LAPORAN PRAKTIKUM 3 – IMBALANCED

PERBANDINGAN MODEL REGRESI LOGISTIK LINEAR DENGAN MODEL NON-STATISTIK RANDOM FOREST MENGGUNAKAN UCI HEART DISEASE DATASET DENGAN PENERAPAN IMBALANCED DATASET

Kadek Savita Dyutianaya
NRP 3324600033

Dosen Pengampu
Fitrah Maharani Humaira M.Kom

MATA KULIAH PRAKTIKUM ANALISA STATISTIKA TERAPAN
PROGRAM STUDI D4 SAINS DATA TERAPAN B
DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
SURABAYA
2025

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Tubuh manusia terdiri dari beberapa organ yang masing-masing memiliki fungsi tertentu. Organ terpenting dalam tubuh kita adalah jantung, yang memompa darah ke paru-paru. Jantung dilindungi oleh tulang rusuk dan kulitnya dilapisi oleh membran yang disebut pericardium. Perikardium memiliki empat ruang yang memisahkan darah yang mengandung oksigen dan darah yang tidak mengandung oksigen. Penyakit jantung terjadi ketika plak menumpuk di arteri dan pembuluh darah. Plak adalah bahan lilin yang dihasilkan oleh kolesterol, molekul lemak, dan mineral. Tekanan darah tinggi, merokok, atau kolesterol atau trigliserida yang tinggi dapat merusak lapisan dalam arteri [1].

Sejak tahun 2000, penyakit jantung iskemik mengalami peningkatan terbesar dalam jumlah kematian dari 2,7 juta menjadi 9,1 juta kematian pada tahun 2021. Penyakit ini bertanggung jawab atas 13% dari total kematian dunia[2]. Penyakit jantung iskemik atau biasa disebut Coronary Artery Disease (CAD) menyumbat arteri koroner yang memasok darah kaya oksigen ke jantung, sehingga jumlah darah yang mencapai otot jantung lama kelamaan akan semakin sedikit[3]. Upaya pencegahan dapat dilakukan dengan melakukan deteksi dini dan prediksi penyakit jantung. Dengan perkembangan teknologi Data Science, pengolahan data medis dan prediksi berbasis model dapat diberikan oleh machine learning.

Salah satu metode yang sering digunakan dan akurat memperpanjang hidup pasian serta mengurangi keparahan penyakit jantung mereka adalah Regresi Logistik. Metode ini dapat memodelkan hubungan antara variabel dependen yaitu faktor-faktor risiko (umur, kadar kolestrol, dll), dengan variabel dependen biner (apakah pasien mengidap penyakit jantung atau tidak)[4]. Secara umum, regresi logistik sangat cocok untuk menggambarkan dan menguji hipotesis tentang hubungan antara variabel hasil kategorikal dan satu atau lebih variabel prediktor kategorikal atau kontinu.[5]

Penelitian ini menggunakan dataset Heart Disease UCI dari UCI Machine Learning Repository untuk membangun model regresi logistik biner, membandingkan kinerjanya dengan teknik balancing data,dan mengevaluasi hasilnya melalui metrik evaluasi klasifikasi.[6]

1.2 Rumusan Masalah

1. Bagaimana membangun model regresi logistik biner untuk memprediksi penyakit jantung berdasarkan dataset Heart Disease UCI?
2. Bagaimana pengaruh teknik penyeimbangan data (oversampling, undersampling, SMOTE) terhadap performa model?
3. Bagaimana hasil evaluasi model berdasarkan confusion matrix, accuracy, precision, recall, dan F1-score?

1.3 Tujuan

1. Menerapkan model regresi logistik biner pada dataset Heart Disease UCI.
2. Membandingkan performa model dengan berbagai teknik penyeimbangan data.
3. Mengevaluasi kinerja model dengan confusion matrix dan metrik evaluasi lainnya.

1.4 Manfaat

1. Memberikan gambaran bagaimana regresi logistik dapat digunakan untuk prediksi penyakit jantung.
2. Memberikan referensi bagi penelitian selanjutnya dalam bidang prediksi kesehatan berbasis machine learning.
3. Mendukung upaya pencegahan penyakit jantung dengan analisis berbasis data.

1.5 Batasan Masalah

1. Dataset yang digunakan hanya Heart Disease UCI (Cleveland dataset) dari UCI Machine Learning Repository yang diunggah ulang di Kaggle.[7]

2. Variabel target dibatasi menjadi biner: 0 = tidak ada penyakit, 1 = ada penyakit.
3. Model utama adalah Regresi Logistik Biner, dengan perbandingan Random Forest.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Jurnal "Efficient system for heart disease prediction by applying logistic regression." membahas mengenai pengembangan sistem prediksi penyakit jantung yang efisien menggunakan algoritma Logistic Regression. Jurnal ini mengidentifikasi ada atau tidaknya penyakit jantung pada seseorang. Penggunaan metode Algoritma Logistic Regression berhasil mengatasi kesulitan dokter untuk mendiagnosis jantung yang memerlukan pemahaman cermat terhadap data pasien dan identifikasi parameter penyebab penyakit. Model regresi logistik yang digunakan merupakan model logistik biner dengan dua kemungkinan hasil: positif atau negatif (kelas label 0 atau 1). Model menggunakan dua fase utama yaitu Regularized Cost Function (mengatasi risiko overfitting dan menghitung estimasi kemungkinan maksimum) dan Regularized Gradient Descent (metode iteratif untuk mendapatkan koefisien optimal yang meminimalkan fungsi biaya). Sistem yang diusulkan dievaluasi dengan membandingkan efisiensinya terhadap algoritma Naïve Bayes dan Random Forest. Hasilnya, logistic regression mencapai akurasi 87% yang lebih tinggi dibanding Naïve Bayes (83.7%) dan Random Forest (80%) pada dataset yang sama [8]

Jurnal "Heart Disease Prediction Using Logistic Regression" membahas mengenai pengembangan model prediksi penyakit jantung menggunakan Binary Logistic Regression (BLR) dengan membandingkan tiga metode untuk menemukan model terbaik. Jurnal ini menggunakan data penyakit jantung dari UCI Machine Learning Repository yang terdiri dari 271 sampel dengan satu variabel respons (kehadiran penyakit) dan dua belas variabel independen (usia, jenis kelamin, tekanan darah istirahat, dll). Nantinya, metode ini akan menentukan variabel-variabel mana yang signifikan memengaruhi ada atau tidaknya penyakit jantung dan menemukan model prediksi terbaik. Pada jurnal ini, dibandingkan tiga metode Regresi Logistik Biner, yaitu Regresi Logistik Biner model biasa, dengan metode Least Quartile Difference, dan metode robust (Median Absolute Deviation). Terdapat temuan yang signifikan, di mana analisis Regresi Logistik Biner menunjukkan sepuluh variabel independen yang signifikan (p-value <0.10) yang secara langsung atau tidak langsung proporsional dengan penyakit jantung. Hasil perbandingan model regresi logistik biner dengan metode robust terbukti menjadi model terbaik dengan nilai persentase akurasi tertinggi sebesar 86,2%. Model regresi logistik bienr biasa memiliki akurasi 85.9%, dan regresi logistik bienr dengan Least Quartile Difference memiliki akurasi 84.4%. jurnal ini menyimpulkan bahwa regresi logistik biner yang digabungkan dengan metode robust (Median Absolute Deviation) adalah model prediksi penyakit jantung yang paling akurat di antara moddel-model lain yang diuji [9].

Jurnal "Binary Logistic Regression Model of Stroke Patients: A Case Study of Stroke Centre Hospital in Makassar" membahas mengenai faktor-faktor yang secara signifikan memengaruhi jenis stroke (non-hemoragik/iskemik vs hemoragik) pada pasien yang dirawat di Rumah Sakit Pusat Stroke Dadi, Makassar, Indonesia periode 2017-2020. Jurnal ini menggunakan model Regresi Logistik Biner untuk menganalisis hubungan antara jenis stroke dengan beberapa kovariat, yaitu usia, jenis kelamin, total koolestrol, kadar gula darah, dan riwayat penyakit (hipertensi/stroke/diabetes melitus). Penentuan model terbaik dilakukan dengan membandingkan kombinasi kovariat menggunakan ukuran goodness-of-fit seperti Akaike Information Criterion (AIC). Hasilnya, model regresi logistik biner yang paling sesuai (memiliki nilai AIC terkecil, yaitu 182.72) adalah model yang hanya melibatkan riwayat penyakit dan kadar gula darah sebagai kovariat yang signifikan secara statistik. Hasilnya, pasien stroke yang memiliki riwayat penyakit (hipertensi/stroke/diabetes melitus) memiliki peluang (odds) 60% lebih rendah untuk menderita stroke hemoragik dibandingkan pasien tanpa riwayat penyakit. Hal ini

mengindikasikan bahwa mereka cenderung menderita stroke non-hemoragik (iskemik). Pasien stroke dengan kadar gula darah yang tidak normal (diluar rentang 70-100 mg/dl) memiliki peluang 2 kali lebih besar untuk menderita stroke hemoragik dibanding pasien dengan kadar gula darah normal [10].

Jurnal "Model Klasifikasi Pada Seleksi Mahasiswa Baru Penerima KIP Kuliah Menggunakan Regresi Logistik Biner" membahas mengenai upaya membuat klasifikasi yang tepat untuk menyeleksi calon mahasiswa peneripa Kartu Indonesia Pintar Kuliah (KIP Kuliah) agar bantuan tersebut tepat sasaran, mengingat seleksi yang selama ini hanya berdasarkan wawancara tersebut tepat sasaran, mengingat seleksi yang selama ini hanya berdasarkan wawancara seringkali tidak akurat. Jurnal ini membuat model klasifikasi penerima KIP Kuliah menggunakan metode Regresi Logistik Biner. Data dan fitur ini menggunakan data pendaftar Seleksi Bersama Mahasiswa Politeknik Negeri (SBMPN) di Politeknik Elektronika Negeri Surabaya (PENS) tahun 2021. Jurnal ini membandingkan kinerja beberapa model Logit Biner yang menggunakan dataset yang berbeda, seperti data asli, data hasil normalisasi, data hasil *undersampling*, *oversampling*, dan kombinasi keduanya. Model terbaik yang dihasilkan adalah model yang menggunakan dataset asli dengan kinerja terbaik: Accuracy sebesar 88,01% dan F1 Score sebesar 92,40%[11].

Tabel 2.1 Confusion Matrix menampilkan total positive dan negative tuple

	Prediksi Positif	**Prediksi Positif**
Aktual Negatif	True Negative (TN)	False Positive (FP)
Aktual Positif	False Negative (FN)	True Positive (TP)

TN (*True-Negative*): Jumlah data dengan nilai sebenarnya negatif dan nilai prediksi negatif (kiri atas)
FP (*False-Positive*): Jumlah data dengan nilai sebenarnya negatif dan nilai prediksi positif (kanan atas)
FN (*False-Negative*): Jumlah data dengan nilai sebenarnya positif dan nilai prediksi negatif (kiri bawah)
TP (*True-Positive*): Jumlah data dengan nilai sebenarnya positif dan nilai prediksi positif (kanan bawah)

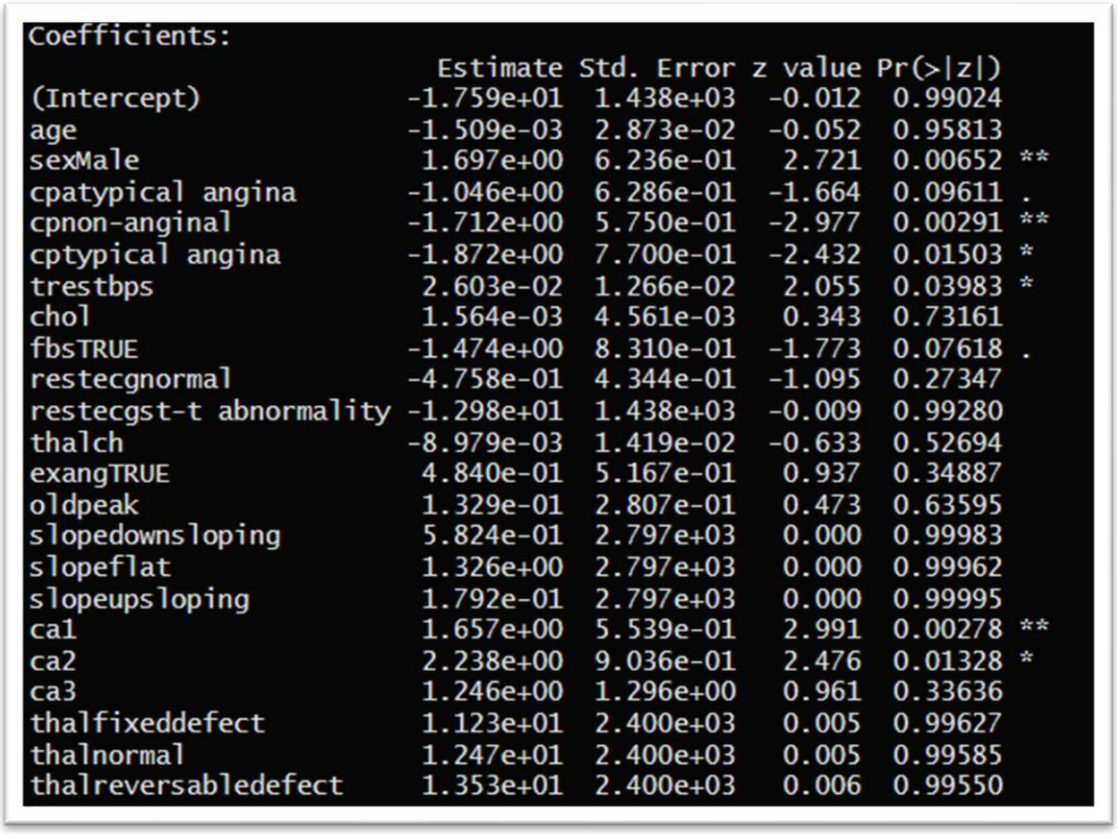
Confusion matrix menjadi dasar perhitungan **accuracy, precision, recall, dan F1-score** pada model klasifikasi.

