

**STUDI KOMPARATIF METODE XGBOOST DAN *RADIAL
BASIS FUNCTION NETWORK* DALAM PERANKINGAN
KONTINGENSI N-1 SISTEM TENAGA LISTRIK UNTUK
PENILAIAN KEAMANAN STATIS**

SKRIPSI

**Disusun Sebagai Persyaratan Memperoleh Gelar Sarjana Teknik
Pada Program Studi S1 Teknik Elektro**



**LISYA LIYA ANGGRAINI
NPM G1D022021**

**PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
2026**

LEMBAR PENGESAHAN SKRIPSI

STUDI KOMPARATIF METODE XGBOOST DAN RADIAL BASIS FUNCTION NETWORK DALAM PERANKINGAN KONTINGENSI N-1 SISTEM TENAGA LISTRIK UNTUK PENILAIAN KEAMANAN STATIS

Oleh:

LISYA LIYA ANGGRAINI

NPM G1D022021

Skrripsi ini telah diuji dan dipertahankan di hadapan Tim Penguji Skripsi
pada tanggal 7 Agustus 2026

1. Dr. Irmanda Priyadi ST. MT

Pembimbing Utama

NIP. 197604102003121001

2. Ir. Afriyastuti Herawati, S.T., M.T., MCE.

Pembimbing Pendamping

NIP. 197911132003121002

3. Ir. Novalio Daratha, S.T., M.Sc., Ph.D.

Ketua Penguji

NIP. 197911132003121002

4. Dr. Natsir Habibullah, S.T., M.T.

Anggota Penguji

NIP. 199312012024061002

Mengesahkan,
Dekan Fakultas Teknik

Dr. Eng. Afidhal Kurniawan Maimil, S.T., M.T.

NIP. 198209262008011007

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan di bawah ini menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

**STUDI KOMPARATIF METODE XGBOOST DAN *RADIAL BASIS FUNCTION NETWORK* DALAM PERANKINGAN KONTINGENSI
N-1 SISTEM TENAGA LISTRIK UNTUK PENILAIAN
KEAMANAN STATIS**

Sepenuhnya merupakan karya saya sendiri dan sepanjang pengetahuan saya bukan merupakan hasil duplikasi dari skripsi dan/atau karya ilmiah lainnya yang pernah dipublikasikan atau pernah dipergunakan untuk mendapatkan gelar kesarjanaan di Perguruan Tinggi atau instansi manapun, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai etika keilmuan, kecuali bagian yang bersumber informasinya dicantumkan sebagaimana mestinya.

Apabila ternyata di dalam skripsi ini terbukti terdapat unsur-unsur plagiasi, saya bersedia menerima sanksi dengan aturan yang berlaku.

Bengkulu, 15 Agustus 2026

Penulis



Lisya Liya Anggraini
G1D022021

MOTO DAN PERSEMBAHAN

MOTTO:

- ❖ “Apa yang ditakdirkan untukmu tidak akan pernah melewatimu, dan apa yang melewatimu tidak akan pernah ditakdirkan untukmu.”
- ❖ “Sesungguhnya bersama kesulitan ada kemudahan.” (QS. Al-Insyirah: 6)

PERSEMBAHAN:

- ❖ Segala puji dan syukur kepada Allah SWT atas segala rahmat, karunia, kesehatan, kekuatan, dan kemudahan yang telah diberikan sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Setiap proses, tantangan, dan pencapaian dalam perjalanan ini tidak terlepas dari pertolongan-Nya.
- ❖ Dengan segala kerendahan hati, karya ini penulis persembahkan kepada Kedua orang tua tercinta, yang selalu menjadi sumber kekuatan dan inspirasi terbesar dalam hidup penulis. Terima kasih atas doa yang tidak pernah putus, kasih sayang yang tulus, pengorbanan, perhatian, serta dukungan yang selalu diberikan dalam setiap langkah perjalanan pendidikan penulis. Skripsi ini penulis persembahkan sebagai bentuk rasa syukur, hormat, dan terima kasih atas segala perjuangan yang telah diberikan.
- ❖ Bapak Dr. Irnanda Priyadi, S.T., M.T., selaku Dosen Pembimbing Utama, dan Ibu Ir. Afriyastuti Herawati, S.T., M.T., MCE., selaku Dosen Pembimbing Pendamping, yang telah meluangkan waktu, tenaga, dan pikiran untuk memberikan bimbingan, arahan, motivasi, kritik, serta saran yang sangat berharga selama proses penelitian dan penyusunan skripsi ini.
- ❖ Bapak Ir. Novalio Daratha, S.T., M.Sc., Ph.D. dan Bapak Dr. Natsir Habibullah, S.T., M.T., selaku dosen penguji yang telah memberikan berbagai masukan, kritik, dan saran yang membangun sehingga skripsi ini dapat menjadi lebih baik.
- ❖ Seluruh dosen dan staf Program Studi Teknik Elektro, Fakultas Teknik, Universitas Bengkulu, yang telah memberikan ilmu pengetahuan, bimbingan, pelayanan akademik serta dukungan dan semangat selama penulis menempuh pendidikan.

- ❖ Teman-teman Teknik Elektro Angkatan 2022, yang telah menjadi rekan seperjuangan dalam menuntut ilmu, berbagi pengalaman, berdiskusi, saling membantu, dan memberikan semangat selama masa perkuliahan hingga penyelesaian skripsi ini.
- ❖ Keluarga Besar Laboratorium Elektronika Dan Telekomunikasi (LAB ELKOM) Agustina, Ridha, Chandra, Arif, Bonafif, Fito dan Azzam yang telah menjadi tempat belajar, berdiskusi, bertukar pengalaman, serta mengembangkan kemampuan akademik maupun nonakademik selama masa perkuliahan. Terima kasih atas kebersamaan, dukungan, dan lingkungan yang positif sehingga memberikan banyak pengalaman berharga bagi penulis dalam menyelesaikan studi dan skripsi ini.
- ❖ Seseorang yang telah menemani dan menjadi bagian dari perjalanan penulis selama masa perkuliahan hingga penyelesaian skripsi ini, Akbar Palito Risman. Terima kasih atas doa, dukungan, perhatian, semangat, serta kebersamaan yang telah diberikan dalam berbagai proses dan tantangan yang penulis hadapi. Kehadiranmu telah menjadi bagian dari cerita yang mewarnai perjalanan akademik ini. Semoga segala kebaikan yang pernah diberikan menjadi amal baik dan kenangan yang berharga sepanjang kehidupan.

KATA PENGANTAR

Alhamdulillahirabbil'alamin, puji dan syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat, karunia, dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul “Studi Komparatif Metode XGBoost dan *Radial Basis Function Network* dalam Perankingan Kontingensi N-1 Sistem Tenaga Listrik untuk Penilaian Keamanan Statis”. Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Bengkulu. Selama proses penelitian, pengolahan data, pengembangan model, hingga penyusunan laporan skripsi ini, penulis memperoleh banyak bantuan, bimbingan, motivasi, serta dukungan dari berbagai pihak. Sehubungan dengan itu, pada kesempatan ini penulis menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT yang telah memberikan nikmat kesehatan, kekuatan, kemudahan, serta kelancaran sehingga penulis dapat menyelesaikan skripsi ini.
2. Kedua orang tua tercinta serta seluruh keluarga, yang senantiasa memberikan doa, kasih sayang, dukungan moral maupun material, motivasi, serta kepercayaan penuh kepada penulis selama menempuh pendidikan hingga penyelesaian skripsi ini.
3. Bapak Dr. Indra Cahyadinata, S.P., M.Si., selaku Rektor Universitas Bengkulu.
4. Bapak Dr. Eng. Afdhal Kurniawan Mainil, S.T., M.T., selaku Dekan Fakultas Teknik Universitas Bengkulu.
5. Ibu Ir. Afriyastuti Herawati, S.T., M.T., MCE., selaku Ketua Program Studi Teknik Elektro Universitas Bengkulu.
6. Bapak Dr. Irnanda Priyadi, S.T., M.T. selaku Dosen Pembimbing Utama dan Ibu Ir. Afriyastuti Herawati, S.T., M.T., MCE. selaku Dosen Pembimbing Pendamping yang telah meluangkan waktu, tenaga, dan pikiran untuk memberikan bimbingan, arahan, motivasi, kritik, saran,

serta ilmu pengetahuan yang sangat berharga sejak awal penelitian hingga selesainya penyusunan skripsi ini.

7. Bapak Ir. Novalio Daratha, S.T., M.Sc., Ph.D. selaku Ketua Penguji dan Bapak Dr. Natsir Habibullah, S.T., M.T. selaku Anggota Penguji yang telah memberikan kritik, saran, dan masukan yang sangat bermanfaat demi penyempurnaan skripsi ini.
8. Seluruh dosen Program Studi Teknik Elektro Universitas Bengkulu, yang telah memberikan ilmu pengetahuan, pengalaman, serta wawasan selama masa perkuliahan.
9. Seluruh staf administrasi Program Studi Teknik Elektro dan Fakultas Teknik Universitas Bengkulu, yang telah membantu kelancaran administrasi selama proses perkuliahan hingga penyelesaian skripsi.
10. Seluruh rekan mahasiswa Teknik Elektro Angkatan 2022, yang telah menjadi teman belajar, berdiskusi, saling memberikan semangat, serta menciptakan pengalaman berharga selama menjalani masa perkuliahan.
11. Keluarga besar Laboratorium Elektro (LAB ELKOM) yang telah menjadi tempat belajar, berdiskusi, berbagi pengalaman, serta memberikan banyak pembelajaran dan kenangan berharga selama masa perkuliahan.
12. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu, yang telah memberikan bantuan, doa, dukungan, dan kontribusi sehingga skripsi ini dapat terselesaikan dengan baik.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun demi penyempurnaan penelitian dan penulisan karya ilmiah di masa yang akan datang.

