

**Analisis Prediktif Keberhasilan Penawaran Penjualan B2B Menggunakan
Artificial Neural Network Pada Industri *Technical Supplier*
(STUDI KASUS : CV Dima Perkasa Teknik)**

TUGAS AKHIR

**Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Strata-1
Program Studi Teknik Industri - Fakultas Teknologi Industri
Universitas Islam Indonesia**



Nama : Dimas Eka Kelana Putra
No. Mahasiswa : 19522302

**PROGRAM STUDI TEKNIK INDUSTRI PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2026**

PERNYATAAN KEASLIAN

Saya mengakui bahwa tugas akhir ini adalah hasil karya saya sendiri kecuali kutipan dan ringkasan yang seluruhnya sudah saya jelaskan sumbernya. Jika dikemudian hari ternyata terbukti pengakuan saya ini tidak benar dan melanggar peraturan yang sah maka saya bersedia ijazah yang telah saya terima ditarik kembali oleh Universitas Islam Indonesia.

Yogyakarta, 20 – 06– 2024



(Dimas Eka Kelana Putra)
19522302

SURAT BUKTI PENELITIAN

CV. DIMA PERKASA TEKNIK

GENERAL TECHNICAL SUPPLIER

JL. A.Yani Km: 7 Komplek Griya pemurus indah Blok: F No: 25
Fax: 0511-4282040 Email: nia2104@yahoo.com / 2104nia@gmail.com

SURAT KETERAGNGAN

No: 076/XI/DPT/2025

Dengan Ini Menrangkan Bahwa :

Nama : Dimas Eka Kelana Putra

Status : MAHASISWA

Jurusan: Teknik Industri

Mahasiswa diatas tersebut telah melaksanakan penelitian tugas akhir di CV. DIMA PERKASA TEHNIK Pada Tanggal 28 November 2025

Demikian surat keterangan ini dibuat untuk dapat dipergunakan sebagaimana mestinya

Banjarmasin, 28 November 2025

CV. Dima Perkasa Tehnik Banjarmasin

CV. DIMA PERKASA TEKNIK

GENERAL TECHNICAL

SUPPLIER

Jl. A. Yani Km.7

KURNIAWATI

Direktur

LEMBAR PENGESAHAN PEMBIMBING

**Analisis Prediktif Keberhasilan Penawaran Penjualan B2B
Menggunakan *Artificial Neural Network* Pada Industri *Technical
Supplier* (STUDI KASUS : CV Dima Perkasa Teknik)**




(Winda Nur Cahyo, S.T., M.T., Ph.D.)

LEMBAR PENGESAHAN DOSEN PENGUJI

Analisis Prediktif Keberhasilan Penawaran Penjualan B2B Menggunakan *Artificial Neural Network* Pada Industri *Technical Supplier* (STUDI KASUS : CV Dima Perkasa Teknik)

TUGAS AKHIR

Disusun Oleh :

Nama : Dimas Eka Kelana Putra

No. Mahasiswa : 19522302

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata-1 Teknik Industri Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 07 - Juli – 2026

Tim Penguji

Winda Nur Cahyo, S.T., M.T., Ph.D.

ketua



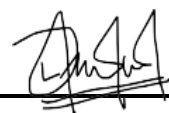
Dr. Dwi Adi Purnama, S.T.

Anggota 1



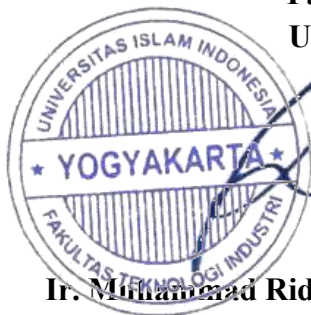
Sayyidah Maulidatul Afraah, S.T., M.T

Anggota 2



Mengetahui,

Ketua Program Studi Teknik Industri Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia




Ir. Muhammad Ridwan Andi Purnomo, ST., M.Sc., Ph.D., IPM
015220101

HALAMAN PERSEMBAHAN

Bismillahirrahmanirrahim

Dengan mengucapkan rasa syukur yang sebesar – besarnya kepada Allah SWT yang maha pemilik segala sesuatu yang ada atas Rahmat dan karunia-Nya. Dengan ini saya mempersembahkan tugas akhir saya kepada diri saya sendiri yang telah berjuang dan berusaha demi menyelesaikan tugas akhir ini. Tak lupa hasil penelitian tugas akhir ini saya persembahkan kepada kedua orang tua saya, yaitu Sugianto dan Kurniawati yang selalu mendukung dan mendoakan saya dari awal saya menempuh pendidikan hingga saya menyelesaikan pendidikan sarjana ini. Dan selesainya tugas akhir ini juga menandakan bahwa saya masih dapat memegang kepercayaan kedua orang tua saya yang diberikan kepada saya.

MOTTO

"Fortune favors the bold"

Virgil

"A mind needs books like a sword needs a whetstone, if it is to keep its edge"

Tyrion Lannister

KATA PENGANTAR

Bismillahirrahmanirrahim

Assalamu'alaikum Warrahmatullahi Wabarakattuh

Alhamdulillahirobil alamin, segala puji dan syukur atas kehadiran Allah SWT karena atas berkah dan rahmatnyalah penulis dapat menyelesaikan penulisan laporan tugas akhir dengan judul “Analisis Prediktif Keberhasilan Penawaran Penjualan B2B Menggunakan *Artificial Neural Network* Pada Industri *Technical Supplier* (Studi Kasus : CV Dima Perkasa Teknik)” yang InsyaAllah bermanfaat untuk penulis, universitas, dan perusahaan. Shalawat serta salam semoga terlimpah curahkan kepada baginda tercinta kita yaitu Nabi Muhammad SAW yang kita nanti - nantikan syafaatnya di akhir nanti.

Laporan Tugas Akhir dibuat guna memenuhi satu prasyarat kelulusan untuk menyelesaikan program studi S-1 di Fakultas Teknologi Industri Universitas Islam Indonesia. Tugas akhir ini sebagai implementasi ilmu yang telah diperoleh dibangku kuliah dengan Realita yang terjadi di dunia industri. Harapan yang ingin dicapai setelah melakukan tugas akhir ini, penulis mampu menerapkan ilmu yang telah diperoleh dengan baik.

Dalam melaksanakan penelitian dan penulisan laporan tugas akhir ini tentunya tak lepas dari bantuan dan dukungan berbagai pihak, untuk itu penulis ingin mengucapkan banyak terima kasih kepada:

1. Allah SWT yang telah memberikan kesehatan dan kelancaran dalam menjalani kegiatan dan penulisan laporan tugas akhir.
2. Ibu Prof. Dr, Sri Kusumadewi, S.Si., M.T. Selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia
3. Bapak Dr. Drs. Imam Djati Widodo, M.Eng.Sc selaku ketua Jurusan Teknik Industri Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Ir. Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D., IPM selaku Ketua Program Studi Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Winda Nur Cahyo, S.T., M.T., Ph.D. selaku dosen pembimbing yang telah meluangkan waktu, tenaga, hingga pikirannya untuk membimbing penulis menyelesaikan laporan tugas akhir.

6. Pihak CV. Dima Perkasa Teknik, yang telah mengizinkan dan memberikan kesempatan serta informasi untuk berjalannya laporan tugas akhir ini.
7. Kedua orang tua saya yaitu Sugianto dan Kurniawati yang selalu memberikan perhatian, kesabaran serta kasih sayang dan pengorbanannya baik berupa moral dan material yang diberikan kepada penulis, sehingga penulis dapat melaksanakan dan menyusun laporan tugas akhir ini dengan baik dan sepenuh hati.
8. Zulfa Aulia Zahra selaku Tunangan yang selalu memberikan perhatian dan semangat sehingga penulis dapat melaksanakan dan menyusun laporan tugas akhir ini.
9. Segenap para teman seperjuangan mahasiswa Teknik Industri 2019 Arya Cahya Mesta yang telah mendukung dan membantu proses laporan tugas akhir. Dan memotivasi penulis untuk menyelesaikan laporan tugas akhir.
10. Dimas Eka Kelana Putra atau saya selaku penulis laporan ini yang di mana telah menjaga semangat, berjuang, berusaha dan bersyukur atas apa yang saya lakukan dalam penelitian tugas akhir ini.

Penulis menyadari bahwa penulisan laporan tugas akhir ini masih jauh dari sempurna, untuk itu penulis mengharapkan kritik, saran dan masukan yang bersifat membangun demi kesempurnaan penulisan di masa yang akan datang. Akhir kata semoga laporan tugas akhir ini dapat digunakan sebagai mana mestinya serta berguna bagi penulis khususnya dan bagi para pembaca yang berminat pada umumnya.

ABSTRAK

Persaingan penjualan B2B pada industri *technical supplier* menuntut perusahaan untuk mampu menentukan peluang keberhasilan penawaran secara lebih objektif. CV Dima Perkasa Teknik memiliki data historis penawaran yang mencakup karakteristik pelanggan, produk, nilai transaksi, margin, skema pembayaran, jumlah item, status pelanggan, dan variasi barang, namun data tersebut belum dimanfaatkan sebagai dasar prediksi. Penelitian ini bertujuan membangun model prediksi keberhasilan penawaran berbasis *Artificial Neural Network*, mengevaluasi performa model, serta mengidentifikasi variabel yang paling berkontribusi terhadap keberhasilan penawaran. Data penelitian berasal dari dokumen penawaran dan data historis perusahaan yang diproses menjadi *dataset* terstruktur. Tahapan metode meliputi pembersihan data, transformasi variabel numerik, pengkodean variabel kategori menggunakan *one-hot encoding*, validasi data, pembagian data latih dan data uji, serta pembangunan model *Artificial Neural Network* dengan arsitektur *Multilayer Perceptron* menggunakan *scikit-learn*. Penelitian ini membandingkan tiga skenario model, yaitu *baseline*, *upsampling*, dan *class weighted*, dengan evaluasi menggunakan *accuracy*, ROC-AUC, *confusion matrix*, dan *classification report*. Hasil penelitian menunjukkan bahwa model *baseline* memberikan performa terbaik dengan *accuracy* 0,958 dan ROC-AUC 0,957. Model *upsampling* menghasilkan *accuracy* 0,854 dan ROC-AUC 0,940, sedangkan model *class weighted* menghasilkan *accuracy* 0,896 dan ROC-AUC 0,943. Hasil *Permutation Importance* menunjukkan bahwa *nilai_penawaran_total*, *TOP_hari*, *margin*, dan *jumlah_item* menjadi variabel yang paling berpengaruh terhadap prediksi keberhasilan penawaran.

Kata kunci: *Artificial Neural Network*, penawaran, model prediksi, *Permutation Importance*, *technical supplier*.

DAFTAR ISI

PERNYATAAN KEASLIAN.....	i
SURAT BUKTI PENELITIAN	ii
LEMBAR PENGESAHAN PEMBIMBING	iii
LEMBAR PENGESAHAN DOSEN PENGUJI.....	iv
HALAMAN PERSEMBAHAN	v
MOTTO	vi
ABSTRAK	ix
DAFTAR ISI	x
DAFTAR TABEL	xii
DAFTAR GAMBAR.....	xiii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	5
1.3 Tujuan Penelitian.....	5
1.4 Manfaat Penelitian.....	5
1.5 Batasan Penelitian.....	6
BAB II TINJAUAN PUSTAKA	7
2.1 Kajian Literatur.....	7
2.2 Landasan Teori	13
2.2.1 Probabilitas.....	13
2.2.2 Logika Fuzzy.....	14
2.2.3 Praproses Data	14
2.2.4 Machine learning.....	15
2.2.5 Klasifikasi	16
2.2.6 Artificial Neural Network	17
2.2.7 Multilayer Preceptron	20
2.2.8 Upsampling	22
2.2.9 Class Weighed	23
2.2.10 Accuracy Score.....	25
2.2.11 Confussion Matrix	25
2.2.12 ROC-AUC.....	26
2.2.13 Classification Report.....	27
2.2.14 Permutation Importance.....	28
2.2.15 Python	29
BAB III METODE PENELITIAN.....	31
3.1 Subjek Penelitian	31
3.2 Objek Penelitian.....	31
3.3 Jenis Data.....	31
3.3.1 Data Primer	31
3.3.2 Data Sekunder	31
3.4 Metode Pengumpulan Data.....	32
3.5 Alur Penelitian	33
3.6 Data Set	36
3.7 Variabel Model.....	37
3.8 Praproses Data	40
3.9 Perancangan Model Artificial Neural Network.....	41

3.9.1	Model <i>Artificial Neural Network Baseline</i>	42
3.9.2	Model <i>Artificial Neural Network Class Weighted</i>	46
3.9.3	Model <i>Artificial Neural Network Upsampling</i>	47
3.10	Arsitektur Model <i>Artificial Neural Network</i>	49
BAB IV PEMBANGUNAN SISTEM		59
4.1	Pengumpulan Data	59
4.2	Pengolahan Data	60
4.2.1	Pembersihan data	61
4.2.2	Transformasi data	62
4.2.3	Pengkodean Data	66
4.2.4	Validasi Data	67
4.3	Perancangan dan Pengujian Model <i>Artificial Neural Network</i>	68
4.3.1	<i>Library</i>	68
4.3.2	Pembangunan Model <i>Artificial Neural Network</i>	69
BAB V PENGUJIAN SISTEM DAN PEMBAHASAN		89
5.1	Pengujian Model <i>Artificial Neural Network</i>	89
5.1.1	Pengujian Model <i>Artificial Neural Network Baseline</i>	90
5.1.2	Pengujian Model <i>Artificial Neural Network Upsampling</i>	94
5.1.3	Pengujian Model <i>Artificial Neural Network Class Weighted</i>	98
5.2	Perbandingan Hasil Pegujian Model <i>Artificial Neural Network</i>	103
5.3	Pengujian Variabel Kontribusi	110
5.4	Implikasi hasil penelitian bagi perusahaan	120
BAB VI PENUTUP		123
6.1	Kesimpulan	123
6.2	Saran	124
DAFTAR PUSTAKA		125
LAMPIRAN		128

DAFTAR TABEL

Tabel 2 1 Kajian Literatur Penelitian Terdahulu	23
Tabel 3 1 Data Set Penelitian.....	53
Tabel 3 2 Variabel Model Artificial Neural Network.....	54
Tabel 4 1 Kolom atribut data penawaran.....	75
Tabel 5 1 Implikasi penelitian bagi perusahaan.....	120

DAFTAR GAMBAR

Gambar 2 1 Struktur <i>Artificial Neural Network</i>	34
Gambar 3 1 Diagram alur penelitian.....	49
Gambar 3 2 Alur Sistematis Praproses Data.....	57
Gambar 3 3 Diagram Alur proses percangan model <i>Artificial Neural Network baseline</i>	59
Gambar 3 4 Diagram alur proses <i>Artificial Neural Network Class Weighted</i>	62
Gambar 3 5 Diagram alur proses <i>Artificial Neural Network Upsampling</i>	64
Gambar 3 6 Lapisan Model Artificial Network Multilayer Preception.....	66
Gambar 3 7 Proses Pelatihan Model.....	68
Gambar 3 8 Proses Pelatihan Model <i>Class Weight</i>	70
Gambar 3 9 Proses Uji Variabel Berpengaruh.....	72
Gambar 4. 1 Tabel Data Uji.....	76
Gambar 4. 2 <i>Import Library</i> dan Definisi Path.....	77
Gambar 4. 3 <i>Input Excel</i>	77
Gambar 4. 4 Bentuk Data	77
Gambar 4. 5 Kode drop <i>Missing Value</i>	78
Gambar 4. 6 Kode menghilangkan Duplikat	78
Gambar 4. 7 Hasil Cleaning.....	78
Gambar 4. 8 Kode datetime	79
Gambar 4. 9 Kode Datatype teks kategori.....	79
Gambar 4. 10 Kode Range Nilai Margin.....	79
Gambar 4. 11 Hasil Transformasi Variabel Margin	80
Gambar 4. 12 Kode Konversi Variabel Numerik	80
Gambar 4. 13 Tipe Data Variabel Numerik.....	80
Gambar 4. 14 Konversi Label Biner	81
Gambar 4. 15 Hasil Pengecekan Label	81
Gambar 4. 16 Kode Mengisi <i>Missing Value</i>	81
Gambar 4. 17 Jumlah Nilai Kosong.....	82
Gambar 4. 18 Kode <i>One-hot encoding</i>	82
Gambar 4. 19 Hasil <i>Encoding</i> data	83
Gambar 4. 20 Kode Validasi Data	83
Gambar 4. 21 Hasil Konfirmasi Validasi Data	84
Gambar 4. 22 Kode <i>Import Library</i>	86
Gambar 4. 23 <i>Import Modul</i> scikit learn	86
Gambar 4. 24 Mendefinisikan <i>Input</i> dan <i>Ouput</i>	87
Gambar 4. 25 Identifikasi Non Variabel.....	87
Gambar 4. 26 Kode Fungsi Load Data	88
Gambar 4. 27 Mendefinisikan Variabel X dan Y	88
Gambar 4. 28 Kode Membangun Model <i>Artificial Neural Network</i>	89
Gambar 4. 29 Split Data Pelatihan dan Data Testing	91
Gambar 4. 30 Pelatihan Model	91
Gambar 4. 31 Uji Prediksi dan Probabilitas Model	92
Gambar 4. 32 Kode Menyimpan Model	93
Gambar 4. 33 <i>Import Library</i> dan Modul.....	93
Gambar 4. 34 <i>Import Modul</i> untuk <i>Upsampling</i>	94
Gambar 4. 35 Mendefinisikan Path dan Variabel.....	94

Gambar 4. 36 Kode Fungsi <i>Upsampling</i>	95
Gambar 4. 37 Fungsi <i>Resample</i>	96
Gambar 4. 38 Kode Pengacakan Data	96
Gambar 4. 39 Memisahkan Variabel kembali	97
Gambar 4. 40 Membagi Data Testing dan Data Pelatihan	98
Gambar 4. 41 Kode Melatih dan Memprediksi Model	98
Gambar 4. 42 <i>Import Library</i> dan Modul	99
Gambar 4. 43 <i>Import Parameter Class Weight</i>	100
Gambar 4. 44 Mendefinisikan Path dan Variabel	100
Gambar 4. 45 Kode Fungsi Build Model <i>Artificial Neural Network</i>	101
Gambar 4. 46 Pembagian data Pelatihan dan data Testing	102
Gambar 4. 47 Kode Menghitung Bobot Kelas	103
Gambar 4. 48 Mendefinisikan <i>Class Weights</i>	103
Gambar 4. 49 Kode Latih dan Prediksi Model	104
Gambar 5 1 Kode Model <i>Artificial Neural Network Baseline</i>	106
Gambar 5 2 Kode Perhitungan Accuracy dan ROC-AUC	107
Gambar 5 3 Kode Perhitungan Confusion Matrix dan Classification Report	107
Gambar 5 4 Hasil Evaluasi Model <i>Artificial Neural Network Baseline</i>	108
Gambar 5 5 Grafik <i>Heatmap Confusion Matrix</i> Model <i>Artificial Neural Network Baseline</i>	108
Gambar 5 6 Hasil <i>Classification Report</i> Model <i>Artificial Neural Network Baseline</i>	109
Gambar 5 7 Grafik Perbandingan <i>Precision, Recall, dan F1-score</i> per Kelas pada Model <i>Artificial Neural Network Baseline</i>	109
Gambar 5 8 Distribusi Data Latih Model <i>Artificial Neural Network Upsampling</i>	110
Gambar 5 9 Perhitungan Accuracy dan ROC-AUC Model <i>Artificial Neural Network Upsampling</i>	111
Gambar 5 10 Kode Confusion Matrix dan <i>Classification Report</i> Model <i>Artificial Neural Network Upsampling</i>	112
Gambar 5 11 Hasil Evaluasi Model <i>Artificial Neural Network Upsampling</i>	112
Gambar 5 12 Grafik <i>Heatmap Confusion Matrix</i> Model <i>Artificial Neural Network Upsampling</i>	113
Gambar 5 13 Hasil <i>Classification Report</i> Model <i>Artificial Neural Network Upsampling</i>	113
Gambar 5 14 Grafik <i>Classification Report</i> Model <i>Artificial Neural Network Upsampling</i>	114
Gambar 5 15 Kode Distribusi dan Bobot Kelas Model <i>Artificial Neural Network Class Weighted</i>	114
Gambar 5 16 Kode Evaluasi Accuracy dan ROC-AUC <i>Class Weighted</i>	116
Gambar 5 17 Kode Evaluasi Klasifikasi Model <i>Artificial Neural Network Class Weighted</i>	116
Gambar 5 18 Distribusi dan Bobot Kelas <i>Class Weighted</i>	116
Gambar 5 19 Hasil Evaluasi Model <i>Artificial Neural Network Class Weighted</i>	117
Gambar 5 20 Grafik <i>Heatmap Confusion Matrix</i> Model <i>Artificial Neural Network Class Weighted</i>	118
Gambar 5 21 Hasil <i>Classification Report</i> Model <i>Artificial Neural Network Class Weighted</i>	118
Gambar 5 22 Grafik <i>Classification Report</i> Model <i>Artificial Neural Network Class Weighted</i>	119

Gambar 5 23 Grafik Dumbbell Chart Perbandingan Accuracy dan ROC-AUC Tiga Model <i>Artificial Neural Network</i>	120
Gambar 5 24 Grafik Perbandingan Komponen Confusion Matrix Tiga Model <i>Artificial Neural Network</i>	121
Gambar 5 25 Perbandingan <i>False Positive</i> dan <i>False Negative</i> Tiga Model <i>Artificial Neural Network</i>	122
Gambar 5 26 Dot Map Probabilitas Kelas 1 terhadap <i>Threshold</i> 0,5 pada Tiga Model <i>Artificial Neural Network</i>	123
Gambar 5 27 Heatmap <i>Classification Report</i> Tiga Model <i>Artificial Neural Network</i> .	124
Gambar 5 28 Heatmap Nilai Asli dan Ranking Performa Tiga Model <i>Artificial Neural Network</i>	125
Gambar 5 29 <i>Import Library Permutation Importance</i>	128
Gambar 5 30 Definisi Path Data dan Model.....	128
Gambar 5 31 Pembentukan Variabel Fitur dan Target untuk Uji <i>Permutation Importance</i>	129
Gambar 5 32 Pembagian Data Latih, Data Uji, dan Pemanggilan Model <i>Baseline</i>	130
Gambar 5 33 Penyesuaian Urutan Fitur Data Uji sesuai Model <i>Baseline</i>	131
Gambar 5 34 Perhitungan ROC-AUC Model <i>Artificial Neural Network Baseline</i>	131
Gambar 5 35 Perhitungan <i>Permutation Importance</i>	132
Gambar 5 36 Penyusunan Hasil <i>Permutation Importance</i> dalam DataFrame.....	132
Gambar 5 37 Menampilkan Hasil <i>Permutation Importance</i>	133
Gambar 5 38 Tabel Ranking Variabel Berdasarkan <i>Permutation Importance</i>	133
Gambar 5 39 Grafik Peringkat Variabel Berdasarkan Rata-rata Penurunan ROC-AUC	134
Gambar 5 40 Grafik Kontribusi Kelompok Variabel terhadap Prediksi Model <i>Artificial Neural Network</i>	135
Lampiran 1 Dokumen Penawaran-1	144
Lampiran 2 Dokumen Penawaran-2	145

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Berdasarkan Statistik Ekonomi dan Keuangan Indonesia yang diterbitkan Bank Indonesia dengan sumber data dari Badan Pusat Statistik, Produk Domestik Bruto Indonesia atas dasar harga berlaku tahun 2025 mencapai Rp23.821.104 miliar. Pada tahun yang sama, sektor industri pengolahan mencatat nilai sebesar Rp4.541.520 miliar atau sekitar 19,07 persen terhadap PDB nasional, sedangkan sektor pertambangan dan penggalan mencatat nilai sebesar Rp2.084.613 miliar atau sekitar 8,75 persen terhadap PDB nasional (Bank Indonesia, 2026, hal. 228-229). Kondisi ini menunjukkan bahwa industri pengolahan dan pertambangan masih memiliki peran penting dalam struktur ekonomi nasional, sehingga kebutuhan terhadap barang penunjang operasional, termasuk ban industri dan ban alat berat. Terutama pada persaingan bisnis B2B (*Business-to-Business*) menunjukkan peningkatan volume transaksi dan intensitas yang semakin tinggi. khususnya pada sektor industri pendukung seperti *technical supplier*.

Pada praktik penjualan B2B, khususnya pada perusahaan pemasok ban untuk sektor tambang dan subkontraktor, proses transaksi hampir selalu diawali dengan pengajuan penawaran yang memuat berbagai keterangan produk, *term of payment* dan, harga yang diajukan. Jenderal Mineral dan Batubara Tahun 2025 menjelaskan bahwa sebagian besar belanja barang pada sektor pertambangan mineral dan batu bara dilakukan melalui distribusi atau vendor dalam negeri. Kondisi ini memperlihatkan bahwa perusahaan pemasok memiliki peran penting dalam rantai pasok pertambangan, terutama dalam penyediaan barang operasional yang mendukung kegiatan produksi (Kementerian Energi dan Sumber Daya Mineral Republik Indonesia, 2025, hlm. 30–31).

Hal ini menunjukkan bahwa satu penawaran ban yang berhasil dapat bernilai signifikan, sementara satu penawaran yang gagal berpotensi menimbulkan kerugian ekonomi yang besar. Permasalahan muncul ketika tingkat keberhasilan penawaran ban bersifat fluktuatif dan sulit diprediksi. Pada sektor tambang dan subkontraktor, pelanggan cenderung memilih pemasok yang mampu memberikan kombinasi harga, kualitas ban, dan skema pembayaran yang paling sesuai dengan kebutuhan. Ketika sebuah penawaran ban gagal dimenangkan, kebutuhan ban tersebut hampir pasti akan dipenuhi oleh kompetitor lain. Publikasi Kementerian Perindustrian Republik Indonesia dalam laporan

industri manufaktur 2024 menyebutkan bahwa salah satu tantangan utama perusahaan pemasok teknis adalah ketidakpastian hasil penawaran akibat banyaknya faktor yang memengaruhi keputusan pembelian pelanggan industri. Ketidakpastian ini menyebabkan perusahaan sering mengalami fluktuasi tingkat keberhasilan penjualan dari waktu ke waktu, meskipun volume penawaran relatif stabil. Situasi tersebut menimbulkan masalah pada perencanaan penjualan dan pengendalian kinerja perusahaan.

Permasalahan semakin kompleks pada perusahaan *technical supplier* yang melayani Perusahaan subkontraktor dan retail tanpa kontrak penjualan. Studi padmapuspita J (2021) menunjukkan bahwa keberhasilan penawaran pada sektor ini sangat dipengaruhi oleh kombinasi variabel finansial dan non finansial yang saling berinteraksi secara non linier. Pendekatan evaluasi penawaran yang hanya berbasis pengalaman dan penilaian subjektif terbukti kurang mampu menangkap pola hubungan antar variabel tersebut. Akibatnya, perusahaan berisiko mengajukan penawaran dengan margin tidak optimal atau mengalokasikan upaya penjualan pada peluang yang memiliki probabilitas keberhasilan rendah.

Kondisi tersebut juga terjadi pada CV Dima Perkasa Teknik sebagai objek penelitian. Perusahaan ini secara rutin mengajukan penawaran penjualan kepada pelanggan industri dan proyek dengan variasi nilai transaksi dan karakteristik pelanggan yang beragam. Data historis penawaran mencakup informasi nilai penawaran total, jenis pelanggan, kategori produk, jumlah item, dan *term of payment*. Namun hingga saat ini data tersebut hanya digunakan sebagai arsip administratif dan laporan deskriptif, tanpa analisis kuantitatif untuk memprediksi peluang keberhasilan penjualan.

CV Dima Perkasa Teknik mengalami kegagalan penawaran dengan total nilai penawaran sebesar Rp. 5.564.499.528 dalam 3 tahun terakhir. Untuk mencegah hal ini secara ideal, perusahaan dapat memanfaatkan data historis penawaran untuk memperkirakan probabilitas keberhasilan sebelum menetapkan harga dan margin, karena keputusan penjualan yang hanya berdasarkan evaluasi subjektif rentan menghasilkan prediksi yang kurang akurat. Penelitian tentang prediksi keberhasilan peluang penjualan menyatakan bahwa pendekatan berbasis evaluasi manusia tradisional cenderung kurang stabil dibandingkan model prediktif berbasis data, sehingga keputusan penjualan yang tidak didukung analisis data berpotensi mengurangi akurasi prediksi hasil penjualan dan efektivitas strategi penjualan. Rezazadeh (2020, hlm. 280–281) menunjukkan bahwa penggunaan model prediktif untuk memproyeksikan hasil penjualan memberikan tingkat

akurasi dan nilai ekonomi yang lebih baik dibanding pendekatan evaluasi subjektif tradisional.

Risiko kerugian akibat kegagalan penawaran ban menjadi semakin besar dalam dan memiliki konsekuensi bisnis yang signifikan, terutama ketika pelanggan berasal dari sektor tambang dan subkontraktor yang tidak terikat kontrak jangka panjang. Dalam kondisi ini, setiap proses pengadaan ban dilakukan melalui penawaran terbuka yang bersifat kompetitif. Ketika satu penawaran ban gagal dimenangkan, kebutuhan pelanggan hampir pasti langsung dipenuhi oleh kompetitor lain. Situasi ini menyebabkan perusahaan kehilangan potensi pendapatan bernilai besar, mengingat kebutuhan ban alat berat bersifat rutin akibat tingkat keausan yang tinggi pada aktivitas tambang dan proyek. Pada tahun 2023, Kementerian ESDM menyoroti kelangkaan ban alat berat di sektor pertambangan batu bara karena ketiadaan ban dapat menghambat produksi. Hal ini menunjukkan bahwa ban alat berat memiliki peran penting dalam menjaga kelancaran operasional pertambangan (Riyandanu, 2023).

Permasalahan utama yang menyebabkan kegagalan penawaran ban adalah ketidakmampuan perusahaan dalam mengidentifikasi peluang penawaran yang paling potensial untuk dimenangkan. Dalam praktik operasional, penentuan harga dan margin sering kali didasarkan pada pengalaman tenaga penjualan tanpa estimasi kuantitatif mengenai peluang keberhasilan. Pendekatan ini menjadi kurang efektif ketika perusahaan harus bersaing dengan banyak kompetitor pada pelanggan yang sama, terutama dalam kondisi subkontraktor bebas memilih *supplier*. Tanpa alat bantu analisis, perusahaan berisiko menetapkan harga yang terlalu tinggi sehingga kalah bersaing, atau terlalu rendah sehingga menurunkan margin secara tidak optimal.

Pemanfaatan data historis penawaran ban menjadi salah satu solusi untuk mengatasi permasalahan tersebut. Data penawaran yang mencakup nilai penawaran, skema pembayaran dan jenis produk sebenarnya menyimpan pola hubungan yang dapat digunakan untuk memprediksi keberhasilan penawaran. Namun pola ini bersifat kompleks dan non linier, sehingga sulit ditangkap menggunakan analisis statistik sederhana.

Artificial Neural Network merupakan metode yang relevan untuk memodelkan permasalahan prediksi kegagalan dan keberhasilan penawaran ban. *Artificial Neural Network* bekerja dengan meniru cara kerja jaringan saraf manusia melalui proses pembelajaran dari data historis. Model ini mampu mengenali pola kompleks antara

variabel *input* dan *output* tanpa asumsi hubungan linear. Zahid (2021) menjelaskan bahwa proses tender B2B dapat dianalisis menggunakan model prediktif untuk memperkirakan risiko kehilangan penawaran. Penelitian tersebut relevan dengan konteks penawaran penjualan karena keputusan menang atau kalah dalam proses B2B dapat dipelajari dari data historis. Selain itu, penelitian tersebut juga membahas kondisi ketidakseimbangan kelas, yaitu ketika jumlah data pada satu kelas lebih dominan dibandingkan kelas lainnya. Kondisi ini sesuai dengan data penawaran, karena jumlah penawaran berhasil dan gagal tidak selalu berada dalam proporsi yang seimbang.

Kurniawan dan Setyaningsih (2025) menunjukkan bahwa *Artificial Neural Network* dapat digunakan untuk mempelajari pola transaksi yang kompleks dan non linier. Kemampuan tersebut membuat *Artificial Neural Network* relevan digunakan pada data bisnis yang memiliki banyak variabel, seperti nilai penawaran, margin, jangka waktu pembayaran, jumlah item, dan kategori produk. Berdasarkan temuan tersebut, *Artificial Neural Network* dapat digunakan sebagai metode prediksi untuk menganalisis peluang keberhasilan penawaran B2B yang memiliki pola hubungan antar variabel tidak sederhana.

Dalam konteks penawaran ban, *Artificial Neural Network* dapat digunakan untuk memprediksi probabilitas keberhasilan setiap penawaran sebelum diajukan ke pelanggan. Variabel seperti harga, margin, serta skema pembayaran diproses sebagai *input* model. Model kemudian menghasilkan keluaran berupa probabilitas keberhasilan penawaran. Dengan informasi ini, perusahaan dapat memprioritaskan penawaran dengan peluang menang lebih tinggi, sekaligus menyesuaikan strategi harga dan margin pada penawaran yang memiliki risiko gagal lebih besar. Pendekatan ini membantu perusahaan mengurangi ketergantungan pada intuisi dan meningkatkan objektivitas pengambilan keputusan. penelitian Atmaja dan Pascima (2022) menunjukkan bahwa *Artificial Neural Network* dapat digunakan untuk membangun aplikasi peramalan penjualan berbasis data historis. Model *Artificial Neural Network* mampu mempelajari pola hubungan antara variabel *input* dan *output* , meskipun pola hubungan tersebut tidak selalu mudah diketahui secara langsung. Penelitian ini mendukung penelitian ini karena data historis penawaran CV Dima Perkasa Teknik juga digunakan untuk mempelajari pola keberhasilan penawaran dan menghasilkan prediksi berbasis data..

Keunggulan *Artificial Neural Network* terletak pada kemampuannya mempelajari pola dari data historis dan digunakan kembali untuk memperkirakan kondisi yang akan datang.

Pada konteks penjualan, *Artificial Neural Network* dapat digunakan untuk data yang berkelanjutan, termasuk data penjualan yang berubah dari waktu ke waktu. Ketika data penawaran ban bertambah, model dapat diperbarui melalui proses pelatihan ulang agar pola terbaru dalam data tetap dapat dipelajari oleh model. Pendekatan ini relevan bagi perusahaan *supplier* ban karena kondisi penawaran dipengaruhi oleh perubahan harga, kebutuhan pelanggan, dan dinamika pasar. Ahmed (2024) menjelaskan bahwa model *deep learning* dalam *sales forecasting* dapat beradaptasi dan belajar dari data baru. menghadapi perubahan pola transaksi

Dengan menerapkan model prediksi berbasis *Artificial Neural Network*, perusahaan seperti CV Dima Perkasa Teknik dapat mengatasi permasalahan kegagalan penawaran secara sistematis. Model ini berfungsi sebagai alat bantu strategis untuk menekan risiko kehilangan transaksi kepada kompetitor, terutama pada pelanggan subkontraktor tanpa kontrak tetap. Selain meningkatkan peluang keberhasilan penawaran ban, pendekatan ini juga mendukung pengelolaan margin yang lebih terkendali dan berkontribusi pada peningkatan kinerja penjualan secara berkelanjutan berbasis data.

1.2 Rumusan Masalah

Berdasarkan Latar belakang masalah yang telah dijelaskan, berikut rumusan masalah dalam penelitian ini:

1. Bagaimana proses pembangunan model prediksi keberhasilan penawaran menggunakan *Artificial Neural Network*?
2. Seberapa baik performa model *Artificial Neural Network* dalam memprediksi keberhasilan penawaran?
3. Variabel apa yang paling berkontribusi terhadap keberhasilan penawaran CV Dima Perkasa Teknik?

1.3 Tujuan Penelitian

Berikut ini merupakan tujuan penelitian berdasarkan rumusan masalah yang sudah dijelaskan di atas sebagai berikut:

1. Mendeskripsikan proses pembangunan model prediksi berbasis *Artificial Neural Network*.
2. Mengevaluasi hasil pengujian model prediksi *Artificial Neural Network*.
3. Mengidentifikasi variabel yang paling berkontribusi dalam menentukan keberhasilan penawaran.

1.4 Manfaat Penelitian

Berikut Adalah manfaat penelitian yang diharapkan dari penelitian ini :

1. Pengembangan ilmu *data science* dengan penerapan *Artificial Neural Network* dan model prediksi terhadap bidang manajemen operasi dan Keputusan penjualan.
2. Sebagai Gambaran untuk Perusahaan mengenai besarnya pengaruh faktor tertentu terhadap keberhasilan yang berguna untuk Menyusun strategi penjualan lebih tepat.
3. Menambah wawasan dan referensi terkait teknik industri modern yang menggabungkan sistem cerdas, *machine learning* dan optimasi data.

1.5 Batasan Penelitian

Batasan Penelitian dilakukan agar memusatkan penelitian yang akan dilakukan, tujuan penelitian bisa tercapai dengan tepat dan cepat. Berikut Adalah Batasan dari penelitian ini:

1. Objek penelitian pada data hasil penawaran CV Dima Perkasa Teknik
2. Pengembangan Model menggunakan *Artificial Neural Network* dengan arsitektur *Multilayer Perceptron* untuk memprediksi keberhasilan penawaran
3. Variabel yang digunakan adalah variabel internal dalam data historis Perusahaan CV Dima Perkasa Teknik.
4. Pengujian Variabel berkontribusi menggunakan Metode *Permutation Importance*.

BAB II

TINJAUAN PUSTAKA

Tinjauan Pustaka berisi kajian literatur dan landasan teori penelitian. Kajian Literatur berisi penelitian terdahulu yang memiliki kesamaan dari metode maupun topik terhadap penelitian kali ini dan terdapat beberapa perbandingan yang dijadikan referensi dan tambahan wawasan untuk penelitian. Landasan teori berisi teori-teori dari para ahli terkait konsep dan keilmuan yang akan digunakan pada penelitian kali ini.

2.1 Kajian Literatur

Berikut merupakan beberapa Literatur penelitian terdahulu yang berhubungan dengan model prediksi dan penerapan *Artificial Neural Network*.

Tabel 2 1 Kajian Literatur Penelitian Terdahulu

No	Nama Peneliti & Tahun	Judul Penelitian	Metode Penelitian	Hasil Penelitian
1	Padmapuspita dkk . (2021)	<i>Identification of Success Factor for B2B E-Commerce in Indonesia</i>	<i>Pairwise comparison matrix</i>	Faktor keberhasilan utama <i>B2B e-commerce</i> adalah <i>IT infrastructure, successful customer relationships</i> , dan <i>marketing and branding</i> . keberhasilan transaksi <i>B2B</i> dipengaruhi oleh kesiapan sistem, hubungan pelanggan, reputasi pemasok, dan proses transaksi yang efisien.
2	Studer dkk. (2021)	<i>Towards CRISP-ML(Q): A Machine learning Process Model with Quality Assurance Methodology</i>	<i>CRISP-ML(Q) process mode</i>	Proses <i>machine learning</i> terdiri dari enam tahap, yaitu <i>business and data understanding, data preparation, modeling, evaluation, deployment</i> , serta

No	Nama Peneliti & Tahun	Judul Penelitian	Metode Penelitian	Hasil Penelitian
				<i>monitoring and maintenance.</i> Tahap evaluasi perlu memakai metrik yang jelas, data uji terpisah, analisis performa, dan pengendalian risiko agar model tidak mengalami penurunan performa saat digunakan.
3	Rahmawati dkk. (2022)	Pengembangan Model Persetujuan Kredit Nasabah Bank dengan Algoritma Klasifikasi <i>Naïve Bayes</i> , <i>Decision Tree</i> , dan <i>Artificial Neural Network</i>	<i>Naïve Bayes</i> , <i>Decision Tree</i> , dan <i>Artificial Neural Network</i>	<i>Artificial Neural Network</i> memperoleh akurasi tertinggi sebesar 99,61% dan AUC 0,983. <i>Decision Tree</i> memperoleh akurasi 99,36% dan AUC 0,999. <i>Naïve Bayes</i> memperoleh akurasi 90,79% dan AUC 0,935.
4	Abdulsalam dkk. (2022)	<i>Customer Churn Prediction in Telecommunication Industry Using Classification and Regression Trees and Artificial Neural Network Algorithms</i>	<i>Relief-F</i> , <i>Artificial Neural Network</i> , dan CART	<i>Relief-F</i> memilih 14 fitur dari 21 atribut data. Data dibagi menjadi 75% data latih dan 25% data uji. <i>Artificial Neural Network</i> menghasilkan akurasi 93,88%, lebih tinggi daripada CART sebesar 91,60%. Nilai <i>precision</i> <i>Artificial Neural Network</i> sebesar 83,02% dan <i>F-score</i> sebesar 77,53%, lebih tinggi daripada

No	Nama Peneliti & Tahun	Judul Penelitian	Metode Penelitian	Hasil Penelitian
				CART dengan <i>precision</i> 69,77% dan <i>F-score</i> 72%.
5	Bakirrarar dan Elhan (2023)	<i>Class Weighting Technique to Deal with Imbalanced Class Problem in Machine learning: Methodological Research</i>	<i>Class weighting, Random Forest, dan Support Vector Machine</i>	<i>Class weighting</i> meningkatkan performa kelas minoritas. Pada data simulasi, metode <i>inverse of square root of number of samples</i> menghasilkan performa terbaik. Pada data nyata, metode <i>sample based class weight</i> pada <i>Support Vector Machine</i> menghasilkan performa terbaik
6	Kaope dan Pristyanto (2023)	<i>The Effect of Class imbalance Handling on Datasets Toward Classification Algorithm Performance</i>	ADASYN, SMOTE, SMOTE-ENN, <i>Random Forest</i> , AdaBoost, dan <i>K-Nearest Neighbor</i>	Kombinasi ADASYN dan <i>Random Forest</i> menghasilkan performa terbaik. Nilai evaluasi meningkat 5% sampai 10% dibandingkan skema model lainnya
7	Thölke dkk. (2023)	<i>Class imbalance Should Not Throw You Off Balance</i>	<i>Logistic Regression, LDA, SVM, dan Random Forest</i>	<i>Accuracy</i> menghasilkan nilai tinggi yang menyedatkan pada data tidak seimbang. AUC dan <i>balanced accuracy</i> menghasilkan evaluasi yang lebih stabil. <i>Random Forest</i> lebih kuat pada

No	Nama Peneliti & Tahun	Judul Penelitian	Metode Penelitian	Hasil Penelitian
				kondisi data tidak seimbang
8	Lestari dkk. (2023)	<i>A Comparison of Artificial Neural Network and Naive Bayes Classification Using Unbalanced Data Handling</i>	<i>Artificial Neural Network, Naïve Bayes, dan SMOTE</i>	Data dibagi menjadi 80% data latih dan 20% data uji. Hasil penelitian menunjukkan bahwa penggunaan SMOTE pada data tidak seimbang menghasilkan performa lebih baik dibandingkan tanpa penanganan data tidak seimbang
9	Soegianto dkk. (2024)	<i>Commparation of Model Performance on Housing Business Using Linear Regression, Random Forest Regressor, SVR, and Neural Network</i>	<i>Linear Regression, Random Forest Regressor, SVR, dan Artificial Neural Network</i>	<i>Artificial Neural Network</i> memperoleh performa terbaik dengan MSE 0,0046, R^2 0,86, dan MAE 0,047. SVR memperoleh R^2 0,83, <i>Random Forest Regressor</i> 0,81, dan <i>Linear Regression</i> 0,67
10	Martha dan Herwindiati (2024)	Prediksi Hujan Menggunakan Metode <i>Artificial Neural Network, K-Nearest Neighbor, dan Naïve Bayes</i>	<i>Artificial Neural Network, K-Nearest Neighbor, dan Naïve Bayes</i>	<i>Artificial Neural Network</i> memperoleh akurasi 85%, <i>K-Nearest Neighbor</i> 84%, dan <i>Naïve Bayes</i> 79%. <i>Artificial Neural Network</i> menjadi metode dengan akurasi tertinggi.
11	Fahrudin dkk. (2024)	<i>Influence of Data Scaling and Train/Test Split Ratios on LightGBM Efficacy for Obesity Rate Prediction</i>	<i>LightGBM</i> , normalisasi data, dan perbandingan rasio data latih dan data uji	Rasio data latih dan data uji berpengaruh terhadap performa model klasifikasi. Hasil penelitian menunjukkan bahwa rasio 80:20 menghasilkan akurasi hingga 97% pada dataset obesitas

No	Nama Peneliti & Tahun	Judul Penelitian	Metode Penelitian	Hasil Penelitian
12	Azhima dkk. (2024)	Penerapan Metode <i>Backpropagation Neural Network</i> untuk Klasifikasi Penyakit Stroke	<i>Backpropagation Neural Network</i> , <i>SMOTE</i> , dan perbandingan rasio data latih dan data uji	Penelitian ini membandingkan rasio 90:10, 80:20, dan 70:30. Hasil terbaik diperoleh pada arsitektur [19:29:1], <i>learning rate</i> 0,001, dan rasio data 80:20 dengan akurasi rata-rata sebesar 96,14%
13	Segara dkk. (2025)	Perbandingan Performa Algoritma <i>Random Tree</i> , <i>K-Nearest Neighbor</i> , dan <i>Artificial Neural Network</i> untuk Deteksi Serangan DDoS pada Software Defined Network	<i>Random Tree</i> , <i>K-Nearest Neighbor</i> , dan <i>Artificial Neural Network</i>	<i>Artificial Neural Network</i> memperoleh akurasi 96,85%, <i>precision</i> 94,35%, <i>Recall</i> 97,79%, <i>F1-score</i> 96,04%, dan AUC 0,994. <i>K-Nearest Neighbor</i> memperoleh akurasi 88,89%, sedangkan <i>Random Tree</i> memperoleh akurasi 86,49%
14	Panji Kurniawan (2025)	Analisis Perbandingan Metode <i>Artificial Neural Network</i> dan <i>XGBoost</i> untuk Prediksi Profit dari Data Transaksi Point of Sale	<i>Artificial Neural Network</i> dan <i>XGBoost</i>	<i>Artificial Neural Network</i> memperoleh R^2 0,9996 dan MAE 1.359. <i>XGBoost</i> memperoleh R^2 0,9978 dan MAE 1.600. <i>Artificial Neural Network</i> menghasilkan prediksi profit lebih dekat dengan nilai aktual

No	Nama Peneliti & Tahun	Judul Penelitian	Metode Penelitian	Hasil Penelitian
15	Arti dan Suherlan (2025)	Evaluasi Kinerja <i>Machine learning</i> dalam Memprediksi Kemampuan Adaptasi Mahasiswa pada Lingkungan Pembelajaran Daring	CRISP-DM dan tujuh algoritma <i>supervised learning</i>	<i>Neural Network</i> memperoleh akurasi tertinggi sebesar 0,946. Durasi kelas menjadi faktor dominan dalam prediksi kemampuan adaptasi mahasiswa
16	Haekal dkk. (2025)	<i>AI Ladder-Based Intelligent Enterprise Resource Planning for Predictive Railway Track Maintenance from UAV Imagery</i>	AI Ladder, UAV, <i>Artificial Neural Network</i> , SVM, <i>Random Forest</i> , dan <i>K-Nearest Neighbor</i>	<i>Artificial Neural Network</i> memperoleh performa tertinggi dengan AUC 0,935, akurasi 0,884, <i>F1-score</i> 0,884, dan MCC 0,768. Hasil klasifikasi terhubung ke modul ERP untuk perintah kerja dan penjadwalan pemeliharaan.
17	Mega dan Olivia Brilliant Chang (2026)	Penerapan Algoritma <i>Artificial Neural Network</i> untuk Klasifikasi Kualitas Buah Apel	<i>Artificial Neural Network</i> dan <i>Multilayer Perceptron</i>	<i>Artificial Neural Network</i> memperoleh akurasi 92% pada data uji. <i>Random Forest</i> memperoleh akurasi 88,3%, sedangkan SVM memperoleh akurasi 74,7%. Kurva ROC dan histogram <i>error</i> memperlihatkan generalisasi model yang baik tanpa <i>overfitting</i> .