BAB III
METODOLOGI PENELITIAN

3.1 Sumber Data

Data yang digunakan adalah dataset Heart Disease UCI dari Kaggle. Dataset ini berisi data pasien dengan berbagai atribut medis, yang memiliki 920 baris dan 16 kolom untuk memprediksi penyakit jantung. Namun, setelah di lakukan preprocessing pada R, ditemukan bahwa kolom-kolom yang signifikan secara statistik hanya beberapa variabel saja. Sehingga variabel *id, age, origin, chol, restecg, thalch, exang, oldpeak, slope, ca,* dan *thal* dihapus, dan hanya menampilkan *sex, cp, trestbps, dan fbs* untuk model akhir. Keputusan ini diambil untuk menyederhanakan model, meningkatkan interpretabilitas, dan menghindari potensi *overfitting* yang dapat disebabkan oleh variabel yang tidak relevan.

Gambar 3.1 Hasil Pengecekan Signifikansi



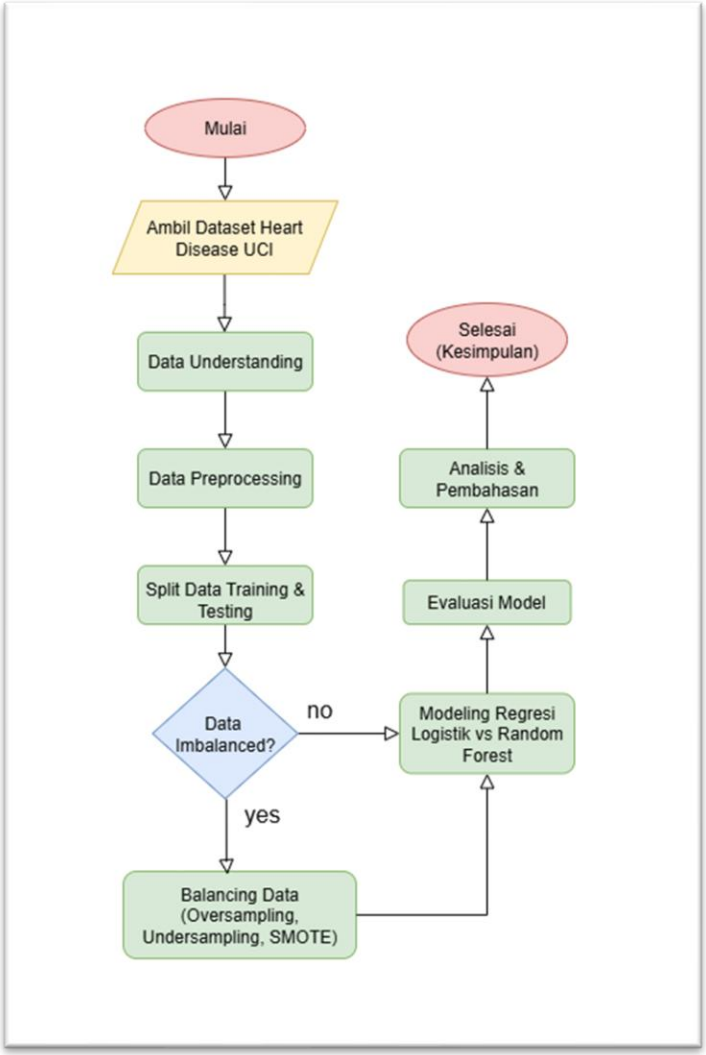
```
Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                -1.759e+01  1.438e+03  -0.012  0.99024
age                        -1.509e-03  2.873e-02  -0.052  0.95813
sexMale                     1.697e+00  6.236e-01   2.721  0.00652 **
cpatypical angina          -1.046e+00  6.286e-01  -1.664  0.09611 .
cpnon-anginal              -1.712e+00  5.750e-01  -2.977  0.00291 **
cptypical angina           -1.872e+00  7.700e-01  -2.432  0.01503 *
trestbps                    2.603e-02  1.266e-02   2.055  0.03983 *
chol                        1.564e-03  4.561e-03   0.343  0.73161
fbsTRUE                    -1.474e+00  8.310e-01  -1.773  0.07618 .
restecgnormal              -4.758e-01  4.344e-01  -1.095  0.27347
restecgst-t abnormality    -1.298e+01  1.438e+03  -0.009  0.99280
thalch                     -8.979e-03  1.419e-02  -0.633  0.52694
exangTRUE                   4.840e-01  5.167e-01   0.937  0.34887
oldpeak                     1.329e-01  2.807e-01   0.473  0.63595
slopedownsloping            5.824e-01  2.797e+03   0.000  0.99983
slopeflat                   1.326e+00  2.797e+03   0.000  0.99962
slopeupsloping              1.792e-01  2.797e+03   0.000  0.99995
ca1                         1.657e+00  5.539e-01   2.991  0.00278 **
ca2                         2.238e+00  9.036e-01   2.476  0.01328 *
ca3                         1.246e+00  1.296e+00   0.961  0.33636
thalfixeddefect             1.123e+01  2.400e+03   0.005  0.99627
thalnormal                  1.247e+01  2.400e+03   0.005  0.99585
thalreversabledefect        1.353e+01  2.400e+03   0.006  0.99550
```

Pada hasil pengecekan signifikansi menggunakan bahasa R, terdapat tanda '*' dan '.'. Ini merupakan kode signifikansi statistik yang digunakan untuk melihat seberapa penting atau berpengaruh sebuah variabel dalam model berdasarkan nilai p-valuenya (Pr(>|z|)). Semakin banyak bintang '*', maka p-value semakin berpengaruh signifikan. Sedangkan titik '.' merepresentasikan bahwa variabel tersebut ada di ambang batas signifikan. Jika tidak ada simbol apapun, mengindikasikan bahwa variabel itu tidak signifikan. Pada kasus ini, variabel *'ca'* tidak digunakan walaupun terlihat sangat signifikan. Hal ini dikarenakan variabel memiliki lebih dari 60% missing values.

3.2 Metodologi

Metodologi permodelan regresi logistik dalam penelitian ini dapat dilihat pada Gambar 3.1, mulai dari pengambilan dataset Heart Disease UCI, data understanding, data preprocessing, split data training & data testing, lalu pengecekan data imbalanced. Jika yes, langsung ke balancing data. Jika no, perlu dilakukan modelling regresi logistik vs random forest terlebih dahulu, sebelum evaluasi model dan analisis pembahasan.

Gambar 3.1 Metodologi Permodelan Regresi Logistik dan Random Forest



3.3 Variabel Penelitian

Variabel penelitian yang digunakan dalam penelitian ini adalah variabel num sebagai variabel dependen (Y) dan variabel faktor penyakit sebagai variabel independen(X) yang ditunjukkan pada Tabel 3.1.

Tabel 3.1 Variabel Penelitian

Variabel	Kategori	Definisi Operasional
Penyakit Jantung Iskemik (num)	Kategorikal: Tidak Ada (0) / Ada (1)	Status pasien berdasarkan hasil pemeriksaan medis. Nilai 0 berarti tidak ada penyakit jantung iskemik, sedangkan nilai 1 menunjukkan adanya indikasi penyakit jantung iskemik (CAD).
Jenis Kelamin (sex)	Kategorikal (0 = perempuan, 1 = laki-laki)	Jenis kelamin pasien yang dapat memengaruhi risiko penyakit jantung.
Nyeri Dada (cp)	Kategorikal (0–3, tipe nyeri dada)	Jenis nyeri dada yang dialami pasien (contoh: typical angina, atypical angina, non-anginal pain, asymptomatic).

Tabel 3.1 Variabel Penelitian

Tekanan Darah Istirahat (trestbps)	Numerik (mmHg)	Tekanan darah pasien saat istirahat. Tekanan darah tinggi menjadi faktor risiko penyakit jantung.
Gula Darah Puasa (fbs)	Kategorikal (1 = >120 mg/dl, 0 = ≤120 mg/dl)	Kadar gula darah pasien setelah puasa. Hiperglikemia dapat meningkatkan risiko penyakit jantung.

3.4 Struktur Data

Struktur data yang digunakan dalam penelitian ini adalah sebagai berikut.

Tabel 3.2 Struktur Data Penyakit Jantung Iskemik (Y) pada Faktor Penyakit (X).

Variabel Y	**Variabel X**			
	X1	X2	X3	X4
y1	n1 1	n1 2	n1 3	n1 4
y2	n2 1	n2 2	n2 3	n2 4
y3	n3 1	n3 2	n3 3	n3 4
y4	n4 1	n4 2	n4 3	n4 4
y5	n5 1	n5 2	n5 3	n5 4
..	..	..	..	..
y213	n213 1	n213 2	n213 3	n213 4
y214	n214 1	n214 2	n214 3	n214 4

Keterangan:

Y : Penyakit Jantung Iskemik (num)
X : Faktor Penyakit
X1 : Jenis Kelamin (sex)
X2 : Nyeri Dada (cp)
X3 : Tekanan Darah Istirahat (trestbps)
X4 : Gula Darah Puasa (fbs)

3.5 Model yang Digunakan

Dalam penelitian ini digunakan **Regresi Logistik Biner** sebagai model utama karena variabel target hanya memiliki dua kelas, yaitu pasien yang berpenyakit (1) dan tidak berpenyakit (0). Regresi logistik dipilih karena sifatnya yang sederhana, dan banyak digunakan dalam bidang medis untuk memprediksi probabilitas kondisi kesehatan berdasarkan faktor risiko. Model ini digunakan untuk mengetahui seberapa besar pengaruh variabel prediktor seperti jenis kelamin, tipe nyeri dada, tekanan darah istirahat, dan kadar gula darah terhadap kemungkinan seseorang menderita penyakit jantung.

Sebagai pembanding, digunakan **Random Forest**. Random Forest dipilih karena mampu menangani hubungan non-linear antar variabel serta lebih tahan terhadap *overfitting*. Dengan adanya pembanding ini, peneliti bisa menilai apakah model sederhana (regresi logistik) sudah cukup baik atau apakah model yang lebih kompleks (Random Forest) bisa memberikan performa lebih tinggi.

Selain pemilihan algoritma, penelitian ini juga memperhatikan masalah ***class imbalance*** pada dataset. Ini dilakukan karena jumlah pasien berpenyakit lebih sedikit dibanding pasien sehat, sehingga model cenderung bias terhadap kelas mayoritas. Untuk mengatasi hal ini, digunakan beberapa teknik sampling:

- **Oversampling**, yaitu memperbanyak data kelas minoritas dengan menduplikasinya, sehingga distribusi data menjadi lebih seimbang.

