yang akan mendapatkan skor yang pasti. Skor untuk sebuah titik data diperoleh melalui rumus berikut [6]:

$$\boldsymbol{p(x)} = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e^{w \times x \times b}}}$$

Dimana w adalah vektor yang memberikan bobot pada setiap *feature* dan b merupakan *offside* bernilai riil. Selanjutnya, dimasukkan ke dalam fungsi sigmoid dengan rumus sebagai berikut:

$$\boldsymbol{\sigma(z)} = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e^{Z}}}$$

Fungsi tersebut digunakan untuk memetakan setiap nilai riil menjadi nilai probabilitas dalam rentang 0 sampai 1.

2.1.6. Decision Tree

Decision Tree atau pohon keputusan secara konseptual merupakan salah satu metode klasifikasi yang paling sederhana. Sesuai dengan namanya, metode ini memiliki struktur seperti pohon dengan cabang yang memiliki prinsip kerja seperti diagram alur (*flowchart*). Konsep dari metode ini diawali dengan *root node*. Selanjutnya, setiap *node* pada pohon akan mengajukan pertanyaan tentang suatu *feature* data. Apabila *feature* tersebut merupakan tipe numerik, pertanyaannya akan berupa apakah nilai data berada di atas atau di bawah ambang tertentu yang kemudian dilanjutkan pada *child node* di bawahnya untuk jawaban iya atau tidak. Namun, jika *feature* berupa kategorik, maka akan dilanjutkan pada *child node* yang berbeda untuk setiap nilai yang dapat diambil. Keputusan akhir atau label kelas dari klasifikasi berada pada cabang terakhir, yang dikenal dengan *leaf node* [6].

2.1.7. K-Nearest Neighbour (KNN)

K-Nearest Neighbour merupakan salah satu metode dasar *instance-based learning*. Algoritma kNN digunakan untuk melakukan klasifikasi data baru berdasarkan jarak dari pembelajaran yang terdekat dengan objek atau disebut dengan tetangga

terdekat. Algoritma tersebut menggunakan pembelajaran *supervised learning* yang dimana hasil dari data yang telah dilakukan pengujian akan dilakukan klasifikasi berdasarkan keanggotaan terdekat yang terbanyak dari data pengujian. Jumlah tetangga terdekat disebut dengan K. Nilai dari K dapat ditentukan dengan ketentuan n+1 yang dimana n merupakan jumlah label, atau menggunakan metode ganjil genap, jika nilai dari n ganjil maka nilai dari k yaitu genap, begitu pula sebaliknya. Langkah yang dapat dilakukan untuk menentukan jarak terdekat dengan cara data dibagi menjadi data latih dan data uji, setelah diperoleh nilai dari data latih dan data uji tersebut akan dilakukan perhitungan jarak dari masing-masing data uji terhadap data latih. Perhitungan jarak dapat menggunakan *Euclidean Distance* dengan persamaan sebagai berikut [4]:

$$\boldsymbol{d}\ (\boldsymbol{x_i x_j})\ = \sqrt{\sum_{\boldsymbol{r=1}}^{\boldsymbol{n}} (\boldsymbol{a_r}(\boldsymbol{x_i}) - \boldsymbol{a_r}(\boldsymbol{x_j}))^{\boldsymbol{2}}}$$

Dengan:

$\boldsymbol{d}\ (\boldsymbol{x_i x_j})$ = jarak *Euclidean*

$\boldsymbol{x_i x_j}$ = data ke 1, record ke j

$\boldsymbol{a_r}$ = data ke-r

n = dimensi objek

2.2. Penelitian Terkait

2.2.1. Klasifikasi pada Dataset Penyakit Hati Menggunakan Algoritma Support Vector Machine, K-NN, dan Naive Bayes

Pada penelitian ini membahas tentang klasifikasi penyakit hati menggunakan algoritma Support Vector Machine, K-Nearest Neighbor, dan Naïve Bayes. Pada penelitian ini dilakukan dengan proses pertama ialah melakukan pembersihan dataset hati yaitu dengan mengatasi missing value dan mendeteksi outlier. Setelah pembersihan data, akan dilakukan klasifikasi dengan tiga algoritma yang berbeda. Algoritma klasifikasi tersebut yaitu Naïve Bayes, KNN, dan SVM. Performa dari ketiga metode tersebut akan dibandingkan untuk mendapatkan metode yang terbaik untuk dataset hati dengan cara menentukan nilai performanya yang berupa nilai accuracy, precision, recall, dan F-measure. Berdasarkan penelitian yg dilakukan didapatkan kesimpulan sebagai berikut [9]:

- Dilakukan beberapa proses pengolahan data sebelum dilakukan penentuan algoritma yang paling baik dengan dengan beberapa proses seperti menghilangkan missing value dan menghilangkan outlier. Setelah data di-cleansing, dilanjutkan dengan menerapkan tiga algoritma yang berbeda.
- Algoritma Naïve Bayes menunjukkan persentase yang baik, yaitu dengan rata-rata 80,97%. K-NN memiliki persentase yang tidak terlalu baik, walaupun sudah diberi tiga skenario yang berbeda. Semakin tinggi nilai K, semakin rendah persentase rata-ratanya.
- Dari penelitian ini, dapat disimpulkan bahwa algoritma SVM mempunyai performa terbaik di antara ketiga algoritma dengan hasil persentase 82,36%.

2.2.2. Analisis Pengaruh PCA Pada Klasifikasi Kualitas Air Menggunakan Algoritma K-Nearest Neighbor dan Logistic Regression

Pada penelitian ini membahas tentang pengaruh metode PCA untuk mengklasifikasikan kualitas air minum menggunakan data water quality yang terdapat pada kaggle. Data yang didapatkan akan dilakukan pembersihan data, selanjutnya dilakukan reduksi dimensi menggunakan metode PCA untuk mendapatkan fitur-fitur baru hasil reduksi dimensi. Fitur hasil reduksi dimensi ini kemudian akan digunakan sebagai input untuk proses klasifikasi kualitas air layak minum menggunakan metode kNN dan Logistic Regression. Berdasarkan penelitian yang dilakukan didapatkan kesimpulan sebagai berikut [4]:

- Dilakukan beberapa proses untuk mengklasifikasikan kualitas air dengan pembersihan data, selanjutnya dilakukan reduksi dimensi menggunakan metode PCA untuk mendapatkan fitur-fitur baru hasil reduksi dimensi.
- Hasil penelitian secara keseluruhan yaitu tingkat performa dari metode kNN dan Logistic Regression menurun saat menerapkan metode reduksi dimensi PCA.
- Metode kNN tanpa PCA memiliki akurasi tertinggi 90.8% pada k=9, sedangkan metode kNN dengan PCA memiliki akurasi 88.5% pada n=5 dan k=5. Metode Logistic Regression tanpa PCA memiliki akurasi tertinggi 90.0%, sedangkan dengan PCA memiliki akurasi 87.9% pada n=5.

2.2.3. Analisis Performa Algoritma K-Nearest Neighbor dan Reduksi Dimensi Menggunakan Principal Component Analysis

Pada penelitian ini membahas tentang performa Algoritma K-Nearest Neighbor dengan reduksi dimensi menggunakan Principal Component Analysis (PCA) pada kasus klasifikasi penyakit diabetes. Banyaknya variabel dan data yang besar pada penyakit diabetes relative memerlukan waktu komputasi yang lama, sehingga dibutuhkan reduksi dimensi untuk mempercepat proses komputasi. Dilakukan dengan normalisasi data dengan menggunakan Z-Score. Selanjutnya melakukan reduksi dimensi dengan membentuk matriks kovarian, menghitung nilai eigen dan vektor eigen, dan membentuk komponen utama dari hasil vektor eigen yang diperoleh. Kemudian dari hasil reduksi dimensi akan dilakukan klasifikasi dengan mencari parameter K, menentukan jarak, dan mencari kelas mayoritas yang hasilnya akan dievaluasi dan dilihat bagaimana penerapan metode ini pada data penyakit diabetes. Berdasarkan penelitian yang dilakukan didapatkan kesimpulan sebagai berikut [10]:

- Hasil penelitian pada studi kasus penyakit diabetes menunjukkan bahwa reduksi dimensi menggunakan PCA menghasilkan 3 komponen utama dari 8 variabel pada data asli yaitu PC1, PC2, dan PC3.
- Hasil klasifikasi menggunakan Algoritma K-Nearest Neighbor menunjukkan bahwa pada pengambilan 3 parameter ketetanggaan (K) terdekatnya yaitu untuk K=3, K=5 dan K=7. Pada kasus K=3, diperoleh hasil akurasi sebesar 67,53%, pada pengambilan K=5 diperoleh akurasi 72,72 %, dan untuk K=7 diperoleh hasil akurasi sebesar 77,92%.