No	Nama Peneliti & Tahun	Judul Penelitian	Metode Penelitian	Hasil Penelitian
18	Vikkurty dkk. (2026)	<i>Toxicity Prediction of Landfill Leachate-Contaminated Crops Using Machine learning Models Based on PAH and Heavy Metal Concentrations</i>	<i>Artificial Neural Network, Random Forest, SVM, dan K-Nearest Neighbor</i>	<i>Artificial Neural Network</i> memperoleh akurasi tertinggi sebesar 97,8% dan AUC 0,98. <i>Random Forest</i> memperoleh akurasi 94,7%, sedangkan SVM memperoleh akurasi 93,6%. Variabel paling berpengaruh terdiri dari Cd, benzo[a]pyrene, dan Pb.

2.2 Landasan Teori

Berikut merupakan Teori terkait konsep dasar yang digunakan dalam penelitian:

2.2.1 Probabilitas

Probabilitas merupakan cabang ilmu statistika yang mempelajari peluang terjadinya suatu kejadian dalam kondisi yang mengandung ketidakpastian. Menurut Musnaini (2024, hlm. 7–8), probabilitas digunakan untuk mengukur tingkat kemungkinan suatu peristiwa berdasarkan informasi yang tersedia. Konsep ini menjadi dasar dalam berbagai analisis statistik karena memungkinkan peneliti melakukan penilaian secara kuantitatif terhadap kejadian yang belum atau akan terjadi. Melalui probabilitas, ketidakpastian dapat diukur secara sistematis sehingga mendukung proses pengambilan keputusan yang lebih objektif.

Dalam teori probabilitas, setiap kejadian memiliki peluang yang dinyatakan dalam rentang nilai antara 0 sampai 1. Nilai probabilitas sebesar 0 menunjukkan bahwa suatu kejadian tidak mungkin terjadi, sedangkan nilai probabilitas sebesar 1 menunjukkan bahwa suatu kejadian pasti terjadi (Mugnaini, 2024, hlm. 8–10). Semakin besar nilai probabilitas suatu kejadian, semakin tinggi peluang kejadian tersebut untuk terjadi. Oleh karena itu, probabilitas berfungsi sebagai ukuran matematis yang menggambarkan tingkat keyakinan terhadap kemungkinan munculnya suatu peristiwa.

Mugnaini (2024, hlm. 9–10) menjelaskan bahwa probabilitas suatu kejadian dapat dihitung berdasarkan perbandingan antara jumlah kejadian yang diinginkan dengan jumlah seluruh kemungkinan kejadian yang terdapat dalam ruang sampel. Secara

matematis, probabilitas suatu kejadian A dapat dinyatakan dengan rumus $P(A) = n(A)/n(S)$, di mana $n(A)$ merupakan jumlah anggota kejadian A dan $n(S)$ merupakan jumlah anggota ruang sampel. Konsep ini menjadi fondasi dalam berbagai metode statistik dan analisis data karena memungkinkan peneliti mengukur peluang suatu hasil secara terstruktur dan terukur

2.2.2 Logika *Fuzzy*

Logika *Fuzzy* Adalah pendekatan logika yang digunakan untuk memodelkan ketidakpastian dan nilai kebenaran yang bersifat bertahap. Sri Kusumadewi (2010) menjelaskan bahwa logika *Fuzzy* tidak membatasi nilai kebenaran hanya pada benar atau salah, melainkan memungkinkan nilai kebenaran berada pada rentang nol sampai satu. Konsep ini digunakan untuk merepresentasikan kondisi nyata yang sulit didefinisikan secara tegas, seperti rendah, sedang, atau tinggi, yang sering dijumpai dalam sistem pengambilan keputusan.

Dasar utama logika *Fuzzy* adalah himpunan *Fuzzy*, yaitu himpunan yang setiap elemennya memiliki derajat keanggotaan. Derajat keanggotaan ini dinyatakan dengan fungsi keanggotaan $\mu_A(x)$ yang bernilai antara 0 sampai 1. Secara umum, derajat keanggotaan dirumuskan sebagai $\mu_A(x) = \text{nilai keanggotaan elemen } x \text{ terhadap himpunan } Fuzzy A$. Jika $\mu_A(x) = 0$ maka x tidak menjadi anggota himpunan A . Jika $\mu_A(x) = 1$ maka x sepenuhnya menjadi anggota himpunan A (Sri Kusumadewi, 2010)

Logika *Fuzzy* dikembangkan untuk menjembatani cara berpikir manusia yang bersifat linguistik dengan sistem komputasi yang bersifat numerik. Logika *Fuzzy* memungkinkan aturan berbentuk pernyataan jika maka yang menyerupai penalaran manusia. Dengan pendekatan ini, sistem dapat mengambil keputusan meskipun informasi yang tersedia tidak lengkap atau tidak pasti. keunggulan logika *Fuzzy* terletak pada kemampuannya mengakomodasi ketidakpastian dan subjektivitas tanpa memerlukan model matematis yang kompleks. Oleh karena itu logika *Fuzzy* banyak digunakan dalam sistem cerdas dan sistem pendukung keputusan yang melibatkan penilaian manusia dan data yang tidak sepenuhnya pasti.

2.2.3 Praproses Data

Praproses data merupakan tahap awal dalam *machine learning* yang dilakukan sebelum data digunakan untuk melatih model. Tim Pembina IOAI Indonesia (2025, hlm. 9) menjelaskan bahwa *dataset* perlu melalui proses pembersihan, penyesuaian, dan transformasi agar sesuai dengan kebutuhan model. Tahap ini mencakup deteksi *outlier*,

imputasi data hilang, normalisasi skala fitur, dan transformasi bentuk data. Praproses data memiliki peran penting karena dapat meningkatkan performa dan kemampuan generalisasi model.

Menurut Tim Pembina IOAI Indonesia (2025, hlm. 9–10), normalisasi diperlukan karena banyak algoritma *machine learning* bekerja lebih baik ketika data memiliki skala yang seragam. Jika terdapat fitur dengan rentang nilai yang terlalu berbeda, fitur dengan skala besar dapat mendominasi proses pembelajaran, meskipun fitur tersebut belum tentu lebih penting. Oleh karena itu, normalisasi atau standardisasi digunakan untuk menyamakan skala fitur agar setiap variabel dapat memberikan kontribusi yang lebih seimbang dalam proses pembelajaran model.

Data hilang juga menjadi bagian penting dalam praproses data. Tim Pembina IOAI Indonesia (2025, hlm. 13) menjelaskan bahwa data hilang dapat muncul karena kesalahan pencatatan, kegagalan sensor, atau data yang tidak terisi. Keberadaan data hilang dapat mengganggu proses pelatihan model sehingga perlu ditangani sebelum data digunakan. Penanganan data hilang dapat dilakukan dengan menghapus data tertentu atau melakukan imputasi menggunakan nilai rata-rata, median, atau nilai yang paling sering muncul sesuai dengan jenis data.

Transformasi data dilakukan untuk mengubah data ke dalam bentuk yang lebih sesuai untuk dianalisis dan digunakan dalam model *machine learning*. transformasi data dapat membantu memperbaiki distribusi data, mengurangi pengaruh *outlier*, menyesuaikan skala data, dan meningkatkan performa model (Tim Pembina IOAI Indonesia, 2025, hlm. 14–15).

2.2.4 *Machine learning*

Machine learning merupakan salah satu cabang dari kecerdasan buatan yang berfokus pada pengembangan algoritma dan model agar komputer dapat belajar dari data. Dalam konsep ini, komputer tidak hanya bekerja berdasarkan instruksi eksplisit, tetapi dilatih untuk mengenali pola, mempelajari hubungan dalam data, dan menghasilkan keputusan berdasarkan data yang dianalisis. Panji Bintoro dkk. (2024, hlm. 1–2) menjelaskan bahwa *machine learning* memberikan kemampuan kepada sistem komputer untuk meningkatkan kinerjanya melalui pengalaman tanpa harus diprogram secara langsung untuk setiap tugas tertentu. Dengan demikian, *machine learning* dapat dipahami sebagai pendekatan komputasi yang menjadikan data sebagai dasar utama proses pembelajaran.

Data memiliki peran utama dalam *machine learning* karena model tidak dapat belajar tanpa data yang memadai. Panji Bintoro dkk. (2024, hlm. 4–6) menjelaskan bahwa data menjadi fondasi dalam pembentukan model, baik dalam bentuk data terstruktur, data tidak terstruktur, maupun data semi terstruktur. Data tersebut kemudian diproses oleh algoritma untuk menemukan pola dan membangun model yang mampu menghasilkan prediksi atau keputusan. Algoritma dalam *machine learning* bekerja sebagai prosedur matematis yang mengolah data, sedangkan model merupakan representasi matematis dari hubungan yang dipelajari algoritma dari data.

Model dalam *machine learning* terbentuk melalui proses pelatihan, yaitu proses ketika algoritma belajar dari data latih untuk menemukan pola dan parameter yang sesuai. Setelah model dilatih, model diuji menggunakan data yang belum pernah dilihat sebelumnya untuk menilai kemampuannya dalam melakukan generalisasi. Panji Bintoro dkk. (2024, hlm. 7–8) menjelaskan bahwa model yang baik harus mampu menjaga keseimbangan antara akurasi dan kompleksitas agar tidak hanya bekerja baik pada data pelatihan, tetapi juga mampu memberikan hasil yang baik pada data baru. Dalam proses ini, evaluasi model dilakukan untuk mengukur kinerja model melalui metrik tertentu, seperti akurasi, *precision*, *Recall*, *F1-score*, MSE, RMSE, serta AUC-ROC.

Machine learning memiliki beberapa kategori utama berdasarkan cara model belajar dari data. Panji Bintoro dkk. (2024, hlm. 9–14) menjelaskan bahwa kategori tersebut meliputi *supervised learning*, *unsupervised learning*, *semi-supervised learning*, dan *reinforcement learning*. *Supervised learning* menggunakan data berlabel untuk mempelajari hubungan antara *input* dan *output*, sedangkan *unsupervised learning* digunakan untuk menemukan pola tersembunyi pada data yang tidak memiliki label. *Semi-supervised learning* menggabungkan data berlabel dan tidak berlabel dalam proses pembelajaran, sedangkan *reinforcement learning* melatih agen untuk mengambil keputusan melalui interaksi dengan lingkungan dan umpan balik berupa reward atau punishment. Pembagian kategori ini menunjukkan bahwa *machine learning* memiliki pendekatan yang berbeda sesuai dengan bentuk data, tujuan analisis, dan mekanisme pembelajaran yang digunakan.

2.2.5 Klasifikasi

Menurut Panji Bintoro dkk. (2024, hlm. 33–34), Klasifikasi merupakan salah satu bentuk *supervised learning* dalam *machine learning* yang digunakan untuk mengelompokkan data ke dalam kelas atau label tertentu. Dalam *supervised learning*, model dilatih

menggunakan data yang telah memiliki pasangan *input* dan *output* , sehingga model dapat mempelajari hubungan antara atribut data dan label yang ditentukan. *supervised learning* bekerja dengan menggunakan data berlabel untuk memprediksi *output* berdasarkan *input* baru setelah model mempelajari pola dari data pelatihan.

Klasifikasi digunakan ketika Input (Y) yang dihasilkan berbentuk kategori atau kelas diskrit. Panji Bintoro dkk. (2024, hlm. 34) menjelaskan bahwa klasifikasi bertujuan memisahkan data ke dalam beberapa kelas, seperti contoh pendeteksian email spam. Dalam proses ini, algoritma mempelajari pola dari data berlabel, kemudian menggunakan pola tersebut untuk menentukan kelas dari data baru. Dengan demikian, klasifikasi dapat dipahami sebagai proses pembentukan model yang memetakan data *input* ke dalam label kelas berdasarkan karakteristik yang dipelajari selama tahap pelatihan.

Proses klasifikasi umumnya terdiri dari dua tahap utama, yaitu tahap pelatihan dan tahap prediksi. Pada tahap pelatihan, model menggunakan data yang telah diberi label untuk mengenali pola pembeda antar kelas. Setelah tahap pelatihan selesai, model digunakan untuk memprediksi kelas dari data baru yang belum memiliki label. klasifikasi melibatkan tahap pelatihan , yaitu pelatihan model menggunakan data berlabel, dan tahap prediksi, yaitu penggunaan model yang telah dilatih untuk memprediksi kelas dari data baru (Panji Bintoro dkk., 2024, hlm. 36).

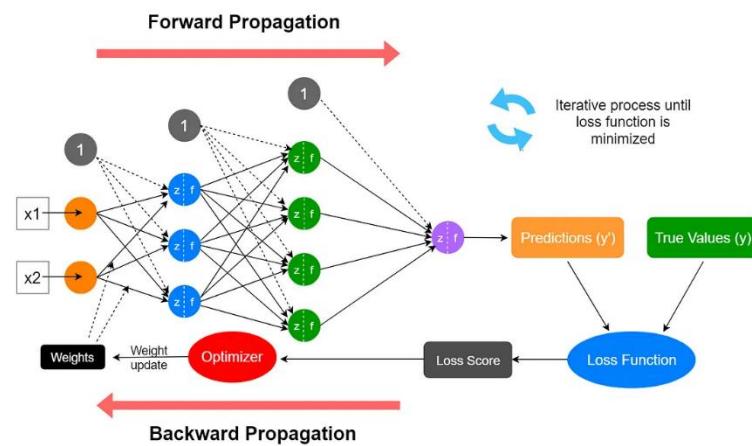
Algoritma klasifikasi memiliki karakteristik yang berbeda sesuai dengan pendekatan pembelajaran yang digunakan. beberapa algoritma klasifikasi yang umum digunakan meliputi *K-Nearest Neighbors*, *Decision Tree*, *Support Vector Machine*, *Naive Bayes*, *Logistic Regression*, dan *Random Forest*. Selain itu, Panji Bintoro dkk. (2024, hlm. 44–46) juga menjelaskan bahwa *Neural Networks* atau jaringan saraf tiruan dapat digunakan untuk tugas klasifikasi, regresi, dan pengenalan pola. *Neural Networks* bekerja melalui lapisan *input*, lapisan tersembunyi, lapisan *output* , bobot, fungsi aktivasi, *forward propagation*, dan *backward propagation*. Dengan demikian, *Artificial Neural Network* dapat dipahami sebagai salah satu pendekatan klasifikasi dalam *machine learning* yang mampu mempelajari pola data melalui proses pelatihan dan menghasilkan prediksi kelas berdasarkan karakteristik *input*.

2.2.6 Artificial Neural Network

Artificial Neural Network adalah model komputasi yang meniru cara kerja jaringan saraf biologis untuk mempelajari pola dari data. *Artificial Neural Network* merupakan sistem pemrosesan informasi yang tersusun dari sejumlah unit sederhana yang saling terhubung,

unit ini disebut neuron, dan koneksinya menyimpan pengetahuan dalam bentuk bobot. Bishop menjelaskan *Artificial Neural Network* sebagai keluarga model fungsi nonlinier yang memetakan *input* ke *output* melalui komposisi beberapa transformasi, sehingga *Artificial Neural Network* mampu mempelajari hubungan kompleks yang sulit direpresentasikan dengan model linier. *Artificial Neural Network* juga dipahami sebagai model probabilistik non linier yang digunakan untuk memetakan hubungan antara variabel *input* dan *output* melalui komposisi beberapa fungsi sederhana. Bishop (2006, hlm. 226–227) menempatkan *Artificial Neural Network* dalam kerangka *pattern recognition* dan *statistical learning*, bukan sekadar sebagai analogi biologis dari otak manusia. Ia menegaskan bahwa *Artificial Neural Network* pada dasarnya adalah model fungsi matematis yang fleksibel, di mana struktur jaringan dan parameter bobot menentukan bentuk fungsi pemetaan yang dipelajari dari data. Dengan pendekatan ini, *Artificial Neural Network* dipandang sebagai alat untuk melakukan aproksimasi fungsi dan inferensi statistik pada permasalahan kompleks.

Menurut Bishop (2006, hlm. 227–228), Struktur dasar *Artificial Neural Network* terdiri dari layer *input*, satu atau lebih *hidden layer*, dan layer *output*. Setiap neuron menerima sejumlah Input (x_1 sampai x_d), lalu mengalikan setiap *input* dengan bobot w , menambahkan bias b , kemudian melewatkannya ke fungsi aktivasi.



Gambar 2.1 Struktur *Artificial Neural Network*

Sumber: https://course-net.com/wp-content/uploads/2024/05/1_ZXAOUqmlyECgfVa81Sr6Ew-L.webp

Pada tingkat neuron tunggal, setiap neuron menerima sejumlah Input (x_1 , x_2 , sampai x_n). Setiap *input* dikalikan dengan bobot w_1 , w_2 , sampai w_n , lalu dijumlahkan dan ditambahkan bias b menghasilkan persamaan (2.1).

$$z = (w_1 \times x_1) + (w_2 \times x_2) + \dots + (w_n \times x_n) + b \quad (2.1)$$

Nilai z disebut net *input* atau *input* linear neuron, nilai z akan diproses oleh fungsi aktivasi F untuk menghasilkan *output* neuron a seperti pada persamaan (2.2). Jika fungsi aktivasi menggunakan fungsi *sigmoid* maka persamaannya menjadi (2.3) dan jika menggunakan fungsi ReLU maka persamaan aktivasi f menjadi persamaan (2.4)

$$a = f(z) \quad (2.2)$$

$$f(z) = \frac{1}{1 + e^{-z}} \quad (2.3)$$

$$f(z) = \max(0, z) \quad (2.4)$$

Untuk layer *output* pada klasifikasi biner, menghasilkan satu nilai Input (Y) yang ditafsirkan sebagai probabilitas kelas positif. Persamaan *output* akhir (2.5).

$$\hat{y} = f(z(L)) \quad (2.5)$$

dengan L menyatakan layer terakhir. Jika fungsi *sigmoid* digunakan, maka $\hat{y}$ merepresentasikan persamaan (2.6). Untuk klasifikasi multikelas, fungsi *softmax* digunakan dengan persamaan

$$P(y = 1|x) \quad (2.6)$$

$$P(y = 1 | x) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2.7)$$

Persamaan ini memastikan bahwa total probabilitas seluruh kelas bernilai satu, sehingga konsisten dengan prinsip probabilitas.

Cara kerja *Artificial Neural Network* pada tahap prediksi mengikuti *forward propagation*. Data masuk ke layer *input*, jaringan menghitung z dan a di setiap layer hingga menghasilkan *output*. Untuk klasifikasi biner, *output* sering berupa probabilitas p , lalu keputusan kelas diperoleh dengan ambang tertentu, misalnya $p \geq 0,5$ sebagai kelas 1. Untuk multikelas, *output softmax* menghasilkan distribusi probabilitas untuk setiap kelas, lalu kelas prediksi diambil dari probabilitas terbesar

Tahap pelatihan *Artificial Neural Network* bertujuan mencari bobot dan bias yang meminimalkan kesalahan prediksi. Kesalahan ini diukur dengan *loss function*. *loss function* mendefinisikan kriteria optimasi yang harus diminimalkan selama proses pembelajaran, sehingga secara langsung menentukan arah penyesuaian bobot dan bias dalam jaringan. Pada permasalahan regresi, *loss function* yang paling umum digunakan adalah *Mean Squared Error*. Fungsi ini mengukur rata rata kuadrat selisih antara nilai prediksi dan nilai aktual Persamaan *Mean Squared Error* (2.8).

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2 \quad (2.8)$$

Untuk klasifikasi biner, *loss function* yang digunakan adalah *Binary Cross entropy* yang berasal dari *negative log likelihood* distribusi Bernoulli. Persamaan ini (2.9), nilai $\hat{y}$ merepresentasikan probabilitas kelas positif yang dihasilkan oleh fungsi *sigmoid* pada layer *output*. *Loss function* ini memberikan penalti besar ketika model memberikan probabilitas tinggi pada kelas yang salah, sehingga sangat efektif untuk melatih model klasifikasi berbasis probabilitas. Pada klasifikasi multikelas, *loss function* yang digunakan adalah *Categorical Cross entropy* yang dikombinasikan dengan fungsi *softmax* dengan persamaan seperti ini (2.10). Bishop menjelaskan bahwa kombinasi *softmax* dan *cross entropy* menghasilkan gradien yang stabil dan interpretasi probabilistik yang konsisten, sehingga sangat sesuai untuk permasalahan klasifikasi dengan lebih dari dua kelas.

$$L = -\frac{1}{N} \sum_{i=1}^N (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)) \quad (2.9)$$

$$L = -\sum_{i=1}^N y_i \log(\hat{y}_i) \quad (2.10)$$

Artificial Neural Network memperbarui parameter menggunakan *backpropagation*, yaitu metode menghitung *gradien loss* terhadap setiap bobot dan bias, sehingga didapatkan pembaharuan nilai Bobot (W) dengan persamaan (2.11) dan pembaharuan Bias (b) dengan persamaan (2.12).

$$W_{baru} = W_{lama} - \eta \times \frac{\partial J}{\partial W} \quad (2.11)$$

$$b_{baru} = b_{lama} - \eta \times \frac{\partial J}{\partial b} \quad (2.12)$$

Secara keseluruhan, *Artificial Neural Network* merupakan kerangka pemodelan yang kuat dan fleksibel untuk pengenalan pola dan prediksi, dengan dasar matematis yang kuat dalam teori probabilitas dan statistika. *Artificial Neural Network* tidak dipandang sebagai sistem heuristik semata, tetapi sebagai model pembelajaran terstruktur yang dapat dianalisis, dioptimalkan, dan dievaluasi secara ilmiah (Bishop, 2006).

2.2.7 Multilayer Perceptron

Menurut Bishop (2006, hlm. 226–228), *Multilayer Perceptron* merupakan salah satu arsitektur *Artificial Neural Network* yang paling dasar dan banyak digunakan dalam

machine learning. *Multilayer Perceptron* merupakan bentuk *feed-forward neural network* yang memetakan variabel *input* menuju variabel *output* melalui fungsi non linier yang dikendalikan oleh parameter bobot dan bias. Model ini bekerja dengan membentuk kombinasi linier dari variabel *input*, kemudian memprosesnya melalui fungsi aktivasi non linier pada *hidden units*. Dengan mekanisme tersebut, mampu merepresentasikan hubungan non linier antara variabel *input* dan *output* secara lebih fleksibel dibandingkan *perceptron* tunggal.

Menurut Bishop (2006), bahwa secara matematis *Multilayer Preceptron* merepresentasikan fungsi komposisi berulang dari transformasi linier dan fungsi aktivasi nonlinier. Setiap neuron pada *hidden layer* menghitung kombinasi linier dari *output* layer sebelumnya, kemudian melewatkannya ke fungsi aktivasi. Proses ini diulang hingga layer *output*. Dengan mekanisme tersebut, *Multilayer Preceptron* mampu membangun representasi bertingkat dari data, di mana layer awal menangkap pola sederhana dan layer yang lebih dalam menangkap pola yang lebih kompleks.

Secara matematis, vektor *input* dinyatakan sebagai $x = [x_1, x_2, \dots, x_d]$. *Output* layer *input* didefinisikan sebagai $a(0) = x$. Pada setiap *hidden layer* ke 1, setiap neuron j menghitung kombinasi linier dari *output* layer sebelumnya. Sehingga menghasilkan Persamaan net *input* neuron (2.13).

$$z_j^{(l)} = \sum_k z_j^{(l)} a_j^{(l-1)} + b_j^{(l)} \quad (2.13)$$

Persamaan ini menunjukkan bahwa setiap neuron mengintegrasikan seluruh informasi dari layer sebelumnya. Nilai net *input* pada persamaan (2.13) kemudian dilewatkan ke fungsi aktivasi non linier untuk menghasilkan *output* neuron dengan persamaan ()

$$z_j^{(l)} = f(z_j^{(l)}) \quad (2.14)$$

Fungsi aktivasi f dapat berupa *sigmoid*, *tanh*, atau ReLU, tergantung pada kebutuhan model. Fungsi aktivasi non linier merupakan syarat utama agar *Multilayer Preceptron* mampu memodelkan hubungan nonlinier antara *input* dan *output*. Proses pelatihan *Multilayer Preception* bertujuan untuk meminimalkan *loss function* yang mengukur selisih *output* dan target. Fungsi untuk regresi menggunakan persamaan *Mean Squared Error* (2.15), fungsi untuk klasifikasi biner menggunakan *Binary Cross Entropy* (2.16) dan untuk klasifikasi multikelas yaitu Input (Y) yang memiliki kelas lebih dari 2 menggunakan *Categorical Cross entropy* (2.17).

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2 \quad (2.15)$$

$$L = -\frac{1}{N} \sum_{i=1}^N (y_i \cdot \log(\hat{y}) + (1 - y_i) \cdot \log(1 - \hat{y})) \quad (2.16)$$

$$L = -\sum_{i=1}^N y_i \log(\hat{y}_i) \quad (2.17)$$

Pembaruan parameter bobot dan bias diperbarui menggunakan algoritma *backpropagation*, Bishop (2006, hlm. 240–242) menjelaskan *backpropagation* sebagai prosedur matematis yang digunakan untuk menghitung turunan *loss function* terhadap setiap parameter jaringan secara efisien dengan menerapkan aturan rantai dari kalkulus. Aturan pembaruan parameter digunakan untuk menyesuaikan bobot (W) dengan persamaan (2.18) dan bias (b) model dengan persamaan (2.19). Penyesuaian ini bertujuan untuk meminimalkan nilai dari *loss function*.

$$W(t+1) = W(t) - a \times \left(\frac{\partial J}{\partial W} \right) \quad (2.18)$$

$$b(t+1) = b(t) - a \times \left(\frac{\partial L}{\partial b} \right) \quad (2.19)$$

Dengan rangkaian persamaan tersebut, *Multilayer Perceptron* dipahami sebagai sistem matematis terstruktur yang mempelajari pemetaan non linier dari data. kekuatan *Multilayer Preception* tidak terletak pada satu persamaan tunggal, melainkan pada komposisi berlapis dari persamaan linier dan nonl inier yang membentuk model pembelajaran yang sangat fleksibel dan kuat (Bishop, 2006)

2.2.8 Upsampling

Upsampling merupakan salah satu teknik penanganan data tidak seimbang pada masalah klasifikasi. Data tidak seimbang terjadi ketika jumlah data pada satu kelas lebih banyak dibandingkan kelas lainnya. Weiss (2013, hlm. 4–5) menjelaskan bahwa pembelajaran dari data tidak seimbang berkaitan dengan kondisi kelas yang memiliki jumlah contoh tidak sama, sehingga kondisi tersebut dapat memengaruhi kemampuan model dalam mempelajari kelas yang lebih jarang muncul. Dampak ini terlihat pada penurunan kinerja klasifikasi kelas minoritas karena model lebih banyak menerima pola dari kelas mayoritas selama proses pelatihan.

Dalam konteks penanganan data tidak seimbang, *upsampling* termasuk ke dalam metode sampling pada level data. Weiss (2013, hlm. 24–25) menjelaskan bahwa metode

sampling banyak digunakan untuk menangani data tidak seimbang. Salah satu bentuk dasarnya adalah *random oversampling*, yaitu metode yang menambah jumlah data kelas minoritas dengan menduplikasi contoh pelatihan dari kelas tersebut. Dengan cara ini, tingkat ketidakseimbangan kelas pada data latih dapat dikurangi karena jumlah data minoritas dibuat lebih dekat dengan jumlah data mayoritas.

secara matematis dapat dijelaskan sebagai proses penyesuaian distribusi kelas pada data latih agar proporsi antar kelas menjadi lebih seimbang. Misalkan suatu *dataset* klasifikasi memiliki dua kelas, yaitu kelas mayoritas C_{maj} dan kelas minoritas C_{min} . Jumlah data pada kelas mayoritas dinyatakan sebagai N_{maj} dan jumlah data pada kelas minoritas dinyatakan sebagai N_{min} , dengan kondisi N_{maj} jauh lebih besar daripada N_{min} . bahwa ketidakseimbangan ini menyebabkan fungsi pembelajaran didominasi oleh kelas mayoritas. tujuan *Upsampling* adalah membentuk distribusi baru N_{min}' sehingga mendekati distribusi kelas mayoritas. Persamaan dasar jumlah data setelah *Upsampling*,

$$N_{min}' = N_{min} + K \quad (2.19)$$

$$K = N_{maj} - N_{min} \quad (2.20)$$

$$N_{min}' = N_{maj} \quad (2.21)$$

Nilai K adalah data tambahan hasil duplikasi dari data minoritas karena tujuan *Upsampling* adalah menyamakan jumlah kelas, maka nilai K dapat ditentukan dengan persamaan (2.20), sehingga setelah proses *Upsampling* diperoleh distribusi data menjadi (2.21), dengan kondisi ini, jumlah data kelas minoritas menjadi setara dengan kelas mayoritas, sehingga distribusi data latih menjadi lebih seimbang. *random oversampling* dapat menurunkan tingkat ketidakseimbangan kelas, tetapi metode ini tidak menambahkan informasi baru karena data tambahan berasal dari duplikasi contoh minoritas yang sudah tersedia (Weiss, 2013, hlm. 25) .

2.2.9 Class Weighed

Class Weighted merupakan teknik penanganan ketidakseimbangan kelas yang bekerja dengan memberikan bobot berbeda pada data latih saat proses pelatihan model. Pada data tidak seimbang, model cenderung lebih mudah mengoptimalkan kelas mayoritas karena jumlah datanya lebih banyak. Weiss (2013, hlm. 12) menjelaskan bahwa penggunaan *accuracy* pada kondisi *class imbalance* dapat menyebabkan penurunan kinerja kelas minoritas karena performa kelas mayoritas lebih dominan dalam proses evaluasi. Kondisi ini menunjukkan bahwa model perlu memperlakukan kesalahan pada tiap kelas secara berbeda agar kelas minoritas tidak diabaikan selama proses pembelajaran.

Pendekatan *Class Weighted* dapat dipahami dalam kerangka *cost-sensitive learning*. Weiss (2013, hlm. 30) menjelaskan bahwa *cost-sensitive learning* dapat diterapkan melalui pemberian bobot pada data latih secara proporsional terhadap biaya kesalahan. Dengan pendekatan ini, setiap sampel tidak lagi memiliki kontribusi kesalahan yang sama. Sampel dari kelas yang dianggap lebih penting atau lebih jarang muncul dapat diberi bobot lebih besar, sehingga kesalahan prediksi pada kelas tersebut memberikan pengaruh yang lebih kuat terhadap proses optimasi model.

Pada klasifikasi biner, *Class Weighted* dapat diterapkan pada *loss function* dengan memberikan bobot berbeda untuk kelas positif dan kelas negatif. Jika y menyatakan label aktual, p menyatakan probabilitas prediksi kelas positif, w_1 menyatakan bobot kelas positif, dan w_0 menyatakan bobot kelas negatif, maka bentuk *weighted binary cross entropy* dapat ditulis dengan persamaan (2.22).

$$L_{\text{weighted}} = -\frac{1}{n} \sum [w_1 y \log(p) + w_0 (1 - y) \log(1 - P)] \quad (2.22)$$

Persamaan (2.22) mengukur kesalahan prediksi model dengan memberikan penalti berbeda pada setiap kelas melalui bobot w_1 untuk kelas positif dan w_0 untuk kelas negatif. Variabel y menunjukkan label aktual, p adalah probabilitas prediksi kelas positif yang dihasilkan model, dan n adalah jumlah data latih. Jika kelas minoritas diberi bobot lebih besar, maka kesalahan prediksi pada kelas tersebut akan menghasilkan nilai *loss* dan gradien yang lebih besar, sehingga proses pembaruan parameter *Artificial Neural Network* lebih berfokus pada peningkatan kemampuan model dalam mengenali kelas minoritas.

Penentuan bobot kelas umumnya mengikuti prinsip berbanding terbalik dengan frekuensi kelas. Misalkan jumlah data pada kelas k adalah N_k dan total data adalah N , persamaan bobot yang sering digunakan (2.23). Untuk model klasifikasi biner kelas mayoritas persamaan dapat ditulis seperti (2.24) dan, persamaan kelas minoritas (2.25).

$$w_k = \frac{N}{(K \times N_k)} \quad (2.23)$$

$$w_{maj} = \frac{N}{(2 \times N_{maj})} \quad (2.24)$$

$$w_{min} = \frac{N}{(2 \times N_{min})} \quad (2.25)$$

Berdasarkan persamaan tersebut, kelas minoritas memperoleh bobot lebih besar karena nilai N_{min} lebih kecil dibandingkan N_{maj} . Bobot yang lebih besar membuat

kesalahan pada kelas minoritas memiliki penalti lebih tinggi dalam *loss function*. Dengan demikian, *Class Weighted* tidak mengubah jumlah data seperti *upsampling*, tetapi mengubah kontribusi tiap sampel dalam proses pelatihan. *oversampling* dapat menimbulkan risiko *overfitting* karena membuat salinan dari contoh kelas minoritas. Oleh karena itu, pendekatan berbasis bobot dapat digunakan sebagai alternatif karena tetap memakai data asli, tetapi mengarahkan proses pembelajaran agar lebih memperhatikan kelas yang lebih jarang muncul (Weiss, 2013, hlm. 36).

2.2.10 Accuracy Score

Accuracy Score merupakan metrik evaluasi yang digunakan untuk mengukur tingkat ketepatan model klasifikasi dalam menghasilkan prediksi. Menurut Panji Bintoro dkk. (2024, hlm. 8), akurasi merupakan persentase prediksi yang benar dari total prediksi yang dibuat oleh model. Dengan demikian, *accuracy* menunjukkan seberapa besar proporsi data uji yang berhasil diklasifikasikan secara benar oleh model.

Dalam evaluasi model klasifikasi, *accuracy* digunakan sebagai ukuran awal untuk menilai performa model secara umum. Panji Bintoro dkk. (2024, hlm. 50) menjelaskan bahwa akurasi merupakan proporsi prediksi yang benar dari semua prediksi dan digunakan pada masalah klasifikasi. Metrik ini mudah dipahami karena langsung menunjukkan perbandingan antara jumlah prediksi benar dan seluruh data yang diuji.

Secara matematis, *accuracy* dapat dihitung dengan membandingkan jumlah prediksi benar terhadap seluruh hasil prediksi. Dalam klasifikasi biner, prediksi benar terdiri dari *True Positive* dan *True Negative*, sedangkan prediksi salah terdiri dari *False Positive* dan *False Negative* dengan persamaan (2.26).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (2.26)$$

Nilai TP menunjukkan data kelas positif yang diprediksi benar sebagai positif. Nilai TN menunjukkan data kelas negatif yang diprediksi benar sebagai negatif. Nilai FP menunjukkan data kelas negatif yang salah diprediksi sebagai positif. Nilai FN menunjukkan data kelas positif yang salah diprediksi sebagai negatif. Berdasarkan persamaan tersebut, *accuracy* menghitung seluruh prediksi benar tanpa membedakan jenis kesalahan yang terjadi.

2.2.11 Confusion Matrix

Confusion Matrix merupakan alat evaluasi yang digunakan untuk melihat hubungan antara hasil prediksi model dan label aktual pada masalah klasifikasi. Evaluasi model

diperlukan untuk mengukur seberapa baik model bekerja setelah proses pelatihan dilakukan. evaluasi menjadi tahap penting dalam *machine learning* karena membantu menilai kinerja model dan menentukan apakah model masih memerlukan perbaikan atau penyesuaian. Dalam konteks klasifikasi, *Confusion Matrix* membantu menyajikan hasil prediksi secara lebih rinci karena model tidak hanya dinilai dari jumlah prediksi benar, tetapi juga dari pola kesalahan yang terjadi pada setiap kelas (Panji Bintoro dkk., 2024, hlm. 8).

Dalam klasifikasi biner, *Confusion Matrix* terdiri dari empat komponen utama, yaitu *True Positive*, *True Negative*, *False Positive*, dan *False Negative*. *True Positive* menunjukkan jumlah data kelas positif yang diprediksi benar sebagai positif. *True Negative* menunjukkan jumlah data kelas negatif yang diprediksi benar sebagai negatif. *False Positive* menunjukkan jumlah data kelas negatif yang salah diprediksi sebagai positif. *False Negative* menunjukkan jumlah data kelas positif yang salah diprediksi sebagai negatif. Keempat komponen tersebut menjadi dasar untuk membaca perilaku model dalam membedakan kelas positif dan kelas negatif.

Menurut Panji Bintoro (2024, hlm. 50), evaluasi klasifikasi dapat menggunakan akurasi, *precision*, *Recall*, *F1-score*, dan AUC-ROC. Metrik tersebut berkaitan dengan hasil perbandingan antara prediksi model dan kondisi aktual data. Oleh karena itu, *Confusion Matrix* dapat digunakan sebagai dasar untuk memahami sumber nilai metrik evaluasi, terutama karena *precision* dan *Recall* membutuhkan informasi tentang prediksi benar dan prediksi salah pada kelas tertentu. Dengan cara ini, evaluasi model tidak hanya menunjukkan ketepatan umum, tetapi juga menjelaskan kesalahan klasifikasi secara lebih terarah. Pada data dengan distribusi kelas tidak seimbang, *Confusion Matrix* penting karena mampu memperlihatkan apakah model hanya bekerja baik pada kelas mayoritas atau juga mampu mengenali kelas minoritas.

2.2.12 ROC-AUC

ROC-AUC merupakan metrik evaluasi yang digunakan untuk menilai kemampuan model klasifikasi dalam membedakan kelas positif dan kelas negatif. Panji Bintoro dkk (2024, hlm. 8) menjelaskan bahwa ROC-AUC digunakan untuk mengukur kemampuan model dalam membedakan antara kelas positif dan negatif. Metrik ini tidak hanya melihat hasil prediksi akhir, tetapi juga memperhatikan skor probabilitas yang dihasilkan model pada berbagai kemungkinan ambang keputusan.

ROC atau *Receiver Operating Characteristic* menggambarkan hubungan antara *True Positive Rate* dan *False Positive Rate*. *True Positive Rate* menunjukkan proporsi data positif yang berhasil diprediksi dengan benar oleh model. *False Positive Rate* menunjukkan proporsi data negatif yang salah diprediksi sebagai positif. Nilai TPR dan FPR diperoleh dari perubahan *threshold* probabilitas, sehingga kurva ROC menunjukkan kemampuan model dalam membedakan kelas pada berbagai kondisi keputusan.

Persamaan *True Positive Rate* (2.27), yang menunjukkan proporsi data kelas positif yang berhasil diprediksi dengan benar oleh model. Sedangkan persamaan *False Positive Rate* (2.28) menunjukkan proporsi data kelas negatif yang salah diprediksi sebagai positif. Kedua persamaan ini digunakan untuk membentuk titik pada kurva ROC ketika nilai *threshold* prediksi diubah.

$$TPR = \frac{TP}{(TP + FN)} \quad (2.27)$$

$$FPR = \frac{FP}{(FP + TN)} \quad (2.28)$$

AUC atau *Area Under the Curve* didefinisikan sebagai luas area di bawah kurva ROC. Secara matematis, AUC dapat dinyatakan sebagai integral dari TPR terhadap FPR dengan persamaan (2.29).

$$AUC = \int_0^1 TPR(FPR) dFPR \quad (2.29)$$

Persamaan tersebut menunjukkan bahwa AUC merangkum seluruh kemampuan model dalam membedakan kelas positif dan negatif pada berbagai nilai *threshold*. Nilai AUC berada pada rentang 0 sampai 1. Semakin tinggi nilai AUC, semakin baik kemampuan model dalam memberi skor yang lebih tinggi pada kelas positif dibandingkan kelas negatif. Weiss (2013, hlm. 21) menjelaskan bahwa ROC analisis dan AUC sering digunakan pada data tidak seimbang karena dapat membantu membandingkan model tanpa hanya bergantung pada *accuracy*. Dengan demikian, ROC-AUC dapat digunakan sebagai metrik tambahan untuk menilai kemampuan diskriminatif model klasifikasi.

2.2.13 Classification Report

Classification report merupakan ringkasan evaluasi model klasifikasi yang menyajikan beberapa metrik performa secara terstruktur pada setiap kelas. Panji Bintoro dkk. (2024, hlm. 8) menjelaskan bahwa *precision*, *Recall*, dan *F1-score* merupakan metrik yang lebih rinci untuk mengevaluasi model klasifikasi, terutama ketika terdapat ketidakseimbangan kelas. Dengan demikian, *classification report* dapat digunakan untuk melihat performa

model secara lebih detail dibandingkan *accuracy* karena hasil evaluasi tidak hanya ditampilkan secara umum, tetapi juga dibaca berdasarkan kemampuan model pada masing masing kelas.