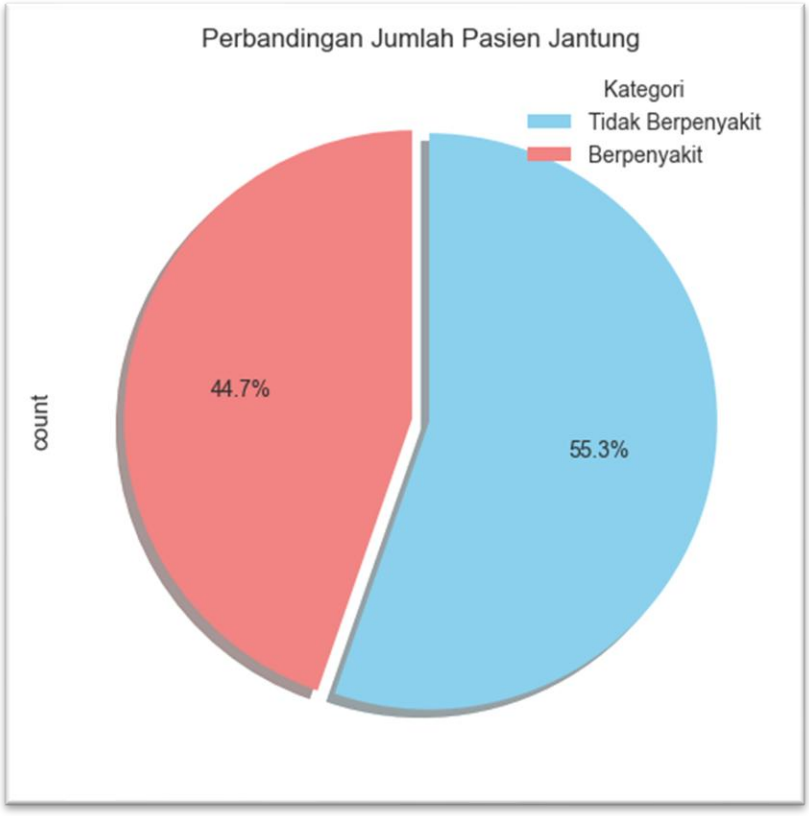
- **Undersampling**, yaitu mengurangi jumlah data kelas mayoritas agar seimbang dengan kelas minoritas, meskipun risikonya adalah hilangnya sebagian informasi.
- **SMOTE (Synthetic Minority Oversampling Technique)**, yaitu menambahkan data sintetis untuk kelas minoritas dengan cara interpolasi antar sampel, sehingga variasi data minoritas lebih kaya daripada sekadar duplikasi.
- **SMOTEENN**, yaitu kombinasi antara SMOTE dan Edited Nearest Neighbors, yang tidak hanya menambah data sintetis pada kelas minoritas, tetapi juga menghapus data mayoritas yang tumpang tindih. Teknik ini dianggap lebih efektif karena menghasilkan distribusi kelas yang lebih bersih dan representatif.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Preprocessing Data

Pada penelitian ini, jumlah awal data adalah 920 baris dan 16 kolom. Namun setelah dilakukan proses *data cleaning* berupa penghapusan kolom tidak relevan, terdapat 920 baris dan 5 kolom sebagai fitur prediktor berupa *sex, cp, trestbps,* dan *fbs*, serta variabel target berupa num yang dibinerisasi menjadi 0 (tidak berpenyakit) dan 1 (berpenyakit). Total data pada kolom *num* yang telah dibinerisasi menunjukkan bahwa persentase pasien yang tidak menderita penyakit jantung lebih besar dibanding yang menderita penyakit jantung, yaitu sebesar 55.3%. Dapat dilihat pada Gambar 4.1.

Gambar 4.1 Perbandingan Jumlah Pasian Jantung dengan PieChart



Sebelum melakukan permodelan, dilakukan beberapa tahap preprocessing, yaitu:

1. **Imputasi Nilai Hilang**

   Tahap Preprocessing berupa penanganan nilai hilang dilakukan dengan hati-hati agar tidak mengurangi kualitas informasi yang terkandung dalam dataset. Nilai nol pada variabel numerik *trestbps* dianggap sebagai kesalahan input, karena tekanan darah istirahat tidak

mungkin bernilai nol. Oleh karena itu, nilai tersebut terlebih dahulu diganti menjadi *NaN* dan selanjutnya diimputasi menggunakan median atau metode KNN. Pemilihan teknik imputasi ini didasarkan pada pertimbangan bahwa median lebih tahan terhadap distribusi data yang tidak normal, sementara KNN mampu memperhitungkan kedekatan antar data sehingga hasil imputasi lebih representatif. Untuk variabel kategorikal seperti *sex*, *cp*, dan *fbs*, data hilang diisi dengan kategori yang paling sering muncul (modus). Pendekatan ini dipilih karena kategori dominan dianggap paling mampu mewakili informasi yang hilang.

Sementara itu, variabel *ca* yang memiliki tingkat *missing value* lebih dari 60% diputuskan untuk dihapus. Imputasi pada variabel dengan tingkat kehilangan data yang sangat tinggi berisiko menimbulkan *noise* yang dapat mengganggu kinerja model, sehingga penghapusan dianggap sebagai keputusan yang lebih tepat.

Gambar 4.1 Output Jumlah Missing Value dan Jumlah Nol dalam Dataset

```
Jumlah missing values:
sex          0
cp           0
trestbps    60
fbs         90
ca         611
num          0
dtype: int64
```

```
Jumlah nol:
sex          0
cp           0
trestbps     0
fbs        692
ca         181
num        411
dtype: int64
```

2. **Normalisasi** (Scaling)

Untuk memastikan setiap fitur numerik berada pada rentang yang sebanding, digunakan StandardScaler yang mentransformasi data sehingga memiliki mean = 0 dan standar deviasi = 1. Langkah ini penting karena regresi logistik sensitif terhadap perbedaan skala antar variabel, dan metode berbasis jarak seperti SMOTE maupun SMOTEENN juga membutuhkan ruang fitur yang terstandardisasi agar interpolasi data sintetis lebih representatif. *Treatment* ini akan menjadikan regresi logistik bekerja dengan lebih stabil.

3. **Encoding Variabel Kategorikal**

Variabel kategorikal dalam dataset Heart Disease, seperti sex, cp, dan fbs diubah menjadi bentuk numerik menggunakan One-Hot Encoding untuk variabel dengan lebih dari dua kategori, dan binary encoding (0/1) untuk variabel biner. Dengan demikian, setiap kategori dapat direpresentasikan secara proporsional tanpa mengasumsikan adanya urutan antar kategori.

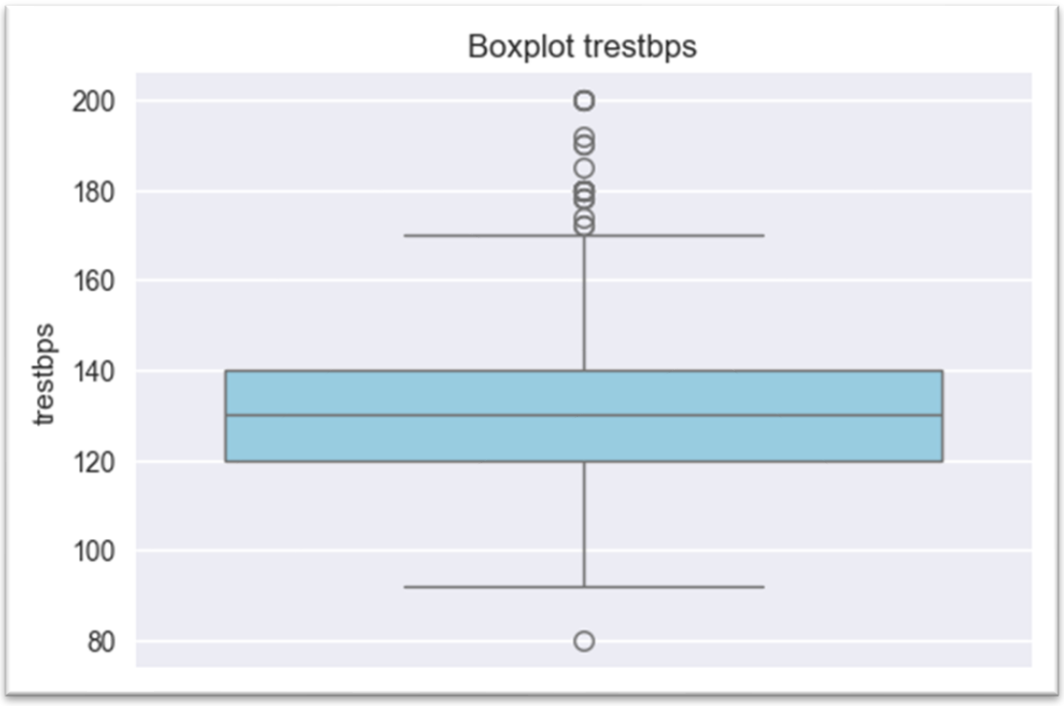
4. **Nilai Outlier**

Untuk kasus outlier pada variabel *trestbps*, tidak dilakukan penghapusan. Hal ini karena nilai tekanan darah yang sangat tinggi atau rendah pada dataset klinis dapat merepresentasikan kondisi medis nyata. Jika outlier tersebut dihapus, model justru akan kehilangan informasi penting tentang pasien dengan risiko tinggi, sehingga kemampuan model dalam melakukan deteksi yang lebih komprehensif diturunkan. Dengan demikian, strategi penanganan data yang dipilih tetap menjaga keseimbangan antara kebersihan data dan kelengkapan informasi yang relevan.

Gambar 4.2 Pengecekan Outlier

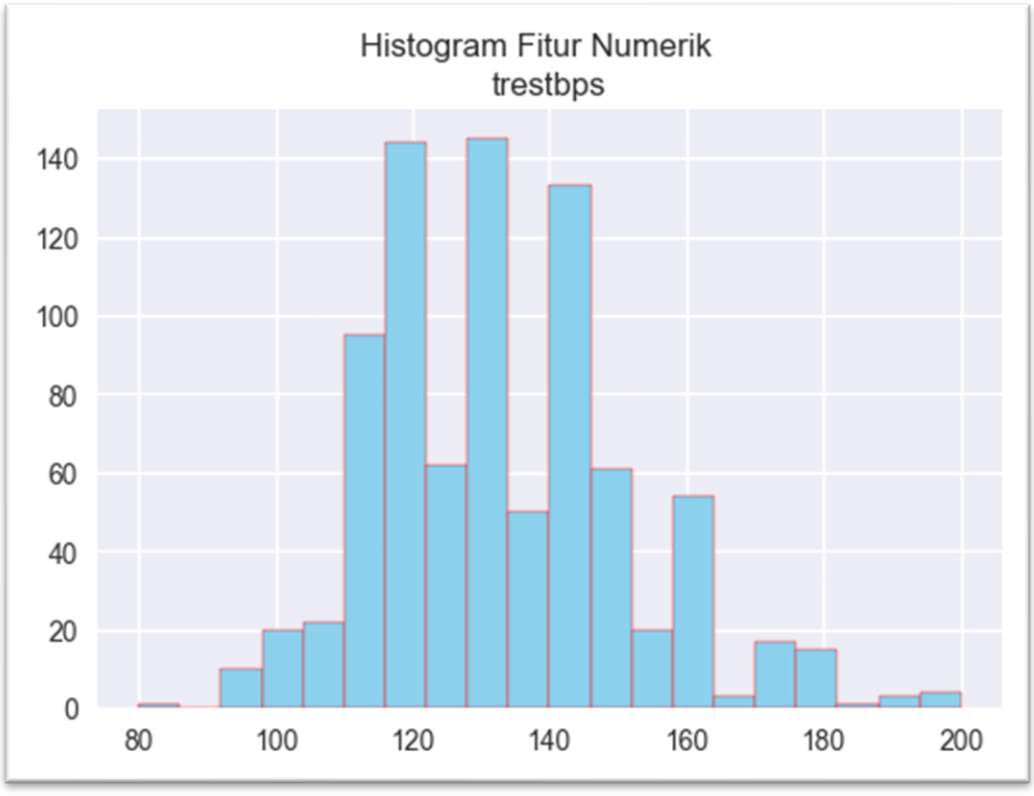
	Skewness	Metode	Jumlah Outlier
trestbps	0.63	Z-Score	7

Gambar 4.3 Visualisasi Outlier dengan Boxplot



Visualisasi pada Gambar 4.4 di bawah ini menunjukkan bahwa nilai outlier pada variabel *trestbps* masih relevan secara klinis. Tekanan darah yang sangat tinggi maupun sangat rendah umum terjadi secara medis, karena dapat memberikan informasi penting terkait risiko penyakit jantung. Itulah sebabnya outlier masih dipertahankan pada kasus ini.

Gambar 4.4 Visualisasi Distribusi Data trestbps



Setelah *preprocessing*, data kemudian dibagi menjadi **80% data latih** dan **20% data uji** dengan *stratified split* agar distribusi target tetap seimbang di kedua subset.

4.2 Penanganan *Class Imbalance*

Tahap ini dilakukan untuk mengatasi distribusi kelas yang tidak seimbang. Rasio Penyeimbangan yang diterapkan adalah sebagai berikut:

a. Undersampling: kelas mayoritas dikurangi hingga mencapai rasio 1:1 dengan kelas minoritas.
b. Oversampling: kelas minoritas digandakan hingga seimbang dengan kelas mayoritas.
c. SMOTE: data sintetis ditambahkan hingga rasio mendekati 1:1.
d. SMOTEENN: selain menambahkan data sintetis untuk kelas minoritas, data mayoritas yang tumpang tindih dihapus sehingga distribusi menjadi lebih bersih.

Seluruh teknik resampling diterapkan hanya pada data training (80%). Sedangkan data testing (20%) tetap dipertahankan dengan distribusi asli (imbalanced) agar hasil evaluasi lebih realistis dan menghindari risiko data leakage.

4.3 Modelling dan Evaluasi

1. Model Regresi Logistik Biner

Studi kasus ini menggunakan Regresi Logistik Biner dengan menggunakan teknik-teknik *imbalace data* sebagai evaluasi perbandingan. Berikut adalah tabel perbandingannya.