- Performa akurasi terbaik untuk klasifikasi penyakit diabetes dicapai pada K=7 dengan akurasi sebesar 77,92%.

BAB III
METODOLOGI PENELITIAN

3.1. *Dataset*

Data yang digunakan dalam penelitian ini merupakan *dataset* di bidang medis, yaitu data pasien penderita hepatitis. *Dataset* ini diperoleh dari situs UCI Machine Learning Repository. Dalam *dataset* tersebut berisi sejumlah atribut profil pasien, tanda dan gejala, hasil pemeriksaan fisik, hasil tes laboratorium, penanganan medis, dan satu atribut kelas (*class*) atau label keputusan yang menyatakan pasien penderita hepatitis hidup atau meninggal.

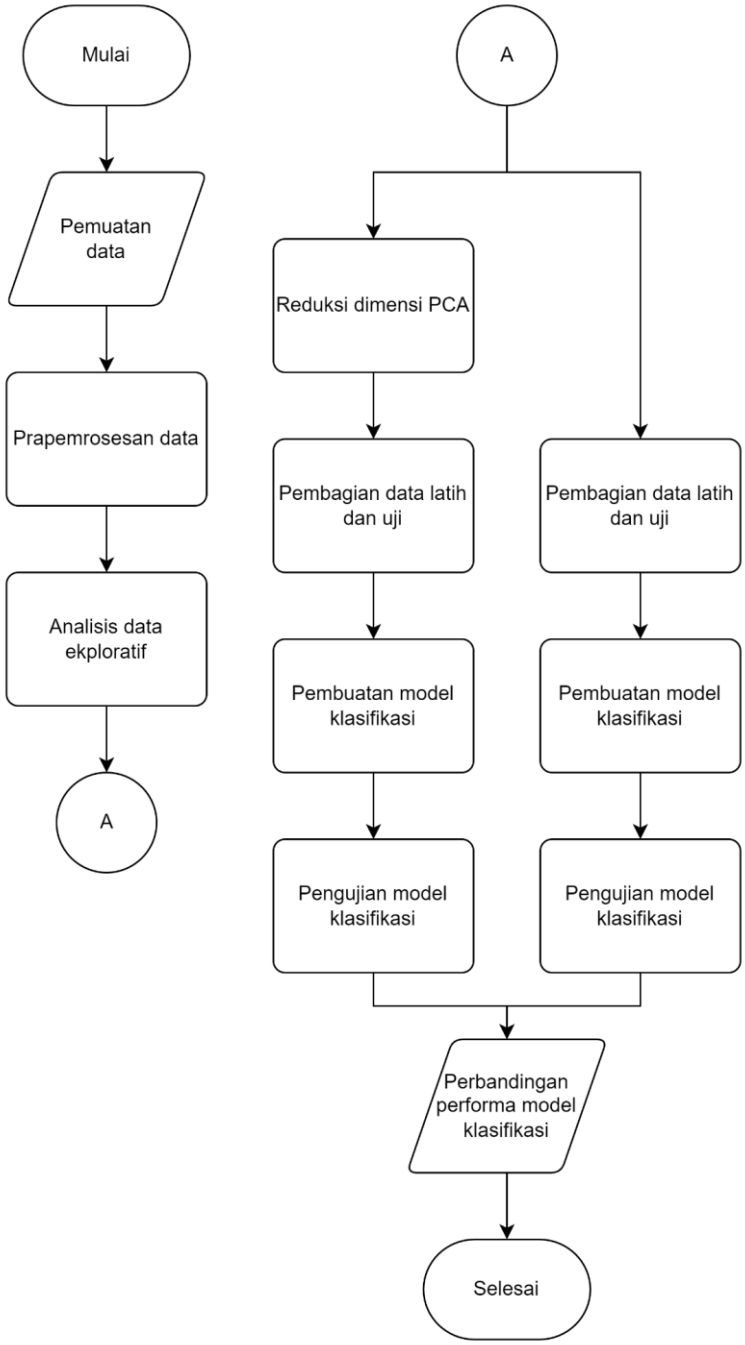
Terdapat 155 *record* data dengan 19 atribut dan 1 atribut kelas (*class*) atau label keputusan. Masing-masing dari atribut tersebut memiliki penjelasan sebagai berikut [11]:

Tabel 1. Deskripsi *Dataset* Hepatitis

Atribut	Tipe data	Deskripsi	Domain Nilai
Age	Numerik	Usia	Bilangan Kontinu
Sex	Binomial	Jenis kelamin	pria, wanita (1, 2)
Steroid	Binomial	Apakah mendapat perawatan steroid?	tidak, iya (1, 2)
Antivirals	Binomial	Apakah mendapat perawatan antivirals?	tidak, iya (1, 2)
Fatigue	Binomial	Apakah mengalami gejala kelelahan akut?	tidak, iya (1, 2)
Malaise	Binomial	Apakah mengalami gejala malaise (rasa tidak nyaman)?	tidak, iya (1, 2)
Anorexia	Binomial	Apakah mengalami gejala anorexia (penurunan nafsu makan yang berlebihan)?	tidak, iya (1, 2)
Liver Big	Binomial	Apakah kondisi hati/liver membesar?	tidak, iya (1, 2)
Liver Firm	Binomial	Apakah kondisi hati/liver mengeras?	tidak, iya (1, 2)
Spleen Palpable	Binomial	Apakah kondisi limfa menjadi lebih besar dari normal?	tidak, iya (1, 2)

Spiders	Binomial	Apakah ada gejala pembuluh darah abnormal pada kulit (pembuluh darah mengumpul dan menonjol pada permukaan kulit)?	tidak, iya (1, 2)
Ascites	Binomial	Apakah terjadi penumpukan cairan pada rongga perut?	tidak, iya (1, 2)
Varices	Binomial	Apakah terjadi pembekakan vena esophagus (varises)?	tidak, iya (1, 2)
Bilirubin	Numerik	Nilai kadar bilirubin (pigmen berwarna kuning kecoklatan dalam empedu) dalam darah	Bilangan Kontinu
Alk Phospat	Numerik	Kadar alkalin phospat dalam liver	Bilangan Kontinu
SGOT	Numerik	Nilai SGOT (enzim-enzim pada hati yang akan meningkat jumlahnya di dalam tubuh jika hati mengalami kerusakan, baik kerusakan fungsi hati secara akut maupun kronis)	Bilangan Kontinu
Albumin	Numerik	Kadar albumin (protein yang paling banyak terdapat dalam aliran darah dan diproduksi oleh hati)	Bilangan Kontinu
Protime	Numerik	Uji masa protrombin (pembekuan hati)	Bilangan Kontinu
Histology	Binomial	Apakah dilakukan pemeriksaan dengan histology (biopsi hati)?	tidak, iya (1, 2)
CLASS	Binomial	Label yang menunjukkan pasien hidup atau mati.	mati, hidup (1, 2)

3.2. Prosedur Analisis



Gambar 1. *Flowchart* Analisis

Proses analisis diawali dengan melakukan prapemrosesan data yang meliputi pembersihan dan transformasi data. Selanjutnya, dilakukan analisis data eksploratif untuk mengetahui statistik deskriptif, penyebaran data, dan korelasi antar variabel. Setelah itu, dilakukan reduksi dimensi menggunakan metode Principal Component Analysis (PCA) untuk mendapatkan data baru hasil reduksi dimensi.