Classification report umumnya diturunkan dari *Confusion matrix* dan memuat *metrik precision, Recall, dan f1 score*. *Precision* didefinisikan sebagai proporsi prediksi positif yang benar terhadap seluruh prediksi positif yang dibuat model, dengan persamaan (2.30). *Recall* didefinisikan sebagai proporsi data positif yang berhasil dikenali oleh model, dengan persamaan (2.31). *Precision* berfokus pada ketepatan prediksi, sedangkan *Recall* berfokus pada kelengkapan pengenalan kelas positif. Kedua metrik ini memberikan sudut pandang yang berbeda terhadap kinerja model.

$$Precision = \frac{TP}{(TP + FP)} \quad (2.30)$$

$$Recall = \frac{TP}{(TP + FN)} \quad (2.31)$$

$$f1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.32)$$

F1-score merupakan metrik gabungan yang menyeimbangkan *precision* dan *Recall*. Panji Bintoro dkk. (2024, hlm. 50) menjelaskan bahwa *F1-score* merupakan rata rata harmonis dari *precision* dan *Recall*, sehingga metrik ini dapat menunjukkan keseimbangan antara ketepatan prediksi dan kemampuan model dalam menemukan kelas positif. Nilai *F1-score* yang tinggi menunjukkan bahwa model tidak hanya tepat dalam memberikan prediksi positif, tetapi juga mampu mengenali data positif aktual dengan baik.

Pada data tidak seimbang, *classification report* penting karena membantu peneliti menilai apakah model hanya unggul pada kelas mayoritas atau mampu mengenali kelas minoritas. (Panji Bintoro dkk., 2024, hlm. 21–22) menjelaskan bahwa *precision* dan *Recall* dapat digunakan untuk menilai performa *classifier* pada kelas minoritas, sedangkan F1-measure menjadi ukuran gabungan yang menempatkan *precision* dan *Recall* secara seimbang. Oleh karena itu, *classification report* dapat digunakan untuk membaca kekuatan dan kelemahan model pada tiap kelas, terutama ketika distribusi data tidak seimbang dan *accuracy* belum cukup untuk menjelaskan performa klasifikasi secara rinci.

2.2.14 Permutation Importance

Menurut Molnar (2022), *Permutation Importance* merupakan metode interpretabilitas model yang bertujuan mengukur seberapa besar ketergantungan kinerja model terhadap suatu fitur tertentu. Molnar (2022) menekankan bahwa pentingnya fitur tidak diukur dari struktur internal model atau nilai bobot parameter, melainkan dari dampak nyata fitur tersebut terhadap kemampuan prediksi model. Pendekatan ini berangkat dari gagasan bahwa suatu fitur dianggap penting apabila informasi yang dikandungnya benar digunakan oleh model untuk menghasilkan prediksi yang akurat. Jika informasi tersebut dirusak melalui proses permutasi, maka Penurunan kinerja model menjadi indikator langsung tingkat kepentingan fitur tersebut.

Molnar (2022) menjelaskan bahwa prosedur *Permutation Importance* dilakukan setelah model selesai dilatih. Langkah pertama adalah mengukur kinerja awal model pada data uji atau data validasi menggunakan metrik tertentu, misalnya M menyatakan kinerja seperti model seperti, *accuracy*, AUC, atau *error*. Nilai kinerja model sebelum permutasi dinyatakan (2.33).

$$M_{\text{asli}} = M(y, \hat{y}) \quad (2.33)$$

$$M_{\text{permutasi}(j)} = M(y, \hat{y}_{\text{permutasi}(j)}) \quad (2.34)$$

$$\text{Importance}(j) = M_{\text{asli}} - M_{\text{permutasi}(j)} \quad (2.35)$$

Langkah berikutnya adalah melakukan permutasi atau pengacakan nilai pada satu fitur tertentu, sementara fitur lain tetap dipertahankan. Permutasi ini memutus hubungan antara fitur tersebut dengan target tanpa mengubah distribusi nilai fitur secara keseluruhan sehingga untuk suatu fitur j , dilakukan permutasi nilai fitur tersebut diperoleh prediksi baru dan kinerja model setelah permutasi dinyatakan persamaan (2.34). Setelah itu, model yang sama digunakan kembali untuk dan kinerja model diukur ulang. Selisih antara kinerja awal dan kinerja setelah permutasi merepresentasikan tingkat kepentingan fitur atau nilai *Permutation Importance* untuk fitur j kemudian dirumuskan sebagai (2.35). Persamaan ini menunjukkan bahwa semakin besar Penurunan kinerja model setelah fitur dipermutasi, semakin besar kontribusi fitur tersebut terhadap prediksi model

2.2.15 Python

Python merupakan bahasa pemrograman yang dapat digunakan dalam proses pengembangan model *machine learning*. Dalam penelitian ini, Python digunakan sebagai alat bantu komputasi untuk mengolah data, membangun model, dan mengevaluasi hasil prediksi. Panji Bintoro dkk. (2024, hlm. 1–4) menjelaskan bahwa *machine learning* merupakan bagian dari kecerdasan buatan yang memungkinkan komputer belajar dari

data, mengenali pola, dan membuat keputusan tanpa diprogram secara eksplisit untuk setiap tugas. Berdasarkan penjelasan tersebut, Python berperan sebagai perangkat implementasi yang membantu peneliti menerapkan proses pembelajaran mesin ke dalam bentuk program yang dapat dijalankan.

Penggunaan Python dalam *machine learning* berkaitan dengan kebutuhan pengolahan data dan pemodelan. komponen utama *machine learning* terdiri dari data, algoritma, model, dan evaluasi. Data digunakan sebagai dasar pembelajaran, algoritma digunakan untuk menemukan pola, model menjadi hasil pembelajaran dari data, dan evaluasi digunakan untuk mengukur kinerja model. Dalam konteks penelitian ini, Python digunakan untuk menjalankan tahapan tersebut secara sistematis, mulai dari membaca *dataset*, melakukan praproses, melatih model, sampai menghitung metrik evaluasi (Panji Bintoro dkk., 2024, hlm. 4–8).

Python juga mendukung proses pengembangan model secara bertahap. Panji Bintoro dkk. (2024, hlm. 14–18) menjelaskan bahwa pengembangan model *machine learning* meliputi pengumpulan data, pemrosesan dan pembersihan data, pembagian data, pemilihan algoritma, pelatihan model, serta evaluasi dan *fine-tuning*. Tahapan tersebut sesuai dengan kebutuhan penelitian prediktif karena model perlu dibangun dari data latih, diuji menggunakan data uji, lalu dievaluasi menggunakan metrik yang sesuai. Dengan demikian, Python dapat digunakan sebagai lingkungan kerja untuk menyusun alur pemodelan secara terstruktur dan dapat diuji ulang.

Dalam implementasi *machine learning*, Python sering digunakan bersama *Library* pendukung. algoritma *machine learning* dapat diimplementasikan dengan bantuan perangkat lunak seperti R, Python melalui *Library* Scikit-learn, atau MATLAB. Hal ini menunjukkan bahwa Python relevan digunakan dalam penelitian ini karena menyediakan dukungan komputasi untuk menjalankan algoritma, melakukan praproses data, dan mengevaluasi hasil model.

BAB III

METODE PENELITIAN

3.1 Subjek Penelitian

Subjek penelitian ini adalah CV Dima Perkasa Teknik, yaitu perusahaan pemasok teknis yang menyediakan kebutuhan operasional pelanggan industri, seperti ban industri, suku cadang, dan perlengkapan teknis lainnya. CV Dima Perkasa Teknik merupakan perusahaan yang berlokasi di JL. A.Yani Km: 7 Komplek Griya pemurus indah Blok: F No: 25. Banjarmasin, Kalimantan Selatan.

3.2 Objek Penelitian

Objek penelitian dari penelitian ini data penawaran penjualan pada Perusahaan CV Dima Perkasa Teknik. Dokumen tersebut merupakan arsip historis perusahaan yang digunakan dalam proses penawaran produk kepada pelanggan. Setiap dokumen penawaran menunjukkan aktivitas penjualan yang pernah diajukan perusahaan, baik penawaran yang berhasil maupun tidak berhasil. Data dari dokumen tersebut kemudian digunakan sebagai dasar penelitian untuk membangun model prediksi keberhasilan penawaran penjualan.

3.3 Jenis Data

Data yang digunakan pada penelitian ini menggunakan 2 jenis data, yaitu data primer dan data sekunder, berikut penjelasan dari kedua data yang akan digunakan dalam penelitian ini:

3.3.1 Data Primer

Data Primer adalah data utama yang digunakan sebagai pembangunan model prediksi. Data didapatkan dari transaksi historis dan arsip dokumen penawaran penjualan dari Perusahaan, wawancara secara langsung juga dilakukan dengan tim pemasaran terkait proses bisnis. Data Primer yang diambil pada penelitian sebagai berikut.

1. Data arsip dokumen penawaran penjualan
2. Data historis *invoice*

3.3.2 Data Sekunder

Data sekunder merupakan data yang didapatkan tidak langsung dalam proses pengambilan data. Data sekunder digunakan sebagai data pendukung seperti dokumentasi resmi Python dan scikit learn untuk pembangunan model. Data sekunder juga didapatkan

dari jurnal penelitian yang telah ada dan *dataset* publik implementasi *Artificial Neural Network* dari Kaggle.

3.4 Metode Pengumpulan Data

Pengumpulan data untuk penelitian diambil dengan metode sebagai berikut.

1. Dokumentasi

Kegiatan dokumentasi dilakukan dengan mengambil data dari arsip internal Perusahaan berupa dokumen penawaran penjualan dan, data historis *invoice* penjualan. Data sekunder juga kumpulkan dengan dokumentasi resmi python dan *Library machine learning* seperti *sckit learn*.

2. Wawancara

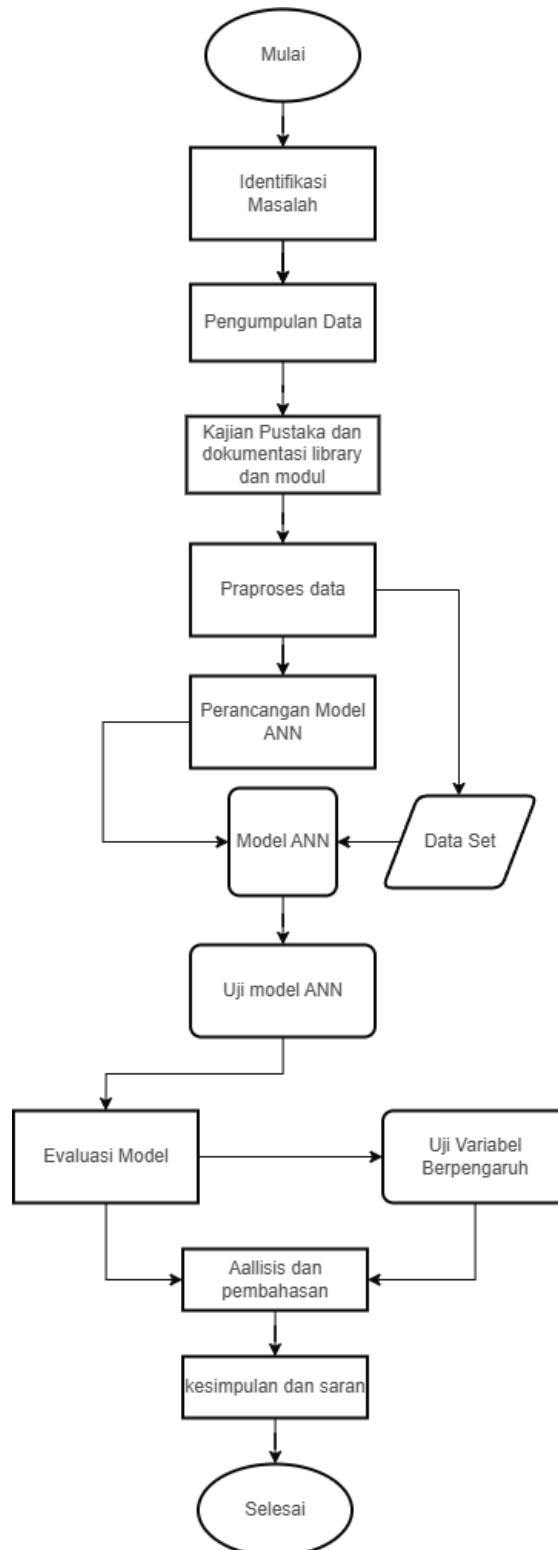
Wawancara dilakukan untuk mengumpulkan informasi terkait proses bisnis penawaran untuk menentukan variabel dalam *dataset* untuk pemodelan *Artificial Neural Network*

3. Studi Pustaka

Studi Pustaka dilakukan sebagai data pendukung penelitian yang dilakukan. Data berupa jurnal penelitian terdahulu dan buku.

3.5 Alur Penelitian

Alur penelitian adalah proses mulai dari sebelum dilakukannya penelitian hingga selesai. Berikut merupakan diagram alir dari penelitian ini.



Gambar 3 1 Diagram alur penelitian

Berikut merupakan detail alur penelitian berdasarkan gambar 3.1.

1. Identifikasi masalah

Pada Tahap ini dilakukan identifikasi masalah terkait kerugian yang dialami, yaitu kegagalan dalam proses penawaran penjualan terhadap klien. Dengan pengamatan terhadap alur penawaran, proses bisnis, serta kendala yang muncul akibat belum adanya pemanfaatan data historis secara optimal. Dari tahap ini dirumuskan permasalahan utama yaitu kebutuhan akan model prediksi berbasis data yang mampu membantu pengambilan keputusan penawaran secara lebih objektif.

2. Pengumpulan Data

Pengumpulan data dilakukan dengan mengambil data berupa dokumen penawaran dan penjualan Perusahaan, data ini digunakan sebagai sumber utama dalam pembangunan *Data Set* penelitian.

3. Kajian Pustaka dan Dokumentasi *Library* dan *Modul*

Tahap ini berisi pengumpulan dan kajian teori yang relevan dengan penelitian, studi literatur terkait *Artificial Neural Network*, *Multilayer Preception*. Pada tahap ini juga dokumentasi *Library* dan modul python yang mendukung dalam membangun mode dan menguji model *Artificial Neural Network*.

4. Praproses data

Tahap Praproses data dilakukan untuk mempersiapkan data set, tahap ini mencakup pembersihan data dari nilai kosong dan duplikat, transformasi data numerik, pengkodean variabel kategori dan validasi data. Sehingga data memiliki kualitas yang baik dan dapat diproses oleh model *Artificial Neural Network*.

5. *Data Set*

Data Set merupakan hasil dari proses praproses data. Data ini telah terstruktur, bersih, numerik, dan tervalidasi. *Data Set* ini menjadi data penelitian yang siap digunakan sebagai *input* pada proses perancangan, pelatihan dan, pengujian model *Artificial Neural Network*.

6. Perancangan Model *Artificial Neural Network*

Pada tahap perancangan Model *Artificial Neural Network* pada penelitian ini dilakukan tiga metode untuk menangani kondisi ketidakseimbangan kelas pada data penawaran, yaitu model *Artificial Neural Network Baseline*, Model *Artificial Neural Network* dengan pembobotan kelas (*Class Weigthed*) dan, model *Artificial Neural Network* dengan *upsampling* kelas minoritas. Semua pendekatan dirancang

menggunakan arsitektur dan parameter pelatihan yang sama agar perbandingan kinerja bersifat objektif, perbedaan hanya terletak pada penanganan *imbalance class*. Model *baseline* dilatih menggunakan data apa adanya sebagai pembanding awal. Model *Class Weighted* dilatih dengan memberikan bobot lebih besar pada kelas minoritas dan, model *Upsampling* dilatih dengan menambahkan jumlah data kelas minoritas agar distribusi kelas lebih seimbang. Ketiga model dilatih dan diuji menggunakan pembagian data pelatihan dan test yang konsisten. Pada tahap ini juga aktivitas meliputi penentuan jumlah layer, jumlah neuron pada setiap layer, fungsi aktivasi, serta parameter pelatihan seperti *learning Rate* dan jumlah iterasi. Perancangan dilakukan secara terstruktur agar model mampu mempelajari hubungan antara variabel *input* dan target secara optimal

7. Model *Artificial Neural Network*

Tahap ini merupakan proses pembentukan dan latih model *Artificial Neural Network* berdasarkan rancangan yang ditetapkan. *Data Set* dimasukkan sebagai *input* ke dalam jaringan saraf dan, variabel target digunakan sebagai fitur yang akan dipelajari oleh model. Model *Artificial Neural Network* pada tahap ini sudah terbentuk secara struktural dan siap untuk menjalani proses pelatihan.

8. Uji Model *Artificial Neural Network*

Pada tahap ini model yang telah dilatih pada pelatihan akan dilatih pada data uji yang telah dibagi sebelumnya. Uji model *Artificial Neural Network* dilakukan untuk mengetahui kemampuan model dalam melakukan prediksi terhadap data yang belum pernah digunakan pada proses pelatihan. Hasil pengujian memberikan pengetahuan mengenai performa model.

9. Evaluasi Model

Evaluasi model dilakukan dengan menganalisis hasil pengujian menggunakan metrik evaluasi yang relevan. Pada tahap ini evaluasi untuk menilai tingkat akurasi, kestabilan dan, kemampuan model.

10. Uji Variabel Berkontribusi

Pada tahap ini dilakukan untuk mengidentifikasi kontribusi dari masing-masing variabel *input* terhadap hasil prediksi model. Uji variabel juga untuk mengetahui variabel mana yang paling dominan mempengaruhi keberhasilan penawaran sehingga hasil bisa mendukung pengambilan Keputusan bisnis.

11. Analisis dan Pembahasan

Pada tahap analisis dan pembahasan dilakukan interpretasi menyeluruh terhadap hasil evaluasi model dan uji variabel berpengaruh. Analisis dilakukan untuk menjelaskan alasan di balik performa model dan makna dari variabel yang berpengaruh terhadap keberhasilan penawaran.

12. Kesimpulan dan Saran

Tahap ini berisi Kesimpulan berdasarkan seluruh hasil analisis yang telah dilakukan. Kesimpulan disusun untuk menjawab rumusan masalah penelitian dan saran berdasarkan keterbatasan penelitian dan hasil yang diperoleh selama proses analisis yang berisi saran praktis bagi Perusahaan dan saran akademik bagi peneliti selanjutnya.

3.6 *Data Set*

Data Set yang digunakan untuk penelitian ini berasal dari dokumen penawaran penjualan CV Dima Perkasa Teknik. Data merupakan arsip Perusahaan yang mencatat aktivitas penawaran kepada user dalam periode tahun 2023 hingga 2025 November. *Data Set* disusun dengan mengonversi dokumen penawaran yang tidak terstruktur dalam bentuk tabel. Setiap baris tabel mewakili 1 dokumen penawaran dan setiap kolom mempresentasikan variabel dan fitur atribut. Berikut merupakan deskripsi dari struktur data set.

Tabel 3 1 *Data Set* Penelitian

No	Kolom Atribut	Jenis Data	Deskripsi
1.	nomor_penawaran	Teks	Nomor identitas dari dokumen penawaran
2.	Tanggal_penawaran	Tanggal	Tanggal penawaran dikeluarkan
3.	Kode_pelanggan	Teks	Kode identitas pelanggan
4.	Jenis_pelanggan	Teks kategori	jenis pelanggan yang terdiri dari subkontraktor, perseorangan dan, retail.
5.	Kategori_produk	Teks kategori	Jenis produk yang memiliki total nominal terbesar dalam 1 penawaran
6.	Nilai_penawaran_total	Numerik	Total omzet nominal dari 1 penawaran
7.	Margin	Numerik	Selisih dari total nominal omzet dan harga <i>supplier</i>
8.	TOP_hari	Numerik	Kesepakatan waktu pembayaran dengan pelanggan
9.	Jumlah_item	Numerik	Total jenis item dalam 1 penawaran
10.	Pelanggan_baru	Biner	Hubungan dengan pelanggan, 1 untuk pelanggan baru, 0 untuk pelanggan lama
11.	Barang_campuran	Biner	Keterangan jenis produk dalam 1 penawaran, 1 untuk penawaran terdiri dari lebih satu jenis barang, 0 untuk penawaran yang terdiri satu jenis barang.
12.	Y1_aktual	Biner	Status aktual penawaran berhasil diterima atau tidak.

3.7 Variabel Model

Variabel model merupakan atribut data yang digunakan dalam proses pembangunan model prediksi *Artificial Neural Network*. Variabel ini dipilih berdasarkan data historis penawaran CV Dima Perkasa Teknik yang telah melalui tahap praproses data. Tidak semua kolom dalam *dataset* digunakan sebagai variabel model, karena terdapat beberapa kolom yang hanya berfungsi sebagai identitas administratif, seperti nomor_penawaran,

tanggal_penawaran, dan kode_pelanggan. Kolom tersebut tidak digunakan sebagai variabel *input* karena tidak merepresentasikan karakteristik penawaran yang secara langsung dipelajari oleh model.

Pada penelitian ini, variabel model dibagi menjadi dua jenis, yaitu variabel *input* dan variabel *output*. Variabel *input* merupakan variabel yang digunakan oleh model untuk mempelajari pola keberhasilan penawaran. Variabel *output* merupakan target prediksi yang menunjukkan status aktual penawaran. Variabel *input* disimbolkan sebagai X, sedangkan variabel *output* disimbolkan sebagai Y. Model *Artificial Neural Network* mempelajari hubungan antara variabel *input* dan variabel *output* untuk menghasilkan prediksi peluang keberhasilan penawaran.

Tabel 3.2 Variabel Model *Artificial Neural Network*

No	Jenis variabel	Nama Variabel	Jenis Data
1	Input (X)	jenis_pelanggan	Kategori
2	Input (X)	kategori_produk	Kategori
3	Input (X)	nilai_penawaran_total	Numerik
4	Input (X)	margin	Numerik
5	Input (X)	TOP_hari	Numerik
6	Input (X)	Jumlah_item	Numerik
7	Input (X)	Pelanggan_baru	Biner
8	Input (X)	barang_campuran	Biner
9	Input (Y)	Y1_aktual	Biner

Pemilihan variabel model pada penelitian ini dilakukan berdasarkan keterkaitan setiap atribut dengan proses penawaran penjualan di CV Dima Perkasa Teknik. Variabel yang digunakan tidak hanya dipilih karena tersedia dalam *dataset*, tetapi juga karena memiliki hubungan dengan keputusan pelanggan dalam menerima atau menolak penawaran. Model *Artificial Neural Network* membutuhkan variabel *input* yang mampu merepresentasikan karakteristik transaksi, pelanggan, produk, harga, margin, dan skema pembayaran. Oleh karena itu, variabel yang bersifat administratif seperti nomor_penawaran, tanggal_penawaran, dan kode_pelanggan tidak digunakan sebagai *input* model karena hanya berfungsi sebagai identitas dokumen, bukan sebagai pola yang dapat menjelaskan peluang keberhasilan penawaran.

Variabel `jenis_pelanggan`, `kategori_produk`, dan `nilai_penawaran_total` digunakan karena ketiganya menggambarkan karakter awal dari suatu penawaran. Jenis pelanggan menunjukkan pihak yang menerima penawaran, seperti subkontraktor, retail, atau perseorangan. Setiap jenis pelanggan dapat memiliki pola kebutuhan, cara negosiasi, dan kemampuan pembayaran yang berbeda. Kategori produk digunakan karena produk yang ditawarkan dapat memiliki tingkat kebutuhan dan urgensi yang tidak sama. `Nilai_penawaran_total` melengkapi dua variabel tersebut karena menunjukkan besarnya nominal transaksi yang diajukan kepada pelanggan. Hubungan ketiga variabel ini penting karena keputusan pelanggan tidak hanya dipengaruhi oleh siapa pelanggan tersebut, tetapi juga oleh jenis produk yang dibutuhkan dan besar nilai penawaran yang harus dipertimbangkan.

Penggunaan `jenis_pelanggan` dan `kategori_produk` juga membantu model mengenali pola yang bersifat kategori dalam data penawaran. Kedua variabel ini tidak dapat langsung diproses dalam bentuk teks, sehingga perlu diubah menjadi bentuk numerik melalui proses *one-hot encoding*. Setelah dikodekan, model dapat mempelajari apakah jenis pelanggan tertentu cenderung memiliki peluang berhasil yang lebih tinggi pada kategori produk tertentu. `Nilai_penawaran_total` kemudian memberikan informasi kuantitatif terhadap pola tersebut. Dengan kombinasi ini, model dapat menangkap hubungan antara karakter pelanggan, karakter produk, dan besar transaksi dalam menentukan status keberhasilan penawaran.

Variabel `margin`, `TOP_hari`, dan `jumlah_item` digunakan karena ketiganya menggambarkan strategi penawaran yang diberikan perusahaan kepada pelanggan. `Margin` menunjukkan selisih antara harga jual dan harga *supplier*, sehingga variabel ini berkaitan langsung dengan strategi keuntungan dan daya saing harga. `TOP_hari` menunjukkan jangka waktu pembayaran yang diberikan kepada pelanggan. Variabel ini penting karena dalam transaksi B2B, pelanggan sering mempertimbangkan kelonggaran pembayaran selain harga. `Jumlah_item` menunjukkan banyaknya item dalam satu dokumen penawaran, sehingga dapat menggambarkan tingkat kompleksitas transaksi yang diajukan.

Hubungan antara `margin`, `TOP_hari`, dan `jumlah_item` menjadi penting karena ketiganya dapat saling memengaruhi dalam proses keputusan pelanggan. Penawaran dengan margin tinggi dapat memberikan keuntungan lebih besar bagi perusahaan, tetapi berpotensi membuat harga kurang kompetitif. Sebaliknya, `TOP_hari` yang lebih panjang

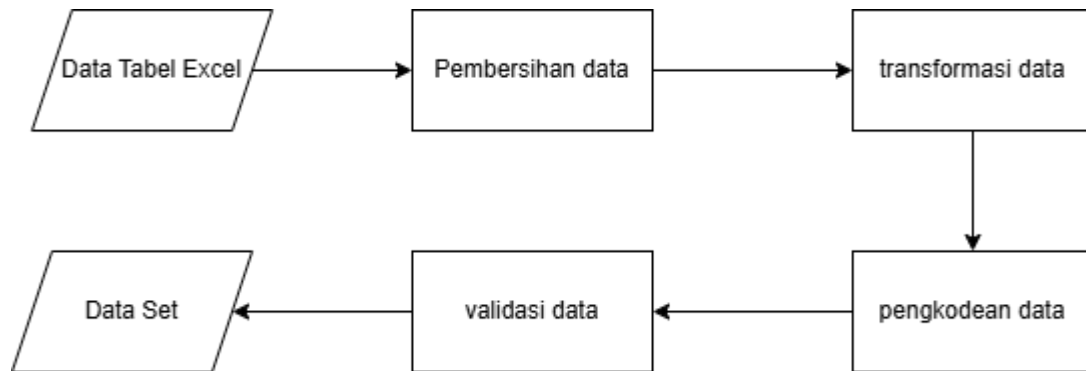
dapat membuat penawaran lebih menarik bagi pelanggan, tetapi dapat meningkatkan risiko bagi perusahaan. Jumlah_item juga dapat memengaruhi proses negosiasi karena semakin banyak item dalam satu penawaran, semakin besar kemungkinan pelanggan melakukan pertimbangan ulang terhadap harga, ketersediaan barang, dan kebutuhan aktual. Dengan memasukkan ketiga variabel ini, model dapat mempelajari pola hubungan antara strategi harga, skema pembayaran, dan kompleksitas penawaran.

Variabel pelanggan_baru, barang_campuran, dan Y1_aktual digunakan untuk melengkapi struktur model dari sisi hubungan pelanggan, variasi penawaran, dan target prediksi. Pelanggan_baru menunjukkan apakah pelanggan sudah pernah bertransaksi atau belum dengan perusahaan. Status ini penting karena pelanggan lama biasanya memiliki riwayat hubungan dan tingkat kepercayaan yang berbeda dibandingkan pelanggan baru. Barang_campuran menunjukkan apakah satu penawaran terdiri dari satu jenis barang atau lebih dari satu jenis barang. Variabel ini digunakan karena penawaran dengan barang campuran dapat memiliki pola keputusan yang berbeda dibandingkan penawaran yang hanya berisi satu jenis barang.

Y1_aktual digunakan sebagai variabel *output* karena menunjukkan hasil akhir dari proses penawaran, yaitu berhasil atau tidak berhasil. Nilai 1 menunjukkan penawaran berhasil menjadi transaksi, sedangkan nilai 0 menunjukkan penawaran tidak berhasil. Variabel pelanggan_baru dan barang_campuran berperan sebagai informasi pendukung yang dapat membantu model memahami konteks hubungan pelanggan dan struktur produk dalam penawaran. Seluruh variabel *input* kemudian dipelajari oleh model *Artificial Neural Network* untuk memprediksi Y1_aktual. Dengan susunan ini, model tidak hanya membaca data sebagai arsip penawaran, tetapi mengubah data historis menjadi dasar prediksi keberhasilan penawaran secara lebih objektif.

3.8 Praproses Data

Proses praproses data dilakukan untuk mengubah data mentah menjadi *Data Set* yang layak untuk pemodelan *Artificial Neural Network*. Karena, model *Artificial Neural Network* memerlukan kualitas *Data Set* yang baik sehingga model dapat mempelajari pola yang benar dan menghasilkan prediksi yang akurat.



Gambar 3 2 Alur Sistematis Praproses Data

Berdasarkan gambar yang menjelaskan alur sistematis praproses data dari data mentah berbentuk Excel menjadi *Data Set* yang layak untuk model prediksi *Artificial Neural Network*. Berikut detail alur dari praproses data.

1. Pembersihan Data

Pada tahap pembersihan data dilakukan pemeriksaan data yang kosong dan data penawaran yang duplikat, baris yang memiliki nilai kolom memiliki atribut kurang dari 3 akan di hapus dan nomor penawaran yang memiliki duplikat akan dihapus

2. Transformasi Data

Pada tahap ini, data akan ditransformasi sesuai tipenya. Jenis data teks menjadi string, jenis data numerik menjadi integer atau float. Filter label pada tipe jenis data biner juga dilakukan untuk mematiskan semua *value* 0 dan 1. Pada tahap ini juga mengisi nilai yang masih kosong dengan median maupun NaN.

3. Pengkodean data

Pada tahap pengkodean data dilakukan proses *one-hot encoding*, proses mengubah data kategori menjadi numerik biner. Tahap ini dilakukan karena model *Artificial Neural Network* hanya dapat memproses data dalam bentuk numerik.

4. Validasi Data

tahap validasi data adalah proses mengecek kembali semua apakah masih ada nilai kosong dan tipe data yang tidak sesuai dengan kolom atribut pada data set. Pada tahap inilah *Data Set* divalidasi bisa digunakan untuk model *Artificial Neural Network*

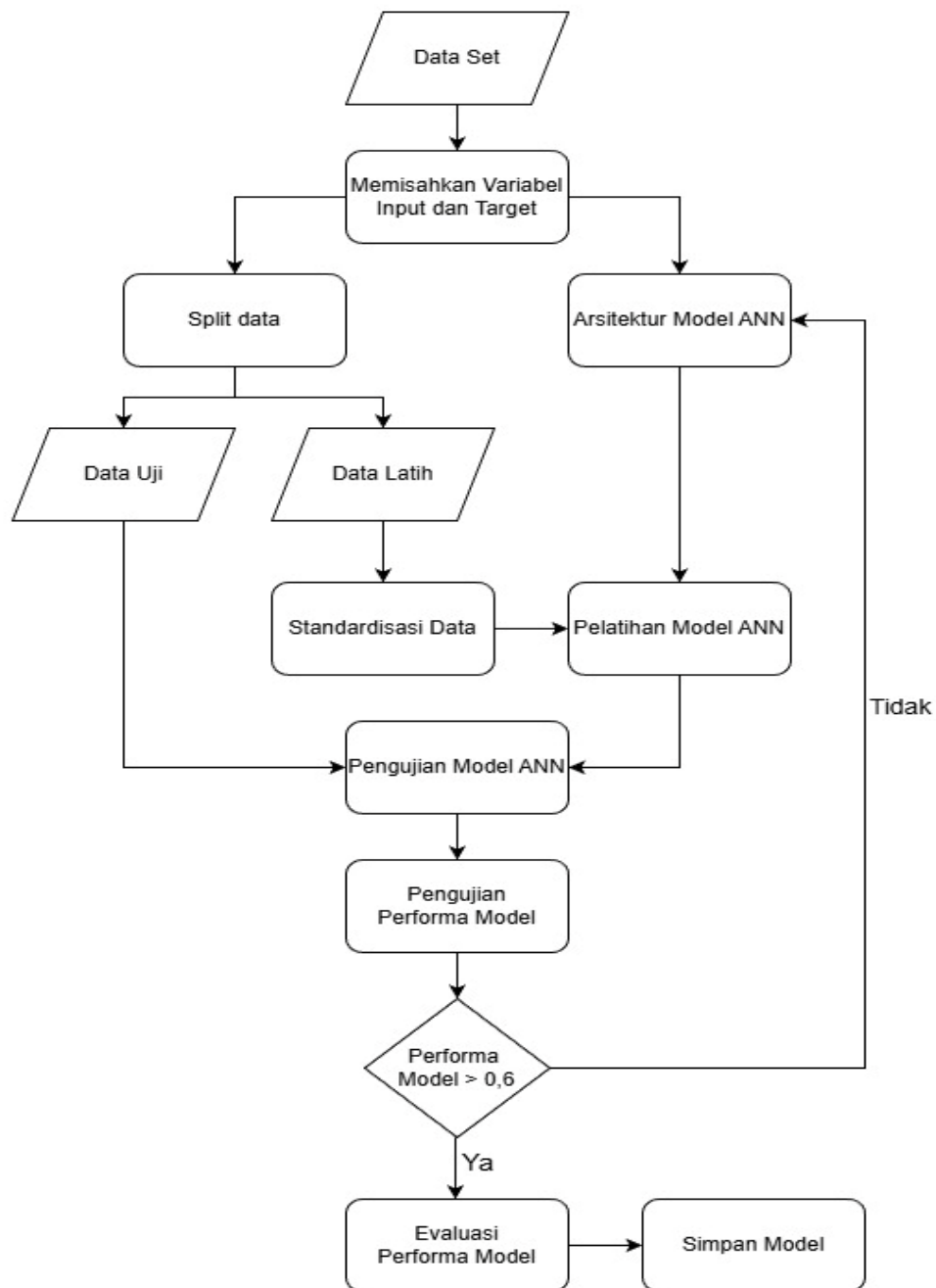
3.9 Perancangan Model *Artificial Neural Network*

Alur Perancangan Model *Artificial Neural Network* di susun secara sistematis untuk menghasilkan model prediksi yang akurat, karena terdapat ketidakseimbangan kelas *Data Set* pada variabel Y1_Aktual, maka dilakukan model alternatif dengan metode yang

berbeda. Perancangan Model prediksi *Artificial Neural Network* menggunakan 3 pendekatan. Model *Artificial Neural Network baseline*, Model *Artificial Neural Network Upsampling* dan, Model *Artificial Neural Network Class Weighted*.

3.9.1 Model *Artificial Neural Network Baseline*

model *Artificial Neural Network baseline* pada penelitian ini disusun sebagai tahapan awal untuk membangun model prediksi yang menjadi acuan utama sebelum dilakukan pendekatan lainnya. Alur proses perancangan model *Artificial Neural Network baseline* pada penelitian ini menggambarkan tahapan kerja yang dimulai dari pengolahan *dataset* hingga diperoleh model akhir yang memenuhi kriteria performa yang ditetapkan.



Gambar 3.3 Diagram Alur proses percangan model *Artificial Neural Network* baseline.

Berdasarkan gambar 3.3 berikut detail alur proses percangan model *Artificial Neural Network* baseline.

1. Data Set

Data Set yang telah melewati proses praproses dimasukkan untuk menjadi sumber data yang akan digunakan dan dipeleajari oleh model *Artificial Neural Network*.

2. Memisahkan Variabel *Input* dan Target

Pada tahap ini dilakukan pemisahan variabel *input* dan variabel target. Variabel *input* berisi fitur yang akan digunakan sebagai layer *input* pada model dan, variabel target merupakan variabel yang ingin diprediksi.

3. Split Data

Data Set dipisahkan menjadi dua bagian, yaitu data latih dan data uji dengan rasio 80% data latih dan 20% data uji. Pemilihan rasio tersebut didasarkan pada kebutuhan model *Artificial Neural Network* untuk memperoleh porsi data latih yang lebih besar, karena jumlah data historis penawaran terbatas dan distribusi kelas tidak seimbang. Fahrudin dkk. (2024) menunjukkan bahwa rasio data latih dan data uji memengaruhi hasil klasifikasi, serta rasio 80:20 menghasilkan akurasi hingga 97% pada dataset obesitas. Azhima dkk. (2024) juga membandingkan rasio 90:10, 80:20, dan 70:30 pada klasifikasi penyakit stroke menggunakan *Backpropagation Neural Network*, dengan hasil akurasi rata-rata tertinggi sebesar 96,14% pada rasio 80:20. Berdasarkan temuan tersebut, rasio 80:20 dipilih karena dapat memberikan jumlah data latih yang lebih dominan untuk proses pembelajaran model, tetapi tetap menyediakan data uji terpisah untuk mengevaluasi kemampuan generalisasi model.

4. Data Latih

Data latih merupakan data yang akan digunakan untuk melatih model *Artificial Neural Network*. Data ini akan melalui proses pelatihan berulang untuk pembelajaran bobot dan bias jaringan.

5. Data Uji

Data uji merupakan data yang tidak digunakan dalam proses pelatihan. Data ini digunakan setelah model selesai dilatih untuk mengevaluasi kinerja prediksi model terhadap data yang belum pernah dilihat sebelumnya

6. Standardisasi Data

Tahap ini dilakukan standardisasi data latih menggunakan *StandardScaler*. Proses ini bertujuan untuk menyamakan skala seluruh variabel *input* atau fitur sehingga memiliki rata-rata nol dan simpangan baku satu untuk menjaga stabilitas proses pembelajaran.

7. Arsitektur Model *Artificial Neural Network*

Tahap ini mendefinisikan struktur *Artificial Neural Network* yang digunakan, termasuk jumlah layer, jumlah neuron pada setiap layer dan, fungsi aktivasi yang diterapkan.

Arsitektur model dirancang sebagai kerangka utama yang menentukan bagaimana data diproses dan dilatih dalam jaringan.

8. Pelatihan Model *Artificial Neural Network*

Pelatihan model dilakukan dengan proses pembelajaran melalui *forward propagation* untuk menghasilkan prediksi awal. Perhitungan nilai kesalahan dan pembaruan bobot dengan *backpropagation*, proses pelatihan diulang hingga diperoleh parameter model yang sesuai dengan pola data.

9. Pengujian Model *Artificial Neural Network*

Pada tahap pengujian, model yang telah di latih digunakan untuk melakukan prediksi pada data uji, kemudian model menghasilkan nilai prediksi berdasarkan bobot dan bias hasil pelatihan tanpa mengubah parameter.

10. Pengujian Performa Model

Hasil prediksi model dibandingkan dengan nilai aktual pada data uji untuk menghitung metrik performa model. Tahap ini bertujuan menilai sejauh mana model mampu memprediksi data secara akurat dan membedakan kelas dengan baik.

11. Performa Model $> 0,6$

Pada tahap ini dilakukan pengambilan keputusan berdasarkan nilai performa yang diperoleh. Jika nilai performa belum memenuhi batas kelayakan yang ditetapkan, maka proses kembali ke tahap perancangan dan pelatihan model untuk dilakukan penyesuaian. Jika nilai performa telah memenuhi kriteria, maka model dianggap layak digunakan. Nilai tersebut digunakan karena ROC-AUC di bawah 0,6 menunjukkan kemampuan klasifikasi yang tidak lebih baik dari klasifikasi acak, sedangkan ROC-AUC 0,6 sampai 0,7 menunjukkan performa yang masih rendah tetapi sudah berada di atas klasifikasi acak. Adityaningrum dkk. (2025, hlm. 159) menjelaskan bahwa AUC 0,6 sampai 0,7 menunjukkan *poor performance*, sedangkan AUC 0,5 sampai 0,6 menunjukkan performa yang tidak lebih baik daripada *random classification*. Oleh karena itu, nilai 0,6 digunakan sebagai batas minimum awal untuk menilai apakah model sudah memiliki kemampuan diskriminasi terhadap kelas berhasil dan gagal

12. Evaluasi Performa Model

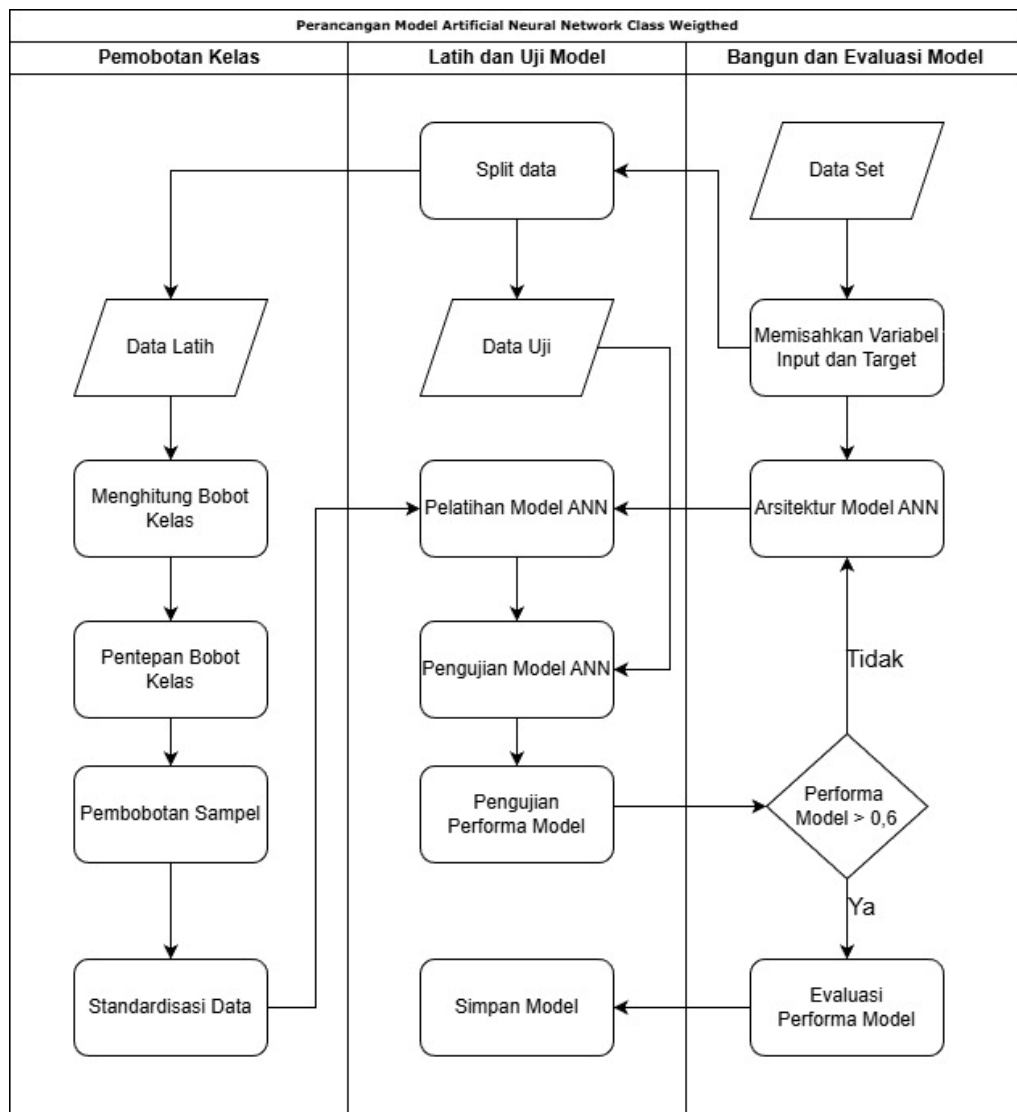
Evaluasi dilakukan untuk menganalisis hasil performa model secara keseluruhan dan mendalam. Evaluasi digunakan sebagai dasar pembahasan kemampuan model dalam memprediksi variabel target.