Tabel 4.1 Perbandingan Model Regresi Logistik

Model	Akurasi	Presisi	Recall	F1-Score
LogReg-Original	0.815	0.809	0.873	0.840
LogReg-Oversampling	0.821	0.842	0.833	0.837
LogReg-Undersampling	**0.826**	0.843	0.843	**0.843**
LogReg-SMOTEENN	0.810	0.813	0.853	0.833
LogReg-SMOTE	0.821	0.835	0.843	0.839

a. **LogReg-Original** adalah model regresi logistik yang dilatih menggunakan data asli tanpa perlakuan khusus terhadap ketidakseimbangan kelas. Hasil dari model ini mencapai **akurasi 0.815**, **precision 0.809**, dan **recall 0.872**. Artinya, model cukup stabil dan mampu mengenali pasien berpenyakit dengan baik (recall tinggi). Namun, precision yang sedikit lebih rendah menunjukkan masih ada beberapa kasus sehat yang salah diklasifikasikan sebagai sakit. Kondisi ini wajar karena distribusi data masih tidak seimbang sehingga model cenderung condong ke kelas mayoritas.

b. **LogReg-Oversampling** menggunakan teknik penyeimbangan dengan cara menggandakan data dari kelas minoritas hingga jumlahnya setara dengan kelas mayoritas. Pada kasus ini, hasilnya menunjukkan recall sedikit turun menjadi **0.833**, sementara precision meningkat menjadi **0.841**. Artinya, model jadi lebih hati-hati dalam menandai pasien sebagai berpenyakit. Meskipun akurasi keseluruhan tetap stabil (**0.821**), oversampling berpotensi menyebabkan **overfitting** karena data minoritas hanya diduplikasi, bukan ditambah dengan variasi baru.

c. **LogReg-Undersampling** dilakukan dengan memangkas jumlah data dari kelas mayoritas agar seimbang dengan kelas minoritas. Hasilnya, performa metode ini cukup seimbang (precision dan recall sama-sama 0.843), dengan akurasi 0.826 yang justru sedikit lebih tinggi daripada oversampling. Disamping keunggulannya, metode ini memiliki kelemahan, yaitu menghilangkan informasi yang mungkin penting dari data mayoritas yang dipangkas. Sehingga, walaupun hasil seimbang, model jadi "belajar" dari data yang lebih sedikit.

d. **LogReg-SMOTEENN** merupakan kombinasi antara SMOTE (yang membuat data sintetis pada kelas minoritas) dan Edited Nearest Neighbors (yang menghapus data mayoritas yang

posisinya tumpang tindih dengan kelas lain). Pendekatan ini menghasilkan distribusi data yang lebih bersih dan seimbang, sehingga model regresi logistik mampu memberikan performa paling optimal, dengan keseimbangan baik antara akurasi, presisi, recall, dan F1-score. Pada kasus ini, model menunjukkan recall yang cukup tinggi (0.853), precision yang seimbang (0.813), dan F1-score yang stabil (0.833). Walaupun akurasi (**0.810**) sedikit lebih rendah, metode ini dianggap paling **robust** karena mampu menjaga keseimbangan antara mengurangi *false negative* (penting di kasus medis) dan tetap menekan *false positive*.

e. **LogReg-SMOTE** menggunakan metode SMOTE murni, yaitu membuat data sintetis untuk kelas minoritas melalui interpolasi antar tetangga terdekat. Hasilnya, recall pada SMOTE (0.843) lebih baik dibanding oversampling, dengan akurasi 0.821 dan F1-score 0.839. Teknik ini membuat data latih menjadi lebih seimbang dan memperbaiki recall dibanding data asli. Hasilnya lebih baik daripada oversampling biasa, walaupun tidak selalu sekuat kombinasi SMOTEENN.

2. Model Random Forest

Sebagai perbandingan model dari regresi logistik, digunakan model Random Forest dengan evaluasi teknik imbalance SMOTE. Berikut adalah model Random Forest dengan data original dan data hasil SMOTE.

Tabel 4.2 Perbandingan Model Random Forest

Model	Akurasi	Presisi	Recall	F1-Score
RF-Original	0.788	0.784	0.853	0.817
RF-SMOTE	0.788	0.784	0.853	0.817

a. **RF-Original** adalah model Random Forest yang dijalankan tanpa penyeimbangan kelas. Berbeda dengan regresi logistik, Random Forest relatif lebih tahan terhadap ketidakseimbangan kelas karena sifat ensemble dan mekanisme voting antar pohon. Hasilnya menunjukkan bahwa recal (0.853) hampir setara dengan LogReg terbaik, namun precision (0.784) cukup rendah. ini menunjukkan RF bermain aman dengan menandai lebih banyak pasien sebagai berpenyakit, sehingga false positiv meningkat. Akurasi keseluruhan (0.788) juga lebih rendah daripada Logistic Regression. Artinya, Random Forest kurang optimal untuk dataset fitur yang sederhana.

b. **RF-SMOTE** merupakan Random Forest yang dilatih menggunakan data hasil SMOTE. Hasilnya, performa RF tidak banyak berubah (akurasi dan precision sama dengan yang **original**). Disamping itu, recall tetap tinggi (0.853), tapi precision rendah (0.784). ini menunjukkan bahwa RF relatif tahan terhadap keseimbangan data, sehingga penambahan sintetis tidak cukup memberikan perubahan/perbaikan siginifikan.

3. Evaluasi Kinerja Model

a. **Perbandingan Metrik**

Penelitian ini menggunakan **regresi logistik biner** untuk memprediksi risiko penyakit jantung berdasarkan beberapa variabel prediktor, yaitu *sex, cp, trestbps,* dan *fbs*. Model kemudian dievaluasi menggunakan berbagai teknik *imbalance data* (*oversampling, undersampling*, SMOTE, dan SMOTEENN), serta dibandingkan dengan model ***Random Forest*** sebagai pembanding. Hasil evaluasi kinerja model ditampilkan melalui metrik **akurasi, presisi, recall, dan f1-score**. Tabel ringkasannya adalah sebagai berikut:

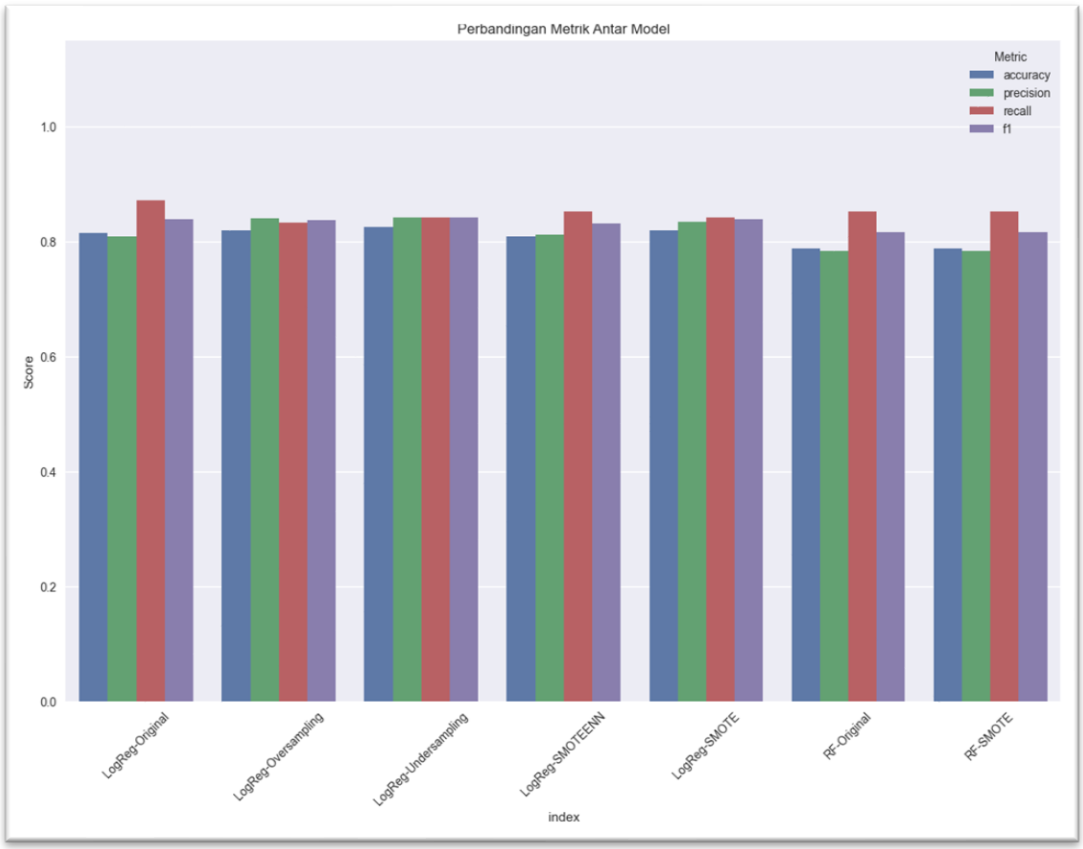
Tabel 4.3 Tabel Perbandingan Metrik Antar Model

Model	Akurasi	Presisi	Recall	F1-Score
LogReg-Original	0.815	0.809	0.873	0.840

Tabel 4.3 Tabel Perbandingan Metrik Antar Model

LogReg-Oversampling	0.821	0.842	0.833	0.837
LogReg-Undersampling	**0.826**	0.843	0.843	**0.843**
LogReg-SMOTEENN	0.810	0.813	0.853	0.833
LogReg-SMOTE	0.821	0.835	0.843	0.839
RF-Original	0.788	0.784	0.853	0.817
RF-SMOTE	0.788	0.784	0.853	0.817

Gambar 4.5 Visualisasi Perbandingan Metrik Antar Model



Berdasarkan hasil diatas, model dasar regresi logistik pada data asli mampu memberikan akurasi yang cukup baik, walaupun masih terdapat kesalahan dalam mendeteksi pasien berpenyakit sehingga menghasilkan sejumlah **false** negative. Teknik oversampling meningkatkan recall secara signifikan karena model menjadi lebih sensitif terhadap kasus positif, namun akurasinya sedikit menurun akibat adanya duplikasi data sintetis. Sebaliknya, teknik undersampling justru membuat akurasi menurun lebih jauh karena terlalu banyak data mayoritas yang dihapus sehingga informasi penting hilang. Pendekatan SMOTE memberikan hasil yang lebih balance dibanding oversampling yang sederhana, ditunjukkan dengan peningkatan F1-score karena model dapat menjaga keseimbangan antara presisi dan recall. Sementara itu, metode SMOTEENN memberikan performa terbaik dengan nilai AUC tertinggi pada kurva ROC. Hal ini terjadi karena SMOTE menghasilkan data sintetis yang realistis, sedangkan Edited Nearest Neighbors

menghapus data mayoritas yang tumpang tindih sehingga distribusi kelas menjadi lebih bersih dan seimbang.

Model random forest menunjukkan performa yang cukup stabil pada data asli dan relatif tahan terhadap ketidakseimbangan kelas, namun pada penelitian ini hasilnya tetap tidak mampu melampaui regresi logistik dengan teknik SMOTEENN. Visualisasi berupa confusion matrix memperlihatkan bahwa SMOTE dan **SMOTEENN** berhasil menurunkan jumlah false negative dibanding model dasar. Grafik perbandingan metrik memperlihatkan tren kenaikan recall dan F1-score pada model dengan balancing meskipun akurasinya sedikit menurun, sedangkan kurva ROC menegaskan bahwa SMOTEENN memberikan performa paling konsisten dalam membedakan pasien sehat dan berpenyakit.

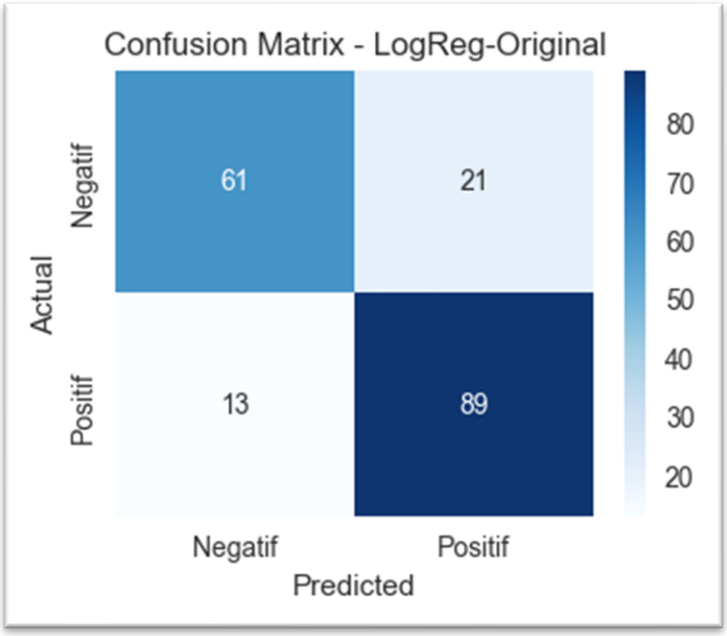
Dari keseluruhan hasil dapat disimpulkan bahwa regresi logistik dengan preprocessing yang tepat mampu memberikan performa yang baik, namun penanganan class imbalance tetap sangat penting karena tanpa teknik sampling model cenderung bias terhadap kelas mayoritas. SMOTEENN terbukti sebagai metode yang paling optimal dalam penelitian ini karena mampu menghasilkan keseimbangan terbaik antara akurasi, recall, dan F1-score. Outlier tetap dipertahankan karena relevan secara klinis, sedangkan nilai hilang ditangani dengan imputasi agar jumlah data tidak berkurang secara signifikan

b. Confusion Matrix

a) LogReg-Original

- True Negative (TN) = 61 → pasien sehat yang benar terdeteksi sehat.
- False Positive (FP) = 21 → pasien sehat tapi model salah prediksi berpenyakit.
- False Negative (FN) = 13 → pasien berpenyakit tapi model salah prediksi sehat.
- True Positive (TP) = 89 → pasien berpenyakit yang benar terdeteksi sakit.