Setelah didapatkan data dengan dimensi yang telah tereduksi, dilakukan pembuatan dan pengevaluasian performa model klasifikasi yang meliputi model Logistic Regression, Decision Tree dan K-Nearest Neighbour (KNN) terhadap data sebelum dan sesudah reduksi dimensi.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Prapemrosesan Data

Merupakan serangkaian proses yang diterapkan terhadap kumpulan data (*dataset*) dengan tujuan mendapatkan data yang bersih untuk digunakan pada tahapan proses selanjutnya. Pada tahap ini, dilakukan pembersihan dan transformasi data. Pembersihan data yang dilakukan meliputi perbaikan format penulisan dan imputasi nilai data yang kosong. Sementara itu, proses transformasi yang dilakukan adalah normalisasi nilai data.

4.1.1. Pembersihan Data

Pada *dataset* yang digunakan, seluruh nama atribut dituliskan dengan menggunakan huruf kapital. Penulisan nama atribut dengan format tersebut akan menyulitkan proses lainnya yang memerlukan penyertaan nama atribut. Oleh karena itu, dilakukan perubahan format penulisan nama atribut menjadi karakter dengan huruf kecil (*lowercase*) dengan menggunakan karakter *underscore* (_) sebagai pengganti spasi.

Selain itu, juga dilakukan perbaikan pada penulisan nilai kosong (*missing value*) yang sebelumnya ditandai dengan menggunakan karakter tanda tanya (?) menjadi Not a Number (NaN). Perubahan tersebut dilakukan untuk memudahkan proses imputasi data.

Setelah itu, dilakukan imputasi dengan mengisikan nilai kosong menggunakan nilai rata-rata atau modus dari variabel yang bersesuaian. Nilai rata-rata digunakan untuk variabel dengan tipe numerik, sedangkan nilai modus digunakan untuk variabel dengan tipe kategorik. Namun, sebelum dilakukan imputasi, data dibagi berdasarkan nilai dari kelas atau variabel target. Hal tersebut dilakukan agar nilai rata-rata atau modus yang digunakan dalam imputasi dapat merepresentasikan nilai dari setiap kelasnya. Setelah itu, data yang terbagi digabungkan kembali menjadi satu. Potongan *dataset* setelah dilakukan pembersihan ditampilkan pada Tabel 2.

Tabel 2. Potongan *Dataset* setelah Pembersihan Data

class	age	sex	steroid	...	albumin	protime	histology
2	30	2	1	...	4.0	66.571	1
2	50	1	1	...	3.5	66.571	1
2	78	1	2	...	4.0	66.571	1
2	31	1	2	...	4.0	80.000	1
2	34	1	2	...	4.0	66.571	1
...	...	...	...	...	...	...	...
1	46	1	2	...	3.3	50.000	2
2	44	1	2	...	4.3	66.571	2
2	61	1	1	...	4.1	66.571	2
2	53	2	1	...	4.1	48.000	2
1	43	1	2	...	3.1	42.000	2

4.1.2. Transformasi Data

Pada tahap ini dilakukan normalisasi data untuk menskalakan dan menyeimbangkan nilai data pada setiap atribut dengan tipe numerik. Hal tersebut dilakukan karena reduksi dimensi dengan menggunakan metode PCA sensitif terhadap skala data. Pada penelitian ini, dilakukan normalisasi data dengan menggunakan metode z-score yang dituliskan dalam rumus perhitungan sebagai berikut:

$$X_{norm} = \frac{X - \mu}{\sigma}$$

Keterangan:
X_{norm} : Nilai hasil normalisasi
X : Nilai dari setiap entri data
μ : Nilai rata-rata dari atribut data
σ : Nilai standar deviasi dari atribut data

Potongan data setelah dilakukan normalisasi dengan menggunakan metode z-score ditampilkan pada Tabel 3.

Tabel 3. Potongan *Dataset* Setelah Transformasi Data

class	age	sex	steroid	...	albumin	protime	histology
2	-0.891	2.938	-1.016	...	0.306	0.261	-0.905
2	0.7	-0.338	-1.016	...	-0.488	0.261	-0.905
2	2.929	-0.338	0.978	...	0.306	0.261	-0.905
2	-0.812	-0.338	0.978	...	0.306	0.996	-0.905
2	-0.573	-0.338	0.978	...	0.306	0.261	-0.905
...	...	...	...	...	...	...	...
1	0.382	-0.338	0.978	...	-0.805	-0.646	1.098
2	0.223	-0.338	0.978	...	0.783	0.261	1.098
2	1.576	-0.338	-1.016	...	0.465	0.261	1.098
2	0.939	2.938	-1.016	...	0.465	-0.756	1.098
1	0.143	-0.338	0.978	...	-1.123	-1.084	1.098

4.2. Analisis Data Eksploratif

Analisis Data Eksploratif (EDA) merupakan suatu metode untuk menganalisis data yang digunakan untuk memahami sebuah dataset. Analisis data eksploratif bertujuan untuk memperoleh informasi awal dari dataset yang akan dianalisis sebelum dilakukannya analisis statistika yang lebih mendalam.

4.2.1. Statistik Deskriptif

Untuk melakukan EDA dapat menggunakan analisis statistika deskriptif dengan mencari mean, jumlah data, standar deviasi, nilai kuartil, nilai maksimum, dan nilai

minimum. Hasil dari perhitungan statistik deskriptif adalah ditampilkan pada Tabel 4 dan 5.

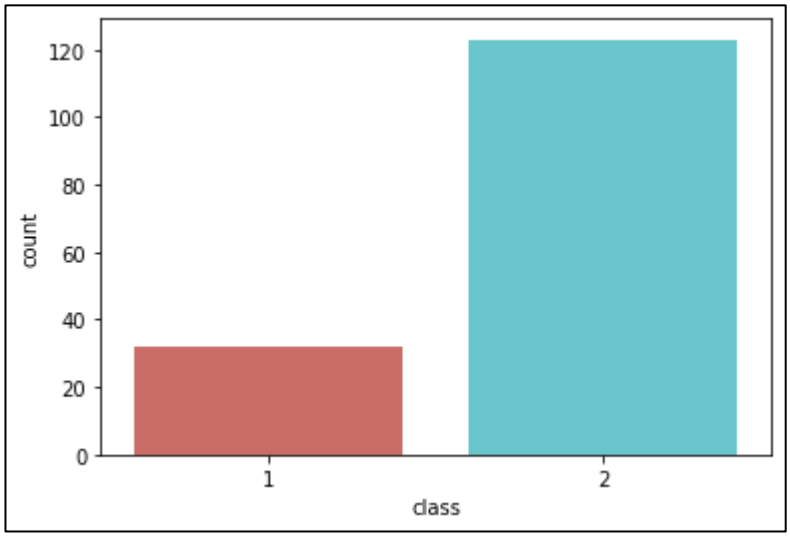
Tabel 4. Statistik Deskriptif Atribut Tipe Numerik

	Jumlah	Missing Value	Mean	Std. Deviasi	Min	25%	50%	75%	Max
age	155	0	41.20	12.57	7.0	32.0	39.00	50.00	78.0
bilirubin	155	0	1.43	1.20	0.3	0.8	1.00	1.50	8.0
alk_ phosphate	155	0	105.66	46.59	26.0	78.0	101.31	122.38	295.0
sgot	155	0	86.03	88.49	14.0	32.5	59.00	99.92	648.0
albumin	155	0	3.81	0.63	2.1	3.4	3.98	4.20	6.4
protime	155	0	61.81	18.27	0.0	46.5	66.57	66.57	100.0