13. Simpan Model

Setelah proses perancangan model dan evaluasi model telah memenuhi kriteria performa. Model disimpan untuk digunakan pada uji variabel yang paling berpengaruh.

3.9.2 Model *Artificial Neural Network Class Weighted*

Model *Artificial Neural Network* dengan pendekatan *Class Weighted* dikembangkan sebagai pengembangan dari model *Artificial Neural Network baseline* untuk menangani permasalahan ketidakseimbangan kelas pada data penelitian. Pada model *baseline*, seluruh data latih diperlakukan memiliki kontribusi yang sama dalam proses pembelajaran, Pendekatan *Class Weighted* memperkenalkan mekanisme pembobotan kelas pada tahap pelatihan model, di mana setiap kelas diberikan bobot yang berbeda sesuai dengan proporsi kemunculannya dalam data latih.



Gambar 3 4 Diagram alur proses *Artificial Neural Network Class Weighted*

Perbedaan utama proses pada model ini terletak pada tahap sebelum dan saat pelatihan, yaitu adanya perhitungan dan penerapan bobot kelas yang memengaruhi perhitungan kesalahan dan pembaruan bobot jaringan, sementara tahapan lain seperti arsitektur model, pengujian, dan evaluasi tetap mengikuti alur yang sama dengan model *baseline*. berikut detail dari alur pembobotan kelas sesuai gambar 3.4.

1. Menghitung Bobot Kelas

Pada Tahap ini data latih dilakukan perhitungan bobot tiap kelas berdasarkan proporsi kemunculannya dalam data latih. Kelas dengan jumlah data lebih sedikit akan memperoleh bobot yang lebih besar, sedangkan kelas mayoritas memperoleh bobot yang lebih kecil untuk menyeimbangkan kontribusi kesalahan antar kelas selama proses pelatihan.

2. Penetapan Bobot Kelas

Bobot kelas yang telah dihitung kemudian ditetapkan sebagai parameter pelatihan model *Artificial Neural Network*. Bobot ini akan digunakan pembelajaran untuk menyesuaikan besarnya penalti kesalahan pada setiap kelas, sehingga kesalahan pada kelas minoritas memiliki pengaruh yang lebih besar terhadap pembaruan bobot jaringan.

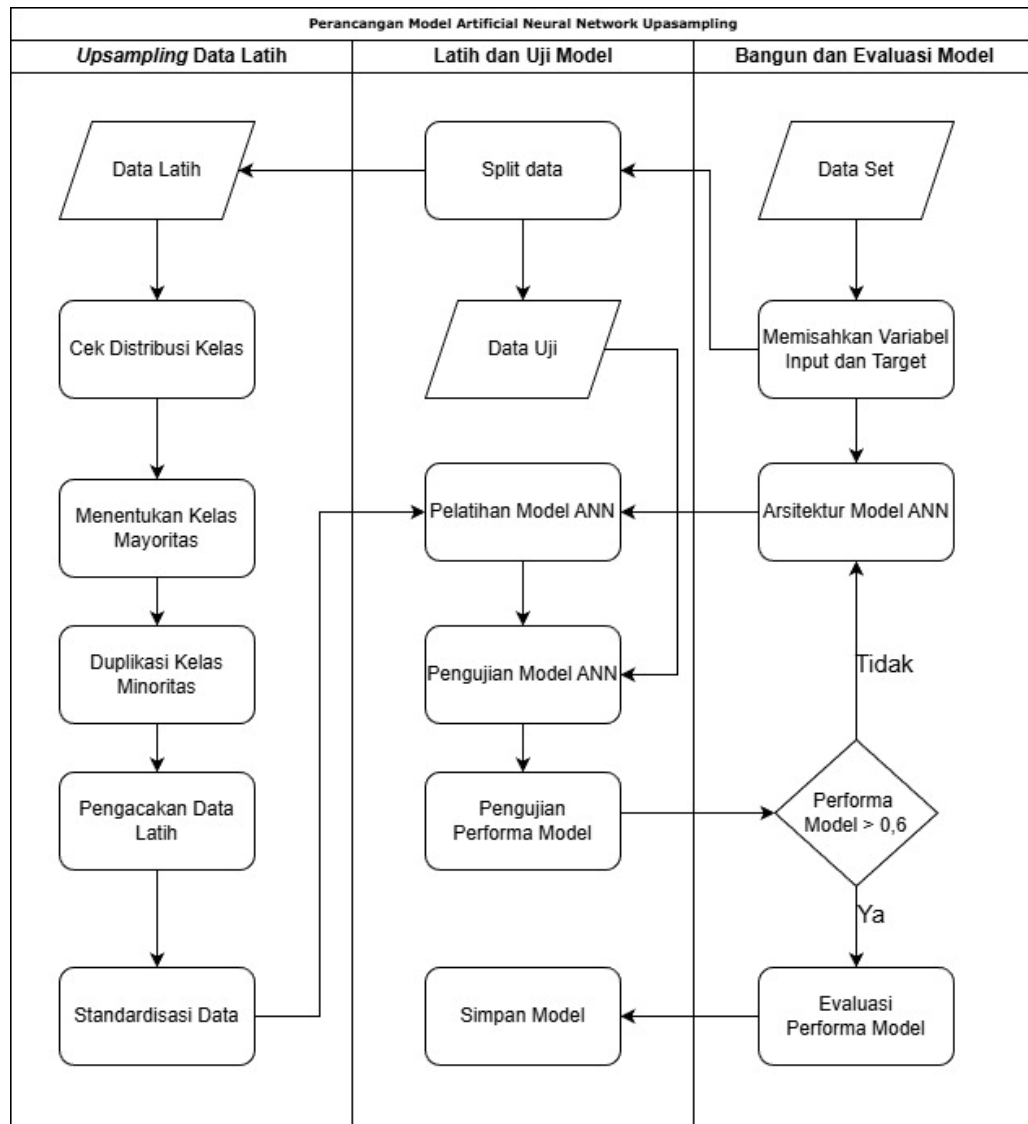
3. Pembobotan Sampel

Pada tahap pembobotan sampel, bobot kelas diterapkan secara tidak langsung pada setiap sampel data latih sesuai dengan kelasnya masing-masing. Sampel dari kelas minoritas akan memiliki kontribusi kesalahan yang lebih besar dalam *loss function*, sedangkan sampel dari kelas mayoritas memiliki kontribusi yang lebih kecil. Proses ini memastikan bahwa model tidak mengabaikan kelas dengan jumlah data yang lebih sedikit.

3.9.3 Model *Artificial Neural Network Upsampling*

Upsampling digunakan sebagai pendekatan alternatif dari model *Artificial Neural Network baseline* untuk menangani ketidakseimbangan kelas pada data latih. Berbeda dengan Pendekatan *Class Weighted* yang memperbaiki kondisi tersebut dengan memberikan bobot kesalahan yang lebih besar pada kelas minoritas tanpa mengubah jumlah sampel data latih. metode *Upsampling* melakukan penyesuaian secara langsung pada struktur data latih dengan menambah jumlah sampel pada kelas minoritas melalui proses duplikasi. Dengan demikian, perbedaan utama metode *Upsampling* terletak pada perubahan distribusi data sebelum proses pelatihan, bukan pada mekanisme perhitungan

kesalahan seperti pada *Class Weighted*. Pendekatan ini bertujuan memberikan representasi data yang lebih seimbang kepada model selama proses pembelajaran, sehingga pengaruh kelas minoritas dapat ditingkatkan tanpa mengubah arsitektur dasar *Artificial Neural Network* yang digunakan.



Gambar 3 5 Diagram alur proses *Artificial Neural Network Upsampling*

Perbedaan proses perancangan model *Artificial Neural Network Upsampling* hanya terdapat perbedaan pada proses *Upsampling* data latih, berikut merupakan detail dari proses *Upsampling* data latih pada gambar 3.5.

1. Data Latih

Data latih menjadi fokus utama dalam metode *Upsampling* karena hanya data inilah yang akan dimodifikasi untuk menyeimbangkan distribusi kelas. Data uji tetap dibiarkan dalam kondisi asli agar evaluasi model tetap objektif.

2. Cek Distribusi Kelas

Pada tahap ini dilakukan analisis distribusi kelas pada data latih untuk mengetahui perbandingan jumlah sampel antar kelas.

3. Menentukan Kelas Mayoritas

Berdasarkan hasil analisis distribusi kelas, kelas dengan jumlah sampel terbanyak ditetapkan sebagai kelas mayoritas. Penentuan ini untuk menyesuaikan jumlah sampel pada kelas minoritas agar seimbang dengan kelas mayoritas.

4. Duplikasi Kelas Minoritas

Pada tahap ini dilakukan proses duplikasi sampel dari kelas minoritas hingga jumlahnya mendekati atau sama dengan jumlah kelas mayoritas. Proses duplikasi dilakukan tanpa mengubah nilai fitur, sehingga karakteristik asli data tetap dipertahankan.

5. Pengacakan Data Latih

Pengacakan data latih dilakukan untuk menghilangkan pola urutan yang mungkin muncul akibat proses *Upsampling*. Pengacakan ini penting agar model tidak mempelajari urutan data tertentu yang dapat memengaruhi stabilitas pembelajaran

6. Standardisasi Data

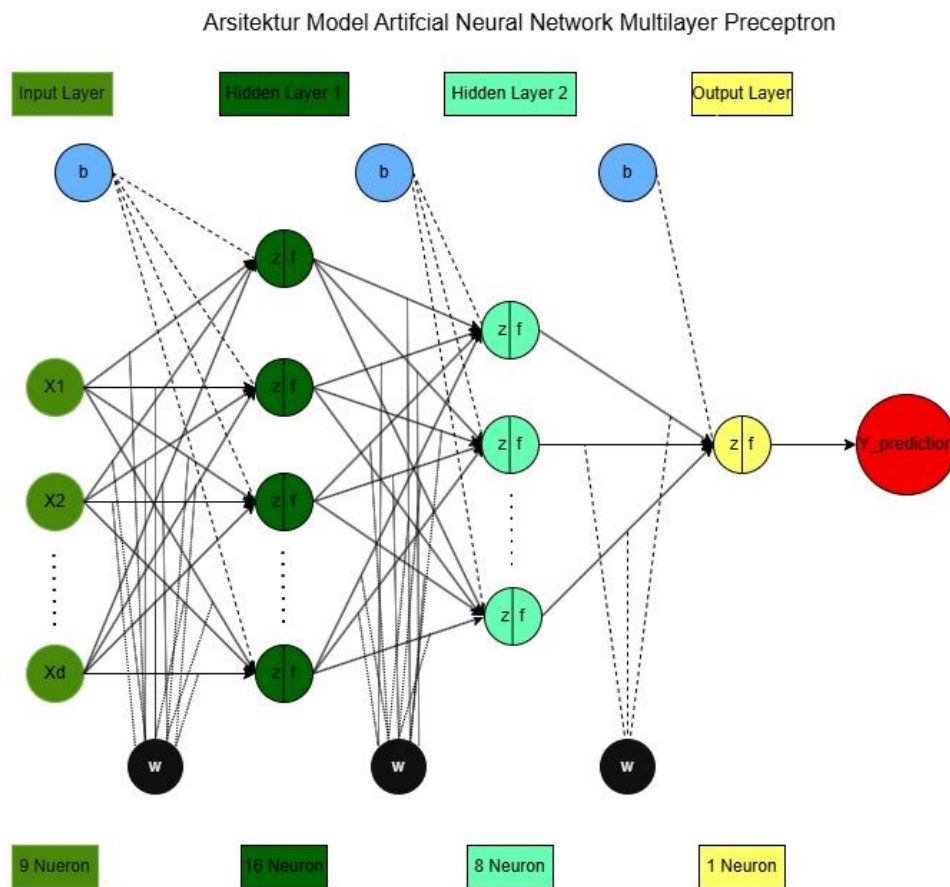
Data latih hasil *Upsampling* kemudian di standardisasi menggunakan *StandardScaler*. Standardisasi dilakukan untuk menyamakan skala seluruh fitur sebelum data dimasukkan ke dalam model *Artificial Neural Network*. Proses ini mengikuti prinsip yang sama dengan model *baseline*, yaitu parameter standardisasi dihitung dari data latih dan digunakan kembali pada data uji.

3.10 Arsitektur Model *Artificial Neural Network*

Arsitektur model *Artificial Neural Network* dirancang menggunakan pendekatan *Multilayer Perceptron* untuk memprediksi keberhasilan penawaran. Arsitektur disusun untuk merepresentasikan hubungan non linier antara variabel penawaran dan status keberhasilan transaksi secara terstruktur dan sistematis. Perancangan arsitektur ini dilakukan sebelum tahap implementasi pemrograman agar struktur jaringan, alur pemrosesan data, serta fungsi setiap komponen dapat dipahami secara konseptual dengan metode yang digunakan pada tahap pelatihan model.

Alur pemrosesan data pada arsitektur ini dimulai dari lapisan *input* menuju lapisan *output* melalui proses *forward propagation*. Data fitur dari *input* layer diteruskan ke lapisan tersembunyi pertama, diproses melalui bobot, bias, dan fungsi aktivasi, lalu diteruskan ke lapisan tersembunyi kedua dengan mekanisme yang sama. Hasil dari

lapisan tersembunyi kedua kemudian masuk ke lapisan *output* untuk menghasilkan nilai prediksi.



Gambar 3.6 Lapisan Model *Artificial Network Multilayer Preception*

Lapisan *input* merupakan lapisan pertama pada arsitektur model yang berfungsi menerima data masukan. Pada penelitian ini, lapisan *input* terdiri dari 9 neuron yang merepresentasikan sembilan fitur hasil seleksi variabel dan praproses data. Setiap neuron pada lapisan *input* menerima satu nilai fitur dan meneruskannya ke lapisan tersembunyi pertama tanpa melakukan proses pembelajaran. Lapisan ini tidak memiliki bobot yang dipelajari, karena perannya hanya sebagai penghubung antara data mentah dan proses komputasi pada jaringan saraf. Pemilihan jumlah neuron berdasarkan jumlah yang sama dengan jumlah fitur bertujuan menjaga seluruh informasi relevan dari data.

Lapisan kedua yaitu *hidden layer* pertama dirancang dengan 16 neuron dan berfungsi mempelajari kombinasi awal serta interaksi non linier antar variabel *input*. Setiap neuron pada lapisan ini melakukan operasi penjumlahan linear antara *input* dan bobot, kemudian ditambahkan dengan bias, yang direpresentasikan dalam diagram sebagai node b seperti pada persamaan (2.1). Hasil operasi ini menghasilkan nilai z yang selanjutnya diproses

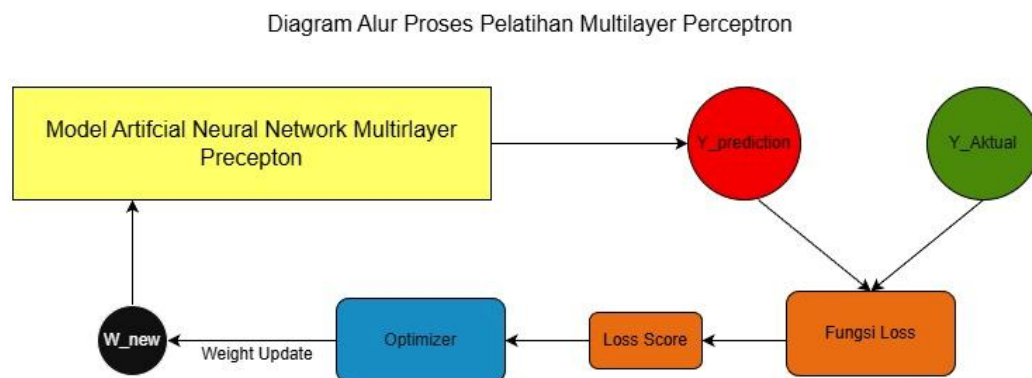
menggunakan fungsi aktivasi ReLU pada node f mengubah z menjadi nilai aktivasi a yang diteruskan ke neuron selanjutnya dengan persamaan (2.4). Penggunaan ReLU dipilih karena mampu memperkenalkan sifat non linier pada model dengan efisien. ReLU juga mempertahankan aliran bernilai *input* positif sehingga proses pelatihan berjalan lebih cepat dibandingkan fungsi aktivasi lain.

Lapisan *hidden layer* kedua terdiri dari 8 neuron dan dirancang sebagai lapisan penyaring informasi. Struktur ini bersifat menyempit dari 16 ke 8 neuron, yang bertujuan mengurangi kompleksitas model sekaligus mempertahankan pola penting yang relevan terhadap target prediksi. Proses komputasi pada lapisan ini sama seperti lapisan sebelumnya, yaitu operasi linear dengan bobot dan bias yang diikuti fungsi aktivasi ReLU. Pengurangan jumlah neuron pada lapisan ini dilakukan secara sengaja untuk mengendalikan kompleksitas model dan menekan risiko *overfitting*, mengingat jumlah data tidak terlalu besar. Struktur menyempit ini membantu model mempertahankan informasi penting sekaligus membuang *noise* yang kurang berkontribusi terhadap prediksi keberhasilan penawaran

Lapisan *output* terdiri dari satu neuron karena tujuan model adalah menghasilkan satu nilai keluaran yang merepresentasikan probabilitas keberhasilan penawaran. Lapisan ini menerima hasil aktivasi dari lapisan tersembunyi kedua dan memprosesnya melalui operasi linear dengan bobot dan bias. Selanjutnya, nilai z pada lapisan *output* diproses menggunakan fungsi aktivasi *sigmoid* menggunakan persamaan (2.3) untuk menghasilkan nilai keluaran kontinu pada rentang 0 hingga 1 yang diinterpretasikan sebagai probabilitas keberhasilan penawaran. Pendekatan ini dipilih karena lebih fleksibel dibandingkan keluaran biner langsung, terutama untuk mendukung analisis lanjutan seperti penentuan ambang keputusan atau prioritas tindak lanjut penawaran. proses penentuan kelas menggunakan *threshold default* 0,5 digunakan oleh fungsi untuk mengubah probabilitas menjadi label kelas biner pada $Y_{\text{prediction}}$ yang akan dibandingkan dengan Y_{Aktual} .

3.11 Proses Pelatihan Model *Artificial Neural Network*

Diagram alur proses pelatihan *Artificial Neural Network* menggambarkan mekanisme pembelajaran model *Multilayer Perceptron* secara iteratif. Proses ini menunjukkan bagaimana model menghasilkan prediksi, menghitung kesalahan, memperbarui bobot, dan mengulangi siklus pembelajaran hingga diperoleh model yang optimal, model yang digunakan adalah *Multilayer Perceptron*. Algoritma ini termasuk jaringan saraf *feedforward*, di mana aliran data bergerak satu arah dari lapisan *input* menuju lapisan *output* pada saat proses prediksi. Proses pembelajaran dilakukan melalui mekanisme *backpropagation*, yaitu algoritma yang menghitung kesalahan prediksi, kembali ke seluruh lapisan jaringan untuk memperbarui bobot dan bias. *Backpropagation* digunakan untuk mengetahui seberapa besar kontribusi setiap bobot terhadap nilai kesalahan yang dihasilkan.



Gambar 3 7 Proses Pelatihan Model

Alur proses pelatihan model seperti gambar 3.7 dimulai dengan data *input* hasil praproses dimasukkan ke dalam model dan diproses melalui mekanisme *forward propagation* yang menghasilkan nilai *Y_Prediction*. Nilai *Y_prediction* kemudian dibandingkan dengan *Y_Aktual* yang berasal dari data historis penawaran. *Y_Aktual* merepresentasikan kondisi sebenarnya dari setiap penawaran, apakah berhasil atau tidak. Perbandingan antara nilai prediksi dan nilai aktual ini menjadi dasar utama dalam mengevaluasi kinerja model.

Perbedaan kedua nilai dihitung menggunakan *loss function*. Pada penelitian ini digunakan *loss function binary cross entropy* dengan persamaan (2.16), karena target prediksi bersifat biner dan model menghasilkan keluaran probabilitas. *Loss function* ini

menghitung besarnya kesalahan prediksi dengan mempertimbangkan tingkat keyakinan model. Prediksi yang salah dengan tingkat keyakinan tinggi akan menghasilkan nilai *loss* yang besar, sedangkan prediksi yang benar dengan tingkat keyakinan tinggi akan menghasilkan nilai *loss* yang kecil. Hasil dari perhitungan ini berupa *loss score* yang merepresentasikan tingkat kesalahan model.

Nilai *loss* yang dihasilkan kemudian digunakan dalam proses *backpropagation* untuk menghitung gradien terhadap setiap parameter jaringan. Gradien ini menunjukkan arah dan besar perubahan bobot yang diperlukan agar nilai *loss* dapat diperkecil pada iterasi berikutnya. Namun, gradien saja belum cukup untuk memperbarui bobot secara langsung. Sehingga diperlukan algoritma optimasi yang mengatur bagaimana bobot diperbarui secara efektif. Pada penelitian ini digunakan *Optimizer* LBFGS.

Optimizer LBFGS berperan pada tahap setelah perhitungan nilai *loss* dan sebelum pembaruan bobot jaringan. Tujuan penggunaan LBFGS adalah memastikan proses pelatihan berjalan lebih stabil dan efisien pada *dataset* berukuran kecil, serta menghasilkan model dengan performa prediksi yang baik, bukan untuk memaksa model mencapai konvergensi penuh.

Hasil dari proses optimasi ini adalah bobot baru yang direpresentasikan pada diagram sebagai W_{new} . Bobot baru tersebut mencakup seluruh parameter jaringan, termasuk bobot dan bias pada setiap lapisan. Bobot yang telah diperbarui kemudian dikembalikan ke model *Artificial Neural Network* untuk digunakan pada proses pelatihan berikutnya.

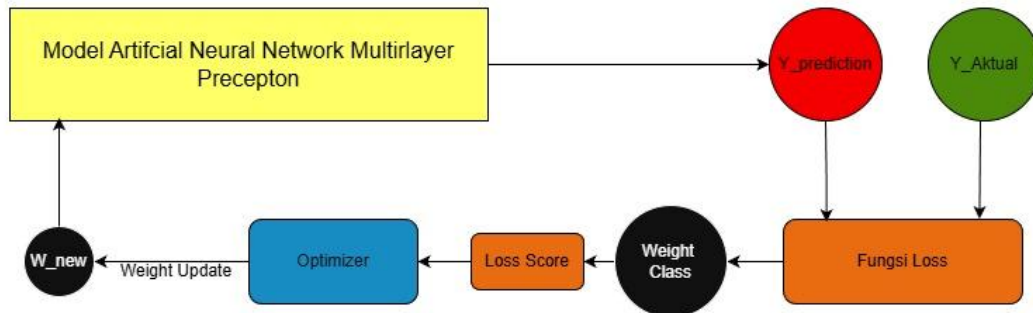
Proses pelatihan berlangsung secara berulang dalam bentuk *loop* iteratif, di mana setiap iterasi digunakan untuk menyesuaikan bobot jaringan berdasarkan kesalahan prediksi yang dihasilkan. Pada setiap siklus pelatihan, bobot diperbarui secara bertahap agar kemampuan model dalam memetakan pola data semakin baik. Pelatihan dihentikan pada kondisi tertentu, seperti tercapainya jumlah iterasi yang telah ditetapkan atau ketika perubahan nilai *loss* tidak lagi memberikan peningkatan kinerja prediksi yang signifikan berdasarkan hasil evaluasi.

3.12 Proses Pelatihan Model *Artificial Neural Network Class Weighted*

Perbedaan mendasar proses pelatihan model *Artificial Neural Network Class Weighted* dibandingkan model *baseline* terletak pada keberadaan komponen *Weight class* di antara *loss function* dan *loss score*. Pada model *baseline*, jalur ini tidak ada sehingga semua kesalahan diperlakukan sama. Dengan adanya *class weight*, proses pelatihan menjadi lebih sensitif terhadap kelas minoritas tanpa mengubah struktur arsitektur jaringan

maupun mekanisme optimasi yang digunakan. Pendekatan ini memungkinkan model belajar secara lebih seimbang pada data dengan distribusi kelas yang tidak merata.

Diagram Alur Proses Pelatihan Multilayer Perceptron Class Weight



Gambar 3 8 Proses Pelatihan Model *Class Weight*

Weight class digunakan untuk menyesuaikan kontribusi kesalahan prediksi berdasarkan kelas data. Setiap kelas diberikan bobot tertentu yang diterapkan pada perhitungan *loss* sehingga kesalahan pada kelas minoritas memiliki pengaruh yang lebih besar terhadap *loss score* dan diterapkan untuk tiap sampel. Perhitungan nilai *loss score* dimulai dengan menghitung nilai *loss* dasar untuk setiap sample menggunakan *loss function* klasifikasi biner untuk menghitung selisih antara nilai *Y_prediction* dan *Y_Aktual*.

Setelah *loss* dasar didapatkan, *loss score* dihitung dengan mengalikan nilai *loss* dasar setiap sampel dengan bobot kelas sesuai label aktualnya sehingga menghasilkan persamaan (2.36). Bobot bersifat tetap selama proses pelatihan dan dipetakan ke setiap sampel data latih sesuai label kelasnya.

$$loss_i = w_k \times loss_dasar_i \quad (2.36)$$

Loss score berbobot kemudian digunakan sebagai dasar perhitungan gradien pada proses *backpropagation*, gradien ini selanjutnya digunakan oleh *optimizer* untuk memperbarui bobot dan bias jaringan. Proses pelatihan berlangsung dalam bentuk loop iteratif sama seperti model *Artificial Neural Network* baseline, proses pelatihan hanya membuat model menjadi lebih sensitif terhadap kelas yang jumlah datanya lebih sedikit tanpa mengubah struktur arsitektur jaringan atau menambah jumlah data latih.

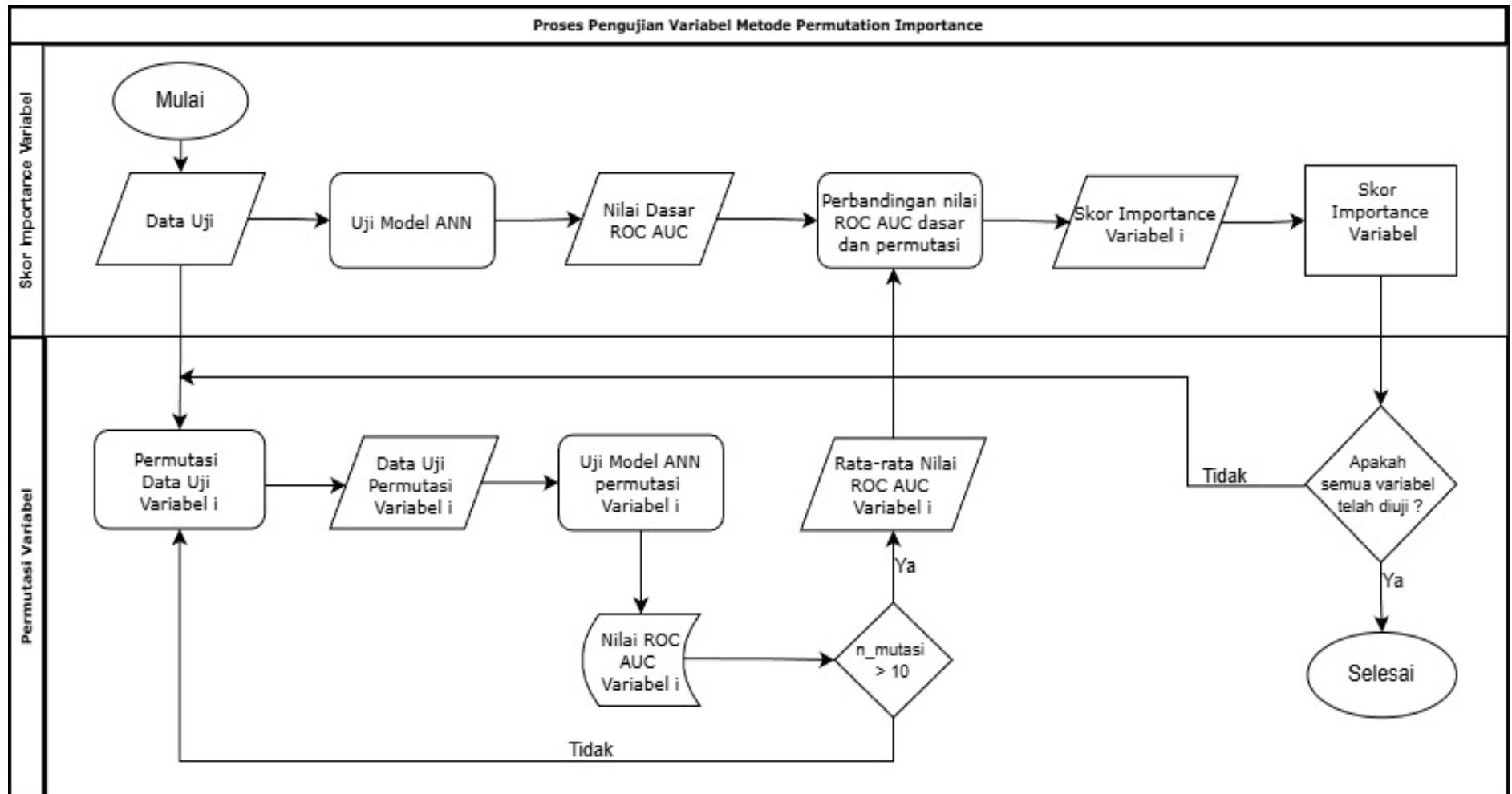
3.13 Pengujian Variabel Kontribusi

Pengujian variabel berkontribusi pada penelitian ini dilakukan untuk mengidentifikasi tingkat kontribusi setiap variabel *input* terhadap kinerja model *Artificial Neural Network*

yang telah dibangun. Model *Artificial Neural Network* tidak menyediakan koefisien yang dapat langsung menunjukkan pengaruh setiap variabel *input*. Model bekerja melalui banyak lapisan dan bobot yang saling berinteraksi sehingga kontribusi variabel tidak dapat dijelaskan secara langsung, maka digunakan pendekatan berbasis kinerja model, yaitu metode *Permutation Importance*. Metode ini mengevaluasi pengaruh variabel dengan mengamati perubahan kinerja model ketika informasi suatu variabel dihilangkan secara sistematis melalui proses permutasi pada data uji.

Metode *Permutation Importance* bekerja dengan mengukur sensitivitas kinerja model terhadap gangguan informasi pada setiap variabel *input*. Pada pendekatan ini, informasi suatu variabel dihilangkan diacak melalui proses permutasi pada data uji, yaitu dengan mengubah nilai variabel tanpa mengubah distribusi statistiknya. Proses ini memastikan bahwa perubahan kinerja model yang terjadi disebabkan oleh hilangnya hubungan variabel dengan target, bukan akibat perubahan distribusi atau karakteristik data.

Seperti pada gambar 3.9 proses pengujian variabel berpengaruh dimulai dari penggunaan data uji yang telah melalui tahap praproses dan memiliki struktur fitur yang sama dengan data pelatihan. Data uji dipilih karena merepresentasikan kemampuan generalisasi model *Artificial Neural Network* terhadap data yang belum pernah dilihat sebelumnya. Pada tahap awal, data uji dimasukkan ke dalam model *Artificial Neural Network* untuk menghasilkan prediksi probabilitas. Hasil prediksi ini digunakan untuk menghitung nilai ROC-AUC dasar yang berfungsi sebagai kinerja acuan model ketika seluruh variabel *input* masih berada dalam kondisi utuh dan informatif.



Gambar 3 9 Proses Uji Variabel Berkontribusi

Nilai ROC-AUC dasar kemudian dijadikan pembanding dalam seluruh proses pengujian variabel. proses dilanjutkan dengan pengujian variabel *input* secara individual menggunakan metode *Permutation Importance*. Pada tahap ini, satu variabel ke i pada data uji dipilih untuk dimutasi. Permutasi dilakukan dengan cara mengubah nilai variabel tanpa mengubah distribusi nilainya. proses ini bertujuan untuk menghilangkan keterkaitan informasi variabel ke i terhadap target tanpa mengubah karakteristik statistik data.

Data uji yang telah mengalami permutasi pada variabel ke i kemudian digunakan kembali untuk menguji model *Artificial Neural Network* yang sama. Model tidak mengalami pelatihan ulang dan seluruh bobot jaringan tetap dipertahankan. Model menghasilkan prediksi baru berdasarkan data uji hasil permutasi, kemudian dihitung nilai ROC-AUC variabel ke i . Nilai ini mencerminkan kinerja model ketika informasi dari variabel ke i tidak lagi tersedia secara efektif dalam proses prediksi.

Untuk menjaga kestabilan dan mengurangi pengaruh acak, proses permutasi pada variabel ke i dilakukan secara berulang hingga jumlah iterasi yang telah ditentukan tercapai, yaitu lebih dari sepuluh kali pengulangan. Setiap iterasi menghasilkan nilai ROC-AUC yang berbeda. Seluruh nilai ROC-AUC hasil permutasi kemudian dirata-ratakan sehingga diperoleh nilai rata-rata ROC-AUC variabel ke i . Nilai rata-rata ini digunakan sebagai representasi kinerja model setelah informasi variabel ke i dihilangkan.

Tahap selanjutnya adalah melakukan perbandingan antara nilai ROC-AUC dasar dan nilai rata-rata ROC-AUC hasil permutasi variabel ke i . Selisih antara kedua nilai tersebut dihitung dan ditetapkan sebagai skor *Importance* variabel ke i . Skor ini menunjukkan besarnya Penurunan kinerja model akibat hilangnya informasi dari variabel ke i . Semakin besar nilai selisih yang dihasilkan, semakin besar pengaruh variabel tersebut terhadap kemampuan prediksi model *Artificial Neural Network*.

Proses pengujian ini dilakukan secara iteratif untuk seluruh variabel *input*. Setiap variabel diuji satu per satu hingga seluruh variabel mengalami proses permutasi dan perhitungan skor *Importance*. Setelah semua variabel selesai diuji, skor *Importance* seluruh variabel dikumpulkan dan diurutkan dari nilai tertinggi

ke nilai terendah. Hasil pengurutan ini digunakan untuk mengidentifikasi variabel yang paling berpengaruh terhadap kinerja model *Artificial Neural Network*.

BAB IV

PEMBANGUNAN SISTEM

4.1 Pengumpulan Data

Data yang telah diambil akan di ubah ke dalam format tabel Excel, data mentah yang semula berbentuk dokumen penawaran menjadi data terstruktur dalam bentuk tabel Excel. Unit analisis yang digunakan adalah satu dokumen penawaran. Artinya, satu baris data di Excel mewakili satu penawaran yang dikirimkan perusahaan kepada satu pelanggan pada satu waktu tertentu. Pemilihan unit ini dilakukan agar setiap baris data langsung berkaitan dengan keputusan bisnis, yaitu apakah penawaran tersebut menjadi *Purchase Order* dan bagaimana perilaku pembayarannya.

Proses ekstraksi data akan mengambil nilai yang terdapat dalam dokumen penawaran dan catatan transaksi untuk mengisi *value* dari beberapa variabel input (X) yang telah dijelaskan pada bab sebelumnya, dan akan digunakan dalam pembentukan model *Artificial Neural Network* untuk menghasilkan prediksi variabel output (Y) sebagai hasil dari pembentukan mode. Excel akan di bagi menjadi beberapa sheet yang berisi beberapa informasi yang telah di ekstraksi dari dokumen. Data yang akan digunakan untuk uji Pembangunan model *Artificial Neural Network* Adalah file excel “data_penawran_dima.xlsx” , dan berikut adalah kolom atributnya:

Tabel 4 1 Kolom atribut data penawaran

Nama Kolom Atribut	Jenis data
nomor_penawaran	Teks
tanggal_penawaran	<i>Date</i>
kode_pelanggan	Teks kategori
kategori_produk	Kategori
nilai_penawaran_total	Numerik
margin	Numerik
TOP_hari	Numerik (integer)
jumlah_item	Numerik (integer)
Pelanggan_baru	Data Bernoulli (0 atau 1)
Barang_campuran	Data Bernoulli (0 atau 1)
Y1_aktual	Data Bernoulli (0 atau 1)

Setelah dilakukan proses ekstraksi data dengan membuat data menjadi atribut yang sesuai variabel input (X), berikut struktur tabel excel yang akan digunakan sebagai variabel uji pada model *Artificial Neural Network* yang akan dibangun:

	A	B	C	D	E	F	G	H	I	J	K	L
1	nomor_penawaran	tanggal_penawaran	kode_pelanggan	jenis_pelanggan	kategori_produk	nilai_penawaran_total	margin	TOP_hari	jumlah_item	pelanggan_baru	barang_campuran	Y1_aktual
76	033/DPT/I/2022											
77	034/DPT/I/2022											
78	035/DPT/V/I/2022	16/06/2023										
79	036/DPT/V/I/2022	16/06/2022	SUB005	subkontraktor	ban	1850000	15%	0	1	1	0	1
80	037/DPT/VII/2022	11/07/2022	SUB001	subkontraktor	bagian_ban			60	6	0	0	1
81	038/DPT/VII/2022	07/07/2022	SUB001	subkontraktor	bagian_ban	196810000	25%	60	19	0	1	0
82	039/DPT/X/2022	20/09/2022	SUB001	subkontraktor	ban	2950000	15%	0	1	0	0	1
83	040/DPT/X/2022	19/09/2022	SUB001	subkontraktor	bagian_ban	186969500	20%	60	26	0	1	1
84	041/DPT/X/2022	04/10/2022	SUB001	subkontraktor	ban	154550000	25%	60	25	0	1	0
85	042/DPT/X/2022	04/10/2022	SUB001	subkontraktor	ban	8850000	25%	60	1	0	0	1
86	043/DPT/X/2022	06/10/2022	SUB001	subkontraktor	ban	74440000	25%	60	10	0	0	1
87	044/DPT/X/2022	26/10/2022	SUB001	subkontraktor	ban	51120000	25%	60	7	0	0	1
88	045/DPT/XI/2022	16/11/2022		subkontraktor	ban	32000000	25%	60	2	0	0	1
89	046/DPT/XI/2022	07/11/2022	SUB005	subkontraktor	ban	1400000	15%	0	1	0	0	1
90	047/DPT/X/2022	14/10/2022	SUB001	subkontraktor	ban	146822500	0,25	60	25	0	1	0
91	048/DPT/X/2022	31/10/2022	SUB001	subkontraktor	bagian_ban	21350000	25%	0	6	0	0	0
92	049/DPT/I/2022											
93	050/DPT/I/2022											
94	051/DPT/XI/2022	17/11/2022	SUB001	subkontraktor	bagian_ban	12700000	15%	0	7	0	1	1
95	052/DPT/I/2022											
96	053/DPT/I/2022											
97	054/DPT/I/2022											
98	055/DPT/XII/2022	02/12/2022	SUB001	subkontraktor	ban	12650000	35%	60	4	0	0	1
99	001/DPT/I/2023	05/01/2023	SUB001	subkontraktor	bagian_ban		35%	60	9	0	1	1
100	002/DPT/I/2023	05/01/2023	SUB001	subkontraktor	bagian_ban		35%	60	9	0	1	1
101	003/DPT/I/2023	05/01/2023	SUB001	subkontraktor	bagian_ban	9450000	35%	60	1	0	0	1
102	004/DPT/I/2023	05/01/2023	SUB001	subkontraktor	ban	2450000	35%	60	1	0	0	1
103	005/DPT/I/2023	06/01/2023	SUB001	subkontraktor	bagian_ban	156500000	20%	60	6	0	0	0

Gambar 4. 1 Tabel data excel

4.2 Pengolahan Data

Sebelum sumber data digunakan sebagai model *Artificial Neural Network* (Artificial Neural Network), data perlu diolah agar sesuai kebutuhan. Pada pengolahan data untuk Model *Artificial Neural Network*, pengolahan data terjadi 4 tahap. Tahap pertama adalah proses Pembersihan data, tahapan tersebut dilakukan untuk membersihkan data yang kosong dan menghilangkan duplikat, tujuan pembersihan data untuk memastikan *Artificial Neural Network* belajar dari data yang konsisten.

Tahap kedua adalah Transformasi data, tujuannya untuk mencegah variabel dengan nilai besar yang mendominasi. *Artificial Neural Network* sebagai algoritma berbasis gradien sangat sensitif terhadap perbedaan skala., maka dari itu variabel numerik perlu dinormalisasi ke rentang 0 hingga 1 agar proses optimasi menjadi stabil.

Tahap ketiga adalah pengkodean data atau *encoding*, digunakan untuk mengubah variabel kategori menjadi bentuk numerik, karena *Artificial Neural Network* tidak dapat memproses data dalam bentuk teks seperti jenis_pelanggan dan kategori_produk maka dari itu perlu menggunakan teknik *one hot Encoding* untuk mengubah setiap kategori

menjadi variabel biner. Tujuannya agar *Artificial Neural Network* dapat mempelajari perbedaan karakteristik tiap kategori.

Tahapan terakhir adalah validasi data. Pada tahap ini dilakukan pengecekan akhir sebelum data digunakan untuk pelatihan dan testing. Memastikan tidak ada lagi nilai kosong, semua variabel numerik pada rentang yang benar, label target hanya berisi nilai 0 dan 1. Tujuan validasi data untuk menjamin bahwa seluruh struktur data sudah siap dan tidak menimbulkan *error* saat model *Artificial Neural Network* dilatih dan diuji.

4.2.1 Pembersihan data

Sebelum data di transformasi menjadi numerik dan rentang 0 dan 1, data perlu di cek dan dibersihkan untuk menghindari data yang berisi nilai kosong, duplikasi penawaran yang sama dapat menyebabkan model mempelajari pola yang salah, Tahap awal pembersihan *Import Library* pandas dan os, *Library* pandas sering digunakan secara umum pada *preprocessing* data, dan *library* os mempermudah interaksi dengan file data.

```
import os
import pandas as pd

def clean_transform(
    input_path="data/data_penawran_dima.xlsx",
    output_path="data/data_model_ann.csv",
):
```

Gambar 4. 2 *Import Library* dan Definisi Path

Kemudian dilakukan dengan mendefinisikan fungsi mentah dan, `output_path`: *output* data csv bersih yang siap digunakan untuk *Artificial Neural Network*. Setelah itu menggunakan pandas untuk membaca file excel yang didefinisikan sebelumnya sebagai `input_path` dengan fungsi `pd.read_excel`.

```
# Baca data
df = pd.read_excel(input_path)
print(df.shape)
```

Gambar 4. 3 *Input Excel*

File excel yang sudah ditambahkan akan diperiksa dengan metode `df.shape`. *DataFrame* memiliki jumlah 349 baris dan 12 kolom dengan *missing value* berjumlah 1222.

```
Shape: 349 baris x 12 kolom
Total missing: 1222
```

Gambar 4. 4 Bentuk Data

Langkah selanjutnya adalah proses pembersihan data dengan membuang baris yang hampir kosong, dengan menggunakan metode `df.notna()` untuk menunjukkan DataFrame mana sel yang bukan NaN (tidak ada *value*). Menentukan parameter baris dan kolom dengan sintak `.sum(axis=1) > 3`], digunakan untuk menghitung beberapa banyak kolom yang terisi per baris dan menyaring hanya baris yang memiliki lebih dari 3 kolom terisi. Baris yang hanya memiliki *value* kurang dari 3 kolom akan dibersihkan karena bukan transaksi valid atau hanya hasil *input* yang salah.

```
# Buang baris hampir kosong
df = df[df.notna().sum(axis=1) > 3].copy()
```

Gambar 4. 5 Kode drop *Missing Value*

Langkah berikutnya adalah menghilangkan nomor_penawaran yang sama karena akan membuat model terbias ke penawaran yang diulang. Menggunakan metode `df.drop_duplicates` untuk menghapus baris duplikat.

```
# Hilangkan duplikat
df = df.drop_duplicates(subset=["nomor_penawaran"])
print(df.shape)
```

Gambar 4. 6 Kode menghilangkan duplikat

Dengan metode yang sama yaitu `df.shape`, untuk melihat baris dan kolom yang tersisa setelah proses pembersihan data dengan membuang data memiliki *value* kurang dari 3 dalam 1 baris dan menghilangkan nomor_penawaran yang sama berjumlah 241 baris dan 12 kolom dengan total *missing value* sebanyak 37.

```
Shape: 241 baris x 12 kolom
Total missing: 37
```

Gambar 4. 7 Hasil *Cleaning*

4.2.2 Transformasi data

Setelah data dibersihkan maka perlu dilakukan transformasi yaitu mengubah data numerik menjadi seragam dan berdistribusi normal, karena model *Artificial Neural Network* hanya bisa membaca data numerik yang berada dalam skala yang seragam, Variabel yang beragam dan memiliki bentuk berbeda seperti nilai margin yang masih dalam persen, angka dengan satuan tidak sama, dapat menyebabkan model tidak akurat dalam menemukan pola karena rentang angka yang tidak terkontrol.