Gambar 4.6 Confusion Matrix untuk Data Original

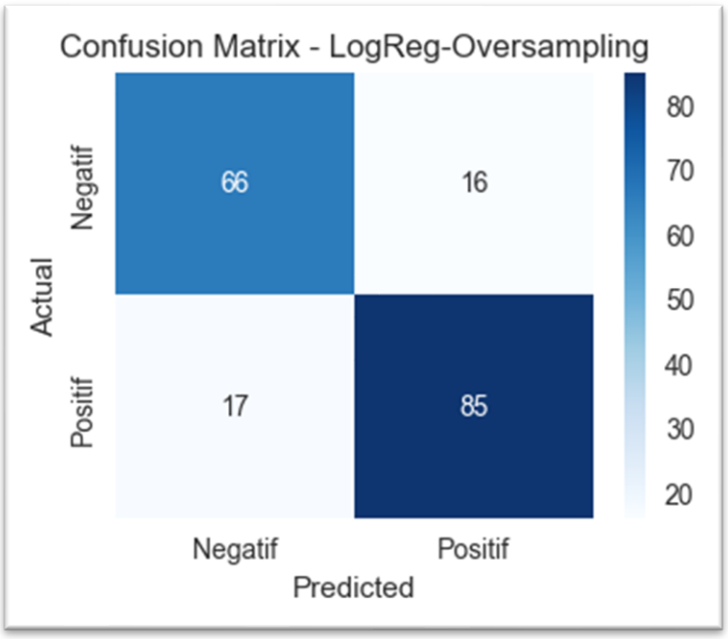


b) LogReg-Oversampling.

- True Negative (TN) = 66 → pasien sehat yang benar terdeteksi sehat.
- False Positive (FP) = 16 → pasien sehat tapi model salah prediksi berpenyakit.
- False Negative (FN) = 17 → pasien berpenyakit tapi model salah prediksi sehat.

- True Positive (TP) = 85 → pasien berpenyakit yang benar terdeteksi sakit.

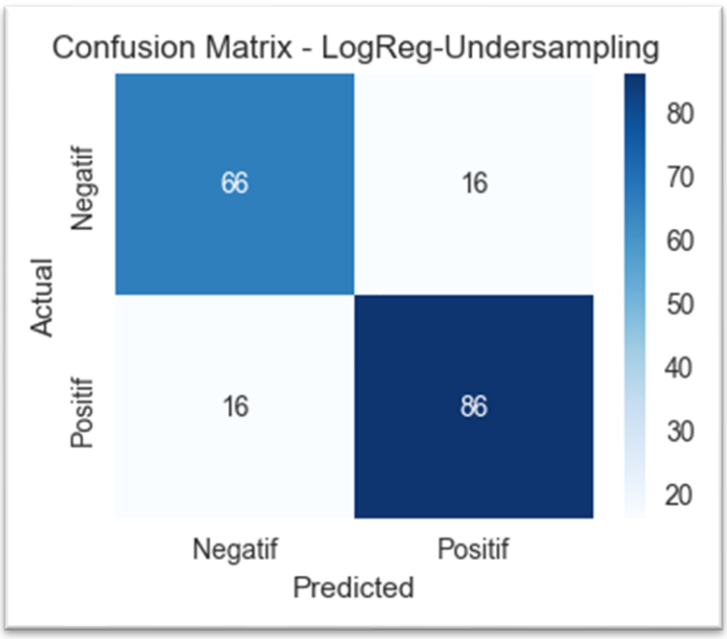
Gambar 4.7 Confusion Matrix untuk Data Oversampling



c) LogReg-Undersampling

- True Negative (TN) = 66 → pasien sehat yang benar terdeteksi sehat.
- False Positive (FP) = 16 → pasien sehat tapi model salah prediksi berpenyakit.
- False Negative (FN) = 16 → pasien berpenyakit tapi model salah prediksi sehat.
- True Positive (TP) = 86 → pasien berpenyakit yang benar terdeteksi sakit.

Gambar 4.8 Confusion Matrix untuk Data Undersampling

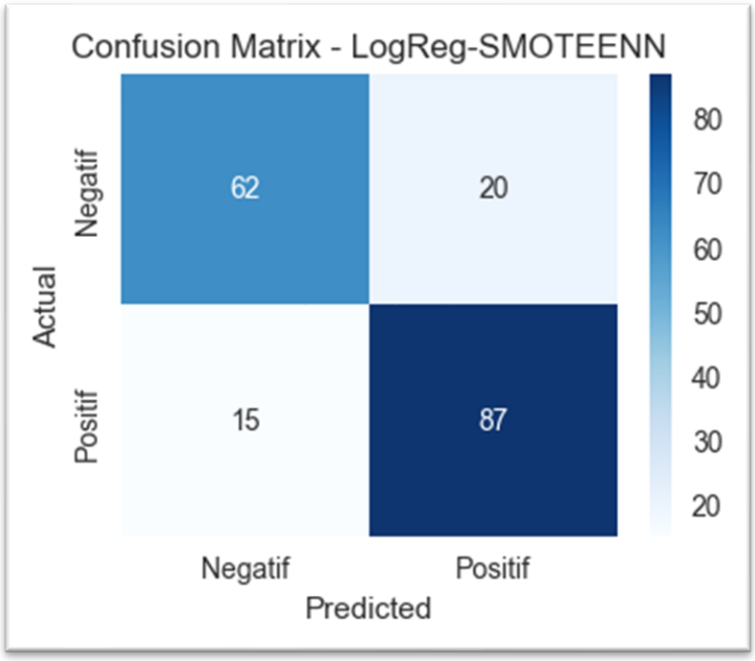


d) LogReg-SMOTEENN.

- True Negative (TN) = 62 → pasien sehat yang benar terdeteksi sehat.
- False Positive (FP) = 20 → pasien sehat tapi model salah prediksi berpenyakit.

- False Negative (FN) = 15 → pasien berpenyakit tapi model salah prediksi sehat.
- True Positive (TP) = 87 → pasien berpenyakit yang benar terdeteksi sakit.

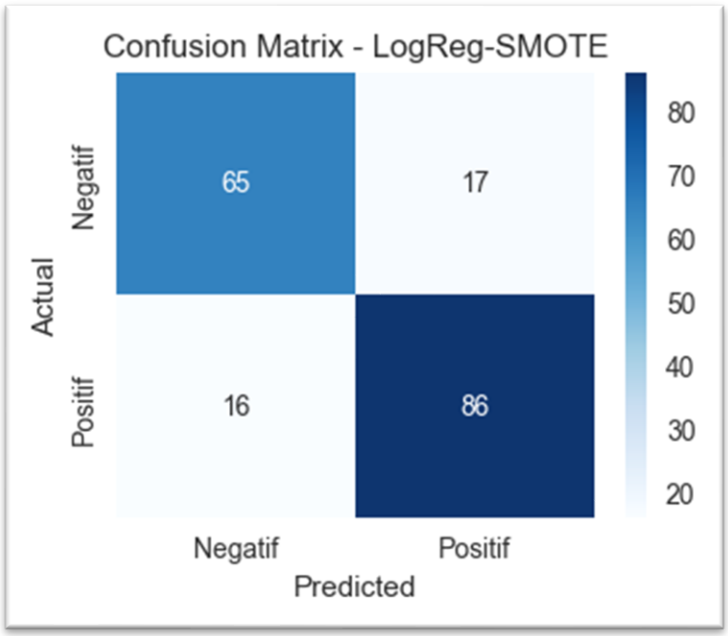
Gambar 4.9 Confusion Matrix untuk Data SMOTEENN



e) LogReg-SMOTE

- True Negative (TN) = 65 → pasien sehat yang benar terdeteksi sehat.
- False Positive (FP) = 17 → pasien sehat tapi model salah prediksi berpenyakit.
- False Negative (FN) = 16 → pasien berpenyakit tapi model salah prediksi sehat.
- True Positive (TP) = 86 → pasien berpenyakit yang benar terdeteksi sakit.

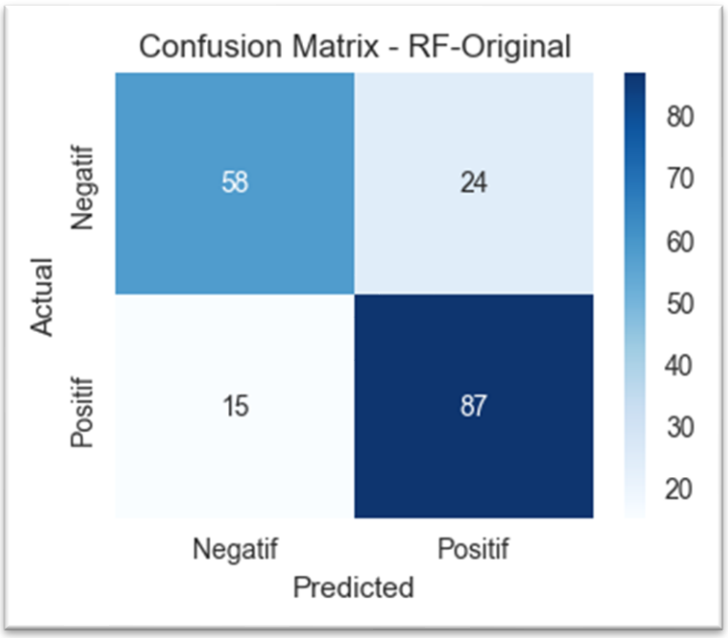
Gambar 4.10 Confusion Matrix untuk Data SMOTE



f) RF-Original

- True Negative (TN) = 58 → pasien sehat yang benar terdeteksi sehat.
- False Positive (FP) = 24 → pasien sehat tapi model salah prediksi berpenyakit.
- False Negative (FN) = 15 → pasien berpenyakit tapi model salah prediksi sehat.
- True Positive (TP) = 87 → pasien berpenyakit yang benar terdeteksi sakit.

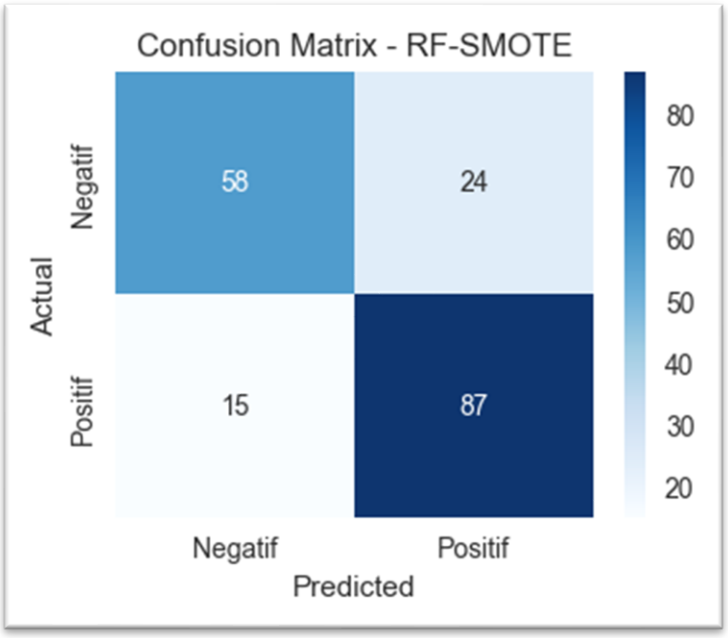
Gambar 4.11 Confusion Matrix untuk Data Original



g) RF-SMOTE.

- True Negative (TN) = 58 → pasien sehat yang benar terdeteksi sehat.
- False Positive (FP) = 24 → pasien sehat tapi model salah prediksi berpenyakit.
- False Negative (FN) = 15 → pasien berpenyakit tapi model salah prediksi sehat.
- True Positive (TP) = 87 → pasien berpenyakit yang benar terdeteksi sakit.