Tabel 5. Statistik Deskriptif Atribut Tipe Kategorik

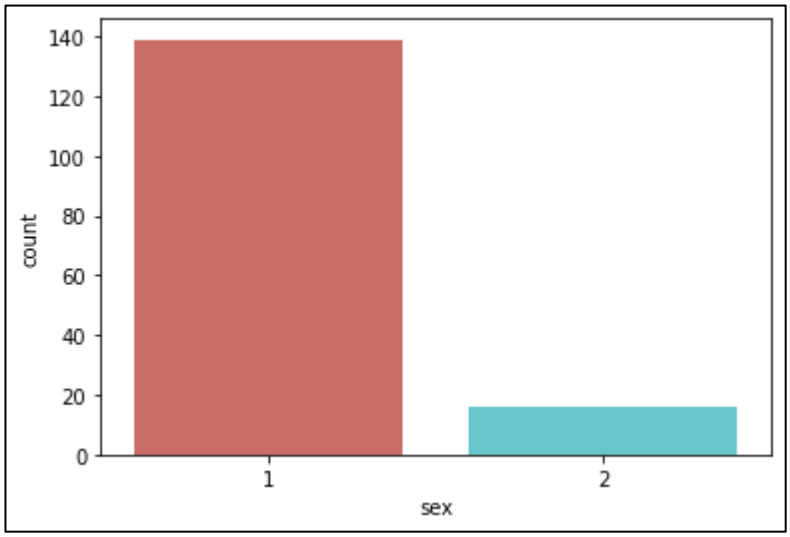
	Jumlah	Missing Value	Jumlah Nilai Unik	Mode	Frekuensi Nilai Mode
class	155	0	2	2	123
sex	155	0	2	1	139
steroid	155	0	2	2	79
antivirals	155	0	2	2	131
fatique	155	0	2	1	101
malaise	155	0	2	2	94
anorexia	155	0	2	2	123
liver_big	155	0	2	2	130
liver_firm	155	0	2	2	95
spleen_palpable	155	0	2	2	125
speiders	155	0	2	2	103
ascites	155	0	2	2	135
varices	155	0	2	2	137
histology	155	0	2	1	85

4.2.2. Jumlah Data



Gambar 2. Grafik Jumlah Nilai Atribut class (kelas)

Berdasarkan grafik pada Gambar 2, dapat terlihat dengan jelas bahwa kelas/label antara pasien meninggal dan hidup tidak seimbang. Terdapat 21% pasien meninggal dan 79% pasien yang hidup.

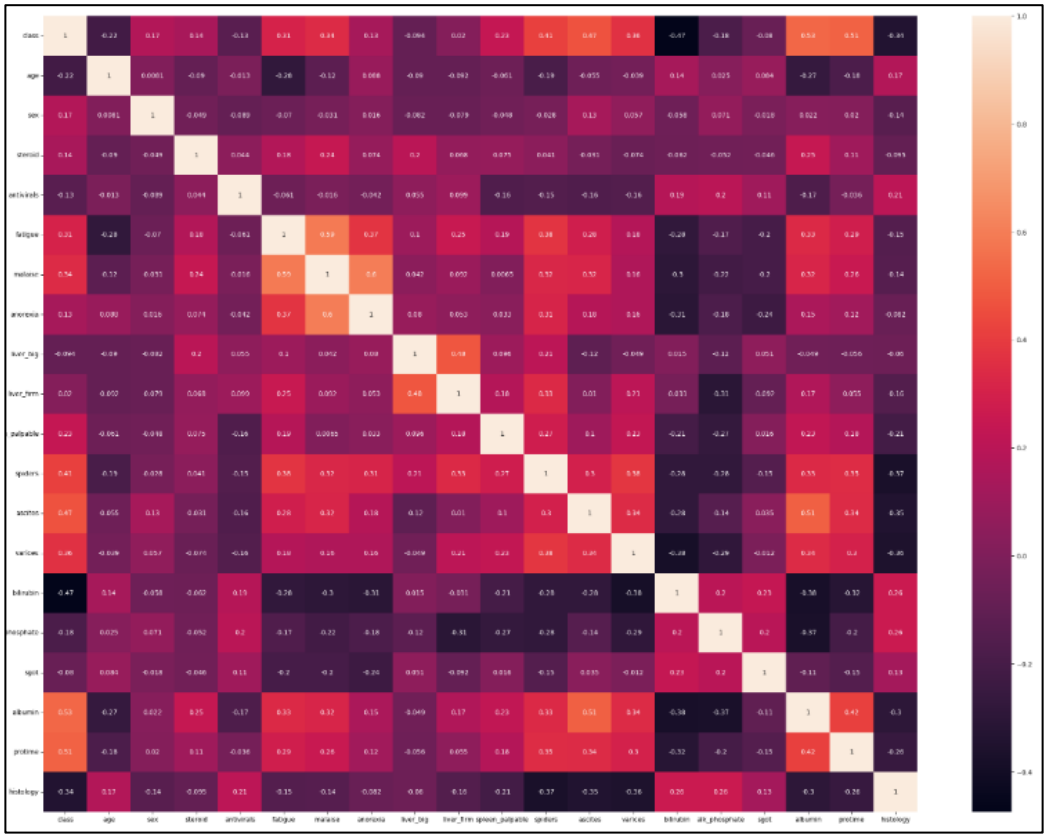


Gambar 3. Grafik Jumlah Nilai Atribut sex (jenis kelamin)

Jenis kelamin pasien pada *dataset* ini didominasi oleh pria dengan jumlah sebanyak 139 orang dibandingkan dengan wanita yang hanya berjumlah 16 orang saja.

4.2.3. Korelasi

Visualisasi matriks korelasi merupakan metode yang digunakan untuk memvisualisasikan korelasi atau hubungan antar variabel dalam sebuah *dataset* menggunakan heatmap atau matriks. Matriks korelasi menunjukkan kondisi hubungan dua variabel yang terkait satu sama lain, apakah berupa korelasi positif atau korelasi negatif.



Gambar 4. Visualisasi matriks korelasi

4.3. Principal Component Analysis (PCA)

Proses reduksi dimensi menggunakan metode PCA dilakukan dengan menggunakan data yang telah dinormalisasikan. Proses penghitungan dilakukan dengan menggunakan SPSS dan Python. Pada Python digunakan bantuan *class* PCA yang terdapat pada *library* sklearn (scikit-learn).

4.3.1. Uji Asumsi

Dilakukan pengujian Kaiser Meyer Olkin (KMO) untuk melihat kecukupan sampel secara keseluruhan dan uji Bartlett untuk mengetahui ada atau tidaknya korelasi antar variabel. Nilai KMO dan Bartlett dianggap sudah mencukupi apabila memiliki nilai lebih dari 0.5 dan signifikansi kurang dari 0.05. Gambar 5 menunjukkan hasil pengujian KMO dan Bartlett yang dilakukan dengan menggunakan SPSS.

KMO and Bartlett's Test

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.746
Bartlett's Test of Sphericity	Approx. Chi-Square	706.878
	df	171
	Sig.	<.001

Gambar 5. Hasil Uji KMO dan Bartlett

Berdasarkan hasil pengujian, didapatkan nilai KMO = 0.746 sehingga dengan nilai tersebut analisis dapat dilanjutkan. Sementara itu, diperoleh signifikansi = 0.001 yang berada jauh di bawah 0.05 sehingga variabel dan sampel sudah bisa dianalisis.