Langkah pertama dalam transformasi data adalah mengubah tipe tanggal dahulu ke tipe tanggal yang standar agar dapat dipakai untuk analisis berbasis waktu, dengan fungsi `pd.to_datetime` untuk mengubah isi kolom menjadi format *datetime*. Parameter

`errors="coerce"` dipakai supaya nilai yang tidak valid, misalnya salah ketik format tanggal atau sel kosong, otomatis diubah menjadi NaT.

```
#Tipe tanggal
df["tanggal_penawaran"] = pd.to_datetime(df["tanggal_penawaran"], errors="coerce")
df["tanggal_penawaran"] = df["tanggal_penawaran"].fillna(df["tanggal_penawaran"].median())
```

Gambar 4. 8 Kode *datetime*

Setelah konversi dilakukan, nilai tanggal yang kosong atau gagal dikonversi diisi menggunakan nilai median dari kolom tanggal. Pemilihan median digunakan karena lebih tahan terhadap nilai ekstrem, misalnya jika terdapat tanggal yang jauh di luar rentang normal akibat *input* salah. Dengan dua langkah ini, kolom `tanggal_penawaran` menjadi bersih dan seragam.

Langkah Berikutnya merapikan teks kategori untuk mempersiapkan variabel kategori yang bersih dan konsisten sebelum proses *encoding*. Menggunakan metode `astype(str)` untuk mengubah kolom untuk memastikan kolom bertipe string, metode `strip()` untuk menghapus spasi di awal dan akhir, metode `str.lower()` untuk mengubah semua huruf jadi kecil. Kolom atribut `jenis_pelanggan` akan dipetakan angka untuk membuat data lebih mudah di dipahami dengan `replace()`.

```
# Rapikan teks kategori
df["jenis_pelanggan"] = df["jenis_pelanggan"].astype(str).str.strip().str.lower().replace(
    {"1": "subkontraktor", "2": "retail", "3": "perseorangan"})
df["kategori_produk"] = df["kategori_produk"].astype(str).str.strip().str.lower()
```

Gambar 4. 9 Kode Datatype teks kategori

Setelah transformasi kolom atribut yang memiliki data kategori selanjutnya, transformasi kolom yang memiliki data tipe numerik. Kolom atribut `margin` dengan metode `astype(str)` untuk mengubah tipe data menjadi string dan mengubah semua karakter yang bukan angka, titik dan, minus menjadi kosong dan yang akan disimpan ke data series variabel `margin`.

```
#Margin ke angka 0-1
margin = df["margin"].astype(str).str.replace("[^0-9.-]", "", regex=True)
margin = pd.to_numeric(margin, errors="coerce")
margin = margin.where(margin <= 1, margin / 100)
df["margin"] = margin
print(df["margin"].sample(5))
```

Gambar 4. 10 Kode *Range* Nilai Margin

Setelah itu menggunakan fungsi `pd.to_numeric` untuk mengubah data `margin` menjadi tipe numerik dan menetapkan parameter `errors="coerce"` yang akan mengubah nilai yang tidak bisa diubah ke angka menjadi NaN. Variabel `margin` yang telah bertipe numerik akan dinormalisasikan dengan metode `where(condition, other)` dengan parameter kondisi

margin bernilai kurang dari 1 “margin <= 1”, jika nilai kondisi tidak terpenuhi maka nilai pengganti akan di bagi 100 “margin / 100”. Logika sintak seperti ini akan membuat variabel margin yang sudah terbentuk desimal akan tetap dan margin yang masih dalam bentuk persen akan diubah menjadi bentuk desimal dengan rentang 0-1. Tujuannya adalah untuk membuat model lebih stabil dan interpretasi menjadi lebih mudah. setelah itu variabel yang margin telah ditransformasi akan dimasukkan ke Dataframe kolom margin `df[“margin”] = margin`.

```
(241, 12)
104    0.35
152    0.30
297    0.25
313    0.25
306    0.20
Name: margin, dtype: float64
```

Gambar 4. 11 Hasil Transformasi Variabel Margin

Dengan metode `df.sample()` untuk mengambil data secara acak untuk melihat proses transformasi margin berhasil, kolom atribut margin sudah bertipe numerik desimal (float). Langkah selanjutnya mengonversi kolom numerik lainnya tujuannya adalah untuk memastikan semua kolom ber tipe numerik karena data dari excel kadang masih terbaca dalam objek tertentu karena *Artificial Neural Network* butuh *input* numerik untuk proses pelatihan .

```
#Konversi kolom numerik
num_cols = ["nilai_penawaran_total", "TOP_hari", "jumlah_item", "pelanggan_baru",
            "barang_campuran", "Y1_aktual"]
for col in num_cols:
    df[col] = pd.to_numeric(df[col], errors="coerce")
print(df[num_cols].dtypes)
```

Gambar 4. 12 Kode Konversi Variabel Numerik

Kolom yang harus berisi numerik akan didefinisikan sebagai 1 variabel yaitu `num_cols`. Setiap kolom dalam list variabel `num_cols` akan dikonversi menjadi tipe numerik menggunakan fungsi `pd.to_numeric` dan nilai yang tidak di konversi menjadi angka akan di ubah menjadi Nan.

```
(241, 12)
nilai_penawaran_total    float64
TOP_hari                 float64
jumlah_item              float64
pelanggan_baru           float64
barang_campuran          float64
Y1_aktual                float64
dtype: object
```

Gambar 4. 13 Tipe Data Variabel Numerik

Transformasi kolom atribut menjadi kategori dan numerik sudah berhasil, selanjutnya adalah memfilter label karena model yang bangun adalah klasifikasi biner maka label harus 0 atau 1.

```
#Filter label valid dan kode_pelanggan
df = df[df["Y1_aktual"].isin([0, 1])]
df = df[df["kode_pelanggan"].notna()]
print("Semua Y1_aktual 0/1? ", df["Y1_aktual"].isin([0, 1]).all())
print("semua kode_pelanggan terisi? ", df["kode_pelanggan"].notna().all())
```

Gambar 4. 14 Konversi Label Biner

Sintak pada baris pertama untuk memfilter baris dengan nilai kolom atribut Y1_aktual yang berada dalam set 0 dan 1. Sintak baris kedua memfilter baris yang kosong dengan metode .notna() dan membuangnya, tujuannya untuk menjamin bahwa *dataset* yang digunakan untuk *Artificial Neural Network* hanya berisi label yang jelas dan identitas pelanggan yang lengkap.

```
(241, 12)
Semua Y1_aktual 0/1? True
semua kode_pelanggan terisi? True
```

Gambar 4. 15 Hasil Pengecekan Label

Langkah terakhir selanjutnya adalah mengisi nilai jika masih ada kolom atribut yang kosong. Tujuannya untuk menghilangkan semua *missing value* sehingga model *Artificial Neural Network* tidak *error* saat pelatihan dan model bisa jalan tanpa NaN.

```
#Isi nilai kosong yang tersisa
df["jenis_pelanggan"] = df["jenis_pelanggan"].fillna(df["jenis_pelanggan"].mode()[0])
df["kategori_produk"] = df["kategori_produk"].fillna(df["kategori_produk"].mode()[0])
for col in ["margin", "nilai_penawaran_total", "TOP_hari", "jumlah_item"]:
    df[col] = df[col].fillna(df[col].median())
df["pelanggan_baru"] = df["pelanggan_baru"].fillna(0)
df["barang_campuran"] = df["barang_campuran"].fillna(0)
df["Y1_aktual"] = df["Y1_aktual"].fillna(0)

print("Jumlah nilai kosong per kolom setelah fillna:")
print(df.isna().sum())
```

Gambar 4. 16 Kode Mengisi *Missing Value*

Menggunakan metode yang sederhana untuk mengubah *missing value* pada kolom atribut bertipe data kategori yaitu; jenis_pelanggan dan kategori_produk menjadi *value* kategori yang sering muncul pada masing-masing kolom atribut dengan metode mode(). Selanjutnya mengubah *missing value* pada kolom atribut bertipe numerik yaitu; margin, nilai_total_penawaran, TOP_hari dan, jumlah item menjadi *value* yang diisi median. Tujuan mengubah ke median karena nilai median membuat skala data mirip dengan data

asli Terakhir mengubah kolom *missing value* pada kolom atribut bertipe biner yaitu; pelanggan_baru, barang_campuran dan, Y1_aktual menjadi 0.

```
(241, 12)
Semua Y1_aktual 0/1? True
semua kode_pelanggan terisi? True
Jumlah nilai kosong per kolom setelah fillna:
nomor_penawaran      0
tanggal_penawaran    0
kode_pelanggan       0
jenis_pelanggan      0
kategori_produk      0
nilai_penawaran_total 0
margin              0
TOP_hari             0
jumlah_item          0
pelanggan_baru       0
barang_campuran      0
Y1_aktual            0
dtype: int64
```

Gambar 4. 17 Jumlah Nilai Kosong

Untuk mengecek Kembali jika masih ada *value* yang hilang menggunakan metode `df.isna()` dan menjumlahkan *missing value* yang hilang dengan `sum()`. Tersisa 241 baris dan 12 kolom dengan seluruh kolom tidak memiliki nilai hilang.

4.2.3 Pengkodean Data

Pengkodean data atau *Encoding* dibutuhkan karena model *Artificial Neural Network* hanya dapat memproses data dalam bentuk numerik, model tidak dapat mengenali hubungan antar kategori pada kolom atribut `jenis_pelanggan` dan, `kategori_produk` karena tidak memiliki makna matematis. *Encoding* mengubah kategori menjadi representasi angka yang konsisten sehingga model dapat mempelajari pola statistik. Proses ini penting karena variabel kategori berperan sebagai indikator perilaku pelanggan, karakteristik barang dan, membaca pola transaksi.

```
# One-hot encoding kategori dan buang tanggal
df_no_date = df.drop(columns=["tanggal_penawaran"])
encoded = pd.get_dummies(df_no_date, columns=["jenis_pelanggan", "kategori_produk"], dtype=int)
print(encoded.sample(5))
```

Gambar 4. 18 Kode *One-hot encoding*

One-hot Encoding dilakukan dengan terlebih dahulu menghapus kolom tanggal karena tidak diperlukan dalam proses pemodelan. Penghapusan dilakukan menggunakan metode `drop` yang menghilangkan kolom `tanggal_penawaran` dari `DataFrame`. Setelah itu, variabel kategori `jenis_pelanggan` dan `kategori_produk` diubah menjadi fitur numerik biner menggunakan fungsi `pd.get_dummies` dengan tipe keluaran `int` agar setiap kategori

direpresentasikan sebagai nilai 0 atau 1. Proses ini menghasilkan DataFrame yang sepenuhnya numerik sehingga dapat digunakan pada tahap pembangunan model.

	nomor_penawaran	kode_pelanggan	...	kategori_produk_bagian_ban	kategori_produk_ban
166	070/DPT/X/2023	SUB001	...	1	0
287	023/DPT/IV/2025	SUB001	...	0	1
238	060/DPT/IX/2024	SUB001	...	0	1
156	060/DPT/IX/2023	SUB001	...	1	0
104	008/DPT/II/2023	SUB001	...	1	0

Gambar 4. 19 Hasil *Encoding* data

Hasil dari *one-hot Encoding* adalah kolom baru kategori dari jenis_pelanggan dan kategori_produk. Variabel kategori dikonversi menjadi bentuk biner 0 dan 1. Representasi ini menghindari terbentuknya urutan numerik semu antar kategori yang tidak bersifat ordinal. Dengan representasi biner, setiap kategori dipandang sebagai fitur independen sehingga jaringan saraf dapat mempelajari pola secara lebih akurat dan stabil

4.2.4 Validasi Data

Sebelum data benar-benar digunakan dalam model *Artificial Neural Network* data akan di Validasi untuk memastikan Kembali tidak ada nilai kosong dan setiap variabel telah berada rentang yang benar sebelum digunakan pada tahap pemodelan.

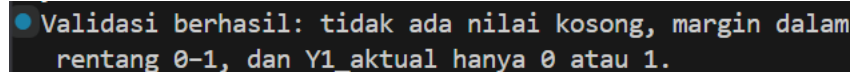
```
#Validasi kolom
missing_total = encoded.isna().sum().sum()
if missing_total > 0:
    raise ValueError(f"Validasi gagal: masih ada {missing_total} nilai kosong.")
if not encoded["margin"].between(0, 1).all():
    raise ValueError("Validasi gagal: margin di luar rentang 0-1.")
if not encoded["Y1_aktual"].isin([0, 1]).all():
    raise ValueError("Validasi gagal: label Y1_aktual bukan 0/1.")
print("Validasi berhasil: tidak ada nilai kosong, margin dalam rentang 0-1, "
      "dan Y1_aktual hanya 0 atau 1.")
```

Gambar 4. 20 Kode Validasi Data

Pemeriksaan nilai kosong dilakukan dengan metode `isna()` yang berfungsi untuk menghitung total seluruh data yang pada DataFrame. Dengan menggunakan `if`, jika ditemukan nilai kosong maka sistem menghentikan proses dan menampilkan pesan kesalahan “Validasi gagal : masih ada {*missing_total*} nilai kosong”. Validasi ini penting agar model tidak menerima data yang kosong karena dapat mempengaruhi akurasi prediksi dan stabilitas pelatihan.

Validasi khusus dilakukan pada dua variabel penting. Kolom `margin` diperiksa menggunakan metode `between(0, 1)` untuk memastikan seluruh nilai telah berada pada

skala yang benar. Label target $Y1_aktual$ juga divalidasi menggunakan $isin([0, 1])$ agar hanya memuat kelas biner yang sesuai. Setelah semuanya kolom atribut memiliki nilai yang sesuai dengan ketentuan, data dapat digunakan dalam untuk model *Artificial Neural Network*.



```
● Validasi berhasil: tidak ada nilai kosong, margin dalam
  rentang 0-1, dan Y1 aktual hanya 0 atau 1.
```

Gambar 4. 21 Hasil Konfirmasi Validasi Data

4.3 Perancangan dan Pengujian Model *Artificial Neural Network*

perancangan dan uji model *Artificial Neural Network*, berisi tahapan pembangunan model prediktif secara sistematis, mulai dari perancangan arsitektur model hingga evaluasi kinerjanya. Pada bab ini, model *Artificial Neural Network* disusun sebagai alat utama untuk mempelajari pola hubungan antara variabel *input* dan variabel target berdasarkan data historis yang telah melalui tahap praproses. Perancangan model dilakukan dengan mempertimbangkan karakteristik data, tujuan prediksi, serta kebutuhan akan hasil yang tidak hanya akurat tetapi juga dapat dianalisis.

Selain perancangan model *Artificial Neural Network* dilakukan proses pengujian model untuk menilai kemampuan generalisasi *Artificial Neural Network* terhadap data uji. Pengujian dilakukan melalui beberapa skenario, yaitu model *Artificial Neural Network baseline*, model *Artificial Neural Network* dengan metode *upsampling*, dan model *Artificial Neural Network* dengan pendekatan *class weighted*. Setiap model diuji menggunakan metrik evaluasi yang sama agar hasilnya dapat dibandingkan secara adil. Melalui rangkaian pengujian ini, bab ini menjadi dasar untuk menilai performa terbaik model *Artificial Neural Network* sekaligus menentukan model yang paling layak digunakan pada tahap analisis lanjutan dan uji variabel berpengaruh.

4.3.1 *Library*

Perancangan Pembangunan model *Artificial Neural Network* pada umumnya menggunakan Tensor flow dan Keras. Tetapi, karena ukuran *dataset* kecil hanya terdiri 241 baris penawaran bersih, seluruh proses pemodelan dilakukan dengan scikit learn. Penggunaan scikit learn karena *Library* ini sudah menyediakan semua fungsi yang dapat melakukan seluruh tahapan dalam satu ekosistem tanpa perlu *framework* tambahan.

Modul utama yang digunakan adalah `sklearn.neural_network` untuk membangun model *Multilayer Perceptron* yang berfungsi sebagai arsitektur *Artificial Neural Network*.

Pemisahan data latih dan uji dilakukan dengan `train_test_split` dari modul `sklearn.model_selection`. Proses pemisahan data merupakan langkah penting dalam penelitian prediktif untuk menghindari *overfitting* dan memastikan generalisasi model terhadap data baru.

Tahap transformasi menggunakan `StandardScaler` serta *Encoding* variabel kategori dilakukan dengan modul `sklearn.preprocessing`. Modul menyediakan sejumlah fungsi evaluasi seperti `accuracy_score`, `classification_report`, `confusion_matrix`, dan `roc_auc_score`. Seluruh metrik ini digunakan untuk menilai kualitas performa model *Artificial Neural Network* dalam melakukan klasifikasi. Evaluasi model menggunakan metrik yang disediakan dalam modul `sklearn.metrics` sehingga performa model dapat ditinjau secara kuantitatif. Struktur ini memungkinkan proses pembangunan model berjalan konsisten.

Karena *dataset* memiliki kelas yang tidak seimbang yang menyebabkan model *Artificial Neural Network* cenderung belajar pola dari kelas mayoritas. Diperlukan model dengan metode *Upsampling* dan *class weighted*, *Upsampling* menambah jumlah sampel kelas lebih sedikit dengan cara menduplikasi baris agar kelas lebih seimbang model dapat kesempatan belajar yang sama dari dua kelas. *Library* Scikit learn menyediakan `RandomOverSampler` dan `Smotes` untuk *Upsampling*. Metode *class weighted* untuk model *Artificial Neural Network* ketiga, model ini akan memberikan bobot lebih besar kepada kelas yang jumlahnya sedikit, metode ini membuat kesalahan prediksi pada kelas minoritas dihitung lebih besar dalam proses pelatihan model. *Library* Scikit learn sendiri menyediakan parameter `class_weight` model `MLPClassifier`.

4.3.2 Pembangunan Model *Artificial Neural Network*

Model *Artificial Neural Network* yang akan dibuat terdiri dari 3 model. Alasan membangun 3 model karena *dataset* memiliki kelas tidak seimbang, jika hanya membuat 1 model tidak ada perbandingan untuk melihat model lebih baik atau tidak. Model pertama yang dibuat adalah Model *Baseline*, model dasar dengan *dataset* sesuai data yang telah diambil, model ini tidak memiliki pemobotan tambahan maupun sintesis data minoritas.

Model kedua yaitu Model *Artificial Neural Network Upsampling*, menggunakan pendekatan pada *dataset*, membuat *Data Set* menjadi seimbang antara kelas minoritas mayoritas dengan cara duplikasi acak data yang sudah ada dari kelas minoritas hingga jumlahnya seimbang, sehingga model bisa mempelajari kelas minoritas sebanyak kelas mayoritas sebelumnya.

Model ketiga yaitu Model *Artificial Neural Network Class Weighted*, model ini menggunakan pendekatan pada pembobotan kelas pada model prediksi, tidak mengubah *dataset* tapi memberi bobot lebih besar pada kelas minoritas sehingga model memperhatikan dan berusaha meminimalkan kesalahan pada kelas minoritas. Ini membantu model mempelajari karakteristik kelas minoritas dengan lebih baik karena keterbatasan *dataset*.

Dengan ketiga model tersebut dibuat, hasil dari pengujian ketiga model *Artificial Neural Network*. Bisa menghasilkan analisis dan perbandingan untuk setiap modelnya.

1. Model *Artificial Neural Network* baseline

Tahap pertama dalam membangun model adalah menyusun kebutuhan pemograman untuk model *Artificial Neural Network* dan memastikan seluruh proses dapat direplikasi secara konsisten. Dengan *Import Library* seperti NumPy dan pandas digunakan untuk mengelola data numerik untuk pembentukan variabel prediktor, pemisahan kolom target, dan penyiapan struktur data yang kompatibel dengan algoritma pembelajaran mesin. *Library* os digunakan untuk mengelola lokasi file sehingga penyimpanan *input* dan *output* penelitian berjalan rapi. *Library* Joblib digunakan untuk menyimpan artefak penting, seperti model yang sudah dilatih dan objek pemrosesan awal, sehingga model dapat digunakan kembali pada tahap pengujian lanjutan atau implementasi sistem tanpa perlu melakukan pelatihan ulang.

```
# import library yang diperlukan
import os
import joblib
import numpy as np
import pandas as pd
```

Gambar 4. 22 Kode *Import Library*

berikutnya menyiapkan komponen scikit learn untuk membangun alur pemodelan end to end, mulai dari pembagian data, tahap praproses, pelatihan awal, hingga evaluasi. *train_test_split* membagi *dataset* menjadi data latih dan data uji agar penilaian kinerja mencerminkan kemampuan generalisasi pada data yang tidak dipakai saat pelatihan.

```
#import sklearn tools
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

Gambar 4. 23 *Import Modul* scikit learn

StandardScaler menstandarkan fitur numerik agar memiliki skala yang sebanding, sehingga proses pembelajaran *Artificial Neural Network* berbasis optimasi gradien menjadi lebih stabil dan konvergen. Pipeline menggabungkan proses standarisasi dan model *Artificial Neural Network* dalam satu rangkaian kerja, sehingga transformasi fitur terjadi dengan urutan yang sama pada fase pelatihan dan prediksi, serta mengurangi risiko ketidakkonsistenan tahap praproses. MLPClassifier berperan sebagai implementasi *Artificial Neural Network* tipe multilayer perceptron untuk klasifikasi.

Tahap berikutnya sebelum membangun model. penetapan DATA_PATH dilakukan untuk menentukan sumber data yang digunakan dalam pembangunan model *Artificial Neural Network*. Bertujuan agar proses pemanggilan data menjadi konsisten dan terstruktur sepanjang tahapan penelitian. Dengan mendefinisikan path ini sejak awal, seluruh proses pengolahan data, pelatihan, dan evaluasi model selalu mengacu pada sumber data yang sama, sehingga alur eksperimen dapat di replikasi dan dikontrol dengan baik.

```
# Definisi input dan output path
DATA_PATH = "data/data_model_ann.csv"
MODEL_PATH = os.path.join("artifacts", "model_ann_baseline.joblib")
```

Gambar 4. 24 Mendefinisikan *Input* dan *Ouput*

Selanjutnya, MODEL_PATH disusun menggunakan fungsi os.path.join untuk menentukan lokasi penyimpanan model hasil pelatihan. Fungsi ini dipilih agar penyimpanan artefak model bersifat rapi dan terpisah dari kode utama. Model *baseline* disimpan dalam format joblib karena format ini mampu menyimpan objek model *Artificial Neural Network* beserta konfigurasi dan parameter yang terbentuk selama proses pelatihan. Dengan cara ini, model yang telah dilatih dapat digunakan kembali pada tahap pengujian lanjutan maupun implementasi sistem tanpa perlu dilakukan pelatihan ulang.

```
# identifikasi kolom bukan fitur
ID_COLS = ["nomor_penawaran", "kode_pelanggan"]
```

Gambar 4. 25 Identifikasi *Non Variabel*

Setelah mendefinisikan *input* dan *output* path, selanjutnya mengidentifikasi kolom yang bukan fitur untuk Pembangunan model uji *Artificial Neural Network*, dengan mendefinisikan ID_COLS untuk mengidentifikasi kolom kolom yang bersifat administratif, yaitu nomor penawaran dan kode pelanggan. Kolom ini tidak digunakan

sebagai variabel prediktor karena hanya berfungsi sebagai penanda unik data. bukan sebagai variabel *input* yang mempengaruhi hasil prediksi. Dengan memisahkan kolom identitas sejak awal, proses pembentukan fitur menjadi lebih terarah dan model *Artificial Neural Network* dilatih menggunakan variabel yang benar mencerminkan kondisi penawaran dan pelanggan, sehingga kemampuan generalisasi model dapat terjaga.

Setelah menetapkan *input* data sebagai sumber data membangun model tahap selanjutnya adalah mendefinisikan fungsi `load_data` sebagai mekanisme standar untuk memuat *dataset* dan mempersiapkan data sebelum masuk ke proses pemodelan *Artificial Neural Network*. Mendefinisikan Fungsi untuk memuat data dengan parameter `path` bertipe string dan nilai default `DATA_PATH` agar proses pemanggilan data tetap konsisten tanpa harus selalu menuliskan lokasi file. Data dimuat menggunakan metode `pd.read_csv` dan disimpan dalam bentuk `DataFrame` karena struktur ini memudahkan pengelolaan data, termasuk seleksi kolom, pembersihan data, dan pemisahan variabel penelitian secara sistematis

```
# Fungsi untuk memuat data
def load_data(path: str = DATA_PATH):
    df = pd.read_csv(path)
    drop_cols = [c for c in ID_COLS if c in df.columns]
```

Gambar 4. 26 Kode Fungsi Load Data

Setelah data masuk, identifikasi kolom yang perlu dikeluarkan dari proses pelatihan melalui pembentukan variabel `drop_cols`, yang berisi kolom identitas yang benar-benar tersedia di dalam *dataset*. Pendekatan ini digunakan agar proses penghapusan kolom berjalan aman meskipun struktur *dataset* mengalami perubahan.

```
# pembentukan variabel fitur dan target
X = df.drop(columns=["Y1_aktual", *drop_cols])
y = df["Y1_aktual"].astype(int)
return X, y
```

Gambar 4. 27 Mendefinisikan Variabel X dan Y

Pembentukan Variabel fitur X dibentuk dengan menghapus kolom target `Y1_aktual` dan kolom identitas, sehingga model *Artificial Neural Network* hanya mempelajari variabel yang merepresentasikan karakteristik penawaran dan pelanggan. Variabel target `y` diekstraksi dari kolom `Y1_aktual` dan dikonversi ke tipe integer dengan metode `astype(int)`, untuk memastikan kesesuaian dengan kebutuhan algoritma klasifikasi. Untuk mengembalikan `X` dan `y` sebagai *output* utama dengan

cara menggunakan fungsi return sehingga ke dua objek utama siap digunakan pada tahap pembagian data, pemrosesan awal, dan pelatihan model. dan uji, scaling, serta pelatihan model *Artificial Neural Network*.

Sebelum proses pelatihan dijalankan di fungsi main, langkah berikutnya adalah mendefinisikan fungsi `build_model` sebagai tempat menyusun rancangan model *Artificial Neural Network* yang akan dipakai. Di dalam fungsi ini, `StandardScaler` dibuat terlebih dahulu untuk menstandarkan semua fitur numerik supaya skala tiap kolom menjadi sebanding. Langkah ini membantu proses belajar *Artificial Neural Network* lebih stabil, karena model tidak perlu “beradaptasi” dengan fitur yang nilainya jauh lebih besar dibanding fitur lain. Pada bagian yang sama, `MLPClassifier` disiapkan sebagai inti model *Artificial Neural Network* yang menangani klasifikasi, sehingga seluruh rancangan model sudah jelas sebelum masuk ke tahap *fitting*.

```
# Fungsi untuk membangun model
def build_model():
    scaler = StandardScaler()
    clf = MLPClassifier(
        hidden_layer_sizes=(16, 8),
        activation="relu",
        solver="lbfgs",
        alpha=0.01,
        max_iter=2000,
        random_state=42,
    )
    return Pipeline([("scaler", scaler), ("clf", clf)])
```

Gambar 4. 28 Kode Membangun Model *Artificial Neural Network*

Berikutnya, konfigurasi `MLPClassifier` ditetapkan secara bertahap agar arsitektur *Artificial Neural Network* sesuai dengan karakteristik data penelitian. Parameter `hidden_layer_sizes=(16, 8)` digunakan untuk membentuk dua lapisan tersembunyi. Lapisan pertama dengan 16 neuron berperan menangkap pola awal dan hubungan kompleks antar variabel *input*, sedangkan lapisan kedua dengan 8 neuron berfungsi memperdalam representasi pola tersebut dalam bentuk yang lebih ringkas. Susunan bertingkat ini dipilih agar model memiliki kapasitas belajar yang cukup tanpa membuat struktur jaringan terlalu besar, sehingga risiko *overfitting* dapat ditekan. Pemilihan arsitektur 18 dan 6 neuron dilakukan setelah model diuji menggunakan arsitektur yang lebih besar, yaitu 64 dan 32 neuron. Arsitektur 64 dan 32 neuron memiliki kapasitas belajar yang lebih besar, tetapi pada data penelitian yang relatif

terbatas struktur tersebut berisiko membuat model terlalu kompleks. Arsitektur 18 dan 6 neuron dipilih karena menghasilkan performa pengujian yang lebih baik dan lebih stabil, sehingga lebih sesuai dengan karakteristik data historis penawaran. Pemilihan ini sejalan dengan Bishop (2006), yang menjelaskan bahwa jumlah neuron pada *hidden layer* merupakan parameter yang dapat disesuaikan untuk memperoleh performa prediksi terbaik dan menjaga keseimbangan antara *underfitting* dan *overfitting*.

Parameter `activation="relu"` digunakan sebagai fungsi aktivasi pada setiap neuron karena mampu menangani hubungan non linear dengan lebih efisien dan membantu proses pembelajaran berjalan lebih stabil. Pemilihan `solver="lbfgs"` digunakan karena penelitian ini memakai data historis penawaran dengan jumlah relatif terbatas. Pada kondisi data kecil, L-BFGS dapat menjadi pilihan yang sesuai karena menggunakan pendekatan optimasi berbasis batch dan mampu mencapai konvergensi lebih cepat.

Parameter `max_iter=2000` ditetapkan untuk memberi ruang iterasi yang cukup agar proses pelatihan dapat mencapai kondisi konvergen, terutama ketika pola data tidak sederhana. Sementara itu, `random_state=42` digunakan untuk menjaga konsistensi hasil pelatihan ketika proses dijalankan ulang, sehingga perubahan kinerja model benar-benar berasal dari perubahan data atau konfigurasi, bukan dari faktor pengacakan internal. Seluruh konfigurasi ini kemudian digabungkan ke dalam Pipeline bersama `StandardScaler`, memastikan bahwa setiap data yang masuk ke model selalu melalui proses standarisasi sebelum diproses oleh *Artificial Neural Network*.

Tahapan selanjutnya adalah membagi data menjadi data latih dan data testing sebelum model *Artificial Neural Network* dilatih. Tujuan dari pembagian data testing untuk menyediakan sekumpulan data yang tidak dilibatkan dalam proses pelatihan model *Artificial Neural Network* sehingga evaluasi kinerja dapat dilakukan secara objektif. Setelah fungsi pemuatan data disusun, proses dilanjutkan dengan pemanggilan `load_data` menggunakan `DATA_PATH` untuk menghasilkan dua komponen utama penelitian, yaitu variabel fitur `X` dan variabel target `y`. fungsi `load_data` untuk memisahkan variabel Input (`X`) serta variabel target `y`. Tahap ini

memastikan data yang digunakan sudah bersih dari kolom identitas dan target berada dalam format numerik yang sesuai untuk klasifikasi.

```
# pembagian data training dan data testing
def main():
    X, y = load_data(DATA_PATH)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )
```

Gambar 4. 29 Split Data Pelatihan dan Data Testing

Setelah data berhasil dimuat, proses selanjutnya adalah pembagian *dataset* menggunakan `train_test_split` menjadi data latih dan data uji. Proporsi pembagian ditetapkan sebesar 80 persen untuk pelatihan dan 20 persen untuk pengujian melalui parameter `test_size=0.2` agar model dapat dievaluasi pada data yang tidak terlibat dalam proses pembelajaran. Parameter `random_state` digunakan untuk memastikan pembagian data bersifat konsisten pada setiap percobaan, sedangkan parameter `stratify` memastikan distribusi kelas pada data latih dan data uji tetap mengikuti distribusi label awal. Dengan pendekatan ini, evaluasi kinerja model mencerminkan kondisi data yang sebenarnya dan mendukung objektivitas hasil penelitian.

Semua fungsi pendukung sudah didefinisikan, proses pemodelan dilanjutkan dengan tahap pembentukan dan pelatihan model *Artificial Neural Network*. Pada tahap ini, fungsi `build_model` dipanggil untuk membangun struktur model yang telah dirancang sebelumnya, termasuk konfigurasi pemrosesan awal dan algoritma *Artificial Neural Network* yang digunakan. Pemisahan logika pembentukan model ke dalam fungsi tersendiri dilakukan agar arsitektur dan parameter model dapat dikelola secara terstruktur serta memudahkan penyesuaian pada tahap eksperimen lanjutan.

```
# Membuat dan melatih model
model = build_model()
model.fit(X_train, y_train)
```

Gambar 4. 30 Pelatihan Model

Setelah model berhasil dibentuk, proses pelatihan dijalankan melalui pemanggilan metode `fit` dengan menggunakan data latih `X_train` dan `y_train`. Tahap ini memungkinkan model *Artificial Neural Network* mempelajari pola hubungan antara variabel *input* dan variabel target berdasarkan data historis. Proses pembelajaran berlangsung hingga parameter model menyesuaikan diri secara optimal terhadap data

latih, sehingga model siap digunakan pada tahap pengujian dan evaluasi kinerja pada data yang belum pernah dilihat sebelumnya.

Tahapan berikutnya adalah pengujian kinerja model menggunakan data uji yang telah disiapkan sebelumnya. Pada bagian ini, prediksi kelas dilakukan melalui pemanggilan `model.predict(X_test)` untuk memperoleh nilai `y_pred` dalam bentuk label diskrit. Hasil prediksi ini merepresentasikan keputusan klasifikasi model terhadap setiap observasi pada data uji dan menjadi dasar untuk membandingkan keluaran model dengan label aktual dalam proses evaluasi kinerja

```
# Melakukan prediksi dan evaluasi model
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[: , 1]
```

Gambar 4. 31 Uji Prediksi dan Probabilitas Model

Selanjutnya proses dilanjutkan dengan menghitung probabilitas prediksi melalui `y_prob = model.predict_proba(X_test)[: , 1]`. Metode `predict_proba` menghasilkan probabilitas untuk setiap kelas, lalu indeks `[: , 1]` mengambil probabilitas untuk kelas positif, yaitu kelas 1. Nilai probabilitas ini digunakan untuk evaluasi berbasis ambang dan kurva seperti ROC-AUC, serta relevan untuk kebutuhan penelitian yang mengarah pada keluaran probabilistik atau fuzzy karena memberikan skor kontinu 0 sampai 1, bukan hanya keputusan kelas akhir.

Tahap terakhir berfokus pada penyimpanan model yang sudah dilatih agar hasil pelatihan tidak hilang dan dapat digunakan kembali pada proses pengujian lanjutan atau implementasi sistem. Fungsi `os.path.dirname(MODEL_PATH)` digunakan untuk mengambil bagian direktori dari `MODEL_PATH`, sedangkan `os.makedirs()` digunakan untuk membuat folder penyimpanan model. Parameter `exist_ok=True` membuat perintah tetap aman dijalankan meskipun folder tersebut sudah tersedia, sehingga program tidak berhenti karena kesalahan pembuatan direktori. Setelah folder siap, fungsi `joblib.dump(model, MODEL_PATH)` digunakan untuk menyimpan objek model terlatih ke dalam berkas sesuai lokasi `MODEL_PATH`. Dengan demikian, artefak model *baseline* dapat disimpan dan dipanggil kembali tanpa perlu melakukan pelatihan ulang. Baris `print` digunakan sebagai bukti eksekusi bahwa berkas model berhasil disimpan, karena jalur penyimpanan ditampilkan langsung pada keluaran program.

```
# Menyimpan model yang telah dilatih
os.makedirs(os.path.dirname(MODEL_PATH), exist_ok=True)
joblib.dump(model, MODEL_PATH)
print("Model baseline disimpan ke:", MODEL_PATH)

if __name__ == "__main__":
    main()
```

Gambar 4. 32 Kode Menyimpan Model

Bagian terakhir adalah pengaturan eksekusi program menggunakan `if name == "__main__": main()`. Struktur ini membuat fungsi `main` hanya dijalankan ketika file Python dieksekusi langsung, bukan ketika file tersebut diimpor sebagai modul dari file lain. Dengan cara ini, Fungsi `main` bisa dipakai sebagai kerangka untuk menguji ulang model dengan metode lain seperti *upsampling* dan *Weighted Class*

2. Model *Artificial Neural Network upsampling*

Perancangan Model *Artificial Neural Network upsampling* memiliki tahapan yang sama seperti pada model *baseline*. dimulai dengan *Import Library* untuk memastikan seluruh kebutuhan pemrosesan data, pemodelan, dan evaluasi tersedia dalam satu lingkungan kerja. Struktur *Import* pada versi ini masih mempertahankan komponen inti seperti `os` untuk pengelolaan path dan folder penyimpanan artefak, `joblib` untuk menyimpan model terlatih, `numpy` dan `pandas` untuk manipulasi data numerik, serta komponen `scikit learn` untuk membangun *Artificial Neural Network* melalui `MLPClassifier`. Pipeline dan `StandardScaler` juga tetap dipakai agar alur preprocessing dan pelatihan *Artificial Neural Network* berjalan konsisten, yaitu fitur distandarkan terlebih dahulu sebelum masuk ke jaringan saraf

```
#masukkan library yang sama seperti model ANN baseline
import os
import joblib
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

Gambar 4. 33 *Import Library* dan Modul

Perbedaan utama dibandingkan *Import Library* sebelumnya adalah penambahan `from sklearn.utils import resample`. Penambahan ini menandakan adanya tahapan *balancing data* melalui *upsampling*, yaitu proses menambah jumlah data pada kelas

minoritas dengan cara melakukan sampling ulang dengan penggantian pada data latih. Jadi, versi sebelumnya fokus pada pemodelan *baseline* dengan distribusi kelas asli, sedangkan versi *upsampling* menambahkan satu modul khusus untuk mengatasi ketidakseimbangan kelas sebelum model dilatih. Dampaknya, alur eksperimen menjadi sedikit lebih panjang karena ada tahap tambahan setelah train test split, yaitu membentuk ulang data latih agar proporsi kelas lebih seimbang

```
# tambahan library untuk upsampling
from sklearn.utils import resample
```

Gambar 4. 34 Import Modul untuk *Upsampling*

Perubahan ini tidak mengubah struktur inti *Artificial Neural Network* dan evaluasinya, tetapi mengubah kondisi data yang dipakai saat pelatihan, sehingga hasil yang diharapkan biasanya terlihat pada metrik yang sensitif terhadap kelas minoritas, seperti *Recall* dan *f1 score* kelas minoritas, serta *confusion matrix* yang menunjukkan penurunan kesalahan pada kelas yang sebelumnya jarang muncul.

Pada tahap selanjutnya, proses yang dijalankan masih mengikuti alur yang sama dengan model *baseline*. *DATA_PATH* digunakan untuk menentukan sumber *dataset* yang dipakai dalam pemodelan, sementara *MODEL_PATH* diarahkan ke nama file model yang berbeda agar hasil pelatihan *Artificial Neural Network* dengan metode *upsampling* tersimpan terpisah dan tidak tertimpa oleh model sebelumnya. Pemisahan lokasi penyimpanan ini memudahkan proses perbandingan kinerja antar skenario pemodelan.

```
# definisi input dan output path dan load data function
DATA_PATH = "data/data_model_ann.csv"
MODEL_PATH = os.path.join("artifacts", "model_ann_upsample.joblib")
ID_COLS = ["nomor_penawaran", "kode_pelanggan"]
def load_data(path: str = DATA_PATH):
    df = pd.read_csv(path)
    drop_cols = [c for c in ID_COLS if c in df.columns]
    X = df.drop(columns=["Y1_aktual", *drop_cols])
    y = df["Y1_aktual"].astype(int)
    return X, y
```

Gambar 4. 35 Mendefinisikan Path dan Variabel

Pendefinisian *ID_COLS* dan fungsi *load_data* tetap dipertahankan tanpa perubahan karena fungsinya sudah sesuai dengan kebutuhan penelitian, yaitu mengeluarkan kolom identitas dan memisahkan data menjadi variabel Input (X) dan Input (Y). Dengan menjaga tahap pemuatan data tetap konsisten, perbedaan hasil yang

diperoleh pada tahap evaluasi dapat dikaitkan langsung dengan penerapan metode *upsampling*, bukan akibat perubahan struktur atau perlakuan awal terhadap data

Berikutnya adalah Tahapan inti dari model *Artificial Neural Network upsampling* yaitu, Menyusun fungsi *upsampling_minority* untuk menyeimbangkan distribusi sebelum model *Artificial Neural Network* dilatih. Fungsi menerima tiga masukan, yaitu X sebagai DataFrame yang berisi fitur, y sebagai Series yang berisi label target, dan random_state sebagai pengunci proses acak agar hasil *upsampling* konsisten ketika eksperimen diulang. Penetapan random_state=42 dipakai supaya perubahan performa model yang terlihat pada tahap evaluasi benar benar berasal dari perlakuan *upsampling* dan konfigurasi model, bukan karena perbedaan acakan sampling.

```
#definisikan fungsi upsample minority class
def upsample_minority(X: pd.DataFrame, y: pd.Series, random_state: int = 42):
    df_tmp = X.copy()
    df_tmp["target"] = y.values
    max_count = df_tmp["target"].value_counts().max()
    pieces = []
    for label, group in df_tmp.groupby("target"):
        pieces.append(
            group.resample(
                group,
                replace=True,
                n_samples=max_count,
                random_state=random_state,
            )
        )
    balanced = (
        pd.concat(pieces)
        .sample(frac=1.0, random_state=random_state)
        .reset_index(drop=True)
    )
    y_bal = balanced["target"].astype(int)
    X_bal = balanced.drop(columns=["target"])
    return X_bal, y_bal
```

Gambar 4. 36 Kode Fungsi *Upsampling*

Fungsi `df_tmp = X.copy()` digunakan untuk membuat salinan data fitur agar proses penambahan kolom target tidak mengubah data asli, lalu fungsi `df_tmp["target"] = y.values` menempelkan label ke dalam tabel yang sama.

Langkah berikutnya menghitung target jumlah sampel untuk setiap kelas melalui `max_count` dan `value_counts` menghitung jumlah baris pada setiap kelas target, kemudian `max` mengambil nilai terbesar, sehingga `max_count` menjadi ukuran kelas mayoritas. Nilai ini dipakai sebagai standar jumlah sampel yang ingin dicapai untuk semua kelas setelah *upsampling*. Setelah itu, `pieces = []` dibuat sebagai wadah untuk menampung hasil resampling dari setiap kela

Selanjutnya adalah menjalankan inti proses *upsampling* per kelas target dengan menggunakan struktur pengulangan dengan loop for. Perulangan for label grup in `df_tmp.groupby("target")` memecah *dataset* sementara `df_tmp` menjadi beberapa grup berdasarkan kelas target, dan pada setiap iterasi loop, grup berisi seluruh baris dari satu kelas tertentu. Variabel label sebenarnya berisi nama kelasnya, tetapi fokus utama ada pada grup yang akan diolah agar ukuran kelasnya disamakan.

```
for label, group in df_tmp.groupby("target"):
    pieces.append(
        resample(
            group,
            replace=True,
            n_samples=max_count,
            random_state=random_state,
        )
    )
```

Gambar 4. 37 Fungsi *Resample*

Kemudian pada setiap grup, proses *resample* dijalankan lalu hasilnya dimasukkan ke list `pieces` melalui fungsi `pieces.append`. Fungsi *resample* menerima *input* grup, lalu `replace=True` membuat pengambilan sampel boleh mengulang baris yang sama, ini yang memungkinkan kelas minoritas bertambah jumlahnya tanpa membuat data baru. Parameter `n_samples=max_count` memaksa hasil *resample* dari setiap kelas memiliki jumlah baris yang sama dengan `max_count`, yaitu jumlah data pada kelas mayoritas. Parameter `random_state=random_state` mengunci proses acak supaya hasil *upsampling* tetap sama ketika kode dijalankan ulang. Dengan pola ini, setiap kelas berakhir dengan jumlah observasi yang setara, sehingga data latih menjadi seimbang dan *Artificial Neural Network* tidak terlalu condong mempelajari kelas yang awalnya dominan.