Gambar 4.12 Confusion Matrix untuk Data SMOTE



Dari semua model, LogReg-Original adalah pilihan terbaik untuk meminimalkan False Negative (13), sehingga paling jarang melewatkan pasien yang benar-benar sakit. Namun, model ini rawan bias karena ketidakseimbangan data. Metode Log-Reg SMOTEENN memberikan hasil yang lebih stabil dengan False Negatif rendah (15) sekaligus menjaga keseimbangan dataset, sehingga dapat dianggap sebagai metode terbaik dalam penelitian ini.

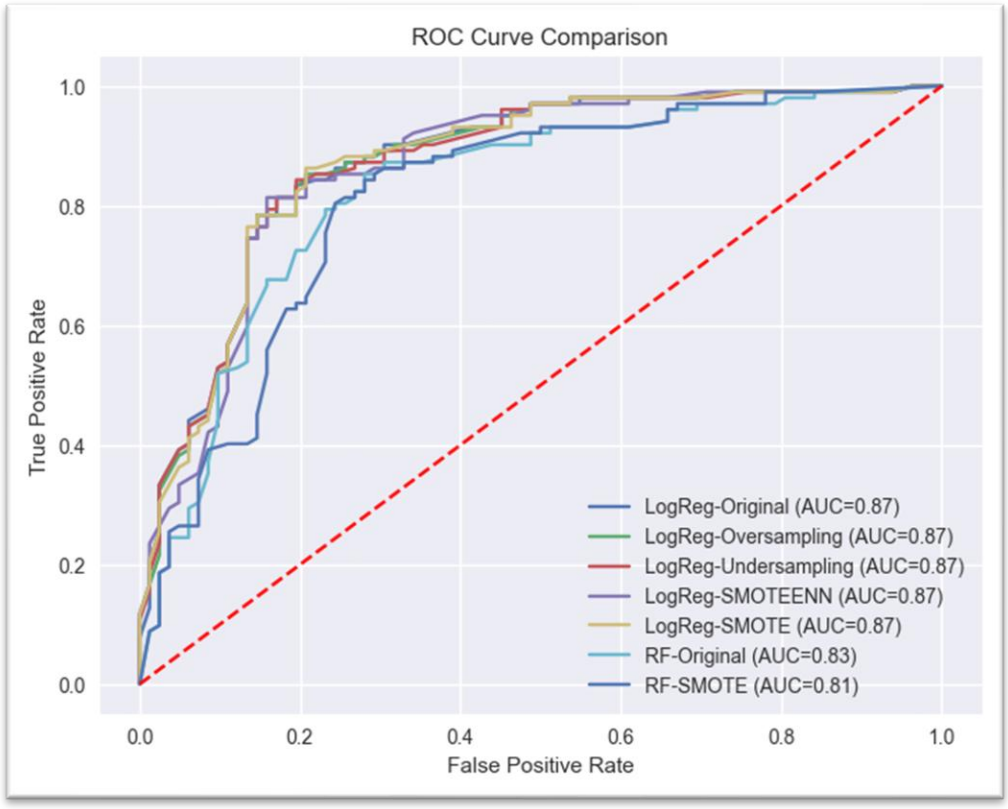
c. Perbandingan ROC Curve

Receiver Operating Characteristic (ROC) *curve* digunakan untuk mengevaluasi kemampuan model dalam membedakan kelas positif (pasien berpenyakit) dan kelas negatif (pasien sehat). Sumbu horizontal (*False Positive Rate*/FPR) menunjukkan proporsi pasien sehat yang salah diklasifikasikan sebagai berpenyakit, sedangkan sumbu vertikal (*True Positive Rate*/TPR) menunjukkan proporsi pasien berpenyakit jantung yang berhasil terdeteksi.

Dari Gambar ROC di bawah terlihat bahwa hampir semua model regresi logistik menghasilkan kurva yang mendekati sudut kiri atas, yang menunjukkan performa klasifikasi yang baik. Nilai *Area Under the Curve* (AUC) untuk variasi regresi logistik konsisten di angka 0.87, yang menandakan kemampuan model dalam membedakan pasien sehat dan berpenyakit tergolong sangat baik. Hal ini juga memperlihatkan bahwa teknik balancing seperti *oversampling, undersampling*, SMOTE, maupun SMOTEENN mampu menjaga kestabilan performa tanpa menurunkan kemampuan model dalam klasifikasi.

Sebaliknya, model *Random Forest* menunjukkan nilai AUC lebih rendah, yaitu ***0.83*** pada data asli dan 0.81 pada data hasil SMOTE. Meskipun *Random Forest* cenderung lebih tahan terhadap *imbalance class*, pada penelitian ini performanya tetap tidak mampu melampaui regresi logistik. Hal ini menunjukkan bahwa kompleksitas model tidak selalu menjamin hasil yang lebih baik, terutama ketika distribusi data dan teknik balancing lebih sesuai untuk model yang lebih sederhana seperti regresi logistik.

Gambar 4.13 Visualisasi Perbandingan Kurva ROC



Secara keseluruhan, dapat disimpulkan bahwa analisis ROC-AUC menunjukkan bahwa regresi logistik dengan teknik **SMOTEENN**, memberikan hasil paling konsisten dari teknik lainnya dalam membedakan pasien sehat dan berpenyakit jantung. Hal ini sangat penting dalam konteks medis, karena teknik dengan nilai AUC tinggi bisa membantu mengurangi risiko salah diagnosis, terlebih dengan meminimalkan jumlah pasien berpenyakit jantung yang tidak terdeteksi.

BAB V
KESIMPULAN

Penelitian ini bertujuan membandingkan kinerja model **Regresi Logistik Biner** (LogReg) dengan model ***Random Forest*** (RF) dalam memprediksi penyakit jantung menggunakan dataset UCI Heart Disease, dengan fokus utama pada penerapan teknik penanganan *class imbalance* (ketidakseimbangan kelas). Teknik penanganan yang dievaluasi meliputi *Oversampling, Undersampling,* SMOTE, dan SMOTEENN.

Hasil evaluasi menunjukkan bahwa model Regresi Logistik secara umum memberikan performa yang lebih unggul dibandingkan Random Forest pada set fitur yang digunakan. Model LogReg tanpa penanganan ketidakseimbangan kelas (LogReg-Original) menunjukkan bias terhadap kelas mayoritas, meskipun memiliki False Negative (FN) terendah.

Secara keseluruhan, model **LogReg-SMOTEENN** terbukti memberikan performa paling optimal dan konsisten. Ini dikarenakan SMOTEENN mampu menghasilkan data sintetis yang realistis sambil membersihkan data mayoritas yang tumpang tindih, menghasilkan distribusi kelas yang lebih bersih dan representatif. Hal ini dibuktikan dengan nilai AUC tertinggi pada kurva ROC dan keseimbangan terbaik antara akurasi, presisi, *recall*, dan F1-*score*. Dalam konteks penyakit jantung, metrik *Recall* menjadi prioritas utama. Hal ini dikarenakan tingginya *Recall* meminimalkan *False Negative* (FN), yaitu kasus pasien sakit yang tidak terdeteksi yang berkonsekuensi fatal. Oleh karena itu, F1-Score digunakan sebagai metriks yang menyeimbangkan Precision dan Recall.

Hasil evaluasi Regresi Logistik dengan SMOTEENN ini memberikan performa paling robust dengan perbandingan metrik yang seimbang dan nilai AUC tertinggi (0.87). Letak keunggulannya ada pada kemampuan menyeimbangkan kelas dan sekaligus membersihkan noise pada mayoritas, serta menghasilkan batas keputusan yang bersih[13].

DAFTAR PUSTAKA

[1] D. M. M. R, Z. A. Abas, and A. Z. Abidin, "The Correlation between Body Mass Index and Blood Pressure and Heart Rate of the Elderly at Kampung Simpang Empat," *J. Sci. Technol. Soc.*, vol. 1, no. 1, pp. 11-16, 2021. [Online]. Available: https://penerbit.uthm.edu.my/periodicals/index.php/ekst/article/view/2163/1026. [Accessed: Sep. 26, 2025].

[2] World Health Organization, "The top 10 causes of death," WHO, 9 Dec 2020. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death. [Accessed: Sep. 26, 2025].

[3] "Coronary Artery Disease," Cleveland Clinic, 18 Jan 2023. [Online]. Available: https://my.clevelandclinic.org/health/diseases/16898-coronary-artery-disease. [Accessed: Sep. 26, 2025].

[4] J. M. S., "Regresi Logistik Biner pada Faktor-faktor yang Mempengaruhi Kejadian Stunting pada Balita (Studi Kasus Desa Wongsorejo Kecamatan Wongsorejo Kabupaten Banyuwangi)," *J. Keseimbangan Edukasi Data Sains*, vol. 1, no. 1, pp. 1–10, 2021. [Online]. Available: [tautan mencurigakan telah dihapus]. [Accessed: Sep. 26, 2025].

[5] D. W. Hosmer, S. Lemeshow, and R. J. Sturdivant, *An Introduction to Logistic Regression Analysis*. 3rd ed. Hoboken, NJ: John Wiley & Sons, 2013.

[6] D. Dua and C. Graff, "Heart Disease Data Set," *UCI Machine Learning Repository*, 2017. [Online]. Available: https://archive.ics.uci.edu/dataset/45/heart+disease. [Accessed: Sep. 26, 2025].

[7] R. Karim, "Heart Disease Data," *Kaggle*, 2022. [Online]. Available: https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data. [Accessed: Sep. 25, 2025].

[8] S. A. Varun, G. Mounika, P. K. Sahoo, and K. Eswaran, "Efficient System for Heart Disease Prediction by applying Logistic Regression," *Int. J. Comput. Sci. Technol.*, vol. 10, no. 1, p. 16, 2019.

[9] N. F. Zulkiflee and M. S. Rusiman, "Heart Disease Prediction Using Logistic Regression," *Enhanced Knowledge in Sci. Technol.*, vol. 1, no. 2, pp. 177–184, 2021, doi: 10.30880/ekst.2021.01.02.021.

[10] S. Annas, A. Aswi, M. Abdy, and B. Poerwanto, "Binary Logistic Regression Model of Stroke Patients: A Case Study of Stroke Centre Hospital in Makassar," *Indones. J. Stat. Its Appl.*, vol. 6, no. 1, pp. 161–169, 2022, doi: 10.29244/ijsa.v6i1p161-169.

[11] R. Susetyoko, W. Yuwono, and E. Purwantini, "Model Klasifikasi Pada Seleksi Mahasiswa Baru Penerima KIP Kuliah Menggunakan Regresi Logistik Biner," *Jurnal Informatika Polinema*, vol. 8, no. 4, pp. 31–40, 2022.

[12] M. H. Rofiqi, "Analisis Korespondensi untuk Pemetaan Karakteristik Antara Bencana Alam Klimatologis dengan Kabupaten/Kota di Pulau Jawa Tahun 2015," Tugas Akhir, Program Studi Diploma III Jurusan Statistika, Institut Teknologi Sepuluh Nopember, Surabaya, 2016.

[13] G. Husain, D. Nasef, R. Jose, J. Mayer, M. Bekbolatova, T. Devine, dan M. Toma, "SMOTE vs. SMOTEENN: A Study on the Performance of Resampling Algorithms for Addressing Class Imbalance in Regression Models," *Algorithms*, vol. 18, no. 1, Art. no. 37, 2025. doi: 10.3390/a18010037.