4.3.2. Nilai Eigen dan Vektor Eigen

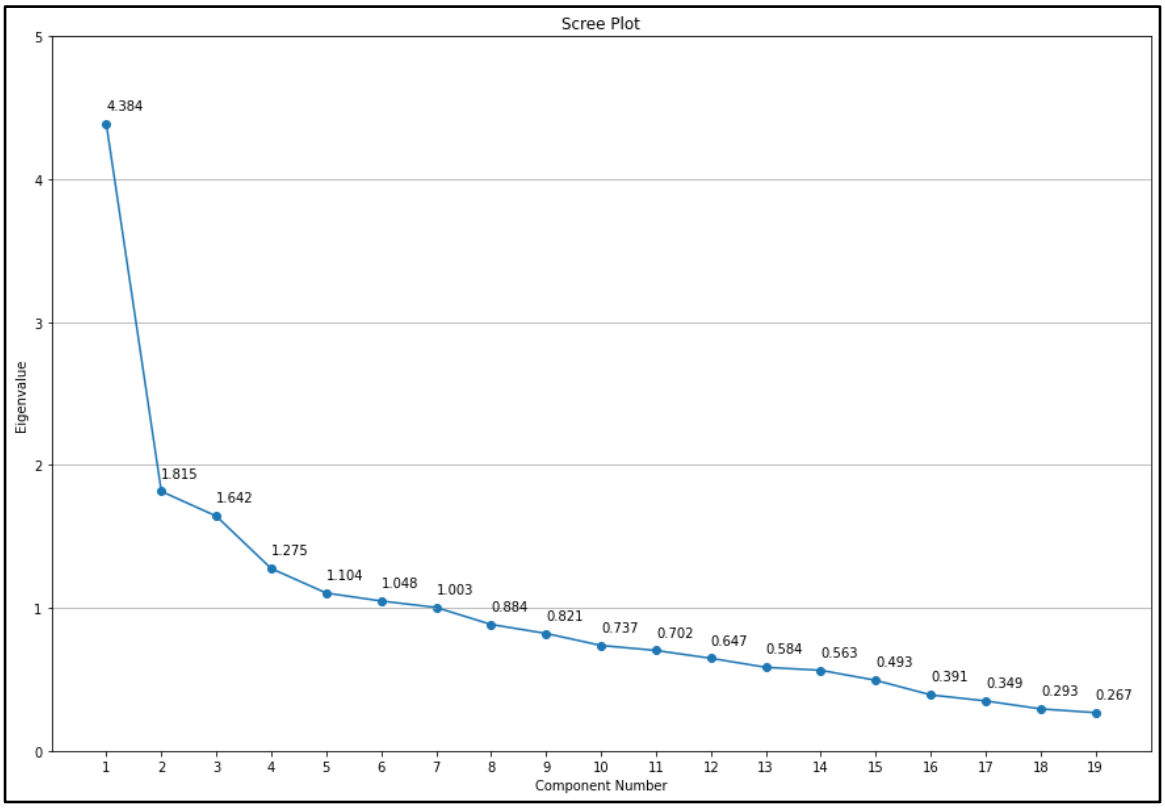
Nilai eigen menyatakan besar keragaman yang mampu dijelaskan oleh suatu variabel. Nilai eigen dan vektor eigen dari matriks kovarians *dataset* ditampilkan pada Tabel 6.

Tabel 6. Hasil Nilai Eigen dan Vektor Eigen

i	Nilai Eigen	Vektor Eigen
1	4.384	[0.136 -0.013 -0.101 0.122 -0.302 -0.293 -0.228 -0.061 -0.165 -0.182 -0.325 -0.269 -0.27 0.286 0.246 0.142 -0.331 -0.271 0.257]
2	1.815	[0.096 0.277 -0.261 -0.266 -0.19 -0.129 -0.104 -0.528 -0.447 -0.029 -0.106 0.308 0.198 -0.161 0.081 0.049 0.121 0.136 -0.139]
3	1.642	[-0.046 0.004 -0.123 -0.144 -0.261 -0.462 -0.43 0.221 0.311 0.343 0.125 -0.025 0.228 0.081 -0.214 0.218 0.049 -0.004 -0.25]
4	1.275	[-0.549 -0.018 0.352 0.266 0.113 -0.033 -0.378 -0.055 -0.111 -0.048 -0.093 0.133 -0.145 0.074 0.274 0.236 0.279 0.26 -0.03]
5	1.104	[0.258 0.034 -0.231 0.389 0.045 0.131 0.154 0.092 0.143 -0.1 0.132 0.358 0.26 0.128 0.202 0.607 -0.047 0.013 0.068]
6	1.048	[-0.193 0.743 -0.032 -0.001 -0.033 -0.009 0.044 0.265 0.156 -0.314 0.088 0.022 -0.046 0.018 0.208 -0.167 -0.147 -0.125 -0.306]
7	1.003	[0.377 0.214 0.64 -0.282 -0.108 0.097 0.089 0.175 -0.145 0.158 -0.164 0.108 -0.139 0.06 -0.067 0.315 0.141 -0.159 -0.083]
8	0.884	[-0.328 -0.084 -0.224 -0.499 0.314 0.068 0.076 0.119 -0.179 0.282 0.176 0.009 -0.123 -0.016 0.314 0.286 -0.184 -0.295 0.06]
9	0.821	[-0.095 -0.178 -0.152 -0.301 0.046 0.118 -0.111 -0.035 0.134 -0.495 -0.048 0.314 -0.157 0.442 -0.345 0.02 0.246 -0.225 -0.047]
10	0.737	[0.053 -0.494 0.19 -0.116 -0.247 -0.043 0.028 0.17 -0.179 -0.399 0.215 -0.012 0.162 -0.149 0.283 -0.013 -0.186 0.045 -0.456]
11	0.702	[0.214 0.013 -0.004 0.041 0.105 0.032 -0.01 -0.16 -0.091 0.269 0.319 0.065 -0.3 0.591 0.082 -0.166 -0.219 0.347 -0.292]
12	0.647	[-0.201 -0.046 0.089 0.328 0.039 0.148 0.101 -0.357 -0.064 0.193 -0.13 -0.121 0.209 0.124 -0.157 0.045 -0.085 -0.544 -0.468]
13	0.584	[0.303 -0.13 0.06 0.003 0.27 -0.126 -0.243 -0.083 0.335 0.121 -0.168 0.352 -0.006 -0.104 0.462 -0.358 0.112 -0.294 -0.05]
14	0.563	[0.055 0.113 0.284 -0.259 0.248 0.038 -0.155 -0.284 0.162 -0.197 0.149 -0.353 0.554 0.269 0.126 0.037 -0.077 0.057 0.227]
15	0.493	[0.091 -0.02 -0.12 -0.099 0.381 0.18 -0.212 0.132 0.045 -0.058 -0.583 -0.053 0.035 -0.106 -0.154 0.139 -0.377 0.315 -0.287]
16	0.391	[-0.214 -0.067 -0.02 -0.065 -0.198 -0.013 0.305 0.309 -0.124 0.193 -0.41 0.182 0.413 0.413 0.168 -0.265 0.045 0.112 0.096]
17	0.349	[-0.021 -0.054 -0.121 -0.175 -0.137 -0.054 0.405 -0.256 0.441 -0.005 -0.223 -0.324 -0.204 0.02 0.278 0.207 0.317 0.189 -0.231]
18	0.293	[-0.189 -0.021 0.122 -0.132 -0.487 0.468 -0.14 -0.224 0.363 0.13

		0.003 0.249 -0.054 -0.094 0.033 -0.01 -0.396 0.054 0.149]
19	0.267	[-0.177 -0.027 0.27 -0.024 0.168 -0.584 0.377 -0.206 0.15 -0.092
		-0.028 0.325 -0.052 -0.054 -0.188 0.066 -0.384 0.058 0.08]

Berdasarkan nilai eigen yang diperoleh dapat ditentukan jumlah komponen yang akan dibentuk, yaitu sejumlah nilai eigen yang nilainya lebih besar dari 1. Gambar 6 merupakan Scree Plot yang menunjukkan nilai eigen dari setiap komponen.