Bengkulu, 15 Agustus 2026

Penulis,

Lisya Liya Anggraini

G1D022021

ABSTRAK

Keamanan sistem tenaga listrik merupakan aspek penting dalam menjamin kontinuitas penyaluran energi, terutama saat terjadi gangguan pada komponen sistem. Salah satu metode penilaian keamanan adalah analisis kontingensi N-1 menggunakan *Newton Raphson Load Flow* (NRLF) untuk menghitung *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV). Meskipun akurat, metode ini memerlukan simulasi pada setiap skenario kontingensi sehingga kurang efisien untuk penilaian berulang. Penelitian ini membandingkan kinerja *Radial Basis Function Network* (RBFN) dan *Extreme Gradient Boosting* (XGBoost) dalam memprediksi PIP dan PIV serta melakukan perankingan kontingensi N-1 pada sistem IEEE 30-Bus. Dataset terdiri atas 222 sampel dari 37 skenario kontingensi pada enam tingkat pembebanan (50%-100%) dengan nilai PIP dan PIV hasil NRLF sebagai *ground truth*. Evaluasi dilakukan menggunakan R^2 , RMSE, MAE, MAPE, dan *Capture Ratio*. Pada pembebanan 100%, XGBoost menunjukkan performa terbaik dengan R^2 sebesar 0,9985 (PIP) dan 0,9980 (PIV), MAPE 0,40% dan 0,35%, serta *Capture Ratio* 100% pada Top-5 dan Top-10 untuk PIP dan Top-10 untuk PIV. Sebaliknya, pada pengujian generalisasi menggunakan 37 skenario pembebanan 85% yang tidak dilibatkan dalam pelatihan, *Optimized Ensemble* RBFN lebih unggul pada prediksi PIP (R^2 0,8978 dibandingkan 0,8314), sedangkan XGBoost tetap memiliki waktu prediksi lebih cepat. Hasil penelitian menunjukkan bahwa XGBoost lebih sesuai untuk kondisi operasi yang representatif terhadap data pelatihan, sedangkan *Optimized Ensemble* RBFN memiliki kemampuan generalisasi yang lebih baik pada kondisi pembebanan baru.

Kata kunci : Keamanan Statis Sistem Tenaga Listrik, Kontingensi N-1, *Active Power Performance Index* (PIP), *Voltage Performance Index* (PIV), *Radial Basis Function Network* (RBFN), *Extreme Gradient Boosting* (Xgboost).

DAFTAR ISI

LEMBAR PENGESAHAN SKRIPSI	i
PERNYATAAN KEASLIAN SKRIPSI.....	iii
MOTO DAN PERSEMBAHAN	v
KATA PENGANTAR.....	vii
ABSTRAK	ix
DAFTAR GAMBAR	xv
DAFTAR TABEL	xvii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan.....	4
1.4 Batasan Masalah.....	4
BAB II TINJAUAN PUSTAKA	5
2.1 Penelitian Terdahulu.....	5
2.2 Sistem Tenaga Listrik.....	7
2.2.1 Keamanan Sistem Tenaga	7
2.2.2 Kontingensi Sistem Tenaga	9
2.2.2.1 Hubungan antara Gangguan (Fault) dan Kontingensi.....	10
2.2.2.2 Definisi Kontingensi N-1	10
2.2.2.3 Jenis-Jenis Gangguan pada Sistem Tenaga Listrik	11
2.2.2.4 Mekanisme Pemutusan Saluran dan Representasi pada Studi Keamanan Statis.....	12
2.2.3 Analisis Kontingensi	13
2.2.4 Contingency Selection dan Ranking	14
2.2.4.1 Power Performance Index (PIP)	15
2.2.4.2 <i>Voltage Performance Index</i> (PIV).....	16
2.2.5 Metrik Kesesuaian Perankingan Kontingensi (Capture Ratio)	17
2.3 Static Security Assessment (SSA).....	18
2.4 XGBoost (<i>Extreme Gradient Boosting</i>)	18

2.4.1 Prinsip Dasar dan Mekanisme Pembelajaran	19
2.4.2 Perumusan Fungsi Optimasi	20
2.4.3 Struktur Model Berbasis Pohon Keputusan.....	22
2.4.4 Tuning Hyperparameter	22
2.5 <i>Radial Basis Function Network</i>	24
2.6 Hipotesis Penelitian	28
BAB III METODE PENELITIAN.....	29
3.1 Metode Pengumpulan Data	29
3.1.1 Studi Literatur	29
3.1.2 Simulasi Numerik	29
3.2 Waktu dan Tempat Penelitian	30
3.3 Alat, Perangkat Lunak, dan Data Penelitian	30
3.3.1 Alat	30
3.3.2 Perangkat Lunak	30
3.3.3 Data Penelitian.....	31
3.4 Alur Penelitian	32
3.5 Simulasi Aliran Daya Metode <i>Newton Raphson</i>	34
3.5.1 Prosedur Simulasi Kontingensi N-1 pada Sistem IEEE 30-Bus	36
3.6 Analisis Keamanan Statis Sistem Tenaga Listrik	37
3.7 Pembentukan dan Pembagian Dataset	38
3.8 Perancangan dan Pelatihan Model RBFN	39
3.9 Perancangan dan Pelatihan Model XGBoost.....	40
3.10 Metode Evaluasi Kinerja Model.....	42
3.11 Data Pendukung.....	45
3.11.1 Klasifikasi dan Parameter Bus.....	46
3.11.2 Data Saluran Transmisi (<i>Branch Data</i>)	47
3.11.3 Data Validasi Newthon Raphson Loadflow	49
BAB IV HASIL DAN PEMBAHASAN	51
4.1 Hasil Simulasi Aliran Daya <i>Newton Raphson Load Flow</i> (NRLF).....	51
4.2 Hasil Simulasi Kontingensi N-1	57
4.3 Hasil Perhitungan Indeks Keamanan Statis (PIP dan PIV)	60
4.4 Hasil Perankingan Kontingensi N-1 Berdasarkan NRLF	62

4.5 Pembentukan dan Pembagian Dataset	64
4.5.1 Normalisasi Data	66
4.6 Hasil Implementasi Model <i>Radial Basis Function Network</i> (RBFN).....	68
4.6.1 Implementasi RBFN dengan Parameter [7]	68
4.6.2 Analisis Implementasi [7] pada Dataset Penelitian	71
4.6.3 Arsitektur Implementasi Model RBFN	74
4.6.4 Hasil Evaluasi Model RBFN.....	74
4.6.5 Perbandingan Implementasi RBFN Referensi dan <i>Optimized Ensemble</i> RBFN	78
4.7 Implementasi Model <i>Extreme Gradient Boosting</i> (XGBoost).....	79
4.7.1 Konfigurasi Parameter Model XGBoost	80
4.7.2 Hasil Evaluasi Model XGBoost	81
4.8 Perbandingan Kinerja Model <i>Extreme Gradient Boosting</i> (XGBoost) dan <i>Optimized Ensemble Radial Basis Function Network</i> (RBFN)	86
4.8.1 Perbandingan Akurasi Prediksi PIP dan PIV	86
4.8.2 Kesesuaian Perankingan Kontingensi N-1 terhadap NRLF Menggunakan Capture Ratio	89
4.8.3 Perbandingan Kemampuan Generalisasi dan Waktu Komputasi	93
BAB V KESIMPULAN DAN SARAN	99
5.1 Kesimpulan	99
5.2 Saran.....	100
DAFTAR PUSTAKA.....	101
LAMPIRAN.....	105

DAFTAR GAMBAR

Gambar 2. 1 Diagram struktur dan proses pembentukan model XGBoost.....	20
Gambar 2. 2 Arsitektur <i>Radial Basis Function Network</i>	26
Gambar 2. 1 Diagram struktur dan proses pembentukan model XGBoost.....	20
Gambar 2. 2 Arsitektur <i>Radial Basis Function Network</i>	26
Gambar 3. 1 Diagram Alir Penelitian.....	32
Gambar 3. 2 Flowchart Perancangan RBFN.....	42
Gambar 3. 3 Flowchart Perancangan Model XGBOOST	43
Gambar 3. 4 Single Line Diagram IEEE 30 Bus	46
Gambar 4. 1 Profil Tegangan Setiap Bus pada Kondisi <i>Base case</i>	55
Gambar 4. 2 Perbandingan Profil Tegangan Sebelum dan Sesudah Kontingensi. 60	
Gambar 4. 3 Implementasi Normalisasi Data Menggunakan <i>RobustScaler</i> dan <i>StandardScaler</i>	66
Gambar 4. 4 Perbandingan Nilai Koefisien Determinasi (R^2) Model XGBoost dan <i>Optimized Ensemble</i> RBFN	87
Gambar 4. 5 Perbandingan Nilai <i>Root Mean Square Error</i> (RMSE) Model XGBoost dan <i>Optimized Ensemble</i> RBFN	87
Gambar 4. 6 Perbandingan Nilai <i>Mean Absolute Error</i> (MAE) Model XGBoost dan <i>Optimized Ensemble</i> RBFN.....	88
Gambar 4. 7 Perbandingan Nilai <i>Mean Absolute Percentage Error</i> (MAPE) Model XGBoost dan <i>Optimized Ensemble</i> RBFN	88
Gambar 4. 8 Perbandingan Hasil Prediksi PIP terhadap NRLF Menggunakan <i>Optimized Ensemble</i> RBFN dan XGBoost	89
Gambar 4. 9 Perbandingan Hasil Prediksi PIV terhadap NRLF Menggunakan <i>Optimized Ensemble</i> RBFN dan XGBoost	89

DAFTAR TABEL

Tabel 2. 1 Penelitian Terdahulu yang Berkaitan	5
Tabel 3. 1 Alat	30
Tabel 3. 2 Perangkat Lunak Penelitian.....	30
Tabel 3. 3 Data Penelitian	31
Tabel 3. 4 Data Bus IEEE 30 Bus	47
Tabel 3. 5 Data Branch IEEE 30 Bus	48
Tabel 4. 2 Hasil Simulasi NRLF pada Kondisi <i>Base case</i>	52
Tabel 4. 3 Profil Tegangan Setiap Bus pada Kondisi <i>Base case</i>	53
Tabel 4. 4 Aliran Daya pada Saluran Transmisi (<i>Base case</i>)	55
Tabel 4. 5 Hasil Simulasi Kontingensi N-1	57
Tabel 4. 6 Perubahan Aliran Daya Saluran Pasca Kontingensi.....	58
Tabel 4. 7 Statistik Nilai PIP pada Setiap Tingkat Pembebanan	61
Tabel 4. 8 Statistik Nilai PIV pada Setiap Tingkat Pembebanan	61
Tabel 4. 9 Perankingan 37 Kontingensi N-1 Berdasarkan PIP dan PIV	62
Tabel 4. 10 Ringkasan Dataset Penelitian	65
Tabel 4. 11 Pembagian Dataset	65
Tabel 4. 12 Perbandingan Implementasi Normalisasi dan Pembagian Data pada Model RBFN dan XGBoost	67
Tabel 4. 13 Parameter Implementasi RBFN [7].....	69
Tabel 4. 14 Perkembangan Nilai Average MSE Selama Pelatihan Model RBFN [7].....	69
Tabel 4. 15 Metrik Evaluasi Model RBFN [7].....	70
Tabel 4. 16 Parameter Implementasi Model RBFN	74
Tabel 4. 17 Tabel Metrik Evaluasi Simulasi Ensemble RBFN untuk PIP	75
Tabel 4. 18 Tabel Metrik Evaluasi Simulasi Ensemble RBFN untuk PIV	76
Tabel 4. 19 Metrik Evaluasi Model Ensemble RBFN.....	78
Tabel 4. 20 Perbandingan Implementasi RBFN Referensi dan <i>Optimized Ensemble</i> RBFN	79
Tabel 4. 21 Parameter Model XGBoost untuk Prediksi PIP	80
Tabel 4. 22 Parameter Model XGBoost untuk Prediksi PIV	80
Tabel 4. 23 Tabel Metrik Evaluasi Simulasi XGBoost untuk PIP	81