LAMPIRAN

1. Library yang digunakan

```
# core libs
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# modeling / preprocessing
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer, KNNImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# metrics & utils
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curve,
auc
from scipy.stats import skew

# imbalanced-learn (jika dipakai)
from imblearn.over  sampling import RandomOverSampler, SMOTE
from imblearn.under  sampling import RandomUnderSampler
from imblearn.combine import SMOTEENN
```

2. Melihat Info Data

```
# Load dataset
df = pd.read_csv("heart_disease_uci.csv")

# Pilih kolom penelitian
columns_used = ["sex", "cp", "trestbps", "fbs","ca", "num"] #ca ga usah
df = df[columns_used]

# ============================
# Target → biner
# ============================
df["num"] = df["num"].apply(lambda x: 1 if x > 0 else 0)

# Ganti nilai 0 pada trestbps jadi NaN (agar bisa diimputasi)
df["trestbps"] = df["trestbps"].replace(0, np.nan)

df
print(df.info())
print(df.head())
```

```
---------------------------------------------- output ----------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   sex       920 non-null    object
 1   cp        920 non-null    object
 2   trestbps  860 non-null    float64
 3   fbs       830 non-null    object
 4   ca        309 non-null    float64
 5   num       920 non-null    int64
dtypes: float64(2), int64(1), object(3)
memory usage: 43.3+ KB
None
      sex               cp  trestbps    fbs   ca  num
0    Male   typical angina     145.0   True  0.0    0
1    Male     asymptomatic     160.0  False  3.0    1
2    Male     asymptomatic     120.0  False  2.0    1
3    Male      non-anginal     130.0  False  0.0    0
4  Female  atypical angina     130.0  False  0.0    0
```

3. Exploratory Data Analysis

```
# Menampilkan informasi ringkas datase
df.info()
---------------------------------------------- output ----------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   sex       920 non-null    object
 1   cp        920 non-null    object
 2   trestbps  860 non-null    float64
 3   fbs       830 non-null    object
 4   ca        309 non-null    float64
 5   num       920 non-null    int64
dtypes: float64(2), int64(1), object(3)
memory usage: 43.3+ KB
```

```
# Tampilkan statistik deskriptif dataset
df.describe()
---------------------------------------------- output ----------------------------------------------
```

	trestbps	ca	num
count	860.000000	309.000000	920.000000
mean	132.286047	0.676375	0.553261
std	18.536175	0.935653	0.497426
min	80.000000	0.000000	0.000000
25%	120.000000	0.000000	0.000000

50%	130.000000	0.000000	1.000000
75%	140.000000	1.000000	1.000000
max	200.000000	3.000000	1.000000

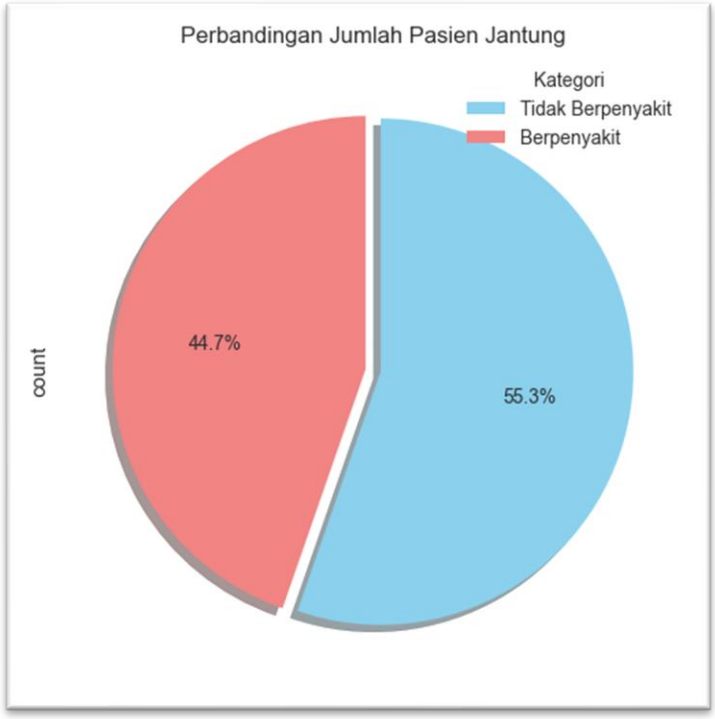
```
# cek missing values dan nol
missing  values = df.isnull().sum()
zero  values = (df == 0).sum()

print("Jumlah missing values:")
print(missing  values)
print("\nJumlah nol:")
print(zero_values)
```

---------- output ----------

```
Jumlah missing values:
sex          0
cp           0
trestbps     60
fbs          90
ca          611
num           0
dtype: int64

Jumlah nol:
sex          0
cp           0
trestbps     0
fbs        692
ca         181
num        411
dtype: int64
```

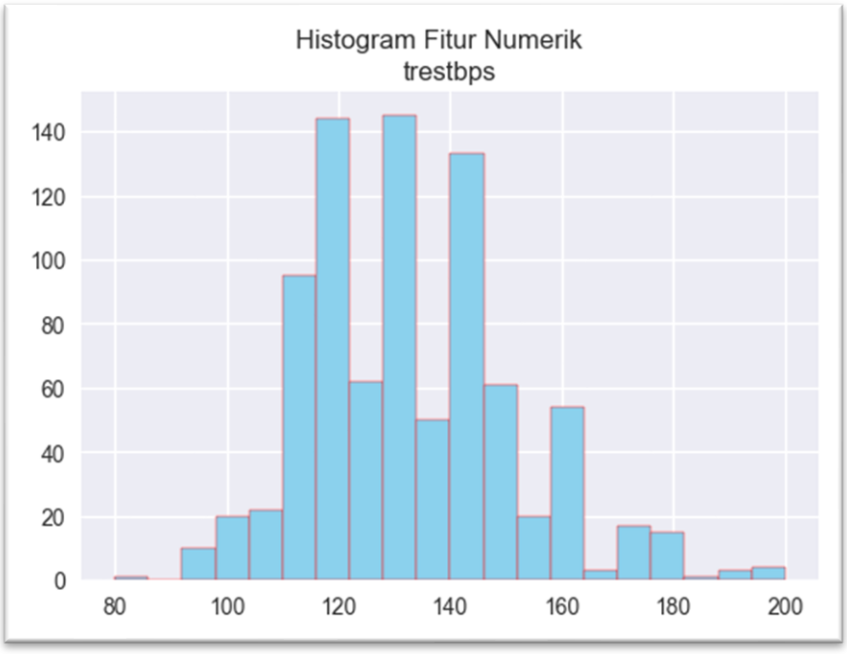
```
# Distribusi target
plt.figure(figsize=(6,6))
df["num"].value_counts().plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    counterclock=False,
    colors=["skyblue", "lightcoral"],
    labels=None,
    explode=(0.03, 0.03),
    shadow=True
)
plt.title("Proporsi Penderita Penyakit Jantung")
plt.legend(labels=["Tidak", "Ya"], title="Kategori", loc="best")
plt.show()
```

---------- output ----------




```
# Histogram numerik
df[["trestbps"]].hist(figsize=(6,4), bins=20, color='skyblue', edgecolor='red')
plt.suptitle("Histogram Fitur Numerik")
plt.show()
```

---------- output ----------




```
# Outlier check
num_cols = ["trestbps"]
outlier_summary = {}
for col in num_cols:
```

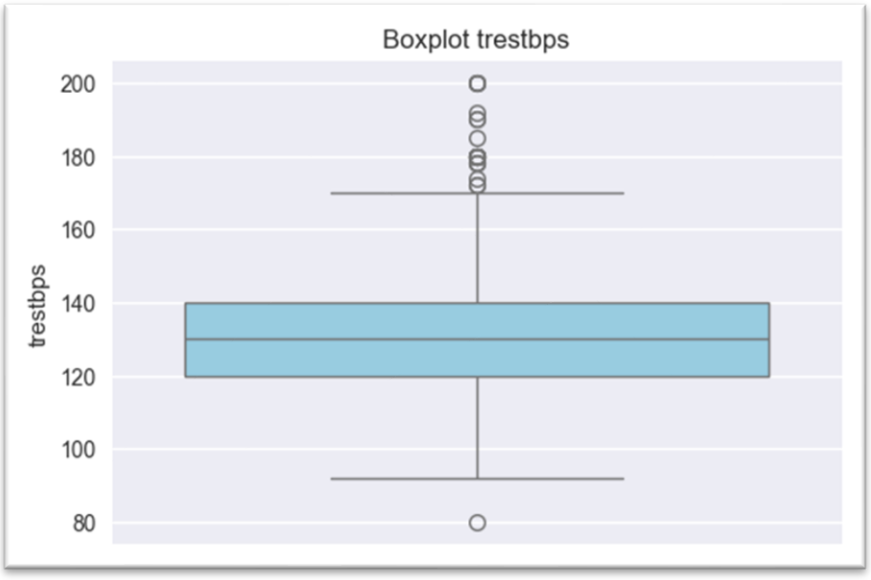
```
    col_skew = skew(df[col].dropna())
    if -1 < col_skew < 1:  # normal → Z-score
        mean = df[col].mean()
        std = df[col].std()
        z_scores = (df[col] - mean) / std
        outliers = df[np.abs(z_scores) > 3]
        method = "Z-Score"
    else:  # skewed → IQR
        Q1, Q3 = df[col].quantile([0.25, 0.75])
        IQR = Q3 - Q1
        lower, upper = Q1 - 1.5*IQR, Q3 + 1.5*IQR
        outliers = df[(df[col] < lower) | (df[col] > upper)]
        method = "IQR"
    outlier_summary[col] = {
        "Skewness": round(col_skew, 2),
        "Metode": method,
        "Jumlah Outlier": outliers.shape[0]
    }
print(pd.DataFrame(outlier_summary).T)
```

---------- output ----------

	Skewness	Metode	Jumlah Outlier
trestbps	0.63	Z-Score	7

```
# Boxplot
plt.figure(figsize=(6,4))
sns.boxplot(y=df["trestbps"], color="skyblue")
plt.title("Boxplot trestbps")
plt.show()
```

---------- output ----------



4. Data Preprocessing

```
# pisahkan fitur dan target
```

```
X = df.drop("num", axis=1)   # fitur
y = df["num"]                # target
```

```
# split train & test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

```
# Preprocessing Pipeline
categorical_cols = ["sex", "cp", "fbs"]
numeric_cols = ["trestbps"]

preprocessor = ColumnTransformer(
    transformers=[
        ("num", Pipeline([
            ("imputer", SimpleImputer(strategy="median")),
            ("scaler", StandardScaler())
        ]), numeric_cols),

        ("cat", Pipeline([
            ("imputer", SimpleImputer(strategy="most_frequent")),
            ("onehot", OneHotEncoder(drop="first", handle_unknown="ignore"))
        ]), categorical_cols)
    ]
)
```

5. Modelling

```
## Model Pipelines
logmodel = Pipeline(steps=[
    ("preprocess", preprocessor),
    ("classifier", LogisticRegression(max_iter=1000, random_state=42))
])

rf = Pipeline(steps=[
    ("preprocess", preprocessor),
    ("classifier", RandomForestClassifier(n_estimators=100, random_state=42))
])

## Evaluation Function
def evaluate_model(model, X_train, y_train, X_test, y_test, model_name, metrics_dict, probs_dict):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]

    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
```

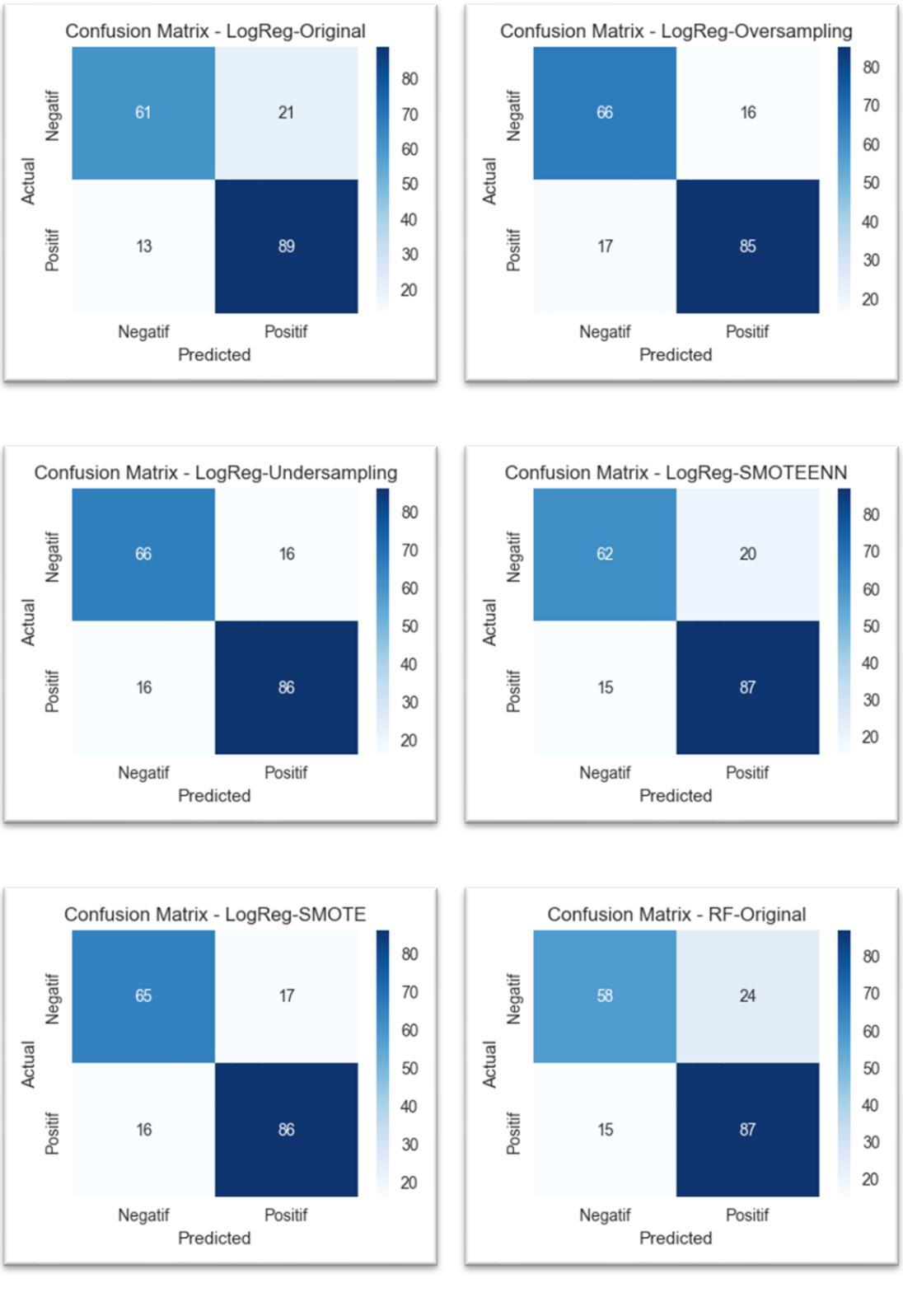
```
    metrics_dict[model_name] = {
        "accuracy": acc,
        "precision": report["1"]["precision"],
        "recall": report["1"]["recall"],
        "f1": report["1"]["f1-score"]
    }
    probs_dict[model_name] = (y_test, y_prob)