Gambar 6. *Scree Plot*

Berdasarkan *Scree Plot* yang terbentuk terdapat 7 komponen yang memiliki nilai eigen lebih dari 1. Oleh karena itu, dapat diartikan akan ada 7 komponen yang terbentuk dari proses PCA.

4.3.3. Transformasi Data

Transformasi data merupakan perubahan data awal menjadi data baru berdasarkan jumlah komponen yang telah ditentukan pada proses sebelumnya. Potongan data baru setelah dilakukan reduksi dimensi dengan menggunakan metode PCA dengan sejumlah 7 komponen ditampilkan pada Tabel 7.

Tabel 7. Potongan *Dataset* Setelah Transformasi Data PCA

PC1	PC2	PC3	PC4	PC5	PC6	PC7	class
-2.006	1.595	-0.619	-0.075	0.022	2.008	-1.22	2
-0.574	1.255	-0.374	-0.996	0.635	-0.372	-1.198	2
-1.17	-0.433	0.058	-1.714	0.701	-0.424	1.516	2

-3.079	-0.433	0.355	-0.205	-1.313	-0.121	0.683	2
-1.91	-1.104	0.082	0.943	1.098	-0.103	0.576	2
…	…	…	…	…	…	…	…
5.224	-2.768	1.589	1.47	-0.283	-0.637	1.47	1
-0.46	0.087	-0.785	0.553	0.729	-0.972	1.05	2
0.99	2.473	-0.613	-1.197	-0.307	-2.009	-0.454	2
1.535	-0.055	-1.83	-0.969	-0.028	2.669	0.601	2
1.87	-2.021	-1.452	-1.053	-0.942	0.471	-0.089	1

4.4. Pengujian Model Klasifikasi

Dalam penelitian ini model klasifikasi yang diuji meliputi model Logistic Regression, Decision Tree, dan K-Nearest Neighbour (KNN). Pengujian dilakukan dengan membagi data latih dan uji dalam proporsi sebesar 80:20. Terdapat 2 tahap pengujian, yaitu pengujian dengan menggunakan *dataset* sebelum diterapkan reduksi dimensi PCA dan pengujian yang menggunakan *dataset* sesudah diterapkan reduksi dimensi PCA. Hal tersebut dilakukan untuk mengetahui pengaruh reduksi dimensi metode PCA pada model klasifikasi dan *dataset* yang diujikan. Seluruh model klasifikasi dibangun dengan menggunakan parameter bawaan yang tersedia dari kelas klasifikasi yang terdapat dalam *module* dan *library* scikit-learn.

Hasil pengujian dengan menggunakan *dataset* sebelum dan setelah diterapkan reduksi dimensi PCA ditunjukkan pada Tabel 8 dan Tabel 9.

Tabel 8. Hasil Pengujian Menggunakan *Dataset* Sebelum Reduksi Dimensi PCA

Metrik	**Model Klasifikasi**		
	Logistic Regression	**Decision Tree**	**KNN**
Accuracy	0.871	0.935	0.839
Recall	0.958	0.958	0.917
Specificity	0.571	0.857	0.571
Precision	0.885	0.958	0.880
NPV	0.800	0.857	0.667
F1-Score	0.920	0.958	0.898
Training Time (s)	0.003000	0.002001	0.001000

Tabel 9. Hasil Pengujian Menggunakan *Dataset* Sesudah Reduksi Dimensi PCA

Metrik	**Model Klasifikasi**		
	Logistic Regression	**Decision Tree**	**KNN**
Accuracy	0.903	0.742	0.839
Recall	0.958	0.792	0.917
Specificity	0.714	0.571	0.571
Precision	0.920	0.864	0.880
NPV	0.833	0.444	0.667
F1-Score	0.939	0.826	0.898
Training Time (s)	0.003307	0.001000	0.000999

Berdasarkan hasil pengujian dari seluruh model klasifikasi yang diujikan, ditemukan bahwa metrik *specificity* dan *F1-Score* menjadi acuan utama dalam menentukan model klasifikasi yang terbaik. Hal tersebut disebabkan oleh tingginya risiko *False Positive* (FP) pada kasus klasifikasi pasien penderita hepatitis. Apabila seorang pasien pada kondisi sebenarnya tidak berisiko meninggal, tetapi diprediksi akan meninggal, maka pada pemeriksaan atau perawatan yang lebih lanjut pasien tersebut dapat mengetahui kondisi yang sebenarnya bahwa ia tidak berisiko meninggal. Namun, apabila terdapat pasien pada kondisi sebenarnya berisiko meninggal, tetapi diprediksi hidup, maka pasien tersebut akan terlambat mengetahui kondisi kesehatannya karena tidak dilakukan pemeriksaan dan penanganan medis yang sesuai. Oleh karena itu, digunakan metrik Specificity untuk menghitung seberapa banyak kasus negatif yang diprediksi benar dari semua kasus negatif yang ada. Sementara itu, metrik F1-Score lebih dipertimbangkan dibandingkan dengan metrik *accuracy* karena distribusi dari kelas positif dan negatif pada *dataset* tidak seimbang.

4.5. Evaluasi Model Klasifikasi

Perbandingan performa model klasifikasi dengan algoritma Logistic Regression, Decision Tree, dan K-Nearest Neighbour (KNN) dengan menggunakan data sebelum maupun sesudah reduksi dimensi PCA disajikan pada Tabel 10.

Tabel 10. Perbandingan Performa Model Klasifikasi

Model Klasifikasi	Metrik							
	Dataset Sebelum Reduksi Dimensi				Dataset Setelah Reduksi Dimensi			
	Accuracy	*F1-Score*	*Specificity*	*Training Time (s)*	*Accuracy*	*F1-Score*	*Specificity*	*Training Time (s)*
Logistic Regression	0.871	0.920	0.714	0.003000	0.903	0.939	0.714	0.003307
Decision Tree	0.935	0.958	0.857	0.002001	0.742	0.826	0.571	0.001000
KNN	0.839	0.898	0.571	0.001000	0.839	0.898	0.571	0.000999

Model Logistic Regression pada *dataset* sesudah reduksi dimensi menunjukkan performa yang lebih baik dibandingkan dengan penerapannya pada *dataset* sebelum reduksi dimensi. Ketika menggunakan *dataset* sesudah reduksi dimensi, model tersebut mencapai *accuracy* sebesar 90.3%, sedangkan penerapannya pada *dataset* sebelum reduksi dimensi hanya mencapai *accuracy* sebesar 87.1%. Seluruh metrik evaluasi pada model ini mengalami peningkatan, kecuali untuk metrik *recall* yang nilainya tetap. Selain itu, waktu pelatihan model menjadi sedikit lebih lama dengan selisih 0.000307 detik.

Sementara itu, model Decision Tree mendapatkan hasil yang berlawanan dengan model Logistic Regression, di mana model ini mengalami penurunan performa yang signifikan ketika diterapkan pada *dataset* setelah reduksi dimensi. Pada penerapannya menggunakan *dataset* sebelum reduksi dimensi, model Decision Tree mencapai *accuracy* sebesar 93.5%, tetapi ketika diterapkan pada *dataset* setelah reduksi dimensi, nilai *accuracy* turun menjadi 74.2%. Penurunan tersebut juga terlihat pada seluruh metrik

evaluasi. Satu-satunya peningkatan yang terjadi pada model ini adalah waktu pelatihan model yang lebih singkat dengan selisih 0.001001 detik.

Berbeda dengan 2 model sebelumnya yang mengalami peningkatan dan penurunan performa, model KNN menunjukkan nilai yang konsisten pada semua metrik evaluasi dengan *accuracy* sebesar 83.9%. Meskipun begitu, terdapat perbedaan pada waktu pelatihan model yang sangat kecil, yaitu sebesar 0.000001 detik lebih cepat pada penerapannya menggunakan *dataset* setelah reduksi dimensi.