Tabel 4. 24 Tabel Metrik Evaluasi Simulasi XGBoost untuk PIV	83
Tabel 4. 25 Metrik Evaluasi Model XGBoost	85
Tabel 4. 26 Perbandingan Metrik Evaluasi XGBoost dan <i>Optimized Ensemble</i> RBFN.....	86
Tabel 4. 27 Perbandingan <i>Capture Ratio</i> Perankingan Kontingensi terhadap NRLF	90
Tabel 4. 28 Perbandingan Sepuluh Kontingensi Teratas Berdasarkan Nilai PIP ..	91
Tabel 4. 29 Perbandingan Sepuluh Kontingensi Teratas Berdasarkan Nilai PIV ..	92
Tabel 4. 30 Perbandingan Metrik Evaluasi Model pada 37 Kontingensi Level Beban 85%.....	94
Tabel 4. 31 <i>Capture Ratio</i> Model pada 37 Kontingensi Level Beban 85%	95
Tabel 4. 32 Perbandingan Waktu Komputasi Model <i>Optimized Ensemble</i> RBFN dan XGBoost	96

BAB I

PENDAHULUAN

1.1 Latar Belakang

Keamanan dan keandalan operasi sistem tenaga listrik merupakan aspek fundamental dalam menjamin kontinuitas penyaluran energi listrik kepada konsumen. Sistem tenaga listrik dituntut untuk tetap beroperasi dalam batasan teknis yang diizinkan, baik pada kondisi normal maupun saat terjadi gangguan. Oleh karena itu, penilaian keamanan sistem tenaga (*power system security assessment*) menjadi proses penting untuk menentukan kemampuan sistem dalam mempertahankan kestabilan operasi serta mengidentifikasi tingkat keparahan gangguan terhadap batas operasi daya dan tegangan sistem [1].

Salah satu pendekatan yang umum digunakan dalam penilaian keamanan sistem tenaga listrik adalah analisis kontingensi N-1, yaitu evaluasi kondisi sistem ketika satu elemen jaringan, seperti saluran transmisi atau peralatan lainnya, mengalami gangguan atau keluar dari sistem [2], [3]. Pendekatan ini digunakan secara luas karena mampu merepresentasikan sebagian besar gangguan tunggal yang berpotensi terjadi dalam sistem tenaga listrik dan berdampak signifikan terhadap keamanan operasi.

Penilaian keamanan statis sistem tenaga umumnya dilakukan dengan menghitung indeks kinerja sistem, seperti *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV), yang diperoleh melalui analisis aliran daya menggunakan metode *Newton Raphson Load Flow* (NRLF) [4]. Metode ini memiliki tingkat akurasi yang tinggi dalam menggambarkan kondisi sistem pasca-kontingensi. Namun demikian, proses perhitungan NRLF memerlukan waktu komputasi yang relatif besar, terutama ketika diterapkan pada berbagai variasi kondisi beban dan sejumlah besar skenario kontingensi N-1. Kondisi ini menyebabkan pendekatan konvensional menjadi kurang efisien untuk aplikasi daring (*online*) yang menuntut kecepatan dan respons real time di pusat pengendalian sistem tenaga listrik [5].

Seiring dengan perkembangan teknologi komputasi, berbagai penelitian telah mengusulkan pemanfaatan teknik kecerdasan buatan, khususnya jaringan saraf tiruan (*Artificial Neural Network*), untuk mengatasi keterbatasan tersebut

dalam penilaian keamanan statis sistem tenaga [6]. ANN memiliki kemampuan generalisasi yang baik serta waktu komputasi yang relatif cepat, sehingga mampu memprediksi kondisi sistem pada skenario yang belum pernah dilatih sebelumnya. Salah satu pendekatan ANN yang banyak digunakan adalah *Radial Basis Function Network* (RBFN), yang telah diterapkan dalam modul *online* power system static security assessment untuk memprediksi nilai PIP dan PIV sekaligus melakukan perankingan kontingensi N-1 secara efisien [7]. Hasil penelitian terdahulu menunjukkan bahwa pendekatan berbasis RBFN mampu memberikan akurasi yang sebanding dengan metode NRLE, namun dengan waktu komputasi yang jauh lebih singkat, sehingga berpotensi mendukung implementasi pemantauan keamanan sistem tenaga secara real time.

Meskipun RBFN telah terbukti efektif sebagai pendekatan ANN konvensional dalam penilaian keamanan statis, perkembangan metode pembelajaran mesin modern berbasis ensemble, seperti *Extreme Gradient Boosting* (XGBoost), menunjukkan potensi peningkatan kinerja dalam berbagai permasalahan regresi dan prediksi nonlinier [8]. XGBoost dikenal memiliki kemampuan pemodelan yang kuat, efisiensi komputasi yang tinggi, serta mekanisme regularisasi yang baik untuk menghindari overfitting. Keunggulan tersebut relevan dengan permasalahan perankingan kontingensi N-1 berbasis indeks PIP dan PIV, karena penentuan tingkat keparahan kontingensi pada dasarnya merupakan permasalahan regresi nonlinier. Nilai PIP dan PIV dipengaruhi oleh interaksi kompleks antara parameter sistem, seperti impedansi saluran, distribusi beban, pembangkitan, serta perubahan topologi jaringan akibat gangguan. Melalui mekanisme gradient boosting dan optimasi fungsi kerugian, XGBoost mampu memodelkan hubungan nonlinier secara efektif sehingga berpotensi menghasilkan prediksi yang lebih akurat dibandingkan metode ANN konvensional [9]. Namun demikian, penerapan XGBoost dalam konteks penilaian keamanan statis sistem tenaga listrik, khususnya untuk perankingan kontingensi N-1 berbasis indeks PIP dan PIV, masih relatif terbatas dalam literatur. Oleh karena itu, diperlukan suatu evaluasi yang sistematis untuk membandingkan kinerja XGBoost dengan metode ANN konvensional seperti RBFN.

Dalam penelitian ini digunakan sistem IEEE 30-bus sebagai sistem uji karena merupakan sistem standar yang umum digunakan dalam studi analisis sistem tenaga listrik. Sistem ini memiliki karakteristik jaringan interkoneksi yang terdiri dari beberapa pembangkit dan beban. Karakteristik tersebut memiliki kemiripan dengan sistem tenaga listrik nyata di Indonesia, seperti pada sistem subtransmisi interkoneksi Jawa Madura Bali, yang juga terdiri dari beberapa unit pembangkit dan jaringan transmisi yang kompleks. Oleh karena itu, penggunaan IEEE 30-bus dianggap representatif untuk mengkaji metode yang diusulkan sebelum diterapkan pada sistem nyata.

Berdasarkan latar belakang tersebut, penelitian ini menyajikan studi komparatif antara metode XGBoost dan *Radial Basis Function Network* dalam perankingan kontingensi N-1 sistem tenaga listrik untuk penilaian keamanan statis. Perbandingan dilakukan dengan mengevaluasi akurasi prediksi indeks kinerja daya aktif dan tegangan, konsistensi peringkat kontingensi terhadap hasil NRLF, serta kompleksitas implementasi kedua metode. Hasil studi ini diharapkan dapat memberikan dasar yang kuat dalam pemilihan metode yang lebih sesuai untuk implementasi modul penilaian keamanan statis sistem tenaga listrik secara *online*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah sebagai berikut

1. Bagaimana kinerja metode XGBoost dan *Radial Basis Function Network* (RBFN) dalam memprediksi indeks keamanan statis sistem tenaga listrik, yaitu *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV), pada analisis kontingensi N-1 ?
2. Bagaimana tingkat kesesuaian hasil perankingan kontingensi N-1 yang dihasilkan oleh metode XGBoost dan RBFN dibandingkan dengan metode *Newton Raphson Load Flow* (NRLF)?
3. Bagaimana perbandingan kompleksitas implementasi metode XGBoost dan *Radial Basis Function Network* (RBFN) dalam penilaian keamanan statis sistem tenaga listrik secara online berdasarkan struktur model dan proses pembentukannya?

1.3 Tujuan

Berdasarkan latar belakang yang telah diuraikan, penelitian ini memiliki tujuan sebagai berikut,

1. Membandingkan kinerja metode XGBoost dan *Radial Basis Function Network* (RBFN) dalam memprediksi nilai indeks keamanan statis sistem tenaga listrik (PIP dan PIV).
2. Menganalisis kesesuaian hasil perankingan kontingensi N-1 yang dihasilkan oleh metode XGBoost dan RBFN terhadap hasil perankingan menggunakan metode NRLF.
3. Membandingkan kompleksitas implementasi serta kemampuan generalisasi metode XGBoost dan *Radial Basis Function Network* (RBFN) untuk mendukung penerapan penilaian keamanan statis sistem tenaga listrik secara online.

1.4 Batasan Masalah

Untuk menjaga fokus penelitian, batasan masalah yang ditetapkan adalah sebagai berikut,

1. Penelitian ini dibatasi pada penilaian keamanan statis sistem tenaga listrik menggunakan analisis kontingensi N-1 berupa gangguan satu saluran transmisi, dengan indeks kinerja yang digunakan adalah *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV) yang diperoleh dari hasil analisis *Newton Raphson Load Flow* (NRLF).
2. Sistem yang digunakan dalam penelitian ini adalah sistem uji standar IEEE 30-bus dengan variasi kondisi beban tertentu sesuai skenario pengujian, sehingga tidak mencakup seluruh kemungkinan kondisi operasi sistem tenaga listrik secara nyata.
3. Metode yang dibandingkan dalam penelitian ini dibatasi pada *Extreme Gradient Boosting* (XGBoost) dan *Radial Basis Function Network* (RBFN), dengan fokus evaluasi pada akurasi prediksi, konsistensi perankingan kontingensi, kompleksitas implementasi model, dan kemampuan generalisasi, tanpa membahas implementasi perangkat keras maupun integrasi sistem secara langsung.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Kajian kontingensi dalam sistem tenaga listrik banyak dilakukan karena gangguan N-1 dapat menyebabkan kelebihan beban saluran dan deviasi tegangan, sehingga berdampak pada keamanan dan keandalan sistem[4]. Analisis kontingensi secara konvensional umumnya menggunakan metode *Newton Raphson Load Flow* (NRLF), namun membutuhkan waktu komputasi yang besar karena setiap skenario harus dianalisis secara terpisah [10].