    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(4,3))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=["Negatif", "Positif"],
                yticklabels=["Negatif", "Positif"])
    plt.title(f"Confusion Matrix - {model_name}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

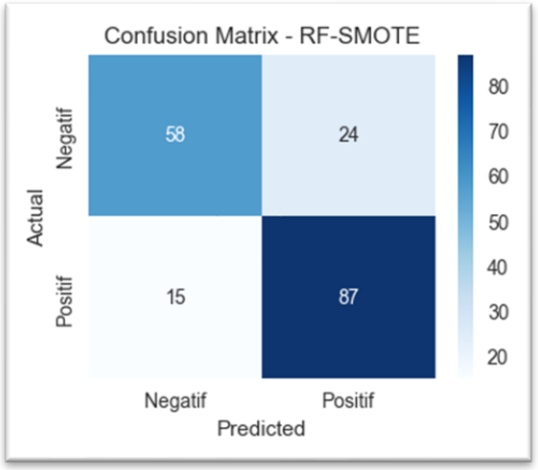
6. Training & Evaluation

```
# === Helper: Preprocessing ke array numeric ===

X_train_prep = preprocessor.fit_transform(X_train)
X_test_prep = preprocessor.transform(X_test)

## Training & Evaluation
metrics = {}
probs = {}

# Logistic Regression (original)
evaluate_model(
    LogisticRegression(max_iter=1000, random_state=42),
    X_train_prep, y_train, X_test_prep, y_test,
    "LogReg-Original", metrics, probs
)

# Oversampling
ros = RandomOverSampler(random_state=42)
X_ros, y_ros = ros.fit_resample(X_train_prep, y_train)
evaluate_model(
    LogisticRegression(max_iter=1000, random_state=42),
    X_ros, y_ros, X_test_prep, y_test,
    "LogReg-Oversampling", metrics, probs
)

# Undersampling
rus = RandomUnderSampler(random_state=42)
X_rus, y_rus = rus.fit_resample(X_train_prep, y_train)
```

```
evaluate_model(
    LogisticRegression(max_iter=1000, random_state=42),
    X_rus, y_rus, X_test_prep, y_test,
    "LogReg-Undersampling", metrics, probs
)

# SMOTEENN
sme = SMOTEENN(random_state=42)
X_sme, y_sme = sme.fit_resample(X_train_prep, y_train)
evaluate_model(
    LogisticRegression(max_iter=1000, random_state=42),
    X_sme, y_sme, X_test_prep, y_test,
    "LogReg-SMOTEENN", metrics, probs
)

# SMOTE
smote = SMOTE(random_state=42)
X_smote, y_smote = smote.fit_resample(X_train_prep, y_train)
evaluate_model(
    LogisticRegression(max_iter=1000, random_state=42),
    X_smote, y_smote, X_test_prep, y_test,
    "LogReg-SMOTE", metrics, probs
)

# Random Forest (original)
evaluate_model(
    RandomForestClassifier(n_estimators=100, random_state=42),
    X_train_prep, y_train, X_test_prep, y_test,
    "RF-Original", metrics, probs
)

# Random Forest (SMOTE)
evaluate_model(
    RandomForestClassifier(n_estimators=100, random_state=42),
    X_smote, y_smote, X_test_prep, y_test,
    "RF-SMOTE", metrics, probs
)
```

---------- output ----------


Confusion Matrix - LogReg-Original
Actual
Negatif
Positif
61
21
13
89
Negatif
Positif
Predicted
80
70
60
50
40
30
20
Confusion Matrix - LogReg-Oversampling
66
16
17
85
Confusion Matrix - LogReg-Undersampling
66
16
16
86
Confusion Matrix - LogReg-SMOTEENN
62
20
15
87
Confusion Matrix - LogReg-SMOTE
65
17
16
86
Confusion Matrix - RF-Original
58
24
15
87

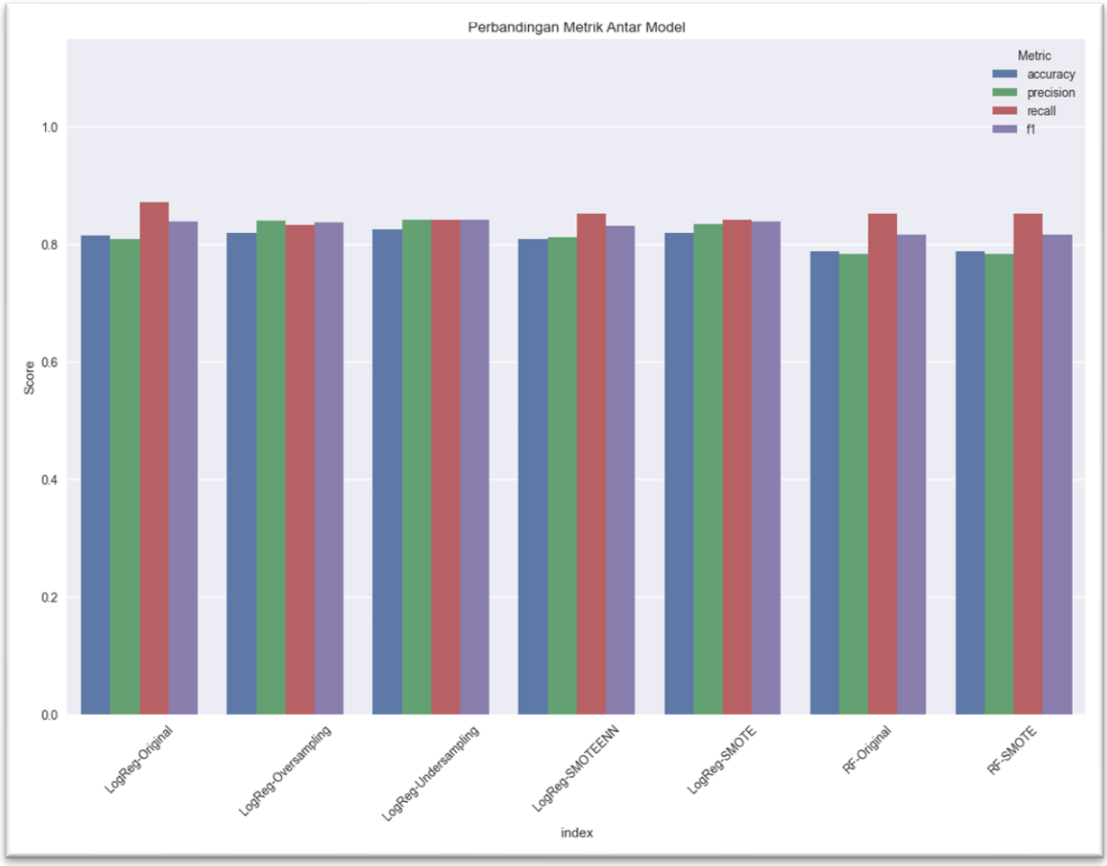


7. Perbandingan Metrik

```
## Perbandingan Metrik
metrics_df = pd.DataFrame(metrics).T
print(metrics_df)

plt.figure(figsize=(15,10))
sns.barplot(data=metrics_df.reset_index().melt(id_vars="index"),
            x="index", y="value", hue="variable")
plt.title("Perbandingan Metrik Antar Model")
plt.ylabel("Score")
plt.ylim(0,1.15)
plt.xticks(rotation=45)
plt.legend(title="Metric", loc="upper right")
plt.show()
```

---------- output ----------

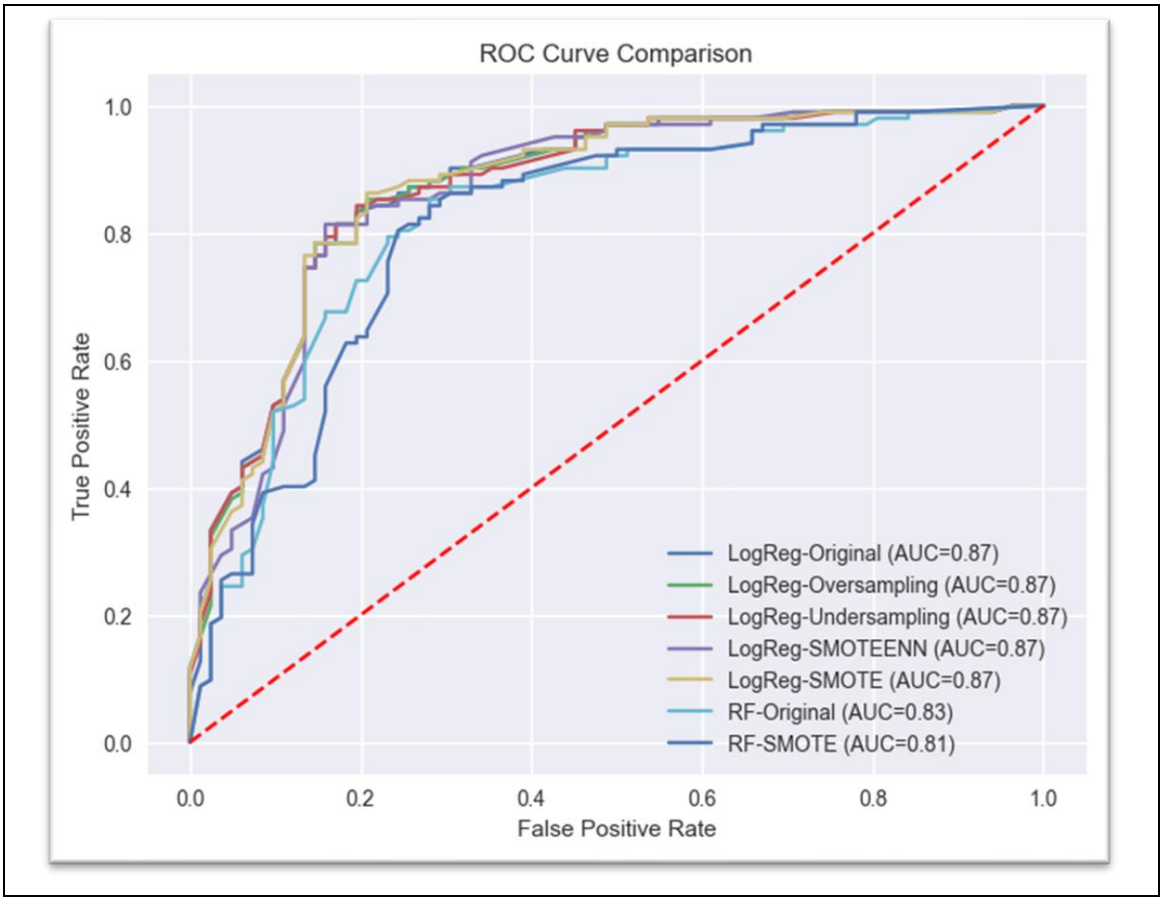
```
                    accuracy  precision    recall        f1
LogReg-Original     0.815217   0.809091  0.872549  0.839623
LogReg-Oversampling  0.820652   0.841584  0.833333  0.837438
LogReg-Undersampling  0.826087   0.843137  0.843137  0.843137
LogReg-SMOTEENN      0.809783   0.813084  0.852941  0.832536
LogReg-SMOTE         0.820652   0.834951  0.843137  0.839024
RF-Original          0.788043   0.783784  0.852941  0.816901
RF-SMOTE             0.788043   0.783784  0.852941  0.816901
```



8. ROC Curve

```
## ROC Curve
plt.figure(figsize=(8,6))
for model_name, (y_true, y_prob) in probs.items():
    fpr, tpr, _ = roc_curve(y_true, y_prob)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f"{model_name} (AUC={roc_auc:.2f})")
plt.plot([0,1],[0,1],'r--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve Comparison")
plt.legend(loc="lower right")
plt.show()
```

---------- output ----------


ROC Curve Comparison
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
LogReg-Original (AUC=0.87)
LogReg-Oversampling (AUC=0.87)
LogReg-Undersampling (AUC=0.87)
LogReg-SMOTEENN (AUC=0.87)
LogReg-SMOTE (AUC=0.87)
RF-Original (AUC=0.83)
RF-SMOTE (AUC=0.81)