Berdasarkan evaluasi performa model klasifikasi yang telah dilakukan, ditemukan bahwa model Decision Tree merupakan model terbaik dengan nilai *accuracy*, F1-Score, dan recall sebesar 93.5%, 95.8%, dan 85.7% ketika diterapkan pada *dataset* sebelum reduksi dimensi. Sementara itu, pada penerapan menggunakan *dataset* setelah reduksi dimensi, model Logistic Regression menjadi model terbaik dengan nilai *accuracy*, *F1-Score*, dan *specificity* sebesar 90.3%, 93.9%, dan 71.4%. Hasil penelitian juga menunjukkan bahwa reduksi dimensi menggunakan metode PCA dapat meningkatkan performa model Logistic Regression dan menurunkan performa model Decision Tree.

BAB V
KESIMPULAN

Berdasarkan pengujian yang telah dilakukan, reduksi dimensi dengan menggunakan metode Principal Component Analysis (PCA) telah berhasil diterapkan pada *dataset* Hepatitis yang menghasilkan *dataset* dengan dimensi baru, yaitu 155 baris dan 8 kolom dari yang sebelumnya sebesar 155 baris dan 20 kolom.

Berdasarkan hasil evaluasi performa dari tiga model klasifikasi yaitu Logistic Regression, Decision Tree, dan K-Nearest Neighbour pada *dataset* Hepatitis, dapat disimpulkan bahwa model Decision Tree memberikan performa terbaik dibandingkan dengan model klasifikasi lainnya saat diuji menggunakan *dataset* sebelum reduksi dimensi, dengan nilai *accuracy*, F1-Score, dan recall sebesar 93.5%, 95.8%, dan 85.7%.

Pada pengujian dengan menggunakan *dataset* setelah reduksi dimensi, model Logistic Regression menjadi model terbaik dengan nilai *accuracy*, *F1-Score*, dan *specificity* sebesar 90.3%, 93.9%, dan 71.4%. Hasil penelitian juga menunjukkan bahwa reduksi dimensi menggunakan metode PCA dapat meningkatkan performa model Logistic Regression dan menurunkan performa model Decision Tree. Secara keseluruhan, performa klasifikasi terbaik didapatkan dengan menerapkan model Decision Tree pada *dataset* sebelum reduksi dimensi.

DAFTAR PUSTAKA

[1] W. D. Septiani, "Penerapan Algoritma C4.5 Untuk Prediksi Penyakit Hepatitis," *Jurnal Techno Nusa Mandiri,* vol. 11, no. 1, pp. 79-78, 2014.

[2] D. Hediyati and I. M. Suartana, "Penerapan Principal Component Analysis(PCA) untuk Reduksi Dimensi pada Proses Clustering Data Produksi Pertanian di Kabupaten Bojonegoro," *Journal of Information Engineering and Educational Technology,* vol. 5, no. 2, pp. 49-54, 2021.

[3] S. F. Putra, R. Pradina and I. Hafidz, "Feature Selection pada Dataset Faktor Kesiapan Bencana pada Provinsi di Indonesia Menggunakan Metode PCA (Principal Component Analysis)," *Jurnal Teknik ITS,* vol. 5, no. 2, pp. 88-92, 2016.

[4] B. N. Azmi, A. Hermawan and D. Avianto, "Analisis Pengaruh PCA Pada Klasifikasi Kualitas Air Menggunakan Algoritma K-Nearest Neighbordan Logistic Regression," *Jurnal Sistem dan Teknologi Informasi,* vol. 7, no. 2, pp. 94-103, 2022.

[5] D. Cielen, A. D. B. Meysman and M. Ali, Introducing Data Science, New York: Manning Publications, 2016.

[6] F. Cady, The Data Science Handbook, New Jersey: Wiley, 2017.

[7] L. Igual and S. Seguí, Introduction to Data Science, Cham: Springer, 2017.

[8] Kusrini and E. T. Luthfi, Algoritma Data Mining, Yogyakarta: Andi Offset, 2009.

[9] C. N. Prabiantissa, "Klasifikasi pada Dataset Penyakit Hati Menggunakan Algoritma Support Vector Machine, K-NN, dan Naïve Bayes," in *Seminar Nasional Teknik Elektro, Sistem Informasi, dan Teknik Informatika*, Surabaya, 2021.

[10] A. Dinanti and J. Purwadi, "Analisis Performa Algoritma K-Nearest Neighbor dan Reduksi Dimensi Menggunakan Principal Component Analysis," *Jambura Journal of Mathematics,* vol. 5, no. 1, pp. 155-165, 2023.

[11] UCI Machine Learning Repository, "Hepatitis," 1988. [Online]. Available: https://doi.org/10.24432/C5Q59J.

LAMPIRAN

Lampiran 1. Kode Program *Import Library*

```
import time
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

Lampiran 2. Kode Program Pemuatan *Dataset*

```
data_df = pd.read_csv('data/hepatitis.csv')
data_df
```

Lampiran 2. Kode Program Pembersihan Data

```
# mengubah nama feature menjadi huruf kecil dan mengubah spasi menjadi
underscore
data_df.columns = data_df.columns.str.replace(' ', '_').str.lower()

# mengubah "?" menjadi NaN
data_df = data_df.replace('?', np.NaN)

# memeriksa tipe data
data_df.dtypes

# menyesuaikan tipe data
# inisialisasi tipe data feature
int_features = ['age']
float_features = ['bilirubin', 'alk_phosphate', 'sgot', 'albumin',
'protime']

# mengubah tipe data menjadi numerik untk feature age, bilirubin,
alk_phosphate, sgot, albumin, and protime
data_df = data_df.astype({col: 'int64' for col in int_features})
data_df = data_df.astype({col: 'float64' for col in float_features})

# mengubah tipe data menjadi object untuk feature lainnya
data_df = data_df.astype({col: 'object' for col in data_df.columns if col
not in int_features + float_features})
```

Output:

```
class               int64
age                 int64
sex                 int64
steroid            object
antivirals          int64
fatigue            object
malaise            object
anorexia           object
liver_big          object
liver_firm         object
spleen_palpable    object
spiders            object
ascites            object
varices            object
bilirubin          object
alk_phosphate      object
sgot               object
albumin            object
protime            object
histology           int64
dtype: object
```

Lampiran 3. Imputasi *Missing Value*

```
# memisahkan data berdasarkan class
class_1_df = data_df[data_df['class'] == 1]
class_2_df = data_df[data_df['class'] == 2]

# menghitung jumlah missing value pada class 1
class_1_df.isnull().sum()

# imputasi missing value dengan mean yang dibulatkan 3 angka di belakang
koma untuk kolom numerik pada class_1_df
numeric_features   =   ['age',   'bilirubin',   'alk_phosphate',   'sgot',
'albumin', 'protime']
class_1_df[numeric_features]                                              =
class_1_df[numeric_features].fillna(class_1_df[numeric_features].mean())

# imputasi missing value dengan mode untuk kolom kategorik pada class_1_df
categorical_columns = [col for col in class_1_df.columns if col not in
numeric_features]
class_1_df[categorical_columns]                                           =
class_1_df[categorical_columns].fillna(class_1_df[categorical_columns].mo
de().iloc[0])

# menghitung jumlah missing value pada class 2
class_2_df.isnull().sum()

# imptasi missing value dengan mean untuk kolom numerik pada class_2_df
class_2_df[numeric_features]                                              =
class_2_df[numeric_features].fillna(class_2_df[numeric_features].mean())

# imputasi missing value dengan mode untuk kolom kategorik pada class_2_df
```

```
categorical_columns = [col for col in class_2_df.columns if col not in
numeric_features]
class_2_df[categorical_columns]                                          =
class_2_df[categorical_columns].fillna(class_2_df[categorical_columns].mo
de().iloc[0])

# menggabungkan class 1 dan class 2
data_df_2 = pd.concat([class_1_df, class_2_df], axis=0)

# mengurutkan data berdasarkan index
data_df_2 = data_df_2.sort_index()
data_df_2
```

Lampiran 4. Kode Program Kode Program Transformasi Data (Normalisasi Data)