Setelah semua kelas selesai diproses dan masuk ke `pieces`, data hasil balancing digabung menjadi satu *dataset* melalui `balanced`. Menggunakan fungsi `pd.concat` untuk menggabungkan `DataFrame` dari setiap kelas yang sudah diseimbangkan menjadi data frame utuh.

```
balanced = (
    pd.concat(pieces)
    .sample(frac=1.0, random_state=random_state)
    .reset_index(drop=True)
)
```

Gambar 4. 38 Kode Pengacakan Data

Selanjutnya mengacak urutan baris dengan `frac=1.0` yaitu, mengambil 100 persen baris dengan urutan baru sehingga data tidak tersusun berkelompok dan berkelas. Tujuannya dari pengacakan data karena data yang berurutan per kelas bisa mempengaruhi proses pelatihan. Kemudian `reset_index(drop=True)` merapikan indeks menjadi urut dari nol dan membuang indeks lama agar struktur *dataset* lebih bersih untuk tahap berikutnya.

Tahap terakhir dalam Menyusun fungsi *upsampling* adalah memisahkan kembali fitur dan label dari *dataset* yang sudah seimbang untuk disiapkan Kembali menjadi format yang dibutuhkan untuk model *Artificial Neural Network*. Baris sintak `y_bal = balanced["target"].astype(int)` mengambil kolom target dari DataFrame `balanced` lalu mengubah tipenya menjadi integer. Konversi ini memastikan label benar benar berbentuk kelas biner 0 dan 1, sehingga tidak ada masalah tipe data saat model melakukan proses klasifikasi dan saat metrik evaluasi menghitung performa. Sedangkan untuk memisahkan fitur dari target, baris sintak `X_bal = balanced.drop(columns=["target"])` menghapus kolom target dari DataFrame `balanced`, sehingga yang tersisa hanya kolom kolom fitur yang akan masuk ke `StandardScaler` dan `MLPClassifier`

```
y_bal = balanced["target"].astype(int)
X_bal = balanced.drop(columns=["target"])
return X_bal, y_bal
```

Gambar 4. 39 Memisahkan Variabel kembali

Bagian `return X_bal, y_bal` mengembalikan data latih versi `balanced`, sehingga fungsi *upsampling_minority* menghasilkan data latih versi seimbang dalam format yang sama seperti *baseline*. Dengan format ini, bagian pelatihan di `main` tidak perlu diubah besar, cukup mengganti `X_train`, `y_train` menjadi `X_bal`, `y_bal` sebelum `model.fit`. Dengan demikian perubahan utama dari *baseline* terjadi pada kondisi data latih yang menjadi seimbang sebelum proses pelatihan, sementara data uji tetap dibiarkan pada distribusi asli agar evaluasi tetap mencerminkan kondisi nyata.

Tahap berikutnya dilanjutkan dengan menyusun fungsi `main` sebagai alur eksekusi utama, sama seperti pada uji model *baseline*. Proses dimulai dengan `X, y = load_data(DATA_PATH)` untuk memanggil *dataset* dan memisahkan variabel *input* serta target. Selanjutnya dilakukan pembagian data menggunakan `train_test_split` menjadi `X_train`, `X_test`, `y_train`, `y_test` dengan proporsi data uji 20 persen melalui `test_size=0.2`. `random_state=42` dipakai agar hasil pembagian data konsisten ketika

pengujian diulang, sedangkan `stratify=y` menjaga proporsi kelas pada data latih dan data uji tetap mengikuti distribusi awal, sehingga evaluasi tetap representatif terhadap kondisi data asli.

```
#Membagi data training dan data testing
def main():
    X, y = load_data(DATA_PATH)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )
    X_train_bal, y_train_bal = upsample_minority(X_train, y_train, random_state=42)
```

Gambar 4. 40 Membagi Data Testing dan Data Pelatihan

Perbedaan utama dari *baseline* muncul pada langkah berikutnya, yaitu penerapan *upsampling* hanya pada data latih. Baris `X_train_bal, y_train_bal = upsample_minority(X_train, y_train, random_state=42)` membentuk ulang data latih menjadi seimbang dengan menambah jumlah sampel pada kelas minoritas sampai setara dengan kelas mayoritas. Penempatan *upsampling* pada data latih saja dipertahankan agar data uji tetap murni dan tidak berubah, sehingga performa model yang dihitung benar-benar menunjukkan kemampuan generalisasi pada distribusi nyata. *Output* `X_train_bal` dan `y_train_bal` kemudian menjadi *input* utama untuk tahap pelatihan model, sehingga *Artificial Neural Network* belajar dari data latih yang proporsi kelasnya sudah lebih adil dibandingkan skenario *baseline*.

Tahapan selanjutnya adalah memproses pembentukan, pelatihan, dan pengujian model menggunakan data yang telah diseimbangkan. Pada tahap ini, fungsi `build_model` dipanggil untuk membangun arsitektur *Artificial Neural Network* beserta alur yang sama seperti pada model *baseline*, sehingga perbedaan hasil kinerja tidak berasal dari perubahan struktur model hanya dari perubahan data yang sudah diseimbangkan. Model kemudian dilatih menggunakan `model.fit` dengan `X_train_bal` dan `y_train_bal`, yaitu data latih yang telah melalui proses *upsampling*, sehingga jaringan saraf mempelajari pola dari distribusi kelas yang lebih seimbang.

```
# membangun, melatih dan, memprediksi model
model = build_model()
model.fit(X_train_bal, y_train_bal)
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]
```

Gambar 4. 41 Kode Melatih dan Memprediksi Model

Setelah proses latih selesai, model diuji menggunakan data uji yang tidak mengalami perubahan. Prediksi kelas dihasilkan melalui `model.predict(X_test)` dan disimpan dalam `y_pred` untuk kebutuhan evaluasi berbasis label diskrit. Selain itu,

probabilitas prediksi dihitung menggunakan `model.predict_proba(X_test)` dengan mengambil probabilitas kelas positif, sehingga diperoleh `y_prob` dalam bentuk nilai kontinu antara 0 dan 1. Nilai probabilitas ini digunakan untuk evaluasi lanjutan seperti ROC-AUC dan analisis berbasis ambang, serta menjadi pembeda penting antara keluaran prediksi keras dan tingkat keyakinan model terhadap setiap hasil klasifikasi.

3. Model *Artificial Neural Network Weighted Class*

Model *Artificial Neural Network* dengan pendekatan *Class weighted* memiliki tahapan awal yang sama. Perbedaan terdapat pada inisialisasi *Multilayer Preceptron Classifier* dengan menambahkan parameter `class_weight`. Parameter digunakan untuk memberikan bobot lebih kecil pada kelas mayoritas dan bobot lebih besar pada kelas minoritas. kesalahan prediksi pada kelas minoritas akan memberikan penalti yang lebih besar Sehingga Model *Artificial Neural Network* lebih memperhatikan pada kelas minoritas. Secara alur dan struktur model *Artificial Neural Network Weighted Class* tetap sama dengan model lainnya. Tujuannya untuk melihat perubahan kinerja yang muncul dapat dikaitkan langsung dengan penggunaan parameter `class_weight` sebagai strategi penanganan ketidakseimbangan kelas.

Tahapan awal kembali dimulai dengan penyusunan *Import Library* sebagai dasar lingkungan pemodelan. *Import os* digunakan untuk pengelolaan path dan direktori *output*, sedangkan *joblib* disiapkan untuk menyimpan artefak model terlatih agar dapat dipakai ulang pada proses pengujian berikutnya. NumPy dan pandas tetap digunakan sebagai fondasi pengolahan data numerik dan data tabular, karena *dataset* penelitian tersusun dalam bentuk tabel dan perlu diproses menjadi fitur serta target yang siap masuk ke model.

```
# masukkan library yang sama seperti model lainnya
import os
import joblib
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

Gambar 4. 42 *Import Library* dan Modul

Menyikan Kembali komponen utama dari scikit learn yang digunakan dalam alur *Artificial Neural Network*. `train_test_split` digunakan untuk membagi data menjadi

data latih dan data uji agar evaluasi generalisasi tetap objektif. MLPClassifier dipakai sebagai implementasi *Artificial Neural Network* untuk klasifikasi, sedangkan Pipeline dan StandardScaler menjaga alur pemrosesan awal berjalan konsisten dengan cara menstandarkan fitur sebelum masuk ke jaringan saraf. Pada pendekatan *class weighted*, struktur *Import* ini memang masih sama seperti *baseline*, karena perbedaan inti tidak terletak pada *Library* tambahan, melainkan pada pengaturan parameter di MLPClassifier.

```
# tambahkan parameter untuk class weight
from sklearn.utils.class_weight import compute_class_weight
```

Gambar 4. 43 Import Parameter Class Weight

Setelah *Library* utama dimasukkan, tahapan berikutnya menambahkan modul `compute_class_weight` dari `sklearn.utils.class_weight` untuk mendukung strategi *class weighted* pada pelatihan *Artificial Neural Network*. Fungsi ini dipakai untuk menghitung bobot kelas secara otomatis berdasarkan distribusi label pada data latih. Dengan pendekatan ini, kelas yang jumlahnya lebih sedikit akan memperoleh bobot lebih besar, sehingga kesalahan prediksi pada kelas minoritas memberi penalti lebih besar.

Berikutnya, variabel `path` dan fungsi pemuatan data kembali disusun agar alur pemodelan *class weighted* tetap konsisten dengan Struktur model sebelumnya. `DATA_PATH` digunakan untuk menetapkan lokasi *data set* yang menjadi sumber *input*, sedangkan `MODEL_PATH` diarahkan ke nama file yang berbeda, yaitu `model_ann_class_weight.joblib`, agar artefak model untuk skenario *class weighted* tersimpan terpisah dan mudah dibandingkan dengan *baseline* maupun *upsampling*. `ID_COLS` tetap didefinisikan untuk menandai kolom identitas seperti `nomor_penawaran` dan `kode_pelanggan`, sehingga kolom tersebut tidak ikut menjadi fitur karena hanya berfungsi sebagai penanda transaksi, bukan variabel *input*.

```
# definisi input dan output path dan load data function
DATA_PATH = "data/data_model_ann.csv"
MODEL_PATH = os.path.join("artifacts", "model_ann_class_weight.joblib")
ID_COLS = ["nomor_penawaran", "kode_pelanggan"]
def load_data(path: str = DATA_PATH):
    df = pd.read_csv(path)
    drop_cols = [c for c in ID_COLS if c in df.columns]
    X = df.drop(columns=["Y1_aktual", *drop_cols])
    y = df["Y1_aktual"].astype(int)
    return X, y
```

Gambar 4. 44 Mendefinisikan Path dan Variabel

Selanjutnya, fungsi `load_data` didefinisikan dengan struktur yang sama untuk menjaga perlakuan data awal tidak berubah antar eksperimen. Data dimuat melalui `pd.read_csv`, lalu `drop_cols` dibentuk untuk memastikan hanya kolom ID yang benar benar ada di *dataset* yang dihapus. Variabel `X` kemudian dibentuk dengan menghapus kolom target `Y1_aktual` dan kolom identitas, sedangkan `y` diambil dari kolom `Y1_aktual` dan dikonversi menjadi integer agar label konsisten sebagai kelas diskrit. Dengan tahap ini, perbedaan kinerja yang muncul pada model *class weighted* bisa dikaitkan ke tahap pembobotan kelas saat pelatihan, bukan karena perubahan cara membaca atau membentuk fitur dari *dataset*.

Tahapan berikutnya adalah menyusun fungsi `build_model` untuk membangun model *Artificial Neural Network*. Struktur yang digunakan tetap sama seperti *baseline*, yaitu membentuk `StandardScaler` untuk menstandarkan fitur numerik dan `MLPClassifier` sebagai model *Artificial Neural Network*. Pemilihan konfigurasi `hidden_layer_sizes=(16, 8)` digunakan untuk membentuk dua *hidden layer* agar model mampu menangkap pola non linear pada data, `activation="relu"` dipakai untuk membantu pembelajaran yang lebih stabil, `solver="lbfgs"` dipilih karena umum dipakai dan cocok untuk optimasi jaringan saraf, `max_iter=2000` memberi ruang iterasi yang cukup agar pelatihan mendekati konvergen, dan `random_state=42` menjaga hasil eksperimen konsisten saat dijalankan ulang.

```
#definisikan fungsi build model
def build_model():
    scaler = StandardScaler()
    clf = MLPClassifier(
        hidden_layer_sizes=(16, 8),
        activation="relu",
        solver="lbfgs",
        alpha=0.01,
        max_iter=2000,
        random_state=42,
    )
    return Pipeline([("scaler", scaler), ("clf", clf)])
```

Gambar 4. 45 Kode Fungsi Build Model *Artificial Neural Network*

Setelah scaler dan classifier disusun, keduanya digabung dalam Pipeline dengan urutan `("scaler", scaler)` lalu `("clf", clf)`. Penyusunan pipeline ini menjaga proses standarisasi selalu terjadi sebelum data masuk ke *Artificial Neural Network*, baik saat pelatihan maupun saat prediksi. Pada bagian `build_model` memang tidak menambahkan parameter bobot kelas karena bobot kelas diterapkan pada tahap

pelatihan melalui `sample_weight` di fungsi `main`. Dengan demikian, model yang dibangun tetap setara dengan *baseline* dari sisi arsitektur, sementara perbedaan eksperimen terjadi pada cara proses pelatihannya memperhitungkan ketidakseimbangan kelas.

Pembagian data latih dan uji tetap dilakukan dengan alur yang sama seperti model *baseline*, karena fungsi utamanya adalah menjaga validitas evaluasi. Proses dimulai dengan memuat *dataset* melalui `load_data` sehingga diperoleh variabel Input (X) dan target y yang sudah bersih dari kolom identitas. Selanjutnya, `train_test_split` digunakan untuk memisahkan data menjadi data latih dan data uji dengan proporsi 80 persen untuk pelatihan dan 20 persen untuk pengujian. Parameter `stratify=y` memastikan distribusi kelas pada data latih dan data uji tetap sebanding dengan distribusi awal, sehingga ketidakseimbangan kelas tetap tercermin secara realistis.

```
# Pembagian data train-test
def main():
    X, y = load_data(DATA_PATH)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )
```

Gambar 4. 46 Pembagian data Pelatihan dan data Testing

Pada model *Artificial Neural Network* dengan pendekatan `class_weight`, hasil pembagian ini menjadi dasar penting untuk tahap berikutnya, yaitu pembobotan kelas yang hanya diterapkan pada data latih. Data uji sengaja dibiarkan tanpa perlakuan apa pun agar evaluasi kinerja mencerminkan kondisi nyata. Dengan demikian, tahapan pembagian train dan test berfungsi sebagai pengunci eksperimen, di mana perbedaan performa antara *baseline* dan *class_weight* benar benar berasal dari mekanisme pembobotan kelas saat pelatihan, bukan dari perubahan distribusi data atau perlakuan pada data uji.

Tahapan selanjutnya yang merupakan tahapan inti dari pembuatan model *Artificial Neural Network* dengan pendekatan `class_weight` karena pada bagian inilah mekanisme penyesuaian pengaruh tiap kelas terhadap proses pelatihan *Artificial Neural Network* diterapkan secara langsung. Proses dimulai dengan `classes = np.array(sorted(y_train.unique()))`, yang bertujuan mengidentifikasi seluruh kelas yang benar-benar muncul pada data latih. Penyusunan kelas dilakukan setelah pembagian data latih dan uji agar pembobotan mencerminkan kondisi distribusi data yang dipakai saat pelatihan model. Dengan mengetahui daftar kelas secara eksplisit,

proses pembobotan dapat dilakukan secara terkontrol dan tidak bergantung pada asumsi awal mengenai jumlah atau jenis kelas.

```
# menghitung bobot kelas
classes = np.array(sorted(y_train.unique()))
weights = compute_class_weight(class_weight="balanced",
                               classes=classes, y=y_train)
```

Gambar 4. 47 Kode Menghitung Bobot Kelas

Langkah berikutnya adalah menghitung bobot kelas menggunakan sintak `weights = compute_class_weight (class_weight="balanced", classes=classes, y=y_train)`. Parameter `class_weight="balanced"` digunakan agar bobot kelas dihitung secara otomatis berdasarkan proporsi kemunculan masing-masing kelas di data latih. Kelas dengan jumlah observasi lebih sedikit akan memperoleh bobot lebih besar, sedangkan kelas dengan jumlah observasi lebih banyak memperoleh bobot lebih kecil. Tujuan dari langkah ini adalah mengoreksi kecenderungan model untuk meminimalkan kesalahan pada kelas mayoritas saja, dengan cara meningkatkan penalti kesalahan pada kelas minoritas selama proses optimasi *Artificial Neural Network*.

Setelah menghitung bobot kelas, Tahapan selanjutnya menyusun bobot kelas ke format yang siap dipakai pada proses pelatihan *Artificial Neural Network*. Baris `class_weights = dict (zip (classes, weights))` menggabungkan dua informasi yang sudah dihitung sebelumnya, yaitu daftar label kelas pada `classes` dan nilai bobotnya pada `weights`. Fungsi `zip` memasangkan elemen secara berurutan, misalnya kelas 0 dipasangkan dengan bobot untuk kelas 0 dan kelas 1 dipasangkan dengan bobot untuk kelas 1, lalu `dict` mengubah pasangan tersebut menjadi kamus. Tujuan penggunaan dictionary adalah supaya bobot dapat dipanggil berdasarkan label kelas secara langsung, sehingga proses penerapan bobot menjadi jelas dan tidak bergantung pada urutan indeks array.

```
# Menyusun class weights
class_weights = dict(zip(classes, weights))
sample_weight = y_train.map(class_weights).to_numpy(dtype=np.float64)
```

Gambar 4. 48 Mendefinisikan *Class Weights*

Baris sintak selanjutnya `sample_weight = y_train.map (class_weights).to_numpy(dtype=np.float64)` untuk membentuk bobot per sampel. Metode `map` pada `y_train` memetakan setiap nilai label di data latih ke bobot yang sesuai di dalam `class_weights`, sehingga setiap baris data latih memiliki nilai bobot sendiri. Alasan membentuk `sample_weight` per baris adalah karena model tidak

menerima bobot per kelas secara langsung pada *Artificial Neural Network*, sehingga bobot perlu diterapkan pada level sampel saat pelatihan. Konversi ke NumPy dengan dtype float64 dilakukan agar formatnya sesuai dan stabil saat diteruskan ke proses optimasi, karena `sample_weight` dipakai sebagai pengali kontribusi *error* setiap sampel di *loss function*, sehingga sampel dari kelas minoritas memberi pengaruh lebih besar pada pembaruan bobot jaringan saraf.

Tahapan selanjutnya yaitu menjalankan inti pelatihan *Artificial Neural Network* dengan pendekatan *class weighted*, lalu menghasilkan *output* prediksi untuk evaluasi. Proses dimulai dari `model = build_model()` untuk membangun pipeline yang berisi `StandardScaler` dan `MLPClassifier` sesuai rancangan sebelumnya. Penempatan scaler di dalam pipeline menjaga fitur numerik selalu distandarkan sebelum masuk ke jaringan saraf, sehingga proses optimasi *Artificial Neural Network* lebih stabil dan tidak bias karena perbedaan skala antar variabel

```
# bangun, latih, dan prediksi model
model = build_model()
model.fit(X_train, y_train, clf__sample_weight=sample_weight)
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]
```

Gambar 4. 49 Kode Latih dan Prediksi Model

Perbedaan model *Artificial Neural Network Baseline* dengan model *Artificial Neural Network* dengan pendekatan *class weighted* ada pada baris sintak `model.fit(X_train, y_train, clf__sample_weight=sample_weight)`. Metode `fit` melatih model menggunakan data latih, tetapi setiap baris data diberi bobot melalui `sample_weight` yang sudah disusun dari distribusi kelas. Penulisan `clf__sample_weight` berarti bobot tersebut dikirim khusus ke langkah classifier di dalam Pipeline yang bernama `clf`. Tujuannya adalah membuat kesalahan prediksi pada kelas minoritas memberi penalti lebih besar selama pelatihan, sehingga model tidak hanya mengejar akurasi pada kelas mayoritas. Setelah model selesai dilatih, `y_pred = model.predict(X_test)` menghasilkan prediksi kelas diskrit pada data uji untuk kebutuhan *confusion matrix* dan *classification report*. Lalu `y_prob = model.predict_proba(X_test)[:, 1]` menghasilkan probabilitas fuzzy, yang dipakai untuk evaluasi seperti ROC-AUC, probabilitas dan, skor keyakinan, bukan hanya keputusan kelas.

BAB V

PENGUJIAN SISTEM DAN PEMBAHASAN

5.1 Pengujian Model *Artificial Neural Network*

Pengujian model *Artificial Neural Network* dilakukan untuk mengukur kemampuan model dalam memprediksi keberhasilan penawaran menggunakan data yang tidak terlibat dalam proses pelatihan. Pada tahap pelatihan, model *Artificial Neural Network* menggunakan data pelatihan untuk mempelajari pola hubungan antara variabel *input* dan variabel target. Setelah proses pelatihan selesai, model diuji menggunakan data testing untuk menilai kemampuan generalisasi terhadap data baru. Hasil prediksi kemudian dibandingkan dengan nilai aktual pada variabel *Y1_aktual* menggunakan beberapa matriks evaluasi, yaitu *accuracy*, *precision*, *Recall*, dan *F1 score*. Penggunaan beberapa matriks tersebut diperlukan karena *dataset* memiliki distribusi kelas yang tidak seimbang. Oleh karena itu, penilaian performa model tidak dapat hanya bertumpu pada nilai akurasi. Melalui pengujian ini, penelitian dapat menilai apakah model *Artificial Neural Network* mampu mengenali pola keberhasilan penawaran secara stabil dan dapat digunakan sebagai alat pendukung pengambilan keputusan.

Pengujian model *Artificial Neural Network* dilakukan dengan tiga pendekatan agar kinerja model dapat dianalisis secara lebih menyeluruh. Pendekatan pertama menggunakan model *Artificial Neural Network baseline*, yaitu model yang dilatih tanpa perlakuan khusus terhadap ketidakseimbangan kelas. Pendekatan kedua menggunakan metode *upsampling* pada data pelatihan untuk meningkatkan jumlah data kelas minoritas sehingga distribusi kelas menjadi lebih seimbang sebelum proses pelatihan. Pendekatan ketiga menggunakan strategi *class weighted*, yaitu pemberian bobot berbeda pada setiap kelas selama proses pelatihan model. Ketiga pendekatan tersebut diuji menggunakan data testing yang sama dan matriks evaluasi yang seragam. Dengan cara ini, perbedaan hasil antar model dapat dianalisis sebagai dampak dari masing-masing strategi dalam menangani ketidakseimbangan kelas.

Seluruh model diuji menggunakan modul yang sama, yaitu *sklearn.matrix*, dengan empat matriks utama, yaitu *accuracy_score*, *classification_report*, *confusion_matrix*, dan *roc_auc_score*. Pemilihan keempat matriks tersebut didasarkan pada kebutuhan untuk mengevaluasi performa model *Artificial Neural Network* secara komprehensif, terutama karena data penelitian memiliki distribusi kelas yang tidak seimbang. *Accuracy_score*

digunakan untuk memberikan gambaran umum mengenai tingkat ketepatan prediksi model secara keseluruhan. Walaupun *accuracy* memiliki keterbatasan pada data tidak seimbang, matriks ini tetap digunakan sebagai indikator awal untuk membandingkan performa global antara model *baseline*, *upsampling*, dan *class weighted*.

Metrik *classification_report*, *confusion_matrix*, dan ROC-AUC digunakan untuk melengkapi keterbatasan *accuracy*. *Classification_report* menampilkan nilai *precision*, *Recall*, dan *F1score* pada setiap kelas, sehingga kemampuan model dalam mengenali kelas mayoritas dan kelas minoritas dapat dianalisis secara terpisah. *Confusion_matrix* digunakan untuk menunjukkan jumlah prediksi benar dan salah pada setiap kelas, sehingga pola kesalahan model dapat diamati secara konkret. Sementara itu, ROC-AUC digunakan untuk mengukur kemampuan model dalam membedakan kelas berdasarkan probabilitas prediksi tanpa bergantung pada satu nilai *threshold* tertentu. Kombinasi keempat matriks tersebut dipilih karena dapat memberikan evaluasi yang lebih seimbang, mulai dari performa umum, performa per kelas, pola kesalahan, hingga kualitas pemisahan kelas. Dengan demikian, hasil pengujian model *Artificial Neural Network* dapat dianalisis secara lebih objektif, menyeluruh, dan sesuai dengan karakteristik data penelitian.

5.1.1 Pengujian Model *Artificial Neural Network Baseline*

Proses perancangan dan pelatihan model *Artificial Neural Network baseline* diselesaikan pada tahap sebelumnya, model kemudian diuji menggunakan data uji yang tidak dilibatkan dalam proses pembelajaran. Tahap pengujian ini bertujuan untuk menilai sejauh mana model mampu melakukan generalisasi, yaitu memprediksi data baru secara akurat berdasarkan pola yang dipelajari dari data latih. Evaluasi dilakukan dengan menggunakan beberapa matriks yang telah ditetapkan agar kinerja model dapat dianalisis secara menyeluruh, baik dari sisi akurasi global maupun kemampuan model dalam membedakan masing-masing kelas.

```
# Menampilkan hasil uji dan evaluasi
print("Distribusi train:", y_train.value_counts().to_dict())
print("Akurasi:", round(accuracy_score(y_test, y_pred), 3))
print("ROC-AUC:", round(roc_auc_score(y_test, y_prob), 3))
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification report:\n", classification_report
      (y_test, y_pred, digits=3))
```

Gambar 5 1 Kode Model *Artificial Neural Network Baseline*

Setelah model *Artificial Neural Network baseline* menghasilkan *y_pred* dan *y_prob* pada data uji, tahap evaluasi dijalankan dengan menampilkan matriks utama melalui

perintah `print`. Sintak `print("Distribusi train:", y_train.value_counts().to_dict())` menampilkan jumlah data per kelas pada `y_train`. Fungsi `value_counts` menghitung frekuensi tiap kelas pada label pelatihan, lalu `to_dict` mengubah hasilnya menjadi format *dictionary* agar lebih ringkas saat dicetak. Tujuan langkah ini adalah mendokumentasikan kondisi distribusi kelas pada data latih, sehingga pembaca dapat melihat apakah data termasuk tidak seimbang sebelum model dilatih.

```
# akurasi dan ROC-AUC
print("Akurasi:", round(accuracy_score(y_test, y_pred), 3))
print("ROC-AUC:", round(roc_auc_score(y_test, y_prob), 3))
```

Gambar 5 2 Kode Perhitungan *Accuracy* dan ROC-AUC

Berikutnya untuk menghitung akurasi menggunakan metrix `accuracy_score` dengan *input* `y_test` sebagai label aktual dan `y_pred` sebagai hasil prediksi kelas. Fungsi `round` dengan parameter 3 digunakan untuk membulatkan hasil sampai tiga angka desimal supaya *output* konsisten dan mudah dibandingkan antar model. Setelah itu menghitung ROC-AUC menggunakan menggunakan `y_test` dan `y_prob`, bukan `y_pred`. Alasan penggunaan `y_prob` adalah karena ROC-AUC membutuhkan skor kontinu probabilitas untuk menilai kemampuan model membedakan kelas di berbagai ambang keputusan. Pembulatan tiga desimal kembali dipakai agar format pelaporan seragam.

```
#confusion matrix dan classification report
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification report:\n", classification_report
      (y_test, y_pred, digits=3))
```

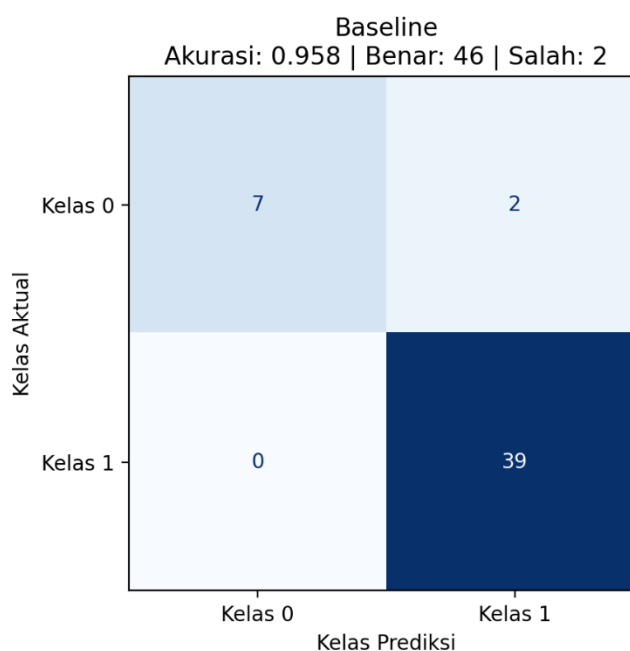
Gambar 5 3 Kode Perhitungan *Confusion Matrix* dan *Classification Report*

Matriks selanjutnya adalah *Confusion matrix* dengan *input* yang sama seperti akurasi, yaitu `y_test` dan `y_pred`. *Confusion matrix* ditulis untuk memperlihatkan jumlah prediksi benar dan salah pada tiap kelas, sehingga pola kesalahan *baseline* dapat dianalisis, misalnya apakah model sering salah mendeteksi kelas minoritas. Sintak terakhir yaitu; `Classification_report (y_test, y_pred, digits=3)` menampilkan laporan klasifikasi yang berisi *precision*, *Recall*, *f1 score*, dan *Support* untuk setiap kelas. Parameter `digits=3` dipakai agar seluruh matriks di report ditampilkan dengan tiga angka desimal, konsisten dengan hasil akurasi dan ROC-AUC yang dibulatkan. Susunan *output* seperti ini dipilih karena memberikan urutan evaluasi yang logis, dimulai dari konteks distribusi data, lalu matriks ringkas global, dilanjutkan matriks pemisahan kelas berbasis probabilitas, kemudian rincian kesalahan prediksi dan ringkasan performa per kelas.

```
Distribusi train: {1: 154, 0: 35}
Akurasi: 0.958
ROC-AUC: 0.957
Confusion matrix:
[[ 8  1]
 [ 1 38]]
```

Gambar 5 4 Hasil Evaluasi Model *Artificial Neural Network Baseline*

Setelah program di jalankan, dengan menggunakan perintah `print()`, didapatkan hasil distribusi kelas pada gambar dengan penyebaran kelas 1 sebanyak 154 dan kelas 2 sebanyak 35. Tingkat akurasi sebesar 0.985, nilai ROC-AUC sebesar 0.957.



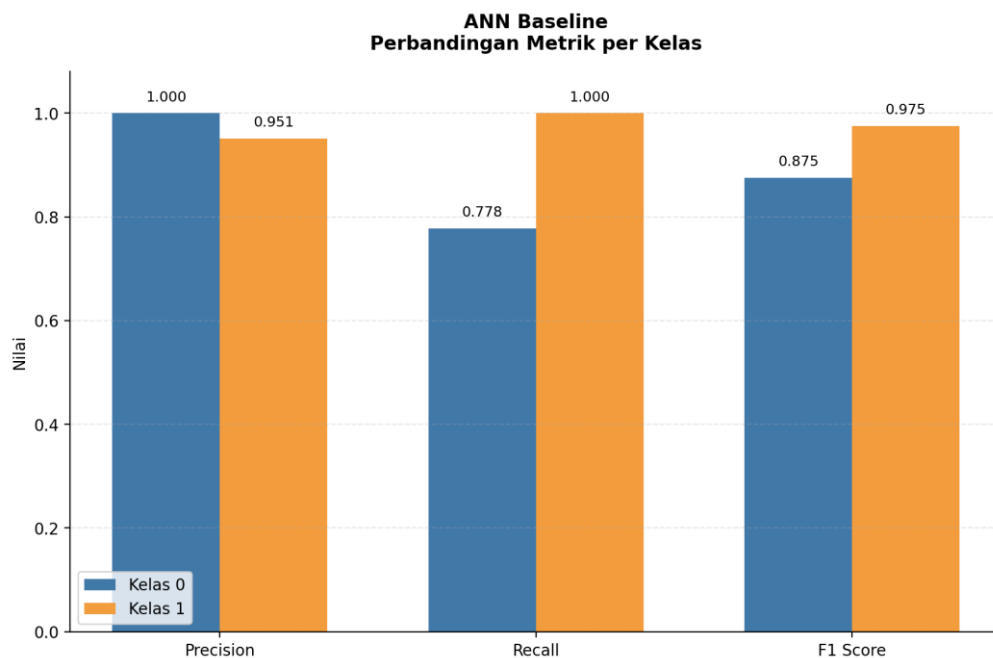
Gambar 5 5 Grafik *Heatmap Confusion Matrix* Model *Artificial Neural Network Baseline*

Hasil dari *confusion matrix* menunjukkan terdapat 8 data kelas 0 yang diprediksi benar sebagai kelas 0 seperti yang tertera dalam angka 8 pada posisi baris pertama kolom pertama. Angka 1 pada baris pertama kolom kedua menunjukkan 1 data kelas 0 yang salah diprediksi sebagai kelas 1. Baris kedua kolom pertama bernilai 1, yang berarti terdapat 1 data kelas 1 yang salah diprediksi sebagai kelas 0. Angka 38 pada baris kedua kolom kedua

Classification report:				
	precision	recall	f1-score	support
0	1.000	0.778	0.875	9
1	0.951	1.000	0.975	39
accuracy			0.958	48
macro avg	0.976	0.889	0.925	48
weighted avg	0.960	0.958	0.956	48

Gambar 5 6 Hasil *Classification Report* Model *Artificial Neural Network Baseline*

Classification report memperlihatkan model *Artificial Neural Network baseline* memperoleh *accuracy* sebesar 0,958 dari total 48 data uji. Nilai tersebut menggambarkan bahwa model mampu memprediksi sebagian besar data penawaran dengan benar. Pada kelas 0, model menghasilkan *precision* sebesar 1,000, *Recall* sebesar 0,778, dan *f1 score* sebesar 0,875 dari 9 data uji. Hasil ini mengindikasikan bahwa prediksi model terhadap kelas 0 memiliki tingkat ketepatan yang sangat baik, tetapi kemampuan model dalam menangkap seluruh data aktual kelas 0 masih belum sempurna. Pada kelas 1, model menghasilkan *precision* sebesar 0,951, *Recall* sebesar 1,000, dan *f1 score* sebesar 0,975 dari 39 data uji. Nilai tersebut mencerminkan bahwa model sangat baik dalam mengenali penawaran yang berhasil, karena seluruh data aktual kelas 1 berhasil diprediksi dengan benar.



Gambar 5 7 Grafik Perbandingan *Precision*, *Recall*, dan *F1-score* per Kelas pada Model *Artificial Neural Network Baseline*

Gambar memperjelas perbedaan karakteristik performa model pada kelas 0 dan kelas 1. Pada matriks *precision*, kelas 0 memiliki nilai lebih tinggi yaitu 1,000 dibandingkan kelas 1 sebesar 0,951, sehingga model lebih selektif ketika memberikan prediksi kelas 0. Namun, pada matriks *Recall*, kelas 1 memiliki nilai lebih tinggi yaitu 1,000 dibandingkan kelas 0 sebesar 0,778. Pola ini menunjukkan bahwa model lebih kuat dalam mengenali seluruh penawaran yang berhasil daripada mengenali seluruh penawaran yang tidak berhasil. Pada matriks *f1 score*, kelas 1 juga lebih unggul dengan nilai 0,975 dibandingkan kelas 0 sebesar 0,875, yang berarti keseimbangan antara *precision* dan *Recall* pada kelas 1 lebih stabil. Perbedaan ini terjadi karena jumlah data kelas 1 lebih besar dibandingkan kelas 0, sehingga model memiliki lebih banyak pola pembelajaran pada penawaran berhasil. Dengan demikian, model *Artificial Neural Network baseline* sudah menunjukkan performa yang baik, tetapi masih memiliki kelemahan pada sensitivitas terhadap kelas 0 sebagai kelas minoritas.

5.1.2 Pengujian Model *Artificial Neural Network Upsampling*

Pengujian selanjutnya yaitu *Artificial Neural Network* dengan metode *upsampling* untuk melihat dampak penyeimbangan kelas terhadap performa model. Pada tahap awal evaluasi *upsampling*, distribusi kelas pada data latih ditampilkan dalam dua kondisi, yaitu sebelum dan sesudah *upsampling*. Baris sintak pertama `print ("Distribusi train asli:", y_train.value_counts().to_dict())` menampilkan jumlah sampel tiap kelas pada data latih asli, sehingga terlihat tingkat ketidakseimbangan kelas yang menjadi alasan diterapkannya *upsampling*. *Output* dibuat ringkas dengan *value_counts* untuk menghitung frekuensi kelas dan *to_dict* agar format tampilannya mudah dibaca di terminal.

```
# tampilkan hasil uji dan evaluasi model upsample
print("Distribusi train asli:", y_train.value_counts().to_dict())
print("Distribusi train upsampling:", y_train_bal.value_counts().to_dict())
```

Gambar 5 8 Distribusi Data Latih Model *Artificial Neural Network Upsampling*

Baris sintak kedua yang berbeda dari Model *Artificial Neural Network Baseline* baris `("Distribusi train upsampling:", y_train_bal.value_counts().to_dict())` menampilkan distribusi kelas setelah proses *upsampling* diterapkan pada data latih. Variabel `y_train_bal` merepresentasikan label data latih yang sudah diseimbangkan, sehingga hasil print ini berfungsi sebagai bukti bahwa metode *upsampling* berhasil menyamakan jumlah sampel antar kelas sebelum pelatihan model dilakukan. Perbandingan dua distribusi ini penting dalam bab uji model *upsampling* karena menjadi dasar bahwa perubahan

performa pada matriks evaluasi berikutnya berasal dari perubahan komposisi data latih, sedangkan data uji tetap dipertahankan sama seperti skenario *baseline*.

Setelah distribusi kelas pada data latih berhasil dibandingkan antara kondisi asli dan kondisi setelah *upsampling*, evaluasi kinerja model dilanjutkan dengan menampilkan matriks utama, yaitu akurasi dan ROC-AUC. menggunakan *accuracy_score* menghitung akurasi model dengan membandingkan label aktual *y_test* dan hasil prediksi kelas *y_pred* pada data uji. Pembulatan dengan *round* sampai tiga angka desimal dipakai agar format pelaporan konsisten dan memudahkan perbandingan langsung dengan hasil *baseline* maupun pendekatan *class weighted*.

```
# hasil akurasi dan ROC-AUC
print("Akurasi:", round(accuracy_score(y_test, y_pred), 3))
print("ROC-AUC:", round(roc_auc_score(y_test, y_prob), 3))
```

Gambar 5.9 Perhitungan Accuracy dan ROC-AUC Model *Artificial Neural Network Upsampling*

Dilanjutkan dengan menghitung nilai ROC-AUC menggunakan *matrix roc_auc_score* dengan *input* *y_test* dan probabilitas kelas positif *y_prob*. Pemakaian *y_prob* penting karena ROC-AUC menilai kemampuan model dalam membedakan kelas berdasarkan skor probabilitas, bukan berdasarkan label prediksi akhir. Dengan menampilkan akurasi dan ROC-AUC pada tahap ini, performa model *upsampling* dapat dinilai dari dua sisi, yaitu ketepatan prediksi keseluruhan serta kualitas pemisahan kelas pada data uji yang tetap menggunakan distribusi asli.

Tahapan uji selanjutnya dilanjutkan dengan menampilkan *confusion matrix* dan *classification report* untuk melihat performa model secara lebih rinci per kelas menampilkan *confusion matrix* dengan membandingkan label aktual *y_test* dan prediksi kelas *y_pred*. *Confusion matrix* dipakai untuk memperlihatkan jumlah prediksi benar dan salah pada setiap kelas, sehingga dapat terlihat apakah metode *upsampling* membantu mengurangi kesalahan pada kelas minoritas, misalnya menurunkan *false negative* atau *false positive* yang sebelumnya lebih dominan pada *baseline*.

```
#hasil confusion matrix dan classification report
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification report:\n",
      classification_report(y_test, y_pred, digits=3))
```

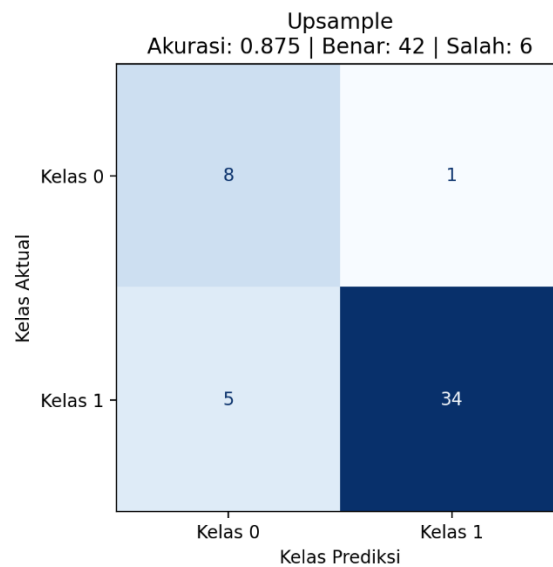
Gambar 5 10 Kode Confusion Matrix dan *Classification Report* Model *Artificial Neural Network Upsampling*

Untuk menampilkan ringkasan matriks klasifikasi per kelas, yaitu *precision*, *Recall*, *f1 score*, serta *Support*, dengan menggunakan *classification_report*. Menggunakan Parameter *digits=3* untuk menampilkan nilai matriks sampai tiga angka desimal agar format pelaporan konsisten dengan matriks sebelumnya dan memudahkan perbandingan antar pendekatan. Tahap ini penting pada model *upsampling* karena fokus utamanya bukan hanya peningkatan akurasi global, tetapi juga perbaikan kemampuan model dalam mengenali kelas minoritas, yang biasanya terlihat lebih jelas melalui *Recall* dan *f1 score* pada kelas tersebut.