Untuk meningkatkan efisiensi komputasi, berbagai metode berbasis pembelajaran mesin, khususnya *Artificial Neural Network* (ANN) dan turunannya seperti RBFN dan MLP, telah digunakan dan terbukti mampu menghasilkan akurasi yang sebanding dengan NRLF dengan waktu komputasi yang lebih singkat [7]. Seiring berkembangnya algoritma pembelajaran mesin, metode ensemble modern seperti *Extreme Gradient Boosting* (XGBoost) menawarkan potensi peningkatan kinerja prediksi[11]. Namun, kajian komparatif yang membandingkan XGBoost dengan metode ANN konvensional, khususnya RBFN, dalam perankingan kontingensi N-1 untuk penilaian keamanan statis sistem tenaga listrik masih terbatas. Oleh karena itu, penelitian ini difokuskan pada perbandingan kinerja NRLF, RBFN, dan XGBoost.

Tabel 2. 1 Penelitian Terdahulu yang Berkaitan

Acuan / Tahun	Masalah yang Dibahas	Metode yang Digunakan	Indeks / Fokus Analisis	Asumsi / Keterbatasan
Saeh, 2014 [12]	Tinjauan penilaian keamanan statis sistem tenaga listrik berbasis pembelajaran mesin	<i>Artificial Neural Network</i> (ANN)	Indeks keamanan statis	Bersifat studi tinjauan, belum membandingkan metode ensemble modern

Alvarez et al., 2024 [13]	Penilaian keamanan statis sistem tenaga untuk skenario kontingensi menggunakan machine learning	Machine Learning (ML-based framework)	Static security assessment dan analisis kontingensi	Fokus pada keamanan statis umum, belum membahas indeks PIP dan PIV
Sekhar & Mohanty, 2016 [7]	Tingginya waktu komputasi metode konvensional dalam penilaian keamanan statis dan perankingan kontingensi	NRLF, RBFN, MLP	PIP dan PIV	Sistem deterministik dan keterbatasan kemampuan generalisasi model
Zhang et al., 2024 [14]	Penilaian keamanan statis sistem tenaga listrik di bawah skenario ketidakpastian dengan pendekatan data-driven	XGBoost, Stochastic Power Flow	Static security assessment dan evaluasi kontingensi	Fokus pada ketidakpastian sistem, belum membandingkan dengan ANN konvensional seperti RBFN
Chen & Guestrin, 2016 [9]	Pengembangan algoritma boosting yang efisien dan akurat untuk pemodelan nonlinier	XGBoost	Fungsi objektif dan ensemble tree	Tidak diaplikasikan secara khusus pada sistem tenaga listrik
Putatunda & Rama, 2018 [11]	Optimasi hyperparameter pada algoritma XGBoost untuk meningkatkan kinerja model	Grid Search, Random Search, Bayesian Optimization	Akurasi dan waktu komputasi	Studi bersifat umum dan tidak spesifik pada aplikasi sistem tenaga
Penelitian ini	Perbandingan metode pembelajaran mesin untuk perankingan kontingensi N-1 pada keamanan statis sistem tenaga	NRLF, RBFN, XGBoost	PIP dan PIV	Fokus pada keamanan statis sistem tenaga listrik

2.2 Sistem Tenaga Listrik

Sistem tenaga listrik merupakan suatu sistem terintegrasi yang terdiri dari pembangkitan, transmisi, dan distribusi energi listrik. Sistem ini dirancang untuk menyalurkan energi listrik dari pusat pembangkitan menuju konsumen secara andal, aman, dan berkesinambungan. Dalam proses penyaluran energi listrik, khususnya pada jaringan transmisi, sistem sangat rentan terhadap gangguan yang dapat menyebabkan terputusnya aliran daya, perubahan profil tegangan, serta kelebihan beban pada peralatan jaringan [15].

Gangguan pada saluran transmisi, baik yang bersifat internal maupun eksternal, dapat berdampak langsung terhadap kinerja dan keandalan sistem tenaga listrik secara keseluruhan. Oleh karena itu, diperlukan perencanaan dan pengoperasian sistem yang memperhatikan batasan teknis serta kondisi keamanan sistem agar gangguan yang terjadi tidak berkembang menjadi kegagalan yang lebih luas [16]

2.2.1 Keamanan Sistem Tenaga

Keamanan sistem tenaga listrik merupakan salah satu aspek fundamental dalam pengoperasian sistem tenaga listrik modern. Keamanan sistem didefinisikan sebagai kemampuan sistem untuk mempertahankan kondisi operasi yang stabil dan andal, baik pada keadaan operasi normal maupun setelah terjadinya gangguan tertentu. Sistem tenaga listrik dikatakan berada dalam kondisi aman apabila seluruh tegangan bus dan aliran daya pada saluran transmisi masih berada dalam batas operasi yang diizinkan, sehingga tidak menimbulkan risiko kerusakan peralatan maupun gangguan kontinuitas suplai energi listrik kepada konsumen [17]

Dalam praktiknya, sistem tenaga listrik beroperasi dalam kondisi yang dinamis, di mana variasi beban, perubahan pola pembangkitan, serta kondisi jaringan dapat berubah sewaktu-waktu. Perubahan ini dapat menyebabkan sistem berada dekat dengan batas operasi tertentu, sehingga gangguan kecil sekalipun berpotensi menimbulkan dampak yang signifikan apabila tidak diantisipasi dengan baik. Oleh karena itu, sistem tenaga listrik harus dirancang dan dioperasikan dengan memperhatikan aspek keamanan agar tetap mampu menahan gangguan yang kredibel tanpa kehilangan stabilitas atau melanggar batas operasional [15].

Gangguan pada sistem tenaga listrik secara umum dapat diklasifikasikan menjadi dua kategori utama, yaitu gangguan terjadwal (*scheduled outages*) dan gangguan tidak terjadwal (*forced outages*)[18].

1. Gangguan terjadwal merupakan gangguan yang direncanakan sebelumnya oleh operator sistem, biasanya untuk keperluan pemeliharaan, inspeksi, atau penggantian peralatan. Karena waktu dan durasi gangguan telah diketahui, operator dapat melakukan pengaturan ulang sistem sehingga dampaknya terhadap keandalan suplai listrik dapat diminimalkan.
2. Gangguan tidak terjadwal terjadi secara tiba-tiba dan sulit diprediksi. Gangguan jenis ini dapat disebabkan oleh kegagalan peralatan internal, kesalahan operasi, maupun faktor eksternal seperti sambaran petir, badai angin, atau kondisi lingkungan ekstrem lainnya. Gangguan tidak terjadwal memiliki potensi lebih besar untuk memicu kondisi tidak aman, karena sistem tidak memiliki waktu persiapan untuk menyesuaikan parameter operasi sebelum gangguan terjadi.

Apabila suatu gangguan menyebabkan satu komponen sistem terlepas dari jaringan, maka aliran daya yang semula melewati komponen tersebut akan dialihkan ke komponen lain. Jika komponen yang tersisa tidak dirancang untuk menanggung beban tambahan tersebut, maka dapat terjadi kelebihan beban yang memicu pemutusan lanjutan oleh sistem proteksi. Fenomena ini dikenal sebagai kegagalan berantai (*cascading failures*), yang dalam kondisi ekstrem dapat mengakibatkan pemadaman sebagian atau seluruh sistem tenaga listrik (*blackout*) [19].

Untuk mencegah terjadinya kondisi tersebut, keamanan sistem tenaga listrik secara umum dijalankan melalui tiga fungsi utama di pusat kendali operasi, yaitu pemantauan sistem (*system monitoring*), analisis kontingensi (*contingency analysis*), dan aliran daya optimal dengan kendala keamanan (*security-constrained optimal power flow*) [20]. Ketiga fungsi ini saling melengkapi dan berperan penting dalam menjaga sistem agar tetap berada dalam kondisi aman dan andal.

2.2.2 Kontingensi Sistem Tenaga

Kontingensi dalam sistem tenaga listrik merujuk pada kondisi gangguan yang terjadi akibat kegagalan satu atau lebih komponen sistem, seperti saluran transmisi, transformator, atau unit pembangkit. Gangguan tersebut dapat menyebabkan perubahan signifikan pada aliran daya, distribusi beban, serta profil tegangan pada bus-bus sistem tenaga listrik [19].

Salah satu jenis kontingensi yang paling umum digunakan dalam kajian keamanan sistem adalah kontingensi N-1, yaitu kondisi di mana satu komponen sistem mengalami gangguan sementara komponen lainnya tetap beroperasi. Analisis kontingensi N-1 digunakan secara luas sebagai standar keandalan karena dianggap mewakili gangguan yang paling mungkin terjadi dalam praktik operasi sistem tenaga listrik. Keadaan operasi sistem tenaga listrik akibat kontingensi dapat diklasifikasikan ke dalam empat kondisi utama.

1. Kondisi pertama adalah optimal dispatch, yaitu kondisi operasi normal tanpa adanya gangguan, di mana tujuan utama pengoperasian sistem adalah meminimalkan biaya pembangkitan dengan tetap memperhatikan batasan teknis sistem. Pada kondisi ini, seluruh parameter sistem berada dalam batas aman.
2. Kondisi kedua adalah *post-contingency*, yaitu keadaan sistem setelah terjadinya gangguan. Pada kondisi ini, sistem sering kali mengalami pelanggaran batas operasi, seperti kelebihan aliran daya pada saluran tertentu atau deviasi tegangan pada beberapa bus. Kondisi ini memerlukan tindakan korektif agar sistem dapat kembali ke keadaan aman.
3. Kondisi ketiga adalah *secure dispatch*, yaitu kondisi operasi normal di mana parameter sistem telah disesuaikan secara preventif. Penyesuaian ini dilakukan agar sistem tetap aman apabila terjadi gangguan tertentu yang telah diperkirakan sebelumnya. Dengan kata lain, sistem dioperasikan dengan mempertimbangkan aspek keamanan sejak awal.
4. Kondisi keempat adalah *secure post-contingency*, yaitu keadaan sistem setelah terjadinya gangguan, namun tetap berada dalam batas operasi yang diizinkan karena telah dilakukan penyesuaian parameter sistem sebelumnya. Kondisi ini umumnya mengacu pada pemenuhan kriteria N-,

di mana satu gangguan tunggal tidak menyebabkan sistem berada dalam kondisi tidak aman [20].

2.2.2.1 Hubungan antara Gangguan (Fault) dan Kontingensi

Dalam kajian keamanan sistem tenaga listrik, istilah gangguan (fault) dan kontingensi (*contingency*) sering digunakan secara bergantian, padahal keduanya merepresentasikan tahapan yang berbeda dalam satu rangkaian sebab-akibat. Gangguan merupakan peristiwa elektrik yang menjadi penyebab awal, sedangkan kontingensi merupakan akibat atau konsekuensi topologis yang timbul setelah gangguan tersebut ditangani oleh sistem proteksi.