```
# mengubah semua tipe data menjadi numerik (diperlukan untuk proses
standardisasi)
data_df_2 = data_df_2.astype({col: 'int64' for col in data_df_2.columns if
col not in numeric_features})

# membuat predifined function untuk standardisasi data
def rataan(data):
    jumlah = 0
    n = 0
    for i in data:
        jumlah+=i
        n += 1
    hasil_rataan = jumlah/n
    return hasil_rataan

def std_dev(data):
    jumlah = 0
    n = 0
    for i in data:
        jumlah += (i - rataan(data))**2
        n += 1
    hasil_varians = jumlah/(n-1)
    hasil_stdev = hasil_varians**(0.5)
    return hasil_stdev

def z_score_standardization(data):
    return (data - rataan(data)) / std_dev(data)

# menyalin data
std_data_df = data_df_2.copy()

# normalisasi semua feature kecuali class
for col in std_data_df.columns:
    if col != 'class':
        std_data_df[col] = z_score_standardization(std_data_df[col])
```

Lampiran 5. Kode Program Analisis Data Eksploratif

```
# menyesuaikan tipe data

# inisialisasi tipe data feature
int_features = ['age']
float_features = ['bilirubin', 'alk_phosphate', 'sgot', 'albumin',
'protime']

# mengubah tipe data menjadi numerik untk feature age, bilirubin,
alk_phosphate, sgot, albumin, and protime
data_df_2 = data_df_2.astype({col: 'int64' for col in int_features})
data_df_2 = data_df_2.astype({col: 'float64' for col in float_features})

# mengubah tipe data menjadi object untuk feature lainnya
data_df_2 = data_df_2.astype({col: 'object' for col in data_df.columns if
col not in int_features + float_features})

# statistik deskriptif
# menampilkan statistik deskriptif
data_df_2.describe()
data_df_2.describe(include='object')

# visualisasi data
# Count Plot untuk atribut class
sns.countplot(x='class', data=data_df_2, palette='hls')
plt.xlabel('class')
plt.show()

# Count Plot untuk atribut sex
sns.countplot(x='sex', data=data_df_2, palette='hls')
plt.xlabel('sex')
plt.show()

# matriks korelasi
data_df_2 = data_df_2.astype({col: 'int64' for col in data_df_2.columns if
col not in numeric_features})

plt.figure(figsize = (30,22))
sns.heatmap(data_df_2.corr(), annot = True)
```

Lampiran 6. Kode Program Principal Component Analysis (PCA)

```
# nilai eigen dan vektor eigen

# membuat objek pca
pca = PCA()

# fit data
pca.fit(x_std_df)

# mendapatkan nilai eigen dan vektor eigen
eigenvalues = np.round(pca.explained_variance_, 3)
eigenvectors = np.round(pca.components_, 3)
```

```
for i, (eigenvalue, eigenvector) in enumerate(zip(eigenvalues,
eigenvectors)):
    print(f"Eigenvalue {i+1}: {eigenvalue}")
    print(f"Eigenvector {i+1}:")
    print(eigenvector)
    print()

# scree plot
# mengatur ukuran plot
plt.figure(figsize=(15,10))

# membuat titik dan garis pada plot
plt.scatter(range(1,x_std_df.shape[1]+1),eigenvalues)
plt.plot(range(1,x_std_df.shape[1]+1),eigenvalues)

# menambahkan judul dan label pada plot
plt.title('Scree Plot')
plt.xlabel('Component Number')
plt.ylabel('Eigenvalue')

# mengatur batas sumbu x dan y
plt.xlim(0,20)
plt.ylim(0,5)

# mengatur interval sumbu x dan y
plt.xticks(np.arange(1,20,1))

# menambahkan nilai di setiap titik pada plot hanya untuk > 1
for i, txt in enumerate(eigenvalues):
    plt.annotate(np.round(txt,3), (i+1, eigenvalues[i]+0.1))

# membuat garis vertikal pada plot
plt.grid(axis='y')

# menampilkan plot
plt.show()

# PCA
# mendapatkan jumlah komponen yang memiliki eigenvalue > 1
n_components = np.sum(eigenvalues > 1)

# membuat objek pca dengan n_components
pca_n = PCA(n_components=n_components)

# fit & transform data
x_pca_n = pca_n.fit_transform(x_std_df)

# mengubah menjadi dataframe
pca_df = pd.DataFrame(x_pca_n, columns=[f'PC{i}' for i in
range(1,n_components+1)])
pca_df['class'] = std_data_df['class']
pca_df
```

Lampiran 7. Pembentukan Model Klasifikasi

```
# membuat objek model klasifikasi

logreg = LogisticRegression()
tree = DecisionTreeClassifier(random_state=0)
knn = KNeighborsClassifier()

# list model
model_list = [logreg, tree,  knn]

# membuat predefined function untuk melakukan training dan testing
def train_test_model(model, xtrain, xtest, ytrain):
    start = time.time()
    # melakukan training model
    model.fit(xtrain, ytrain)
    end = time.time()

    # melakukan testing model
    y_pred = model.predict(xtest)

    # menghitung waktu training
    training_time = end - start

    # mengembalikan hasil testing
    return y_pred, training_time

# membuat predefined function untuk melakukan evaluasi model
def evaluate_model(ytest, ypred):
    report = classification_report(ytest, ypred, output_dict=True)

    # accuracy
    accuracy = report['accuracy']

    # recall & specificity
    recall = report['2']['recall']
    specificity = report['1']['recall']

    # precision & npv
    precision = report['2']['precision']
    npv = report['1']['precision']

    # f1-score
    f1_score = report['2']['f1-score']

    return accuracy, recall, specificity, precision, npv, f1_score

# membuat predefined function untuk melakukan training, testing, dan
evaluasi model
def model_report(models, xtrain, xtest, ytrain, ytest):
    eval_list = []

    for model in models:
```

```
        # training dan testing model
        y_pred, training_time = train_test_model(model, xtrain, xtest,
ytrain)
        # evaluasi model
        accuracy, recall, specificity, precision, npv, f1_score =
evaluate_model(ytest, y_pred)
        # menampilkan hasil evaluasi dalam bentuk dictionary
        model_name = model.__class__.__name__
        eval_dict = {'Model': model_name,
                     'Accuracy': accuracy,
                     'Recall': recall,
                     'Specificity': specificity,
                     'Precision': precision,
                     'NPV': npv,
                     'F1-Score': f1_score,
                     'Training Time (s)': training_time
                     }

        eval_list.append(eval_dict)

    # membuat dataframe dari hasil evaluasi
    eval_df = pd.DataFrame(eval_list)
    eval_df = eval_df.set_index('Model')
    eval_df.index.name = 'Metrik'
    eval_df = eval_df.T

    return eval_df
```

Lampiran 8. Kode Program Pengujian Model Klasifikasi

```
# dataset sebelum reduksi dimensi PCA
X = std_data_df.drop(['class'], axis=1)
y = std_data_df['class']

# membagi data menjadi training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# menampilkan jumlah data training dan testing
print(f"Jumlah data training: {X_train.shape[0]}")
print(f"Jumlah data testing : {X_test.shape[0]}")

withouth_pca = model_report(model_list, X_train, X_test, y_train, y_test)
withouth_pca

# dataset setelah reduksi dimensi PCA
X_pca = pca_df.drop(['class'], axis=1)
y_pca = pca_df['class']

# membagi data menjadi training dan testing
X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(X_pca, y_pca,
test_size=0.2, random_state=42)
```

```
# menampilkan jumlah data training dan testing
print(f"Jumlah data training: {X_train.shape[0]}")
print(f"Jumlah data testing : {X_test.shape[0]}")

with_pca = model_report(model_list, X_train_2, X_test_2, y_train_2,
y_test_2)
with_pca
```