```
Distribusi train asli: {1: 154, 0: 35}
Distribusi train upsample: {1: 154, 0: 154}
Akurasi: 0.875
ROC-AUC: 0.9
Confusion matrix:
[[ 8  1]
 [ 5 34]]
```

Gambar 5 11 Hasil Evaluasi Model *Artificial Neural Network Upsampling*

Hasil uji model *Artificial Neural Network* dengan metode *upsampling* menunjukkan bahwa distribusi data latih awal tidak seimbang, dengan kelas 1 sebanyak 154 data dan kelas 0 sebanyak 35 data, kemudian setelah *upsampling* distribusi data latih menjadi seimbang, yaitu kelas 1 dan kelas 0 masing-masing sebanyak 154 data. Nilai akurasi yang diperoleh sebesar 0.854 dan nilai ROC-AUC sebesar 0.940.



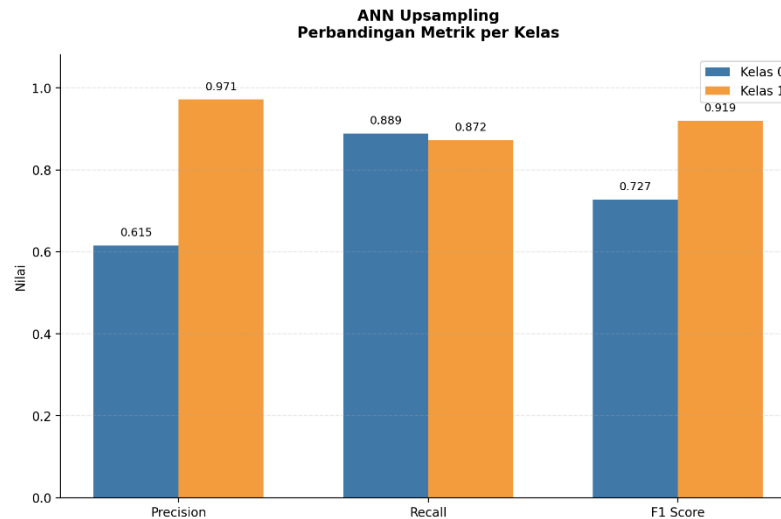
Gambar 5 12 Grafik *Heatmap Confusion Matrix* Model *Artificial Neural Network Upsampling*

Confusion matrix menunjukkan 7 data kelas 0 diprediksi benar sebagai kelas 0, 2 data kelas 0 salah diprediksi sebagai kelas 1, 5 data kelas 1 salah diprediksi sebagai kelas 0, dan 34 data kelas 1 diprediksi benar sebagai kelas 1.

Classification report:				
	precision	recall	f1-score	support
0	0.583	0.778	0.667	9
1	0.944	0.872	0.907	39
accuracy			0.854	48
macro avg	0.764	0.825	0.787	48
weighted avg	0.877	0.854	0.862	48

Gambar 5 13 Hasil *Classification Report* Model *Artificial Neural Network Upsampling*

Selanjutnya hasil dari *classification report* model *Artificial Neural Network* dengan metode *upsampling* menampilkan matriks per kelas dan matriks agregat. kelas 0 menampilkan *precision* 0.583, *Recall* 0.778, *f1 score* 0.667, dan *Support* 9. Angka *precision* 0.583 berarti dari seluruh prediksi yang diberi label 0 oleh model, sekitar 58,3 persen benar sebagai kelas 0. Angka *Recall* 0.778 berarti dari seluruh data aktual kelas 0 pada data uji, sekitar 77,8 persen berhasil terdeteksi sebagai kelas 0. Nilai *f1 score* 0.667 merupakan gabungan *precision* dan *Recall* untuk kelas 0. *Support* 9 menunjukkan jumlah data aktual kelas 0 pada data uji adalah 9 baris.



Gambar 5 14 Grafik *Classification Report* Model *Artificial Neural Network Upsampling*

kemampuan model terhadap kelas 0 meningkat pada sisi *Recall*, yaitu sebesar 0,889, bahkan sedikit lebih tinggi dibandingkan kelas 1 sebesar 0,872. Gambar menunjukkan bahwa model menjadi lebih sensitif dalam mengenali penawaran tidak berhasil setelah data kelas minoritas diperbanyak pada tahap pelatihan. Namun, *precision* kelas 0 hanya sebesar 0,615, jauh lebih rendah dibandingkan kelas 1 sebesar 0,971, sehingga model masih menghasilkan cukup banyak prediksi kelas 0 yang tidak tepat. Nilai *f1 score* kelas 0 sebesar 0,727 juga lebih rendah dibandingkan kelas 1 sebesar 0,919, yang mencerminkan bahwa peningkatan kemampuan mendeteksi kelas 0 belum diikuti oleh ketepatan prediksi yang seimbang. Secara keseluruhan, metode *upsampling* membantu model memperhatikan kelas 0 dengan lebih baik, tetapi menimbulkan penurunan kestabilan prediksi pada kelas minoritas dibandingkan kelas mayoritas.

5.1.3 Pengujian Model *Artificial Neural Network Class Weighted*

Uji model *Artificial Neural Network* dengan pendekatan *Class Weighted* diawali dengan menampilkan kondisi data latih dan bobot kelas yang digunakan saat pelatihan. Dengan fungsi *Value_counts* untuk menghitung jumlah kemunculan setiap kelas pada variabel *y_train*. Dan fungsi *to_dict* untuk mengubah hasil perhitungan ke dalam bentuk dictionary agar *output* lebih ringkas.

```
# menampilkan hasil uji dan evaluasi
print("Distribusi train:", y_train.value_counts().to_dict())
print("Bobot kelas:", class_weights)
```

Gambar 5 15 Kode Distribusi dan Bobot Kelas Model *Artificial Neural Network Class Weighted*

Baris print ke-2 pada gambar untuk menampilkan nilai bobot yang sudah dihitung dari *y_train*. Bobot ini menunjukkan pengaturan penalti yang diterapkan saat pelatihan, di

mana kelas minoritas memiliki nilai bobot lebih besar dibanding kelas mayoritas. Penampilan bobot kelas pada *output* menjadi bukti bahwa mekanisme *class weighted* benar-benar diterapkan dalam pelatihan model *Artificial Neural Network* karena pada pendekatan *weighted class* jumlah data tidak diubah., sehingga hasil evaluasi berikutnya dapat ditelusuri langsung pada strategi pembobotan ini.

Hasil akurasi dari model *Artificial Neural Network Class Weighted* menggunakan data uji yang sama seperti model *Artificial Neural Network baseline*. Menggunakan *matrix accuracy_score* yang sama seperti model sebelumnya untuk menghitung tingkat ketepatan prediksi model dengan membandingkan label aktual *y_test* dan hasil prediksi kelas *y_pred*. Fungsi *accuracy_score* digunakan untuk memperoleh proporsi prediksi benar secara keseluruhan, lalu round dengan angka 3 digunakan untuk membulatkan hasil sampai tiga desimal agar format pelaporan konsisten dan mudah dibandingkan dengan hasil *baseline* dan *upsampling*.

```
# menampilkan akurasi dan ROC-AUC
print("Akurasi:", round(accuracy_score(y_test, y_pred), 3))
print("ROC-AUC:", round(roc_auc_score(y_test, y_prob), 3))
```

Gambar 5 16 Kode Evaluasi *Accuracy* dan *ROC-AUC Class Weighted*

Untuk menghitung ROC-AUC dengan *matrix* `roc_auc_score` dengan label aktual `y_test` dan probabilitas prediksi `y_prob`. Pembulatan tiga desimal juga diterapkan untuk menjaga konsistensi tampilan hasil. Pada pendekatan *weighted class*, dua matriks ini dipakai untuk melihat performa global model dan kualitas pemisahan kelas setelah proses pelatihan yang memperhitungkan bobot sampel melalui `sample_weight`.

Berikutnya untuk menampilkan *confusion matrix* dan *classification report* untuk melihat detail performa model per kelas. Menggunakan fungsi `confusion_matrix` dengan input `y_test` sebagai label aktual dan `y_pred` sebagai hasil prediksi kelas.

```
# menampilkan confusion matrix dan classification report
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification report:\n",
      classification_report(y_test, y_pred, digits=3))
```

Gambar 5 17 Kode Evaluasi Klasifikasi Model *Artificial Neural Network Class Weighted*

`Classification_report` untuk menampilkan ringkasan matriks klasifikasi per kelas, yaitu *precision*, *Recall*, *f1 score*, dan *Support*. Parameter `digits=3` digunakan untuk menyeragamkan tampilan angka hingga tiga desimal, agar pelaporan hasil konsisten dengan matriks akurasi dan ROC-AUC yang juga dibulatkan. Pada pendekatan *weighted class*, keluaran dua evaluasi ini dipakai untuk mendokumentasikan perubahan performa per kelas setelah pelatihan model dilakukan dengan `sample_weight`, tanpa mengubah komposisi data latih seperti pada metode *upsampling*.

Hasil awal uji model *Artificial Neural Network* dengan pendekatan *class weighted* menunjukkan bahwa distribusi data latih masih tidak seimbang, dengan kelas 1 sebanyak 154 data dan kelas 0 sebanyak 35 data. Kondisi ini sama seperti pada model *baseline* karena pendekatan *class weighted* tidak mengubah jumlah data latih, melainkan hanya mengubah kontribusi tiap kelas saat proses pelatihan

```
Distribusi train: {1: 154, 0: 35}
Bobot kelas: {np.int64(0): np.float64(2.7), np.int64(1): np.float64(0.6136363636363636)}
```

Gambar 5 18 Distribusi dan Bobot Kelas *Class Weighted*

Bobot kelas yang digunakan menunjukkan bahwa kelas 0 memiliki bobot sebesar 2.7, sedangkan kelas 1 memiliki bobot sebesar sekitar 0.614. Nilai bobot ini menunjukkan

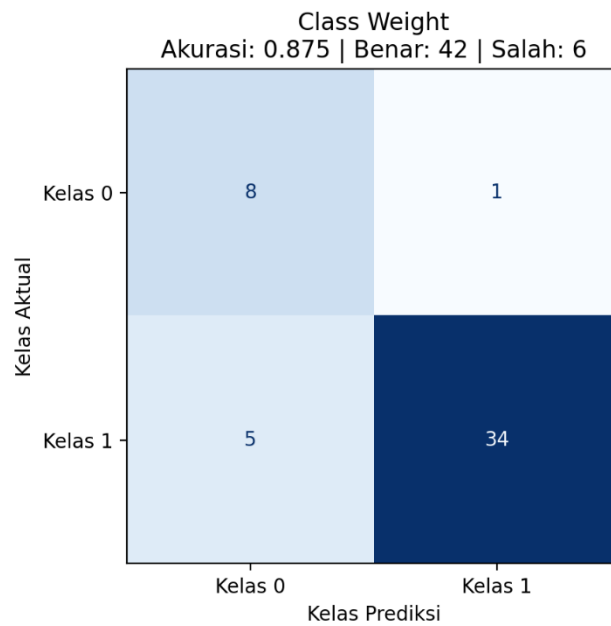
bahwa setiap kesalahan prediksi pada kelas 0 diberi penalti yang jauh lebih besar dibanding kelas 1 selama proses pelatihan model *Artificial Neural Network*

Selanjutnya nilai akurasi dari Hasil evaluasi model *Artificial Neural Network* dengan pendekatan *class weighted* menunjukkan nilai sebesar 0.896, yang berarti model memprediksi dengan benar sekitar 89,6 persen dari seluruh data uji. Nilai ROC-AUC sebesar 0.943 menunjukkan kualitas pemisahan kelas yang tinggi berdasarkan skor probabilitas, sehingga model mampu memberi peringkat yang baik antara kelas 0 dan kelas 1 pada berbagai ambang keputusan. Kedua nilai ini ditampilkan sebagai ringkasan performa global model pada data uji, dengan akurasi berbasis prediksi kelas dan ROC-AUC berbasis probabilitas.

```
Akurasi: 0.875
ROC-AUC: 0.96
Confusion matrix:
[[ 8  1]
 [ 5 34]]
```

Gambar 5 19 Hasil Evaluasi Model *Artificial Neural Network Class Weighted*

Hasil dari *confusion matrix* menunjukkan 8 data kelas 0 dan 35 data kelas 1 berhasil diprediksi dengan benar. Terdapat 1 data kelas 0 yang salah diprediksi sebagai kelas 1 dan 4 data kelas 1 yang salah diprediksi sebagai kelas 0. Matriks ini merepresentasikan distribusi kesalahan dan prediksi benar model *Artificial Neural Network* dengan pendekatan *class weighted* pada data uji.



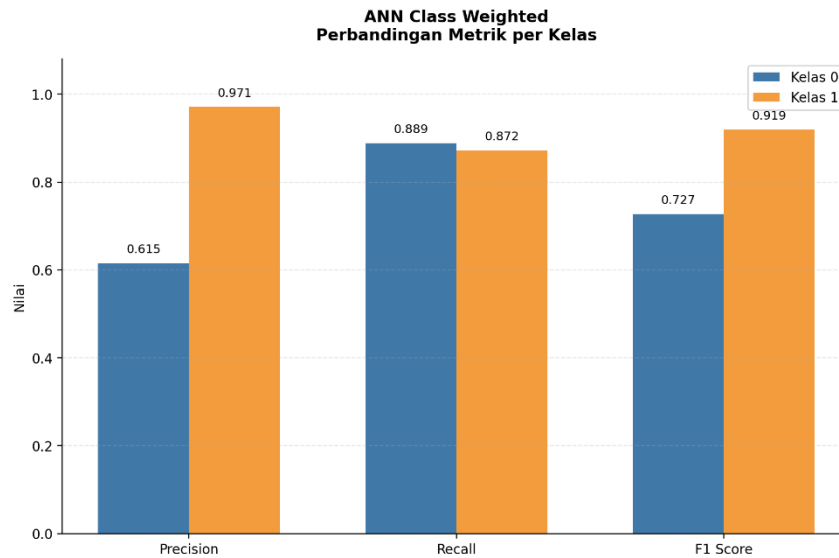
Gambar 5 20 Grafik *Heatmap Confusion Matrix Model Artificial Neural Network Class Weighted*

Berikutnya hasil dari *Classification report* menunjukkan kinerja yang berbeda pada masing-masing kelas. Untuk kelas 0, nilai *precision* sebesar 0.667 menunjukkan bahwa sekitar dua pertiga prediksi kelas 0 merupakan prediksi yang benar. Nilai *Recall* sebesar 0.889 menunjukkan bahwa sebagian besar data aktual kelas 0 berhasil dikenali oleh model. Nilai *f1 score* sebesar 0.762 mencerminkan keseimbangan antara *precision* dan *Recall* pada kelas ini, dengan jumlah data uji sebanyak 9 observasi.

Classification report:				
	precision	recall	f1-score	support
0	0.615	0.889	0.727	9
1	0.971	0.872	0.919	39
accuracy			0.875	48
macro avg	0.793	0.880	0.823	48
weighted avg	0.905	0.875	0.883	48

Gambar 5 21 Hasil *Classification Report Model Artificial Neural Network Class Weighted*

Hasil *classification report* menunjukkan *accuracy* sebesar 0,875 dari total 48 data uji. Pada kelas 0, nilai *precision* sebesar 0,615, *Recall* sebesar 0,889, *F1-score* sebesar 0,727, dan *Support* sebanyak 9 data. Pada kelas 1, nilai *precision* sebesar 0,971, *Recall* sebesar 0,872, *F1-score* sebesar 0,919, dan *Support* sebanyak 39 data. Nilai *macro average* terdiri dari *precision* 0,793, *Recall* 0,880, dan *F1-score* 0,823. Nilai *weighted average* terdiri dari *precision* 0,905, *Recall* 0,875, dan *F1-score* 0,883.



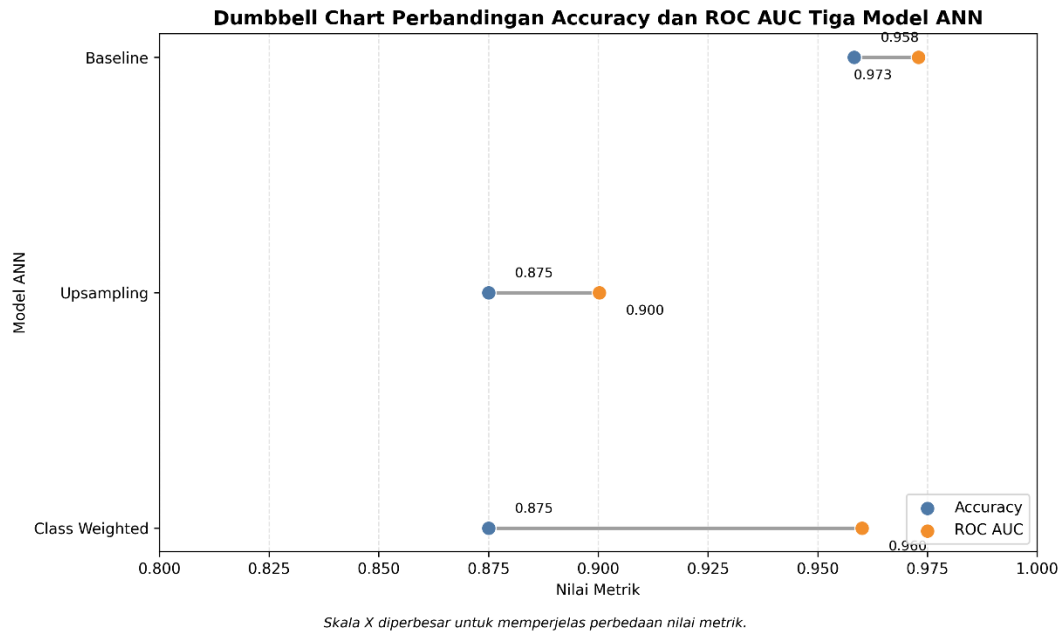
Gambar 5 22 Grafik *Classification Report* Model *Artificial Neural Network Class Weighted*

Gambar memperlihatkan pola performa yang lebih sensitif terhadap kelas 0, tetapi belum sepenuhnya stabil pada ketepatan prediksinya. Nilai *Recall* kelas 0 sebesar 0,889 lebih tinggi dibandingkan kelas 1 sebesar 0,872, sehingga model lebih mampu mengenali data aktual kelas 0 setelah bobot kesalahan pada kelas minoritas diperbesar. Namun, *precision* kelas 0 hanya sebesar 0,615, jauh di bawah *precision* kelas 1 sebesar 0,971, yang mengindikasikan bahwa peningkatan kemampuan mengenali kelas 0 masih diikuti oleh banyak prediksi kelas 0 yang kurang tepat. Nilai *f1 score* kelas 0 sebesar 0,727 juga lebih rendah dibandingkan kelas 1 sebesar 0,919, sehingga keseimbangan antara ketepatan dan kemampuan deteksi pada kelas 0 belum sebaik kelas 1. Hasil ini menggambarkan bahwa pendekatan *class weighted* berhasil mengarahkan model agar lebih memperhatikan kelas minoritas, tetapi performa keseluruhan masih lebih kuat pada kelas 1 sebagai kelas mayoritas.

5.2 Perbandingan Hasil Pegujian Model *Artificial Neural Network*

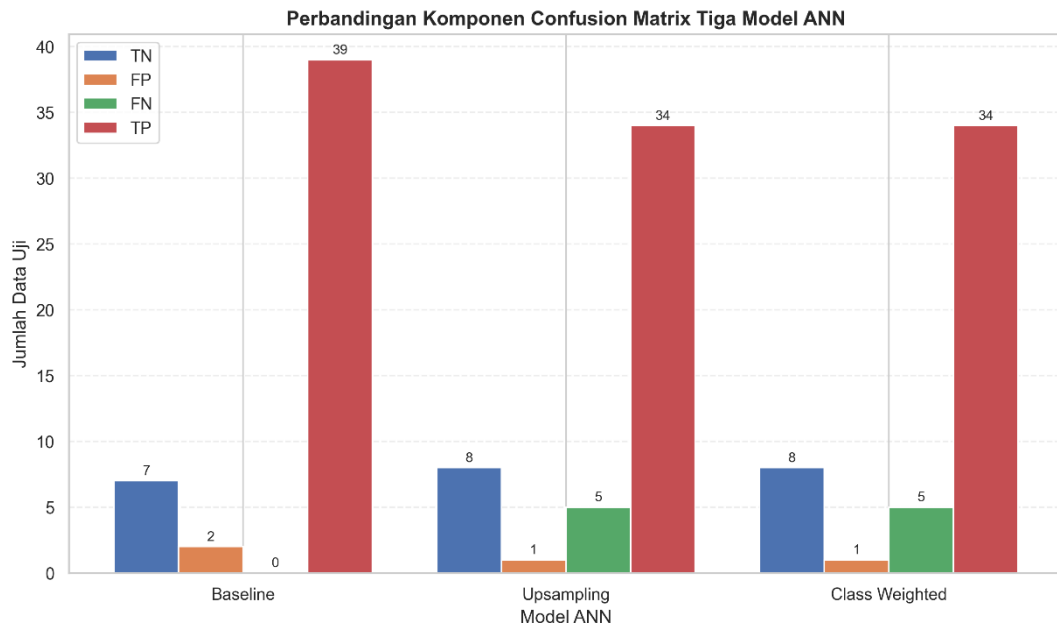
Hasil pengujian ketiga model *Artificial Neural Network* dibandingkan untuk mengetahui model yang memiliki performa paling sesuai dengan tujuan penelitian. Perbandingan dilakukan terhadap model *Artificial Neural Network baseline*, *Artificial Neural Network upsampling*, dan *Artificial Neural Network class weighted*, karena ketiga pendekatan tersebut memiliki perlakuan berbeda terhadap distribusi kelas pada data penawaran. Model *baseline* digunakan sebagai acuan awal, model *upsampling* digunakan untuk menyeimbangkan jumlah data antar kelas, sedangkan model *class weighted* digunakan untuk memberi bobot lebih besar pada kelas minoritas saat pelatihan. Ketiga model diuji

menggunakan data uji yang sama dan matriks evaluasi yang konsisten, sehingga hasil perbandingan dapat menunjukkan pengaruh masing-masing pendekatan terhadap performa prediksi model.



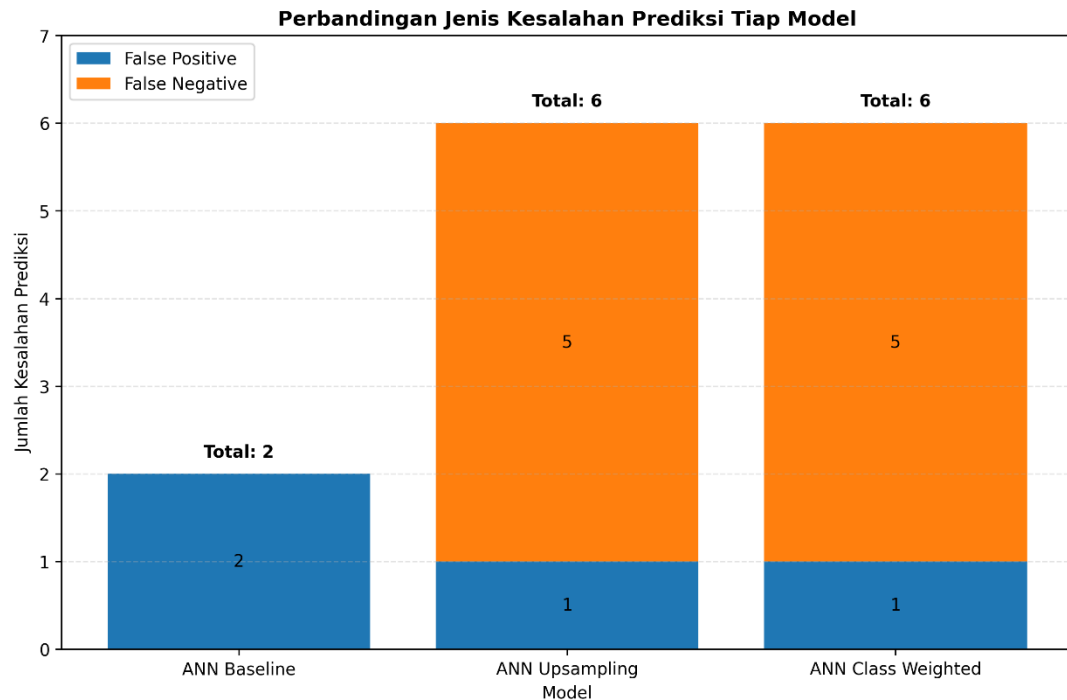
Gambar 5.23 Grafik *Dumbbell Chart* Perbandingan *Accuracy* dan ROC-AUC Tiga Model *Artificial Neural Network*

Grafik *dumbbell* pada gambar 5.23 memperlihatkan jarak antara *accuracy* dan ROC-AUC pada setiap model, sehingga pola kestabilan performa dapat dibaca lebih jelas. Model *Artificial Neural Network baseline* memiliki posisi paling kanan dengan *accuracy* 0,958 dan ROC-AUC 0,973, serta jarak kedua matriks yang kecil. Kondisi ini menunjukkan bahwa prediksi kelas akhir dan kemampuan pemisahan probabilitas model berjalan searah. Model *upsampling* berada pada posisi terendah dengan *accuracy* 0,875 dan ROC-AUC 0,900, serta tidak menunjukkan peningkatan yang kuat setelah distribusi kelas diseimbangkan melalui duplikasi data. Model *class weighted* memiliki *accuracy* yang sama dengan *upsampling*, yaitu 0,875, tetapi ROC-AUC naik jauh hingga 0,960. Jarak yang lebar pada *class weighted* menunjukkan bahwa model sebenarnya cukup baik dalam membedakan peluang antar kelas, tetapi keputusan akhir pada *threshold* masih belum menghasilkan ketepatan prediksi setinggi *baseline*. Berdasarkan pola jarak tersebut, *baseline* lebih stabil sebagai model utama, sedangkan *class weighted* dapat dipahami sebagai model yang kuat pada skor probabilitas tetapi belum optimal pada klasifikasi akhir.



Gambar 5 24 Grafik Perbandingan Komponen *Confusion Matrix* Tiga Model *Artificial Neural Network*

Setelah perbandingan *accuracy* dan ROC-AUC menunjukkan bahwa model *baseline* memiliki performa global paling kuat, grafik komponen *confusion matrix* memberikan gambaran lebih rinci mengenai sumber perbedaan performa tersebut. Pada model *baseline*, prediksi benar pada kelas 1 mencapai 39 data dan tidak terdapat *false negative*, meskipun masih muncul 2 *false positive* pada kelas 0. Model *upsampling* dan *class weighted* sama-sama menghasilkan prediksi benar kelas 0 yang lebih tinggi, yaitu 8 data, serta *false positive* yang lebih rendah, yaitu 1 data. Namun, kedua model tersebut mengalami penurunan pada kelas 1 karena hanya menghasilkan 34 *true positive* dan memiliki 5 *false negative*. Pola ini menunjukkan bahwa *upsampling* dan *class weighted* lebih baik dalam mengenali sebagian data kelas 0, tetapi kehilangan kemampuan dalam menjaga prediksi benar pada kelas 1 sebagai kelas penawaran berhasil.

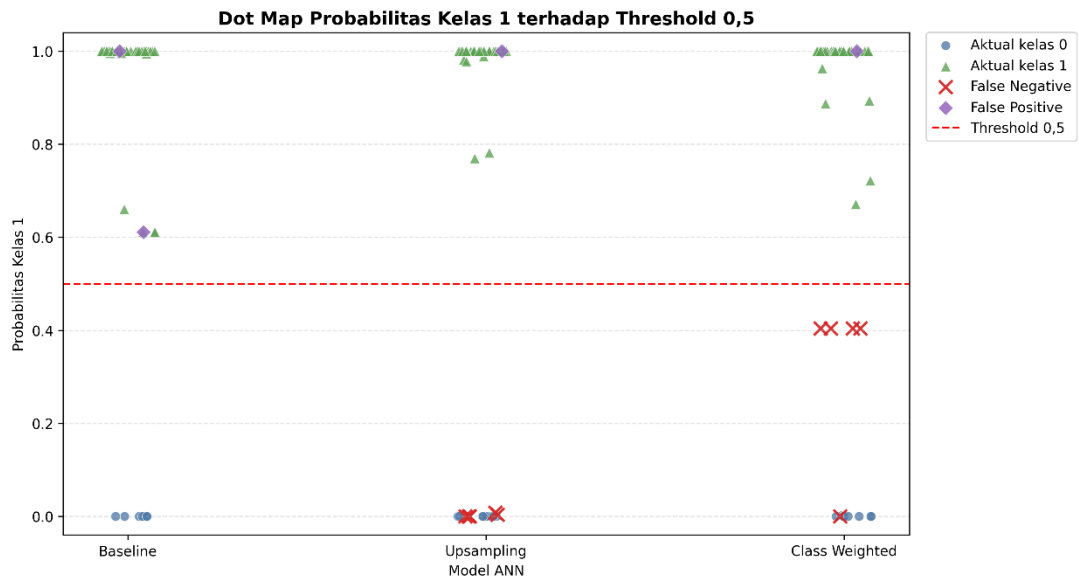


Gambar 5.25 Perbandingan *False Positive* dan *False Negative* Tiga Model *Artificial Neural Network*

Berdasarkan grafik *error comparison* pada gambar 5.25, perlakuan terhadap ketidakseimbangan kelas melalui *upsampling* dan *class weighted* belum memberikan hasil yang lebih sesuai terhadap kebutuhan prediksi keberhasilan penawaran. Kedua pendekatan tersebut memang mampu menekan kesalahan *false positive*, tetapi peningkatan *false negative* menunjukkan bahwa model menjadi lebih ketat dalam mengklasifikasikan penawaran sebagai berhasil. Kondisi ini berdampak pada meningkatnya risiko model membaca penawaran yang sebenarnya berhasil sebagai tidak berhasil

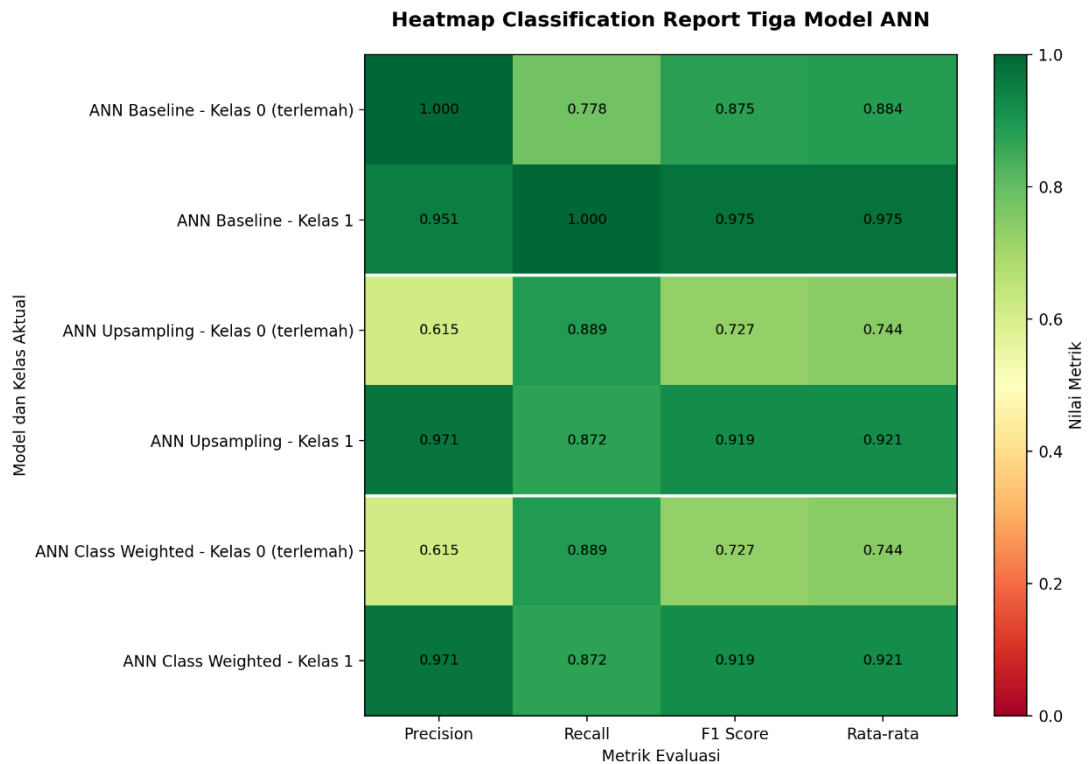
False negative menjadi perhatian utama karena kesalahan ini membuat model membaca penawaran yang sebenarnya berhasil sebagai penawaran tidak berhasil. Dalam proses bisnis, kondisi tersebut dapat membuat perusahaan salah menurunkan prioritas pada penawaran yang memiliki peluang menjadi *Purchase Order*. Risiko ini lebih besar daripada *false positive*, karena perusahaan berpotensi kehilangan peluang transaksi yang seharusnya dapat ditindaklanjuti. Berdasarkan pola tersebut, model *baseline* menjadi model yang paling aman untuk prediksi keberhasilan penawaran karena mampu mempertahankan *false negative* pada nilai 0 dan menghasilkan *true positive* paling tinggi. Meskipun *baseline* memiliki *false positive* lebih besar dibandingkan *upsampling* dan

class weighted, kesalahan tersebut masih lebih dapat diterima karena tidak menghilangkan peluang penawaran berhasil yang menjadi fokus utama penelitian.



Gambar 5.26 Dot Map Probabilitas Kelas 1 terhadap *Threshold* 0,5 pada Tiga Model *Artificial Neural Network*

Setelah risiko kesalahan prediksi pada grafik sebelumnya dianalisis, sebaran probabilitas terhadap *threshold* 0,5 pada gambar 5.26 memberikan penjelasan lebih rinci mengenai penyebab munculnya *false negative* pada tiap model. Pada model *baseline*, data aktual kelas 1 banyak berada di atas *threshold* dan data aktual kelas 0 berada di bawah *threshold*, sehingga pemisahan kelas terlihat lebih stabil. Pada model *upsampling*, *false negative* berada sangat rendah di bawah *threshold*, bahkan mendekati probabilitas 0. Kondisi ini menunjukkan bahwa model memberikan keyakinan yang kuat pada prediksi kelas tidak berhasil, padahal data aktualnya merupakan penawaran berhasil. Sebaliknya, pada model *class weighted*, *false negative* terlihat lebih dekat ke *threshold* 0,5, terutama pada area sekitar 0,4. Pola ini menunjukkan bahwa pembobotan kelas mampu menaikkan probabilitas sebagian data aktual kelas 1, tetapi belum cukup untuk melewati batas keputusan. Perbedaan ini menjelaskan mengapa *class weighted* memiliki kemampuan pemisahan probabilitas yang lebih baik dibandingkan *upsampling*, tetapi tetap menghasilkan kesalahan klasifikasi akhir. Dampaknya, kedua model masih berisiko membuat perusahaan menurunkan prioritas pada penawaran yang sebenarnya berhasil, sedangkan *baseline* lebih aman karena peluang penawaran berhasil lebih konsisten berada pada area keputusan yang tepat.

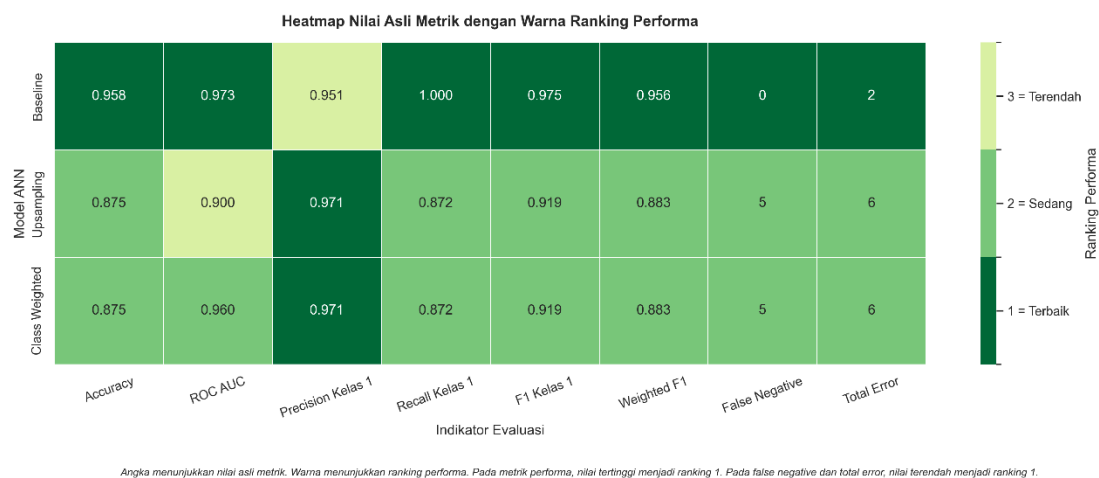


Gambar 5 27 Heatmap *Classification Report* Tiga Model *Artificial Neural Network*

Pola probabilitas terhadap *threshold* 0,5 menunjukkan bahwa setiap model memiliki tingkat kestabilan prediksi yang berbeda. Untuk memperjelas kualitas prediksi pada masing-masing kelas, heatmap *classification report* pada gambar 5 27, memperlihatkan kualitas prediksi setiap model pada kelas 0 dan kelas 1 secara lebih rinci. Pada kelas 0, *Artificial Neural Network baseline* memiliki *precision* tertinggi sebesar 1,000, tetapi *Recall* lebih rendah sebesar 0,778. Pola ini menunjukkan bahwa ketika *baseline* memprediksi kelas tidak berhasil, prediksinya sangat tepat, tetapi masih ada sebagian data kelas 0 yang terbaca sebagai kelas 1. *Artificial Neural Network upsampling* dan *Artificial Neural Network class weighted* memiliki *Recall* kelas 0 lebih tinggi sebesar 0,889, tetapi *precision* turun menjadi 0,615. Kondisi ini menunjukkan bahwa kedua model lebih sering menangkap data kelas 0, tetapi ketepatan prediksi kelas 0 lebih lemah. Pada kelas 1, *Artificial Neural Network baseline* menghasilkan *Recall* tertinggi sebesar 1,000 dan *F1-score* 0,975, sedangkan *upsampling* dan *class weighted* memiliki *Recall* kelas 1 sebesar 0,872 dan *F1-score* 0,919. Perbedaan ini memperlihatkan bahwa *baseline* lebih kuat dalam mempertahankan kemampuan mengenali penawaran berhasil.

Hasil *heatmap* memperkuat pembahasan sebelumnya bahwa model terbaik tidak hanya dipilih dari kemampuan mengenali kelas minoritas, tetapi juga dari risiko kesalahan terhadap kelas yang menjadi fokus penelitian. Pada penelitian ini, kelas 1 lebih

penting karena merepresentasikan penawaran berhasil. Nilai *Recall* kelas 1 pada *baseline* yang mencapai 1,000 menunjukkan bahwa model mampu mengenali seluruh penawaran berhasil pada data uji. Hal ini sejalan dengan hasil *error comparison* dan dot map *threshold*, di mana *baseline* tidak menghasilkan *false negative* dan sebaran probabilitas kelas 1 berada pada area keputusan yang lebih aman. *Upsampling* dan *class weighted* memang lebih sensitif terhadap kelas 0, tetapi penurunan *Recall* kelas 1 menunjukkan adanya risiko melewatkan penawaran yang sebenarnya berhasil. Berdasarkan pola tersebut.



Gambar 5 28 Heatmap Nilai Asli dan Ranking Performa Tiga Model *Artificial Neural Network*

Rangkuman seluruh matriks evaluasi yang ditunjukkan gambar 5.28 memperlihatkan bahwa *Artificial Neural Network baseline* memiliki performa paling dominan dibandingkan dua model lainnya. *Baseline* unggul pada *accuracy*, ROC-AUC, *Recall* kelas 1, *F1-score* kelas 1, *weighted F1*, *false negative*, dan *total error*. Pola warna ranking juga menunjukkan bahwa sebagian besar indikator *baseline* berada pada posisi terbaik, sehingga performanya terlihat lebih konsisten. *Artificial Neural Network upsampling* dan *Artificial Neural Network class weighted* memiliki keunggulan pada *precision* kelas 1, yang menunjukkan bahwa prediksi kelas berhasil pada kedua model tersebut lebih selektif. Namun, keunggulan ini belum diikuti oleh kemampuan mengenali seluruh data kelas 1, karena *Recall* kelas 1 masih lebih rendah dibandingkan *baseline*. *Class weighted* memiliki ROC-AUC lebih tinggi daripada *upsampling*, tetapi hasil pada *Recall* kelas 1, *F1-score* kelas 1, *false negative*, dan *total error* masih sama dengan *upsampling*. Kondisi ini menunjukkan bahwa peningkatan kemampuan pemisahan probabilitas pada *class weighted* belum mampu memperbaiki keputusan klasifikasi akhir.

Pemilihan model terbaik mempertimbangkan hubungan antar matriks, terutama performa global, kemampuan mengenali penawaran berhasil, dan risiko kesalahan yang dapat merugikan proses penjualan. *Baseline* memenuhi kebutuhan tersebut karena memiliki *Recall* kelas 1 tertinggi dan *false negative* terendah, sehingga model lebih aman dalam menjaga penawaran yang sebenarnya berpotensi menjadi Purchase Order agar tetap teridentifikasi. *Upsampling* dan *class weighted* memang lebih unggul pada *precision* kelas 1, tetapi penurunan *Recall* kelas 1 dan peningkatan total *error* membuat kedua model tersebut kurang sesuai sebagai model utama. Berdasarkan pola performa tersebut, *Artificial Neural Network baseline* menjadi model yang paling layak dipilih karena memberikan hasil paling stabil dan paling sesuai dengan tujuan prediksi keberhasilan penawaran.

5.3 Pengujian Variabel Kontribusi

Uji variabel berkontribusi dilakukan untuk memahami peran masing masing variabel *input* dalam proses pengambilan keputusan oleh model *Artificial Neural Network*. Pengujian ini bertujuan untuk mengetahui variabel mana yang memberikan kontribusi paling besar terhadap kemampuan model dalam melakukan klasifikasi, serta variabel mana yang kontribusinya relatif kecil. Dengan adanya uji variabel berpengaruh, hasil pemodelan tidak hanya dipandang sebagai keluaran prediksi semata, tetapi juga sebagai alat untuk menelusuri pola hubungan antara variabel *input* dan *output* model. Fungsi utama dari uji ini adalah memberikan dasar pemahaman terhadap perilaku model *Artificial Neural Network* yang bersifat kompleks dan nonlinear, sehingga keputusan model dapat dipahami secara lebih sistematis.

Selain untuk mengetahui hubungan variabel uji ini juga berfungsi sebagai alat evaluasi tambahan terhadap kualitas fitur yang digunakan. Variabel dengan pengaruh rendah dapat menjadi indikator bahwa informasi yang dikandungnya kurang relevan atau tumpang tindih dengan variabel lain, sedangkan variabel dengan pengaruh tinggi menunjukkan bahwa informasi tersebut memiliki peran penting dalam membedakan kelas target. Dengan demikian, hasil uji ini dapat dimanfaatkan untuk penyederhanaan model, perbaikan struktur data, serta sebagai dasar penyusunan rekomendasi analitis yang lebih terarah. Pendekatan ini juga membantu memastikan bahwa model *Artificial Neural Network* tidak hanya bekerja secara akurat, tetapi juga selaras dengan logika hubungan variabel yang masuk akal dalam konteks data yang digunakan.

Metode yang digunakan untuk menguji variabel berpengaruh adalah *Permutation Importance*. Pendekatan ini bekerja dengan cara mengacak nilai satu variabel pada data uji, kemudian mengukur penurunan kinerja model setelah pengacakan dilakukan. Jika pengacakan suatu variabel menyebabkan penurunan nilai ROC-AUC yang besar, maka variabel tersebut dinilai memiliki pengaruh yang kuat terhadap prediksi model. Sebaliknya, jika penurunan kinerja relatif kecil, kontribusi variabel tersebut dianggap lebih rendah. Metode *Permutation Importance* dipilih karena sesuai untuk model *Artificial Neural Network* yang bersifat non inear dan sulit diinterpretasikan secara langsung, serta karena penilaiannya berbasis dampak nyata terhadap performa model, sehingga hasilnya relevan untuk analisis dan pembahasan lanjutan dalam penelitian

Model yang dipilih sebagai uji variabel adalah model *Artificial Neural Network baseline* karena dasar uji variabel berpengaruh didasarkan pada nilai ROC-AUC yang paling tinggi dibandingkan dengan pendekatan *upsampling* dan *class weighted*. ROC-AUC dipilih sebagai indikator utama karena matriks ini menilai kemampuan model dalam membedakan kelas berdasarkan skor probabilitas secara menyeluruh, tanpa bergantung pada satu ambang keputusan tertentu. Nilai ROC-AUC yang tinggi menunjukkan bahwa model memiliki kualitas pemisahan kelas yang baik dan stabil, sehingga perubahan kinerja yang muncul saat variabel diuji dapat dikaitkan langsung dengan peran variabel tersebut, bukan akibat kelemahan model dalam membedakan kelas.

Tahapan pertama dalam uji variabel berpengaruh adalah menyiapkan *Library* yang dibutuhkan untuk memuat model *baseline*, mengelola data, membagi data uji, dan menghitung matriks kinerja yang akan dipakai sebagai acuan pengaruh variabel. *Import joblib* digunakan untuk memanggil kembali model *Artificial Neural Network baseline* yang sebelumnya sudah disimpan dalam bentuk file, sehingga uji variabel berpengaruh dapat dilakukan tanpa melatih ulang model. *Import pandas as pd* digunakan untuk membaca dan mengolah *dataset* dalam bentuk *DataFrame*, karena proses pengacakan nilai variabel dilakukan pada level kolom.

Menggunakan modul dari Scikit-learn *Import* fungsi *train_test_split* untuk membentuk pembagian data yang konsisten antara data latih dan data uji, sehingga proses pengukuran penurunan kinerja akibat pengacakan variabel dilakukan pada data uji yang sama. *roc_auc_score* diimpor karena matriks ini dipakai sebagai skor utama yang dipantau sebelum dan sesudah variabel diacak. ROC-AUC dipilih pada tahap ini karena pengujian *Permutation Importance* membutuhkan matriks yang stabil berbasis probabilitas,

sehingga perubahan skor dapat mencerminkan seberapa besar kontribusi variabel terhadap kemampuan model membedakan kelas.

Selanjutnya menambahkan modul `permutation_importance` dari `sklearn.inspection` sebagai alat utama untuk mengukur pengaruh variabel pada model *Artificial Neural Network baseline*. Fungsi `permutation_importance` digunakan untuk menjalankan mekanisme pengujian dengan cara mengacak nilai satu variabel pada data uji secara bergantian, kemudian menghitung penurunan skor kinerja model setelah pengacakan dilakukan. Semakin besar penurunan skor yang terjadi, semakin besar kontribusi variabel tersebut terhadap kemampuan model dalam melakukan klasifikasi.