Gangguan pada sistem tenaga listrik dapat bersifat terencana (*planned outage*), yaitu ketika suatu komponen sengaja dikeluarkan dari operasi untuk keperluan pemeliharaan atau perbaikan, maupun tidak terencana (*forced outage*), yaitu ketika komponen keluar dari operasi secara tiba-tiba akibat kegagalan internal, kesalahan operasi, atau faktor eksternal seperti sambaran petir dan cuaca ekstrem [18]. Pada kedua kondisi tersebut, sistem proteksi yang terdiri dari relai dan pemutus tenaga (*circuit breaker*) bekerja mengisolasi bagian sistem yang mengalami gangguan agar dampaknya tidak menyebar ke bagian jaringan lain.

Dengan demikian, kontingensi bukan merupakan gangguan itu sendiri, melainkan keadaan sistem setelah satu komponen jaringan baik saluran transmisi, transformator, maupun unit pembangkit terlepas dari topologi akibat kerja sistem proteksi dalam merespons suatu gangguan. Perbedaan ini penting untuk ditegaskan karena penelitian ini tidak memodelkan proses fisik terjadinya gangguan, melainkan memodelkan kondisi tunak (*steady-state*) sistem setelah satu elemen jaringan tersebut hilang dari operasi.

2.2.2.2 Definisi Kontingensi N-1

Notasi N pada kontingensi N-1 merepresentasikan jumlah total komponen kritis dalam sistem yang berpotensi mengalami gangguan, yaitu keseluruhan saluran transmisi, transformator, atau unit pembangkit yang menjadi elemen kandidat kontingensi. Kontingensi N-1 didefinisikan sebagai kondisi operasi

system ketika tepat satu (1) komponen dari total N komponen tersebut keluar dari operasi, sementara N-1 komponen sisanya tetap beroperasi secara normal [3], [4].

Kriteria N-1 digunakan secara luas sebagai standar keandalan (reliability criterion) dalam operasi sistem tenaga listrik karena kegagalan tunggal merupakan skenario gangguan yang paling mungkin terjadi secara statistik dibandingkan dengan kegagalan ganda (N-2) atau kegagalan majemuk lainnya. Sistem tenaga listrik yang memenuhi kriteria N-1 harus tetap mampu mempertahankan kondisi operasi dalam batas aman baik dari sisi pembebanan saluran maupun profil tegangan bus meskipun kehilangan satu elemen jaringan mana pun secara tunggal [4].

Analisis kontingensi N-1 dilaksanakan dengan menguji setiap kemungkinan elemen yang dapat hilang secara bergantian dan berurutan, satu per satu, terhadap seluruh elemen kandidat dalam sistem. Setiap pengujian menghasilkan satu skenario kontingensi yang independen, sehingga jika sistem memiliki NL saluran transmisi, maka akan diperoleh sejumlah NL skenario kontingensi N-1 yang masing-masing merepresentasikan hilangnya satu saluran transmisi tertentu dari topologi jaringan.

2.2.2.3 Jenis-Jenis Gangguan pada Sistem Tenaga Listrik

Sebagai penyebab yang memicu terjadinya kontingensi, gangguan pada sistem tenaga listrik dapat diklasifikasikan ke dalam beberapa kategori berikut.

- Gangguan hubung singkat (short circuit fault), yang merupakan jenis gangguan paling umum terjadi dan terbagi lagi menjadi empat jenis berdasarkan fasa yang terlibat, yaitu gangguan satu fasa ke tanah (single line-to-ground), gangguan fasa ke fasa (line-to-line), gangguan dua fasa ke tanah (double line-to-ground), serta gangguan tiga fasa simetris (three-phase fault). Gangguan satu fasa ke tanah merupakan jenis yang paling sering terjadi di lapangan, sedangkan gangguan tiga fasa simetris jarang terjadi namun menimbulkan dampak yang paling parah karena melibatkan ketiga fasa secara bersamaan.
- Gangguan hubung terbuka (open conductor fault), yaitu terputusnya konduktor tanpa disertai hubung singkat, misalnya akibat kawat penghantar putus atau kegagalan sambungan.

- Gangguan peralatan, yang mencakup kegagalan transformator, kegagalan pemutus tenaga, maupun kegagalan isolator.
- Pembebanan lebih (overload), yaitu kondisi ketika aliran daya pada suatu saluran atau transformator melampaui kapasitas nominalnya, yang dalam praktiknya juga dapat memicu kerja proteksi arus lebih meskipun bukan merupakan gangguan seketika.
- Penyebab eksternal, seperti sambaran petir, pohon tumbang, cuaca ekstrem, serta kegagalan isolasi akibat usia peralatan yang telah menurun.

Keempat kategori pertama umumnya bersumber dari kondisi internal sistem, sedangkan kategori terakhir berasal dari faktor lingkungan di luar sistem. Terlepas dari jenis gangguan yang terjadi, dampak akhirnya terhadap studi keamanan statis adalah sama, yaitu hilangnya satu elemen jaringan dari topologi sistem, sebagaimana dijelaskan pada Subbab 2.2.2.4.

2.2.2.4 Mekanisme Pemutusan Saluran dan Representasi pada Studi Keamanan Statis

Sistem tenaga listrik pada dasarnya beroperasi dalam bentuk tiga fasa. Meskipun gangguan awal yang memicu suatu kontingensi seringkali hanya melibatkan satu fasa misalnya pada gangguan satu fasa ke tanah yang merupakan jenis gangguan paling umum terjadi skema proteksi pada sistem transmisi pada umumnya dirancang dengan prinsip three-phase tripping. Prinsip ini berarti bahwa relai proteksi akan memerintahkan pemutus tenaga untuk memutus ketiga fasa saluran secara bersamaan, bukan hanya fasa yang mengalami gangguan. Pendekatan ini diterapkan agar kondisi sistem tetap seimbang (balanced) setelah gangguan diisolasi, serta untuk mencegah timbulnya komponen urutan negatif dan nol yang dapat mengganggu kestabilan sistem apabila hanya satu fasa yang terputus.

Berdasarkan prinsip tersebut, sistem tenaga listrik tiga fasa yang seimbang dapat direpresentasikan secara ringkas dalam bentuk diagram satu garis (single-line diagram), di mana satu garis saluran pada diagram tersebut mewakili keseluruhan sirkuit tiga fasa. Dengan representasi ini, ketika suatu skenario kontingensi N-1 mensimulasikan keluarnya suatu saluran transmisi dari sistem, yang dimaksud

adalah lepasnya seluruh sirkuit tiga fasa saluran tersebut secara utuh dari topologi jaringan, bukan hanya salah satu fasanya. Analisis terhadap gangguan yang bersifat tidak seimbang, seperti hubung singkat satu fasa ke tanah, memerlukan pendekatan komponen simetris (*symmetrical components*), yaitu penguraian sistem tiga fasa menjadi komponen urutan positif, negatif, dan nol. Analisis semacam ini umumnya menjadi kajian tersendiri dalam bentuk studi hubung singkat (*short-circuit study* atau *fault analysis*) yang terpisah dari studi keamanan statis berbasis aliran daya. Penelitian ini berfokus pada studi keamanan statis dengan asumsi sistem tetap berada dalam kondisi tiga fasa seimbang, baik sebelum maupun sesudah kontingensi terjadi. Variabel yang berubah akibat kontingensi adalah topologi jaringan, yaitu hilangnya satu elemen saluran transmisi, dan bukan kondisi ketidakseimbangan fasa pada sistem.

2.2.3 Analisis Kontingensi

Analisis kontingensi merupakan tahapan penting dalam kajian keamanan sistem tenaga listrik yang bertujuan untuk mengevaluasi dampak gangguan terhadap kondisi operasi sistem. Gangguan yang dianalisis umumnya berupa pemadaman satu atau lebih komponen sistem, seperti saluran transmisi atau unit pembangkit, untuk mengetahui apakah sistem masih dapat beroperasi dalam batas aman yang diizinkan [4]. Melalui analisis ini, komponen yang paling kritis serta potensi pelanggaran batas operasi dapat diidentifikasi sejak dini.

Analisis kontingensi dilakukan dengan menerapkan setiap skenario gangguan ke dalam model sistem tenaga listrik, kemudian mengevaluasi kembali kondisi sistem menggunakan analisis aliran daya. Pendekatan ini memungkinkan penentuan profil tegangan bus, aliran daya pada saluran transmisi, serta rugi-rugi daya sistem setelah terjadinya gangguan. Informasi tersebut menjadi dasar dalam penilaian tingkat keparahan kontingensi terhadap keamanan sistem tenaga listrik [10].

Metode analisis aliran daya yang paling banyak digunakan dalam analisis kontingensi adalah *Newton Raphson Load Flow* (NRLF). Metode ini dipilih karena memiliki tingkat akurasi yang tinggi dan konvergensi yang cepat pada sistem nonlinier. NRLF menyelesaikan persamaan keseimbangan daya aktif dan reaktif

pada setiap bus, yang secara umum dirumuskan dalam persamaan berikut.

$$P_i = \sum_{j=1}^N |V_i| |V_j| (G_{ij} \cos \delta_{ij} + B_{ij} \sin \delta_{ij}) \quad (2.1)$$

$$Q_i = \sum_{j=1}^N |V_i| |V_j| (G_{ij} \sin \delta_{ij} - B_{ij} \cos \delta_{ij}) \quad (2.2)$$

dengan,

- $|V_i|, |V_j|$ = magnitudo tegangan pada bus i dan j,
- δ_{ij} = selisih sudut fasa bus i dan j,
- G_{ij} = elemen konduktansi dan susceptansi dari matriks admitansi bus (*Radial Basis Function Networkbus*),
- N = jumlah total bus dalam sistem.

Pada metode *Newton Raphson*, persamaan nonlinier di atas diselesaikan secara iteratif dengan menggunakan matriks Jacobian yang menghubungkan mismatch daya aktif dan reaktif terhadap perubahan sudut dan magnitudo tegangan, sebagaimana dinyatakan pada Persamaan berikut,

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = J \begin{bmatrix} \Delta \delta \\ \Delta |V| \end{bmatrix} \quad (2.3)$$

Proses iterasi dilakukan hingga nilai mismatch daya berada di bawah batas toleransi yang ditentukan. Setelah konvergensi tercapai, diperoleh nilai tegangan bus dan aliran daya saluran yang merepresentasikan kondisi sistem pasca-kontingensi, NRLF memberikan hasil yang akurat dan reliabel, penerapannya dalam analisis kontingensi menghadapi kendala dari sisi waktu komputasi. Setiap skenario gangguan harus dianalisis secara terpisah, sehingga pada sistem berskala besar dengan jumlah kontingensi yang banyak, total waktu perhitungan menjadi sangat besar. Hal ini menjadikan pendekatan konvensional kurang sesuai untuk aplikasi *online* atau real time [10].

2.2.4 Contingency Selection dan Ranking

Dalam praktik operasi sistem tenaga listrik, jumlah kemungkinan kontingensi yang dapat terjadi relatif sangat besar, khususnya pada sistem berskala

menengah hingga besar. Tidak seluruh kontingensi tersebut memberikan dampak yang signifikan terhadap keamanan sistem. Oleh karena itu, diperlukan suatu mekanisme untuk menyeleksi dan mengurutkan kontingensi berdasarkan tingkat keparahannya agar analisis dapat difokuskan pada gangguan yang paling kritis. Proses ini dikenal sebagai Contingency selection dan contingency ranking [21].

Contingency selection bertujuan untuk menyaring kontingensi yang berpotensi menimbulkan pelanggaran batas operasi sistem, baik berupa kelebihan aliran daya pada saluran transmisi maupun deviasi tegangan bus. Setelah kontingensi yang relevan teridentifikasi, langkah selanjutnya adalah *contingency ranking*, yaitu mengurutkan kontingensi berdasarkan tingkat dampaknya terhadap keamanan sistem. Dengan adanya proses perankingan, operator sistem dapat menentukan prioritas penanganan terhadap gangguan yang memiliki risiko paling tinggi terhadap keandalan sistem tenaga listrik.

Perankingan kontingensi umumnya dilakukan menggunakan indeks kinerja (*performance index*) yang merepresentasikan kondisi sistem secara kuantitatif setelah terjadinya gangguan. Indeks kinerja ini dihitung berdasarkan hasil analisis aliran daya pasca-kontingensi. Kontingensi dengan nilai indeks yang lebih besar menunjukkan tingkat pelanggaran batas operasi yang lebih serius, sehingga dikategorikan sebagai kontingensi yang lebih kritis dan memerlukan perhatian prioritas [4].

Dalam penelitian ini, indeks kinerja yang digunakan adalah *Power Performance index* (PIP) dan *Voltage Performance Index* (PIV). Kedua indeks tersebut dihitung berdasarkan hasil *Newton Raphson Load Flow* (NRLF) pasca-kontingensi dan digunakan sebagai dasar dalam proses *Contingency selection* dan ranking. Selain itu, nilai PIP dan PIV juga dimanfaatkan sebagai target keluaran dalam pelatihan model *Radial Basis Function Network* (RBFN).

2.2.4.1 Power Performance Index (PIP)

Power Performance index (PIP) digunakan untuk mengevaluasi dampak kontingensi terhadap aliran daya aktif pada saluran transmisi. Indeks ini menggambarkan tingkat pembebanan saluran setelah terjadinya gangguan dibandingkan dengan kapasitas maksimum yang diizinkan. Aliran daya saluran

yang melampaui batas operasional dapat meningkatkan risiko kegagalan peralatan serta memicu gangguan lanjutan pada sistem tenaga listrik. Secara matematis, PIP dirumuskan sebagai berikut,

$$PIP = \sum_{l=1}^{NL} \frac{W}{2n} \left(\frac{Pl}{Pl^{max}} \right)^{2n} \quad (2.4)$$

dengan,

- Pl = aliran daya aktif pada saluran ke l ,
- Pl^{max} = kapasitas maksimum saluran ke l ,
- NL = jumlah total saluran transmisi,
- W = faktor bobot (umumnya = 1),
- n = eksponen fungsi penalti (biasanya = 1).

Nilai PIP yang semakin besar menunjukkan bahwa semakin banyak saluran yang mengalami pembebanan tinggi atau mendekati batas maksimumnya. Oleh karena itu, kontingensi yang menghasilkan nilai PIP terbesar dapat dikategorikan sebagai kontingensi paling kritis dari sisi keamanan aliran daya dan menjadi prioritas utama dalam proses perankingan kontingensi [7].

2.2.4.2 Voltage Performance Index (PIV)

Voltage Performance Index (PIV) digunakan untuk menilai dampak kontingensi terhadap profil tegangan bus pada sistem tenaga listrik. Indeks ini mengukur tingkat deviasi tegangan bus dari nilai referensi atau nilai spesifikasi yang telah ditetapkan. Tegangan bus yang berada di luar batas yang diizinkan dapat menyebabkan gangguan kualitas daya serta meningkatkan risiko kerusakan peralatan sistem. PIV dirumuskan sebagai berikut,

$$PIV = \sum_{i=1}^{NB} \frac{W}{2n} \left(\frac{|Vi| - |Vi^{sp}|}{\Delta Vi^{lim}} \right)^{2n} \quad (2.5)$$

dengan,

- $|Vi|$ = magnitudo tegangan pada bus ke- i setelah kontingensi

- $|Vi^{sp}|$ = tegangan referensi (tegangan nominal) pada bus ke-i,
- ΔVi^{lim} = toleransi deviasi tegangan
- NB = jumlah total bus,
- W dan n sama seperti pada rumus PIP.

Nilai PIV yang tinggi mengindikasikan adanya penyimpangan tegangan yang signifikan pada satu atau lebih bus akibat kontingensi [18]. Dengan demikian, kontingensi dengan nilai PIV terbesar diklasifikasikan sebagai kontingensi yang paling berbahaya dari sisi kestabilan tegangan sistem.

2.2.5 Metrik Kesesuaian Perankingan Kontingensi (Capture Ratio)

Selain akurasi prediksi nilai indeks kinerja, kesesuaian hasil perankingan kontingensi terhadap metode acuan juga menjadi aspek penting dalam evaluasi metode *contingency* ranking. Salah satu metrik yang digunakan untuk mengukur kesesuaian tersebut adalah *Capture Ratio* (CR), [22] dan dikembangkan lebih lanjut dalam konteks algoritma metaheuristik (Algoritma Genetika dan Ant Colony Optimization) untuk seleksi kontingensi pada sistem tenaga listrik berskala besar[23]. *Capture Ratio* mengukur seberapa besar kesesuaian antara daftar N kontingensi paling kritis hasil metode yang diuji dengan daftar N kontingensi paling kritis hasil metode acuan (ground truth), yang dirumuskan sebagai,

$$CR = \left(\frac{K_p}{N} \right) \times 100\% \quad (2.6)$$

Dengan K_p adalah jumlah kontingensi pada N posisi teratas hasil metode yang diuji yang juga termasuk dalam N kontingensi paling kritis menurut metode acuan, dan N adalah jumlah kontingensi teratas yang dibandingkan. Metode acuan yang digunakan adalah metode ekshaustif yang menghitung seluruh kemungkinan kontingensi, sedangkan metode yang diuji berupa algoritma metaheuristik[22], [23]. Pada penelitian ini, konsep *Capture Ratio* diadaptasi dengan menempatkan hasil simulasi *Newton Raphson Load Flow* (NRLF) sebagai metode acuan, sedangkan metode yang diuji adalah hasil prediksi RBFN dan XGBoost. Metrik ini

digunakan untuk melengkapi evaluasi akurasi numerik (R^2 , RMSE, MAE, MAPE) dengan evaluasi konsistensi urutan prioritas kontingensi.

2.3 Static Security Assessment (SSA)

Static Security Assessment (SSA) merupakan metode evaluasi keamanan sistem tenaga listrik yang dilakukan pada kondisi tunak, tanpa mempertimbangkan respons dinamis sistem terhadap gangguan. SSA bertujuan untuk menentukan apakah sistem berada dalam kondisi aman, alarm, atau tidak aman berdasarkan hasil analisis aliran daya [24].

Pada SSA konvensional, evaluasi keamanan dilakukan dengan membandingkan tegangan bus dan aliran daya saluran terhadap batas yang diizinkan. Meskipun akurat, pendekatan ini menjadi kurang efisien untuk aplikasi *online* karena tingginya kebutuhan komputasi.

2.4 XGBoost (*Extreme Gradient Boosting*)

XGBoost (Extreme Gradient Boosting) merupakan algoritma machine learning berbasis pohon keputusan yang dikembangkan untuk meningkatkan efisiensi komputasi dan akurasi prediksi. Algoritma ini merupakan pengembangan dari metode Gradient Boosting Decision Tree (GBDT) [25] dengan berbagai peningkatan pada aspek optimasi dan pengendalian kompleksitas model.

XGBoost termasuk ke dalam metode ensemble learning, yaitu pendekatan yang menggabungkan beberapa model lemah (*weak learners*) untuk membentuk satu model prediksi yang lebih kuat. Proses pembelajaran dilakukan melalui mekanisme boosting, di mana model dibangun secara bertahap dengan menambahkan pohon keputusan baru pada setiap iterasi untuk memperbaiki kesalahan prediksi dari model sebelumnya. Pada penelitian ini, XGBoost digunakan sebagai model regresi (XGBoost Regressor) untuk memprediksi nilai Performance Index of Power (PIP) dan Performance Index of Voltage (PIV) yang bersifat kontinu, bukan untuk permasalahan klasifikasi.

Struktur umum algoritma XGBoost ditunjukkan pada Gambar 2.1, yang memperlihatkan bahwa setiap data masukan x_i diproses secara berurutan oleh sejumlah pohon keputusan. Setiap pohon menghasilkan nilai prediksi parsial, dan

seluruh keluaran pohon tersebut kemudian dijumlahkan untuk memperoleh nilai prediksi akhir $\hat{y}_i$. Prediksi model pada iterasi ke- t dirumuskan sebagai [26],

$$\widehat{y}_i^{(t)} = \sum_{k=1}^t f_k(x_i) = \widehat{y}_i^{(t-1)} + f_t(x_i) \quad (2.6)$$

Dengan f_k adalah fungsi prediksi (pohon keputusan) yang dibangun pada iterasi ke- k , dan $f_t(x_i)$ adalah pohon baru yang ditambahkan pada iterasi ke- t untuk memperbaiki prediksi dari iterasi sebelumnya. Mekanisme ini mencerminkan prinsip ensemble learning pada XGBoost, di mana kombinasi beberapa pohon keputusan mampu menghasilkan prediksi yang lebih akurat dibandingkan penggunaan satu pohon keputusan tunggal [27].

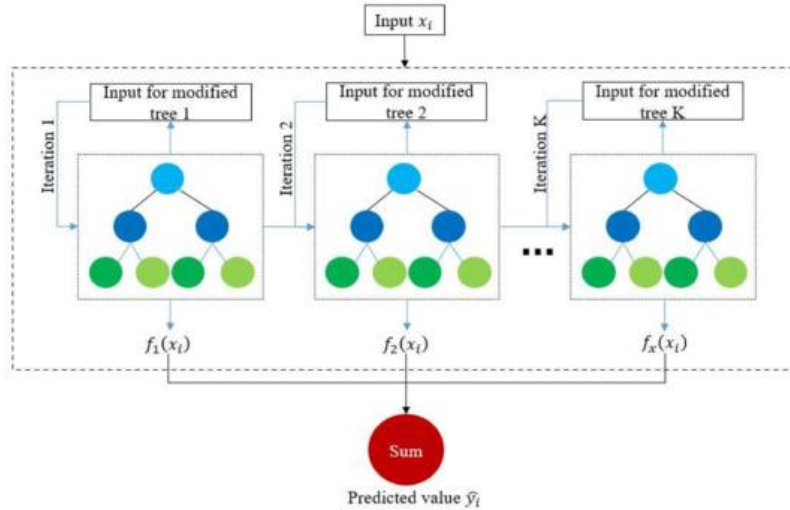
Selain itu, XGBoost menerapkan ekspansi Taylor orde kedua pada fungsi kerugian untuk mempercepat proses optimasi serta menggunakan regularisasi eksplisit untuk mengendalikan kompleksitas model dan mengurangi risiko overfitting. Algoritma ini juga menggunakan pendekatan histogram (histogram-based split finding) dalam proses pembentukan pohon, yang mempercepat pencarian titik pemisahan terbaik pada setiap node dibandingkan pendekatan exact greedy konvensional, sehingga XGBoost mampu menangani dataset berukuran besar secara lebih efisien dibandingkan metode boosting konvensional [28].