```
# library permutation importance
from sklearn.inspection import permutation_importance
```

Gambar 5 29 Import Library Permutation Importance

Alasan menggunakan modul ini karena *Artificial Neural Network* tidak menyediakan koefisien atau bobot interpretatif yang langsung menunjukkan tingkat pengaruh tiap variabel seperti pada model linear. Dengan `permutation_importance`, pengukuran pengaruh variabel didasarkan pada dampak nyata terhadap performa model, sehingga hasilnya dapat dipakai sebagai dasar pemeringkatan variabel yang paling berperan dalam meningkatkan ROC-AUC pada data uji.

Langkah berikutnya adalah menetapkan lokasi file data dan file model *baseline* yang akan dipakai pada uji variabel berpengaruh, serta mendefinisikan kolom identitas yang harus dikeluarkan dari fitur. Variabel DATA menunjuk ke file *dataset* yang sama dengan proses pelatihan, sehingga fitur yang diuji tetap konsisten dengan fitur yang digunakan saat model dibangun. Variabel MODEL menunjuk ke file artefak model *Artificial Neural Network baseline* yang sudah disimpan sebelumnya, sehingga proses uji variabel berpengaruh dapat langsung memakai model terlatih tanpa melakukan pelatihan ulang.

```
# Definisi path data dan model
DATA = "data/data_model_ann.csv"
MODEL = "artifacts/model_ann_baseline.joblib"
ID_COLS = ["nomor_penawaran", "kode_pelanggan"]
```

Gambar 5 30 Definisi Path Data dan Model

Variabel `ID_COLS` didefinisikan untuk menandai kolom identitas seperti `nomor_penawaran` dan `kode_pelanggan`. Kolom ini dipertahankan sebagai informasi administratif, tetapi tidak boleh masuk sebagai variabel *input* saat menghitung *Permutation Importance*, karena kolom identitas tidak merepresentasikan karakteristik

yang relevan untuk prediksi. Dengan tahap ini, proses uji variabel berpengaruh berjalan pada fitur yang valid dan sejalan dengan fitur yang digunakan pada model *baseline*.

Langkah selanjutnya yaitu memuat data dan membentuk struktur dasar yang digunakan dalam uji variabel berpengaruh, yaitu variabel Input (X) dan variabel target y. menggunakan fungsi `pd.read_csv` yang berasal dari *Library* *pandas* untuk membaca file *dataset* dari path DATA dan menyimpannya ke dalam sebuah DataFrame bernama `df`. Penggunaan *pandas* dipilih karena struktur DataFrame memudahkan pengelolaan kolom, penghapusan variabel tertentu, serta manipulasi data yang diperlukan pada tahap selanjutnya, khususnya saat melakukan permutasi nilai variabel

```
# masukan data dan pisahkan fitur dan target
df = pd.read_csv(DATA)
X = df.drop(columns=[c for c in ["Y1_aktual", *ID_COLS] if c in df.columns])
y = df["Y1_aktual"].astype(int)
```

Gambar 5 31 Pembentukan Variabel Fitur dan Target untuk Uji *Permutation Importance*

Variabel X didefinisikan sebagai kumpulan seluruh fitur *input* yang digunakan oleh model *Artificial Neural Network baseline*. Pembentukan X dilakukan dengan metode `drop` pada DataFrame, yang berfungsi untuk menghapus kolom-kolom tertentu agar tidak ikut masuk sebagai fitur. Kolom yang dihapus mencakup kolom target `Y1_aktual` dan kolom identitas yang tercantum dalam `ID_COLS`, yaitu `nomor_penawaran` dan `kode_pelanggan`. Penghapusan kolom target karena target tidak boleh digunakan sebagai *input* dalam proses evaluasi pengaruh variabel. Sementara itu, kolom identitas dihapus karena tidak mengandung informasi prediktif, hanya berfungsi sebagai penanda administratif. Dengan proses ini, variabel X berisi murni variabel prediktor yang sebelumnya digunakan oleh model *Artificial Neural Network* dalam melakukan klasifikasi, sehingga setiap variabel di dalam X dapat diuji kontribusinya secara individual melalui metode *Permutation Importance*.

Variabel y didefinisikan sebagai variabel target yang merepresentasikan label kelas aktual. Variabel ini dibentuk dengan mengambil kolom `Y1_aktual` dari DataFrame, kemudian dikonversi ke tipe integer menggunakan `astype(int)`. Konversi tipe data ini bertujuan memastikan bahwa label target berada dalam format numerik untuk kebutuhan perhitungan matriks klasifikasi dan ROC-AUC. Variabel y berperan sebagai acuan dalam proses evaluasi, karena nilai inilah yang dibandingkan dengan hasil prediksi model sebelum dan sesudah dilakukan pengacakan variabel. Dengan struktur X sebagai fitur dan y sebagai target, uji variabel berpengaruh dapat dilakukan secara sistematis untuk menilai

sejauh mana perubahan pada satu variabel *input* memengaruhi kinerja model *Artificial Neural Network baseline*.

Tahapan berikutnya membagi data menjadi data latih dan data uji, lalu memanggil model *Artificial Neural Network baseline* yang sudah disimpan, fungsi `train_test_split` dipakai untuk memecah pasangan data X dan y menjadi empat keluaran, yaitu `X_train`, `X_test`, `y_train`, dan `y_test`. Struktur fungsi ini menerima *input* fitur X dan target y, lalu mengatur proporsi data uji melalui parameter `test_size=0.2`, artinya 20 persen data dialokasikan sebagai data uji dan sisanya menjadi data latih. Parameter `random_state=42` berfungsi mengunci proses pengacakan saat pembagian data, sehingga komposisi data latih dan data uji akan sama ketika kode dijalankan ulang. Target dari step ini adalah menyediakan data uji yang konsisten untuk menghitung ROC-AUC *baseline* dan untuk menjalankan *Permutation Importance*, karena proses permutasi harus diuji pada data yang tidak dipakai melatih mode.

```
# Bagi data uji dan latih
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = joblib.load(MODEL)
```

Gambar 5 32 Pembagian Data Latih, Data Uji, dan Pemanggilan Model *Baseline*

Selanjutnya, fungsi `joblib.load(MODEL)` digunakan untuk memuat kembali objek model dari file yang tersimpan pada path `MODEL`. Tujuannya untuk menggunakan kembali model *Artificial Neural Network baseline* yang sudah terlatih tanpa melakukan pelatihan ulang, sehingga uji variabel berpengaruh benar benar mengukur dampak perubahan fitur terhadap model yang sudah final. Dengan urutan ini, data uji sudah siap, model sudah tersedia, lalu tahap berikutnya bisa langsung menghitung skor AUC dan menjalankan *Permutation Importance*.

Sebelum menghitung Skor ROC-AUC dan melakukan pengacakan variabel dengan *Permutation Importance*, harus menyelaraskan fitur pada data uji agar cocok dengan struktur *input* model yang tersimpan. fungsi `getattr()` untuk mengambil atribut `feature_names_in_` dari objek model. Objek yang diperiksa adalah model. Nama atribut yang dicari adalah `feature_names_in_` dalam bentuk string, dan nilai `default` diisi `None` agar kode tetap aman jika atribut tidak ada. Target dari langkah ini adalah memperoleh daftar fitur yang direkam model saat pelatihan

```
# menyesuaikan fitur jika model menyimpan nama fitur
feature_names = getattr(model, "feature_names_in_", None)
if feature_names is not None:
    X_test = X_test[feature_names]
```

Gambar 5 33 Penyesuaian Urutan Fitur Data Uji sesuai Model *Baseline*

struktur `if feature_names is not None` digunakan sebagai filter agar proses penyesuaian hanya dilakukan jika model memang menyimpan daftar fitur. `X_test = X_test[feature_names]` digunakan untuk memilih kolom berdasarkan daftar nama kolom, sekaligus menyusun ulang urutan kolom mengikuti urutan pada list `feature_names`. Target dari indexing ini adalah memaksa `X_test` memiliki format *input* yang identik dengan data saat model dilatih, sehingga setiap nilai fitur masuk ke *input* yang tepat. Tujuannya adalah mencegah fitur tidak masuk ke *input* yang tepat yang menyebabkan evaluasi AUC dan hasil *Importance* menjadi keliru.

```
# hitung ROC-AUC baseline
auc = roc_auc_score(y_test, model.predict_proba(X_test)[: , 1])
```

Gambar 5 34 Perhitungan ROC-AUC Model *Artificial Neural Network Baseline*

selanjutnya menghitung ROC-AUC *baseline* dengan fungsi `roc_auc_score`. Parameter `y_true` diisi `y_test` sebagai label aktual pada data uji. Parameter `y_score` diisi skor probabilitas yang dihasilkan model melalui `model.predict_proba(X_test)`. Fungsi `predict_proba` menghasilkan array dua dimensi, dengan baris mewakili observasi dan kolom mewakili kelas. Indexing `[: , 1]` digunakan untuk mengambil seluruh baris pada kolom indeks 1, yaitu probabilitas untuk kelas positif atau kelas 1. Tujuan dari Target dari indexing ini adalah memperoleh skor kontinu 0 sampai 1 yang dibutuhkan oleh ROC-AUC, karena ROC-AUC menilai kemampuan model membedakan kelas berdasarkan ranking probabilitas, bukan berdasarkan label prediksi akhir. Nilai AUC yang dihasilkan menjadi *baseline score* yang akan dibandingkan dengan skor setelah fitur diacak pada *Permutation Importance*, sehingga penurunan skor dapat diinterpretasikan sebagai besarnya kontribusi fitur terhadap kemampuan pemisahan kelas.

Langkah berikutnya menjalankan fungsi `permutation_importance` sebagai inti uji variabel berpengaruh, mengukur seberapa besar kontribusi setiap fitur terhadap kinerja model *Artificial Neural Network baseline* dengan cara mengacak nilai fitur satu per satu pada data uji. Fungsi ini menerima beberapa *input* utama, yaitu model sebagai estimator yang dievaluasi, `X_test` sebagai data fitur yang dipakai untuk pengujian, dan `y_test` sebagai label aktual. Target penggunaan `X_test` dan `y_test` adalah memastikan

pengukuran dilakukan pada data yang tidak dipakai melatih model, sehingga penurunan skor yang terjadi saat fitur diacak mencerminkan perubahan kemampuan generalisasi, bukan sekadar kecocokan pada data latih.

```
# hitung permutation importance
res = permutation_importance(model, X_test, y_test, scoring="roc_auc",
                             n_repeats=10, random_state=42)
```

Gambar 5 35 Perhitungan *Permutation Importance*

Parameter `scoring="roc_auc"` digunakan untuk menetapkan matriks yang dipantau selama proses permutasi, sehingga setiap kali satu fitur diacak, fungsi akan menghitung ulang ROC-AUC dan membandingkannya dengan skor awal. `n_repeats=10` berfungsi mengulang proses pengacakan untuk setiap fitur sebanyak 10 kali, sehingga hasil *Importance* tidak bergantung pada satu kali pengacakan yang bisa saja menghasilkan penurunan skor yang terlalu ekstrem atau terlalu kecil. `random_state=42` digunakan untuk mengunci proses acak agar hasil dapat di replikasi *output-nya* dan disimpan dalam variabel `res` yang berisi nilai penurunan skor untuk tiap fitur pada setiap pengulangan, serta ringkasan berupa rata-rata dan variasinya yang akan dipakai pada tahap penyusunan tabel *Importance*.

Langkah Berikutnya adalah menyusun *output Permutation Importance* menjadi tabel yang rapi agar mudah dibaca, diurutkan, dan dipakai untuk pelaporan. Fungsi utama yang dipakai adalah `pd.DataFrame` dari `pandas`, yang strukturnya menerima sebuah dictionary berisi pasangan nama kolom dan data kolom. Pada bagian ini, kolom feature diisi dengan `X_test.columns`, yaitu daftar nama kolom fitur yang benar-benar masuk ke model saat evaluasi. Targetnya adalah memastikan setiap nilai *Importance* punya label variabel yang jelas, sehingga hasilnya tidak hanya berupa angka, tetapi langsung terhubung dengan nama variabel di *dataset*.

```
# menyusun hasil importance dalam DataFrame
imp = (pd.DataFrame({"feature": X_test.columns,
                     "rata-rata penurunan": res.importances_mean,
                     "kestabilan fitur": res.importances_std}))
     .sort_values("rata-rata penurunan", ascending=False))
```

Gambar 5 36 Penyusunan Hasil *Permutation Importance* dalam DataFrame

Kolom Rata-rata penurunan diisi dari `res.Importances_mean`, yaitu nilai rata rata penurunan skor evaluasi ketika suatu fitur diacak. Angka ini menjadi ukuran utama pengaruh variabel karena menunjukkan seberapa besar kinerja model turun saat informasi pada fitur tersebut “dirusak” melalui permutasi. Kolom Kestabilan fitur diisi dari `res.Importances_std`, yaitu simpangan baku penurunan skor dari proses pengulangan

n_repeats. karena fitur yang pengaruhnya konsisten akan memiliki variasi penurunan skor yang lebih kecil dibanding fitur yang hasilnya fluktuatif.

Langkah terakhir untuk menampilkan hasil uji variabel berpengaruh agar mudah dibaca. Perintah print pertama digunakan untuk menampilkan nilai ROC-AUC *baseline* yang sebelumnya sudah dihitung. Penulisan f string dengan format {auc:.3f} bertujuan membatasi tampilan angka hingga tiga desimal agar konsisten dengan pelaporan matriks pada bab uji model sebelumnya. Nilai ini menjadi titik acuan utama, karena seluruh nilai *Importance* dihitung berdasarkan penurunan ROC-AUC dari skor *baseline* tersebut.

```
# menampilkan hasil
print(f"ROC-AUC baseline: {auc:.3f}")
print("Top fitur:")
print(imp.head(10))
```

Gambar 5 37 Menampilkan Hasil *Permutation Importance*

Perintah print(imp.head(10)) menampilkan sepuluh variabel teratas dari DataFrame imp yang sebelumnya sudah diurutkan berdasarkan nilai mean *Importance* secara menurun. Fungsi head(10) dipakai untuk membatasi tampilan hanya pada variabel paling berpengaruh, sehingga fokus analisis tidak melebar ke variabel dengan kontribusi kecil. Tahap ini menjadi jembatan antara perhitungan teknis *Permutation Importance* dan tahap pembahasan, karena daftar fitur teratas inilah yang kemudian diinterpretasikan sebagai variabel paling dominan dalam memengaruhi kinerja prediksi model *Artificial Neural Network baseline*.

Berikut hasil dari uji variabel berpengaruh dari pengacakan fitur dengan metode *Permutation Importance*.

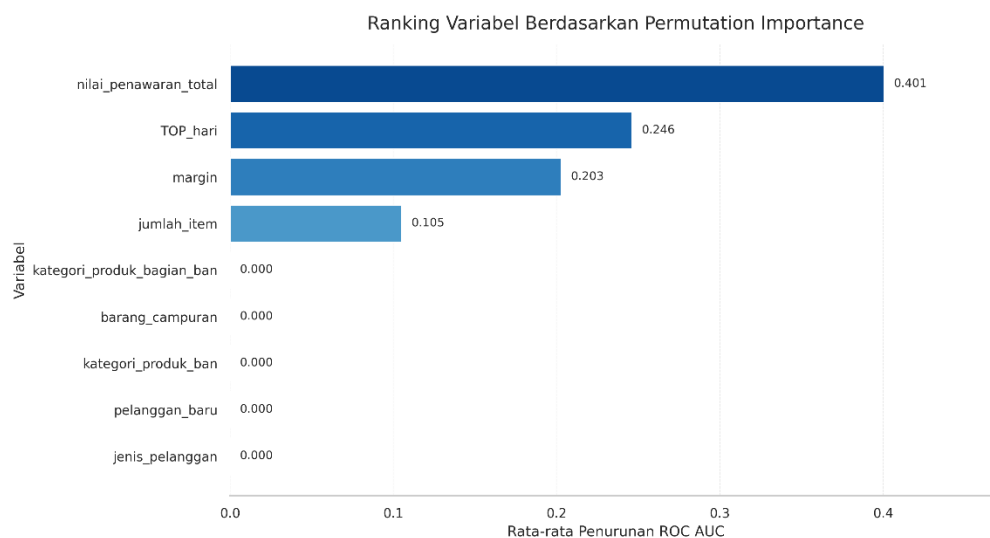
Ranking	Variabel	Rata-rata Penurunan ROC AUC	Kestabilan Fitur
1	nilai_penawaran_total	0.401	0.000
2	TOP_hari	0.246	0.000
3	margin	0.203	0.000
4	jumlah_item	0.105	0.000
5	kategori_produk_bagian_ban	0.000	0.000
6	barang_campuran	0.000	0.000
7	kategori_produk_ban	0.000	0.000
8	pelanggan_baru	0.000	0.000
9	jenis_pelanggan	0.000	0.000

Gambar 5 38 Tabel Ranking Variabel Berdasarkan *Permutation Importance*

Nilai_penawaran_total menempati peringkat pertama dengan rata-rata penurunan ROC-AUC sebesar 0,401, sehingga variabel ini menjadi informasi paling penting dalam membedakan penawaran yang berhasil dan tidak berhasil. TOP_hari berada pada

peringkat kedua dengan nilai 0,246, diikuti margin sebesar 0,203 dan jumlah_item sebesar 0,105. Keempat variabel tersebut menjadi faktor utama yang digunakan model dalam proses klasifikasi. Sementara itu, kategori_produk_bagian_ban, barang_campuran, kategori_produk_ban, pelanggan_baru, dan jenis_pelanggan memiliki nilai 0,000, sehingga pengacakan variabel tersebut tidak menyebabkan penurunan ROC-AUC pada hasil pengujian ini. Kondisi ini menunjukkan bahwa model lebih banyak memanfaatkan informasi nilai transaksi, skema pembayaran, margin, dan jumlah item dibandingkan karakteristik produk maupun pelanggan

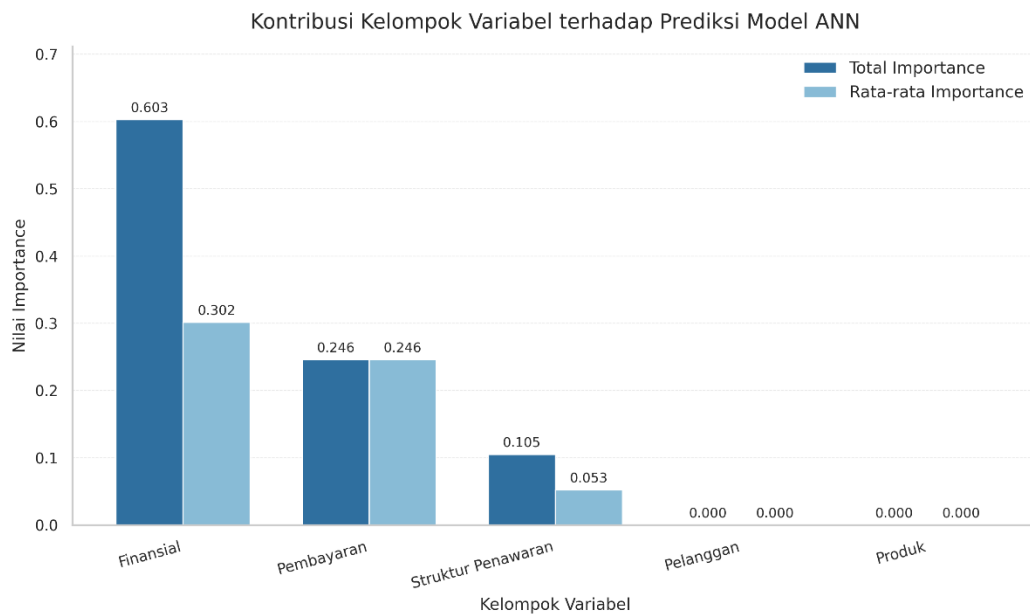
Kolom kestabilan fitur digunakan untuk melihat konsistensi nilai *Importance* ketika proses permutasi dilakukan secara berulang. Nilai kestabilan yang kecil menunjukkan bahwa penurunan ROC-AUC pada setiap pengulangan relatif seragam, sehingga pengaruh variabel dapat dianggap lebih stabil. Pada tabel ini, seluruh nilai kestabilan fitur tercatat sebesar 0,000. Nilai tersebut menunjukkan bahwa hasil pengulangan tidak menghasilkan variasi yang terlihat pada pembulatan tiga angka desimal, sehingga urutan variabel pada tabel dapat dibaca secara lebih konsisten. Dampaknya, variabel dengan nilai rata-rata penurunan ROC-AUC tertinggi, terutama nilai_penawaran_total, TOP_hari, margin, dan jumlah_item, dapat dibahas sebagai variabel dominan karena memiliki kontribusi yang besar dan tidak menunjukkan fluktuasi pada kolom kestabilan fitur.



Gambar 5 39 Grafik Peringkat Variabel Berdasarkan Rata-rata Penurunan ROC-AUC

Grafik peringkat memperjelas hasil pada tabel karena perbedaan panjang bar menunjukkan jarak kontribusi antar variabel secara lebih mudah diamati. Nilai_penawaran_total memiliki jarak paling besar dibandingkan variabel lain, sehingga model sangat bergantung pada informasi nilai transaksi dalam menentukan peluang

keberhasilan penawaran. TOP_hari dan margin juga memberikan kontribusi kuat, meskipun nilainya masih berada di bawah nilai_penawaran_total. Jumlah_item memberikan pengaruh lebih kecil, tetapi tetap berkontribusi terhadap kinerja model. Setelah empat variabel tersebut, seluruh variabel lain berada pada nilai 0,000, sehingga tidak terlihat adanya kontribusi tambahan terhadap penurunan ROC-AUC. Pola ini menjelaskan bahwa model *Artificial Neural Network* lebih banyak menangkap pola dari aspek nominal penawaran, jangka waktu pembayaran, margin, dan jumlah item dibandingkan karakteristik pelanggan atau kategori produk



Gambar 5 40 Grafik Kontribusi Kelompok Variabel terhadap Prediksi Model *Artificial Neural Network*

Visualisasi berdasarkan kategori pada gambar 5.40 menunjukkan bahwa kelompok finansial menjadi kelompok variabel paling dominan, dengan total *Importance* sebesar 0,603 dan rata-rata *Importance* sebesar 0,302. Nilai ini berasal dari gabungan nilai_penawaran_total dan margin, sehingga aspek keuangan menjadi dasar utama yang dipelajari model dalam memprediksi keberhasilan penawaran. Kelompok pembayaran berada pada posisi kedua dengan total dan rata-rata *Importance* sebesar 0,246, yang berasal dari TOP_hari sebagai satu-satunya variabel dalam kelompok tersebut. Kelompok struktur penawaran berada pada posisi berikutnya dengan total *Importance* 0,105 dan rata-rata 0,053, struktur penawaran masih berpengaruh, tetapi tidak sekuat aspek finansial dan pembayaran. Kelompok pelanggan dan produk memiliki nilai 0,000, sehingga belum menjadi pembeda utama dalam hasil prediksi model. Hasil ini menunjukkan bahwa

keberhasilan penawaran pada model lebih ditentukan oleh nilai ekonomi penawaran dan skema pembayaran daripada identitas pelanggan atau jenis produk.

5.4 Implikasi hasil penelitian bagi perusahaan

Berdasarkan hasil pengujian model dan hasil uji variabel kontribusi, penelitian ini memberikan beberapa rekomendasi bagi CV Dima Perkasa Teknik dalam menyusun strategi penawaran. Rangkuman hasil penelitian dan arahan bagi perusahaan disajikan pada Tabel 5.1 sebagai berikut.

Tabel 5.1 Implikasi penelitian bagi perusahaan

No	Hasil Penelitian	Implikasi terhadap perusahaan	Strategi perusahaan
1	Model <i>Artificial Neural baseline</i> menghasilkan performa terbaik dibandingkan model <i>upsampling</i> dan <i>class weighted</i> .	Perusahaan dapat mulai menggunakan data historis penawaran sebagai dasar evaluasi penawaran, bukan hanya sebagai arsip administrasi.	Menyusun daftar prioritas penawaran berdasarkan peluang keberhasilan. Penawaran yang berpeluang tinggi dapat diprioritaskan, sedangkan penawaran berisiko rendah perlu dievaluasi kembali sebelum dikirim.
2	Nilai_penawaran_total menjadi variabel paling dominan dalam hasil uji kontribusi	Perusahaan perlu lebih hati-hati dalam menyusun nilai transaksi, terutama pada penawaran bernilai besar	Mengevaluasi harga total sebelum penawaran dikirim dengan mempertimbangkan kebutuhan pelanggan, daya saing harga, dan risiko kehilangan omzet
3	TOP_hari menjadi salah satu variabel yang berkontribusi besar terhadap keberhasilan penawaran.	Perusahaan perlu memperhatikan jangka waktu pembayaran sebagai bagian dari strategi penawaran.	Menyesuaikan skema pembayaran berdasarkan nilai transaksi, riwayat pelanggan, dan risiko pembayaran. Pelanggan lama dengan riwayat baik dapat dipertimbangkan untuk memperoleh opsi pembayaran yang lebih fleksibel.
4	Margin termasuk variabel penting dalam prediksi keberhasilan penawaran.	Perusahaan perlu menentukan margin secara lebih terukur agar	Menentukan margin berdasarkan nilai penawaran, tingkat

No	Hasil Penelitian	Implikasi terhadap perusahaan	Strategi perusahaan
		terap kompetitif dan tetap menjaga keuntungan.	persaingan, dan peluang penawaran diterima. Margin yang terlalu tinggi dapat menurunkan daya saing, sedangkan margin yang terlalu rendah dapat menekan keuntungan.
5	Jumlah_item ikut berkontribusi terhadap hasil prediksi model.	Jumlah item dapat menjadi dasar evaluasi dalam menyusun struktur penawaran karena banyaknya item berkaitan dengan tingkat kompleksitas penawaran, ketersediaan barang, ketepatan harga, dan kemudahan pelanggan dalam memahami isi penawaran	Mengelompokkan item berdasarkan prioritas kebutuhan pelanggan, ketersediaan barang, dan fungsi produk. Item utama ditempatkan dalam penawaran utama, sedangkan item pelengkap atau substitusi dapat dibuat sebagai opsi terpisah agar penawaran lebih ringkas dan mudah dievaluasi pelanggan.
6	Jenis_pelanggan, kategori_produk, pelanggan_baru, dan barang_campuran belum menjadi pembeda utama dalam hasil model.	Perusahaan jangan menilai peluang penawaran hanya dari status pelanggan lama, pelanggan baru, atau kategori produk.	Tetap mengevaluasi semua penawaran melalui nilai penawaran total, skema pembayaran, margin, dan jumlah item, meskipun pelanggan sudah dikenal atau produk sudah sering ditawarkan.
7	Data dokumen penawaran berhasil diolah menjadi <i>data set</i> terstruktur.	Hasil ini menunjukkan bahwa dokumen penawaran dapat diubah menjadi sumber data evaluasi penjualan yang lebih terukur. Data penawaran tidak hanya berfungsi sebagai arsip, tetapi dapat digunakan untuk membaca pola keberhasilan dan kegagalan penawaran.	Menyusun format pencatatan penawaran yang konsisten agar setiap penawaran berhasil dan gagal dapat dievaluasi pada periode berikutnya.
8	Model dapat diperbarui jika data penawaran bertambah.	Perusahaan dapat melihat hasil model sebagai alat evaluasi yang dapat dikembangkan sesuai perubahan kondisi bisnis.	Melakukan pembaruan data dan evaluasi berkala, misalnya setiap 6

No	Hasil Penelitian	Implikasi terhadap perusahaan	Strategi perusahaan
			bulan atau 1 tahun, agar hasil analisis tetap sesuai dengan kondisi penawaran terbaru.

Berdasarkan Tabel 5.1, hasil penelitian menunjukkan bahwa perusahaan dapat memperbaiki proses evaluasi penawaran melalui penyusunan penawaran yang lebih terstruktur dan pencatatan data yang lebih konsisten. Dengan cara tersebut, setiap penawaran yang berhasil maupun gagal dapat menjadi bahan evaluasi untuk strategi penjualan berikutnya.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan hasil dari Pembangunan model dan pengujian model yang telah dilakukan, berikut adalah kesimpulan dari penelitian ini.

1. Proses pembangunan model prediksi keberhasilan penawaran menggunakan *Artificial Neural Network* dilakukan melalui beberapa tahapan, yaitu pengumpulan data historis penawaran, pembersihan data, transformasi data, pengkodean variabel kategori, validasi data, pemisahan data latih dan data uji, serta pelatihan model menggunakan pendekatan *Multilayer Perceptron*. Model dibangun dalam tiga skenario, yaitu *Artificial Neural Network baseline*, *Artificial Neural Network upsampling*, dan *Artificial Neural Network class weighted*. Ketiga skenario tersebut digunakan untuk membandingkan performa model pada kondisi data yang memiliki ketidakseimbangan kelas, sehingga proses pembangunan model tidak hanya berfokus pada pembentukan prediksi, tetapi juga pada pencarian model yang paling stabil dan sesuai dengan karakteristik data penawaran CV Dima Perkasa Teknik.
2. Hasil pengujian menunjukkan bahwa model *Artificial Neural Network baseline* memberikan performa terbaik dibandingkan *Artificial Neural Network upsampling* dan *Artificial Neural Network class weighted*. Model *Artificial Neural Network baseline* menghasilkan *accuracy* sebesar 0,958 dan ROC-AUC sebesar 0,957, sehingga model mampu mengklasifikasikan sebagian besar data uji dengan benar dan memiliki kemampuan yang kuat dalam membedakan penawaran berhasil dan tidak berhasil. Model *Artificial Neural Network upsampling* menghasilkan *accuracy* sebesar 0,854 dan ROC-AUC sebesar 0,940, sedangkan model *Artificial Neural Network class weighted* menghasilkan *accuracy* sebesar 0,896 dan ROC-AUC sebesar 0,943. Berdasarkan hasil tersebut, model *Artificial Neural Network baseline* dipilih sebagai model utama karena memberikan performa global paling tinggi dan lebih stabil dibandingkan dua pendekatan penanganan *imbalance class* lainnya.
3. Hasil uji variabel kontribusi menggunakan *Permutation Importance* menunjukkan bahwa variabel yang paling berkontribusi terhadap keberhasilan penawaran adalah *nilai_penawaran_total* dengan rata-rata penurunan ROC-AUC sebesar 0,401. Variabel berikutnya adalah *TOP_hari* sebesar 0,246, margin sebesar 0,203, dan

jumlah_item sebesar 0,105. Sementara itu, kategori_produk_bagian_ban, barang_campuran, kategori_produk_ban, pelanggan_baru, dan jenis_pelanggan memiliki nilai *Importance* sebesar 0,000. Hasil ini menunjukkan bahwa prediksi keberhasilan penawaran pada model lebih banyak dipengaruhi oleh nilai ekonomi penawaran, skema pembayaran, margin, dan jumlah item, sedangkan karakteristik pelanggan dan kategori produk belum menjadi pembeda utama dalam hasil prediksi model.

6.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, saran yang dapat diberikan adalah sebagai berikut.

1. Hasil model prediksi dapat digunakan sebagai alat bantu untuk mengevaluasi peluang keberhasilan penawaran sebelum diajukan kepada pelanggan. Hasil utama dari penelitian ini menunjukkan bahwa nilai_penawaran_total, TOP_hari, margin, dan jumlah_item menjadi faktor paling dominan dalam prediksi keberhasilan penawaran. Oleh karena itu, perusahaan perlu lebih teliti dalam menyusun nilai penawaran, menyesuaikan margin agar tetap kompetitif, serta mempertimbangkan skema pembayaran yang sesuai dengan karakteristik pelanggan. Hasil ini juga dapat membantu perusahaan menentukan prioritas penawaran, terutama pada penawaran bernilai besar yang memiliki risiko kehilangan omzet jika gagal dimenangkan.
2. Penelitian selanjutnya dapat memfokuskan pada pendalaman variabel yang terbukti berpengaruh dalam penelitian ini dan perbaikan kualitas data pada kelas minoritas. Hasil penelitian menunjukkan bahwa pendekatan *upsampling* dan *class weighted* belum memberikan performa lebih baik dibandingkan model *baseline*.
3. Model prediksi yang dikembangkan dalam penelitian ini dapat digunakan sebagai acuan awal bagi perusahaan dan penelitian lain yang memiliki kebutuhan analisis prediktif berbasis data historis. Namun, model perlu dilatih kembali menggunakan data dari perusahaan terkait karena setiap perusahaan memiliki karakteristik variabel, pola transaksi, jenis pelanggan, dan kondisi bisnis yang berbeda. Pelatihan ulang diperlukan agar model dapat mempelajari pola data yang sesuai dengan kebutuhan dan konteks operasional masing-masing perusahaan.

DAFTAR PUSTAKA

- Abdulsalam, S. O., Arowolo, M. O., Saheed, Y. K., & Afolayan, J. O. (2022). Customer Churn Prediction in Telecommunication Industry Using Classification and Regression Trees and Artificial Neural Network Algorithms. *Indonesian Journal of Electrical Engineering and Informatics (IJEI)*, 10(2), 431–440. <https://doi.org/10.52549/ijeei.v10i2.2985>
- Adityaningrum, A., Payu, M. R. F., & Silfana, B. (2025). *Factors Associated with Waste Management Behavior in Coastal Communities: Evidence from Binary Logistic Regression*.
- Ahmed, R. S., Hasnain, M., Mahmood, M. H., & Mehmood, M. A. (2024). *Comparison of Deep Learning Algorithms for Retail Sales Forecasting*.
- Arti, S., & Suherlan, E. (2025). Evaluasi Kinerja Machine Learning dalam Memprediksi Kemampuan Adaptasi Mahasiswa pada Lingkungan Pembelajaran Daring. *Jurnal Pustaka AI (Pusat Akses Kajian Teknologi Artificial Intelligence)*, 5(1), 50–57. <https://doi.org/10.55382/jurnalpustakaai.v5i1.901>
- Atmaja, K. J., & Pascima, I. B. N. (2022). *Implementation of Artificial Neural Network on Sales Forecasting Application*. 5(4).
- Azhima, M., Afrianty, I., Budianita, E., & Gusti, S. K. (2024). *Penerapan Metode Backpropagation Neural Network untuk Klasifikasi Penyakit Stroke*.
- Bakirrarar, B., & Elhan, A. H. (2023). Class Weighting Technique to Deal with Imbalanced Class Problem in Machine Learning: Methodological Research. *Turkiye Klinikleri Journal of Biostatistics*, 15(1), 19–29. <https://doi.org/10.5336/biostatic.2022-93961>
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Fahrudin, N. F., Putra, K. R., Umaroh, S., & Lautan, G. B. (2024). Influence of Data Scaling and Train/Test Split Ratios on LightGBM Efficacy for Obesity Rate Prediction. *MIND Journal*, 9(2), 220–234. <https://doi.org/10.26760/mindjournal.v9i2.220-234>
- Haekal, J., Mu'min, R., Adriansyah, A., Paduloh, Sjarifudin, D., Nuryono, A., Rosihan, R. I., Tambunan, E. B. M., & Yaakop, S. N. K. (2025). AI Ladder-Based Intelligent Enterprise Resource Planning for Predictive Railway Track Maintenance from UAV Imagery. *Mathematical Modelling of Engineering Problems*, 12(10), 3577–3592. <https://doi.org/10.18280/mmep.121023>
- Kaope, C., & Pristyanto, Y. (2023). The Effect of Class Imbalance Handling on Datasets Toward Classification Algorithm Performance. *MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, 22(2), 227–238. <https://doi.org/10.30812/matrik.v22i2.2515>
- Kementerian Energi dan Sumber Daya Mineral Republik Indonesi. (2025). *Laporan Kinerja Ditjen Mineral dan Batubara Tahun 2025*. Kementerian Energi dan Sumber Daya Mineral Republik Indonesia.
- Lestari, N., Indahwati, I., Erfiani, E., & Julianti, E. D. (2023). A COMPARISON OF ARTIFICIAL NEURAL NETWORK AND NAIVE BAYES CLASSIFICATION USING UNBALANCED DATA HANDLING. *BAREKENG: Jurnal Ilmu Matematika Dan Terapan*, 17(3), 1585–1594. <https://doi.org/10.30598/barekengvol17iss3pp1585-1594>
- Martha, R., & Herwindiati, D. E. (2024). Prediksi Hujan Menggunakan Metode Artificial Neural Network, K-Nearest Neighbors, dan Naïve Bayes. *Jurnal Teknologi Dan*

- Sistem Informasi Bisnis*, 6(4), 859–865.
<https://doi.org/10.47233/jteksis.v6i4.1650>
- Mega, N. M., & Olivia Brilliant Chang. (2026). PENERAPAN ALGORITMA ARTIFICIAL NEURAL NETWORK UNTUK KLASIFIKASI KUALITAS BUAH APEL. *HOAQ (High Education of Organization Archive Quality) : Jurnal Teknologi Informasi*, 17(1).
- Molnar, C. (2022). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (2 ed.). <https://christophm.github.io/interpretable-ml-book/>
- Musnaini. (2024). *Statistika Ekonomi: Metode Probabilitas*. Universitas Jambi. <https://repository.unja.ac.id/70149/1/2024-C1a1a.%20Buku%20refrensi%20Statistika%20Ekonomi%20Metode%20Probabilitas.pdf>
- Padmapuspita J, L., Nurcahyo, R., & Sihono Gabriel, D. (2021a, Maret 7). Identification of Success Factor for B2B E-Commerce in Indonesia. *Proceedings of the International Conference on Industrial Engineering and Operations Management*. 11th Annual International Conference on Industrial Engineering and Operations Management. <https://doi.org/10.46254/AN11.20210111>
- Panji Bintoro, R., Edy Wihardjo, I. P. P., & Andi Asari. (2024). *Pengantar Machine Learning*. PT Mafy Media Literasi Indonesia.
- Panji Kurniawan, P. W. S. (2025). *Building of Informatics, Technology and Science (BITS)*. 7(1).
- Rahmawati, P., Larasati, A., & Marsono, M. (2022). PENGEMBANGAN MODEL PERSETUJUAN KREDIT NASABAH BANK DENGAN ALGORITMA KLASIFIKASI NAÏVE BAYES, DECISION TREE, DAN ARTIFICIAL NEURAL NETWORK. *J@ti Undip: Jurnal Teknik Industri*, 17(1), 1–12. <https://doi.org/10.14710/jati.1.1.1-12>
- Rezazadeh, A. (2020). A Generalized Flow for B2B Sales Predictive Modeling: An Azure Machine-Learning Approach. *Forecasting*, 2(3), 267–283. <https://doi.org/10.3390/forecast2030015>
- Riyandanu, M. F. (2023, Juli 21). *ESDM Soroti Kelangkaan Ban Alat Berat Ancam Produksi Batu Bara*. Katadata.co.id. <https://katadata.co.id/berita/energi/64ba6a7b5802d/esdm-soroti-kelangkaan-ban-alat-berat-ancam-produksi-batu-bara>
- Segara, A. P., Saputra, M. A. W., & Ranggianto, N. A. (2025). *Perbandingan Performa Algoritma Random Tree, K-NN, dan A-NN untuk Deteksi Serangan DDoS pada Software Defined Network (SDN)*. 08(2).
- Soegianto, L. M., Hinandra, A. T., Suri, P. A., & Fajar, M. (2024). Comparison of Model Performance on Housing Business Using Linear Regression, Random Forest Regressor, SVR, and Neural Network. *Procedia Computer Science*, 245, 1139–1145. <https://doi.org/10.1016/j.procs.2024.10.343>
- Sri Kusumadewi, H. P. (2010). *Aplikasi Logika Fuzzy untuk Pendukung Keputusan* (Ed. 2). Graha Ilmu.
- Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K.-R. (2021). Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Machine Learning and Knowledge Extraction*, 3(2), 392–413. <https://doi.org/10.3390/make3020020>
- Thölke, P., Mantilla-Ramos, Y.-J., Abdelhedi, H., Maschke, C., Dehgan, A., Harel, Y., Kemtur, A., Mekki Berrada, L., Sahraoui, M., Young, T., Bellemare Pépin, A., El Khantour, C., Landry, M., Pascarella, A., Hadid, V., Combrisson, E., O’Byrne, J.,

- & Jerbi, K. (2023). Class imbalance should not throw you off balance: Choosing the right classifiers and performance metrics for brain decoding with imbalanced data. *NeuroImage*, 277, 120253. <https://doi.org/10.1016/j.neuroimage.2023.120253>
- Tim Pembina IOAI Indonesia. (2025). *Modul 2: Pengenalan Machine Learning*. https://ioai.toki.id/assets/modul/Modul_2_IOAI__Pengenalan_Machine_Learning__ML_.pdf?
- Vikkurty, S., Vani, M. S., Sravani, D., M., S. S., & Durga, K. (2026). Toxicity Prediction of Landfill Leachate-Contaminated Crops Using Machine Learning Models Based on PAH and Heavy Metal Concentrations. *Nature Environment and Pollution Technology*, 25(2), B4376. <https://doi.org/10.46488/NEPT.2026.v25i02.B4376>
- Weiss, G. M. (2013). Foundations of Imbalanced Learning. Dalam H. He & Y. Ma (Ed.), *Imbalanced Learning* (1 ed., hlm. 13–41). Wiley. <https://doi.org/10.1002/9781118646106.ch2>
- Zahid, E., Ong, Y. J., Megahed, A., & Nakamura, T. (2021). *Predicting Loss Risks for B2B Tendering Processes* (arXiv:2109.06815). arXiv. <https://doi.org/10.48550/arXiv.2109.06815>

LAMPIRAN

CV. DIMA PERKASA TEKNIK

GENERAL TECHNICAL SUPPLIER

JL. A.Yani Km: 7 Komplek Griya pemurus indah Blok: F No: 25

Fax: 0511-4282040 Email; nia2104@yahoo.com / 2104nia@gmail.com

Banjarmasin, 05 April 2025
 No:019/DPT/IV/2025
 Hal: Revisi Penawaran

Kepada Yth,
 PT.Widya Sapta Cotractors
 Up. Bapak Moses
 Di. Tempat

Dengan Hormat

Bersama ini kami dari CV.DIMA PERKASA TEKNIK menawarkan ban Kepada PT. Widya Sapta Contractors (Site Adaro) , adapun harga yang kami tawarkan adalah sebagai berikut:

1. Ban Uk 245/75 R16 Savero MT GT	Rp. 1.775.000,-/Pcs
2. Ban Uk 245/75 R16 MT Bridgestone	Rp. 2.100.000,-/Pcs
3. Ban Uk 265/65 R17 Dualer MT 674 Bridgestone	Rp. 2.740.000.-/Pcs
4. Ban Uk 265/70 R17 Savero Komodo MT Plus GT	Rp. 2.300.000.-/Pcs
5. Ban Luar Uk 750/16 Bias SU88N Cacing GT	Rp 2.150.000./Set
6. Ban Uk 27x8.50 R14 Savero Komodo M/T Plus GT	Rp. 1.275.000./Pcs
7. Oring R25 8 mm	Rp. 400.000./Pcs

Penawaran harga dari kami, Atas perhatian dan kerjasamanya kami ucapkan

Terima kasih

- Harga belum termasuk ppn 11%
- Pembayaran 2 Minggu-3 Minggu
- Franco tanjung
- Penawaran berlaku 1 minggu
- Barang Ready stock

Hormat Kami

KURNIAWATI

CV. DIMA PERKASA TEKNIK

GENERAL TECHNICAL SUPPLIER

JL. A.Yani Km: 7 Komplek Griya pemurus indah Blok: F No: 25
Tel/ Fax: 0511-4282040 Email; nia2104@yahoo.com / 2104nia@gmail.com

Banjarmasin, 5 Januari 2025
No:019/DPT/III /2025
Hal: Penawaran

Kepada Yth,
PT.Widya Sapta Cotractors
Up. Bapak Moses
Di. Tempat

Dengan Hormat

Bersama ini kami dari CV.DIMA PERKASA TEKNIK menawarkan ban Kepada PT. Widya Sapta Contractors (Site Adaro) , adapun harga yang kami tawarkan adalah sebagai berikut:

1. Ban Uk 17.5/R25 **Triangle Radial + Oring** Rp. 23.950.000,-/Set



2. Ban Uk 17.5/R25 **Techking Radial+ Oring** Rp. 26.900.000,-/Set



3. Ban Uk 17.5/25-16Ply **VL2 Bridgestone Nylon** Rp. 28.850.000,-/Pcs

Tipe Tube Assy