2.4.1 Prinsip Dasar dan Mekanisme Pembelajaran

Pada XGBoost, proses pembelajaran dilakukan secara aditif melalui penambahan pohon keputusan pada setiap iterasi. Setiap pohon keputusan baru dibangun untuk mempelajari kesalahan prediksi atau gradien negatif dari fungsi kerugian yang dihasilkan oleh model pada iterasi sebelumnya. Dengan mekanisme ini, kesalahan prediksi dapat dikurangi secara bertahap hingga hasil prediksi mendekati nilai aktual [29].

Prinsip pembentukan pohon keputusan pada XGBoost dapat dipahami melalui contoh struktur pohon seperti yang ditunjukkan pada Gambar 2.2. Pada gambar tersebut, setiap node internal merepresentasikan proses pengambilan keputusan berdasarkan nilai suatu fitur dengan aturan pemisahan tertentu. Proses

pemisahan ini membagi data ke dalam cabang-cabang yang lebih homogen hingga mencapai node akhir atau *leaf*.



Gambar 2. 1 Diagram struktur dan proses pembentukan model XGBoost

Setiap *leaf* pada pohon keputusan memiliki nilai keluaran tertentu yang merepresentasikan hasil prediksi dari pohon tersebut. Nilai inilah yang menjadi kontribusi prediksi parsial dari satu pohon dalam model XGBoost. Dengan demikian, setiap pohon berperan dalam memperbaiki kesalahan prediksi secara bertahap pada proses pembelajaran.

2.4.2 Perumusan Fungsi Optimasi

Pelatihan model XGBoost didasarkan pada fungsi optimasi yang bertujuan untuk menyeimbangkan antara akurasi prediksi dan kompleksitas model. Fungsi optimasi (objective function) XGBoost pada iterasi ke- t dirumuskan sebagai berikut[26],

$$\text{Obj}^{(t)} = \sum_{i=1}^n l(y_i, \widehat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \quad (2.7)$$

Dengan l adalah fungsi kerugian (loss function) yang mengukur perbedaan antara nilai aktual y_i dan nilai prediksi, dan $\Omega(f_t)$ adalah fungsi regularisasi yang mengendalikan kompleksitas pohon f_t yang ditambahkan pada iterasi ke- t . Pada

penelitian ini, karena permasalahan yang dikaji berupa regresi terhadap nilai PIP dan PIV, fungsi kerugian yang digunakan adalah kesalahan kuadrat (squared error),

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 \quad (2.8)$$

Untuk mempercepat proses optimasi, fungsi kerugian pada Persamaan (2.9) didekati menggunakan ekspansi Taylor orde kedua di sekitar $\hat{y}_i(t-1)$, sehingga diperoleh,

$$\text{Obj}^{(t)} \approx \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t) \quad (2.9)$$

dengan g_i dan h_i masing-masing adalah turunan pertama (gradien) dan turunan kedua (hessian) dari fungsi kerugian terhadap $\hat{y}_i(t-1)$,

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \quad (2.10)$$

$$h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2} \quad (2.11)$$

Fungsi regularisasi $\Omega(f)$ digunakan untuk mengendalikan kompleksitas pohon, sehingga model tidak terlalu menyesuaikan diri dengan data pelatihan. Regularisasi ini dirumuskan sebagai,

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_j w_j^2 + \alpha \sum_j |w_j| = T \gamma + \frac{1}{2} \lambda \sum_j w_j^2 + \alpha \sum_j |w_j| \quad (2.12)$$

dengan T adalah jumlah leaf (node akhir) pada pohon, w_j adalah nilai keluaran (bobot) pada leaf ke- j , γ adalah parameter penalti terhadap jumlah leaf (gamma), λ adalah parameter regularisasi L2 (reg_lambda), dan α adalah parameter regularisasi L1 (reg_alpha). Ketiga parameter (γ , λ , α) ini merupakan hyperparameter yang secara eksplisit dioptimasi pada penelitian ini, sebagaimana dijelaskan pada Subbab 2.4.4. Kombinasi antara fungsi kerugian dan fungsi

regularisasi memungkinkan XGBoost menghasilkan model dengan tingkat akurasi yang tinggi sekaligus memiliki kemampuan generalisasi yang baik pada data yang belum pernah dilihat sebelumnya [27].

2.4.3 Struktur Model Berbasis Pohon Keputusan

Model XGBoost dibangun menggunakan *Classification and Regression Tree* (CART) sebagai unit dasar pembelajaran. Karena permasalahan pada penelitian ini bersifat regresi, setiap pohon CART menghasilkan nilai prediksi kontinu pada tiap leaf, bukan label kelas. Setiap pohon CART terdiri atas node keputusan dan node akhir (*leaf*) yang menghasilkan nilai prediksi tertentu.

Dalam XGBoost, beberapa pohon keputusan CART digabungkan dalam suatu struktur ensemble. Setiap pohon dilatih secara berurutan, di mana pohon yang dibangun pada iterasi berikutnya berfokus pada perbaikan kesalahan prediksi yang dihasilkan oleh pohon-pohon sebelumnya. Proses pembentukan pohon pada penelitian ini menggunakan algoritma histogram (*hist*), yaitu pendekatan yang mengelompokkan nilai fitur ke dalam sejumlah bin sebelum mencari titik pemisahan (*split*) terbaik, sehingga proses pelatihan menjadi lebih efisien dibandingkan pendekatan pencarian split secara eksak (*exact greedy*) [31].

2.4.4 Tuning Hyperparameter

Hyperparameter merupakan parameter yang ditentukan sebelum proses pelatihan dimulai dan tidak dipelajari secara langsung dari data. Oleh karena itu, proses penyesuaian hyperparameter memegang peranan penting dalam menentukan performa akhir suatu algoritma pembelajaran mesin.

Optimasi hyperparameter merupakan proses untuk mencari kombinasi nilai hyperparameter yang menghasilkan kinerja terbaik berdasarkan kriteria evaluasi tertentu. Setiap algoritma memiliki jenis dan jumlah hyperparameter yang berbeda, sehingga proses optimasi sering kali memerlukan waktu yang cukup lama apabila dilakukan secara manual. Pada penelitian ini, optimasi hyperparameter dilakukan menggunakan random search, khususnya melalui *RandomizedSearchCV*, yang mengambil sejumlah kombinasi hyperparameter secara acak dari ruang pencarian yang telah ditentukan, alih-alih mengevaluasi seluruh kombinasi secara sistematis

seperti pada grid search. Pendekatan ini dipilih karena lebih efisien secara komputasi ketika ruang pencarian hyperparameter berukuran besar [28].

Proses validasi dilakukan menggunakan skema Group K-Fold Cross Validation yang dikelompokkan berdasarkan saluran (branch) kontingensi, dengan kriteria evaluasi berupa koefisien determinasi (R^2), untuk mencegah kebocoran data (data leakage) akibat munculnya saluran kontingensi yang sama pada data pelatihan dan pengujian sekaligus.

Pada algoritma XGBoost, terdapat sembilan hyperparameter utama yang dioptimasi pada penelitian ini karena berpengaruh signifikan terhadap kinerja model, antara lain sebagai berikut,

1. *n_estimators*, yaitu jumlah pohon keputusan atau jumlah iterasi dalam proses pelatihan. Nilai *n_estimators* yang terlalu kecil dapat menyebabkan underfitting, sedangkan nilai yang terlalu besar berpotensi menyebabkan overfitting.
2. *min_child_weight*, yang digunakan untuk menentukan bobot minimum sampel pada node atau cabang daun. Parameter ini berfungsi untuk mengendalikan kompleksitas model dan mencegah terjadinya overfitting.
3. *max_depth*, yaitu kedalaman maksimum pohon keputusan. Semakin besar nilai *max_depth*, semakin kompleks struktur pohon dan semakin tinggi kemampuan adaptasi model, namun risiko overfitting juga meningkat.
4. *subsample*, yang menyatakan proporsi data pelatihan yang digunakan untuk membangun setiap pohon keputusan. Penggunaan nilai *subsample* yang lebih kecil dapat meningkatkan variasi model dan mengurangi overfitting.
5. *colsample_bytree*, yaitu proporsi fitur yang digunakan saat membangun setiap pohon keputusan. Parameter ini membantu meningkatkan keacakan dan stabilitas model.
6. *learning_rate*, yang mengatur kontribusi setiap pohon keputusan terhadap model secara keseluruhan. Nilai *learning_rate* yang lebih kecil membuat proses pembelajaran menjadi lebih stabil, namun memerlukan jumlah pohon yang lebih banyak.

7. *gamma* (γ), yaitu ambang batas minimum penurunan fungsi kerugian yang disyaratkan agar suatu node bersedia dipecah menjadi dua leaf baru, sebagaimana Persamaan (2.11). Semakin besar nilai *gamma*, semakin konservatif proses pemisahan node, sehingga pohon yang terbentuk cenderung lebih sederhana.

8. *reg_alpha* (α), yaitu koefisien regularisasi L1 pada Persamaan (2.12) yang mendorong sebagian bobot leaf bernilai nol, sehingga dapat berfungsi sebagai seleksi fitur implisit dan mengurangi kompleksitas model.

9. *reg_lambda* (λ), yaitu koefisien regularisasi L2 pada Persamaan (2.12) yang menekan besar nilai bobot leaf secara merata, sehingga membantu mencegah overfitting akibat bobot leaf yang terlalu besar.

Pada penelitian ini, kesembilan hyperparameter tersebut dioptimasi bersama-sama pada ruang pencarian yang telah ditentukan menggunakan `RandomizedSearchCV` dengan validasi `Group K-Fold` [31].

2.5 Radial Basis Function Network

Radial Basis Function Network (RBFN) merupakan salah satu metode *Artificial Neural Network* (ANN) yang banyak digunakan untuk memodelkan hubungan nonlinier antara variabel masukan dan keluaran. RBFN dikenal memiliki struktur yang lebih sederhana serta proses pelatihan yang relatif cepat dibandingkan jaringan saraf berlapis seperti *Multi-Layer Perceptron* (MLP), sehingga sesuai untuk aplikasi prediksi dan estimasi pada sistem tenaga listrik [7]

Secara umum, arsitektur RBFN terdiri dari tiga lapisan utama. Lapisan input bertugas menerima variabel sistem, seperti aliran daya dan tegangan bus pada sistem tenaga listrik. Lapisan tersembunyi menggunakan neuron dengan fungsi Gaussian yang terpusat pada titik tertentu atau centroid. Setiap neuron menghitung jarak antara data input dengan pusatnya, lalu mengubah jarak tersebut menjadi keluaran nonlinier melalui fungsi Gaussian. Lapisan output bersifat linear, menggabungkan hasil keluaran lapisan tersembunyi untuk menghasilkan prediksi akhir [28], [29]. Gambar 2.4 memperlihatkan ilustrasi arsitektur dasar RBFN, yang terdiri dari lapisan input, lapisan tersembunyi berbasis Gaussian, dan lapisan output linear. Lapisan tersembunyi pada RBFN menggunakan fungsi aktivasi radial, yang

paling umum berbentuk fungsi Gaussian. Setiap neuron pada lapisan ini memiliki pusat (center) tertentu yang merepresentasikan suatu wilayah dalam ruang data masukan. Respon neuron akan semakin besar apabila vektor masukan berada dekat dengan pusat neuron tersebut. Fungsi aktivasi radial dinyatakan dalam persamaan berikut[27],

$$G_j(x) = \exp\left(-\frac{\|x-c_j\|^2}{2\sigma^2}\right) \quad (2.13)$$

dengan,

- x adalah vektor masukan,
- c_j adalah pusat neuron radial ke- j ,
- σ^2 adalah parameter sebaran (*spread*) yang mengatur lebar fungsi Gaussian.

Untuk menjaga kestabilan skala keluaran antar neuron radial, keluaran setiap neuron pada lapisan tersembunyi kemudian dinormalisasi terhadap jumlah total keluaran seluruh neuron radial pada lapisan tersebut, sebagaimana dinyatakan pada persamaan berikut,

$$\hat{G}_j(x) = \frac{G_j(x)}{\sum_{k=1}^M G_k(x)} \quad (2.14)$$

Normalisasi ini menghasilkan bentuk RBFN ternormalisasi (normalized RBFN), di mana total kontribusi seluruh neuron radial terhadap suatu data masukan bernilai konsisten, sehingga proses kombinasi linear pada lapisan keluaran menjadi lebih stabil, terutama ketika pusat neuron radial memiliki kerapatan yang tidak merata pada ruang data masukan.

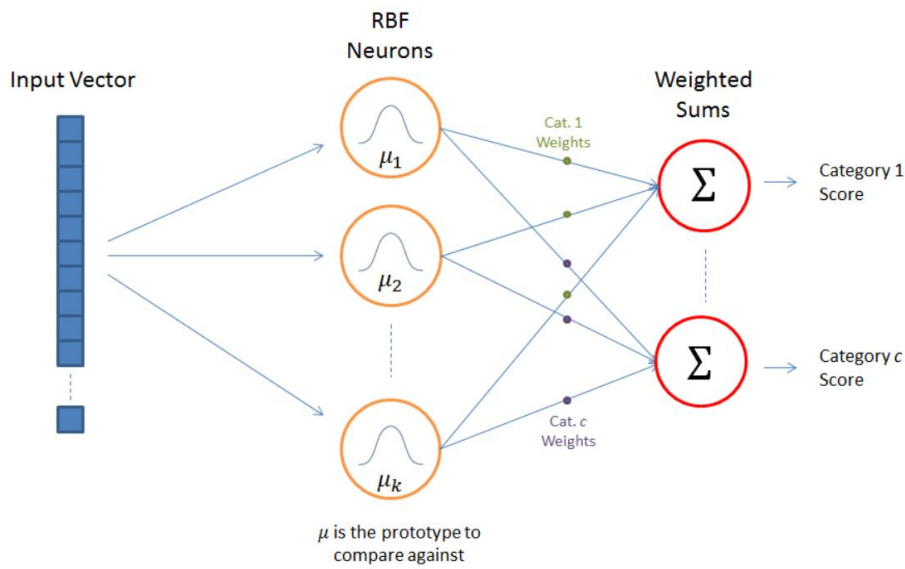
Pemilihan pusat radial dan nilai parameter sebaran sangat memengaruhi kemampuan jaringan dalam mempelajari pola data. Pusat radial umumnya diperoleh dari data pelatihan menggunakan pendekatan clustering atau pemilihan sampel representatif, sehingga setiap neuron radial mewakili kondisi operasi tertentu dari sistem. Lapisan keluaran pada RBFN bersifat linier dan berfungsi mengombinasikan keluaran neuron-neuron radial yang telah dinormalisasi.

Keluaran jaringan dirumuskan sebagai kombinasi linier dari seluruh fungsi aktivasi radial ternormalisasi, yang dapat dilihat pada persamaan berikut,

$$y(x) = \sum_{j=1}^M w_j \hat{G}_j(x) \quad (2.15)$$

dengan,

- w_j adalah bobot koneksi neuron ke- j ,
- M adalah jumlah neuron pada lapisan tersembunyi.



Gambar 2. 2 Arsitektur Radial Basis Function Network

Proses pelatihan RBFN pada penelitian ini dilakukan melalui dua tahap. Tahap pertama, nilai bobot keluaran awal ditentukan secara analitik menggunakan pendekatan regularized least squares (ridge regression atau regularisasi Tikhonov). Pendekatan ini bertujuan untuk memperoleh solusi bobot awal yang stabil secara numerik sekaligus mengurangi risiko overfitting akibat kolinearitas antar neuron radial. Bobot awal dirumuskan sebagai,

$$W_0 = (G^T G + \lambda I)^{-1} G^T t \quad (2.16)$$

dengan,

- G adalah matriks keluaran lapisan tersembunyi (nilai $\hat{G}_j(x)$ untuk seluruh data pelatihan),
- t adalah vektor nilai target,
- λ adalah parameter regularisasi yang mengendalikan tingkat penalti terhadap besar bobot,
- I adalah matriks identitas.

Tahap kedua, bobot keluaran yang diperoleh dari tahap pertama disempurnakan secara iteratif menggunakan algoritma Least Mean Square (LMS), untuk lebih meminimalkan selisih antara keluaran jaringan dan nilai target pada setiap data pelatihan. Error pada iterasi ke- m dinyatakan sebagai,

$$e(m) = y(m) - t(m) \quad (2.17)$$

dengan $t(m)$ merupakan nilai target pada iterasi ke- m . Bobot jaringan kemudian diperbarui menggunakan persamaan berikut,

$$w_j(m+1) = w_j(m) - \eta \hat{G}_j(x(m))e(m) \quad (2.18)$$

di mana η adalah learning rate yang mengatur laju konvergensi proses pelatihan [12]. Kombinasi kedua tahap pelatihan ini bertujuan untuk mempercepat konvergensi model sekaligus menjaga kestabilan numerik proses pembelajaran, karena inisialisasi bobot awal secara analitik pada Persamaan (2.15) dapat mengurangi jumlah iterasi LMS yang dibutuhkan untuk mencapai kondisi konvergen dibandingkan apabila bobot dimulai dari nilai acak.

Keunggulan utama RBFN terletak pada kemampuannya dalam mempelajari hubungan nonlinier secara cepat dan stabil, serta kebutuhan waktu pelatihan yang relatif singkat. Oleh karena itu, RBFN banyak diterapkan dalam berbagai studi sistem tenaga listrik, khususnya untuk estimasi dan prediksi parameter sistem yang memerlukan respons cepat dan akurasi yang memadai [31]

2.6 Hipotesis Penelitian

Dilandasi oleh penelitian sebelumnya [7] dan [14] yang membahas penilaian keamanan statis sistem tenaga listrik menggunakan metode konvensional dan pendekatan berbasis *Artificial Neural Network*, penelitian ini melakukan studi komparatif antara metode *Extreme Gradient Boosting* (XGBoost) dan *Radial Basis Function Network* (RBFN) dalam perankingan kontingensi N-1 . pada sistem tenaga listrik.

XGBoost merupakan metode *ensemble learning* berbasis *gradient boosting*, sedangkan RBFN merupakan jaringan saraf tiruan berbasis fungsi basis radial. Perbedaan mekanisme pembelajaran kedua metode tersebut diperkirakan menghasilkan perbedaan pada akurasi prediksi indeks keamanan, kualitas perankingan kontingensi, dan kompleksitas implementasi model.

Pengujian hipotesis dilakukan melalui studi komparatif dengan mengevaluasi kinerja kedua metode berdasarkan tiga aspek utama, yaitu,

1. Akurasi prediksi indeks keamanan statis, yang diukur menggunakan nilai koefisien determinasi (R^2), Root Mean Square Error (RMSE), dan Mean Absolute Error (MAE) terhadap nilai Performance Index of Power (PIP) dan Performance Index of Voltage (PIV) hasil perhitungan *Newton Raphson Load Flow* (NRLF).
2. Kualitas perankingan kontingensi N-1, yang dievaluasi menggunakan Capture Ratio, yaitu ukuran yang menunjukkan kemampuan metode dalam menangkap kontingensi paling kritis dibandingkan hasil perankingan referensi yang diperoleh melalui perhitungan NRLF.
3. Kompleksitas implementasi model, yang dievaluasi berdasarkan tahapan pembentukan model serta kebutuhan mekanisme tambahan (seperti penentuan center, regularisasi, dan proses ensemble) pada masing-masing metode.

Berdasarkan karakteristik metode *Extreme Gradient Boosting* (XGBoost) dan *Radial Basis Function Network* (RBFN), diduga terdapat perbedaan kemampuan kedua metode dalam memprediksi nilai Performance Index of Power (PIP), Performance Index of Voltage (PIV), dan perankingan kontingensi N-1 pada sistem tenaga listrik.

BAB III

METODE PENELITIAN

3.1 Metode Pengumpulan Data

Metode pengumpulan data pada penelitian ini dilakukan melalui dua pendekatan utama, yaitu studi literatur dan simulasi numerik/eksperimental.

3.1.1 Studi Literatur

Studi literatur dilakukan dengan menelaah berbagai jurnal ilmiah, buku teks, prosiding, serta penelitian terdahulu yang berkaitan dengan sistem tenaga listrik, static security assessment, *contingency* analysis, metode aliran daya *Newton Raphson Load Flow* (NRLF), serta metode machine learning, khususnya *Radial Basis Function Network* (RBFN) dan XGBoost.

Hasil studi literatur digunakan sebagai dasar teoritis dalam pemodelan sistem tenaga listrik, perumusan indeks keamanan statis, serta perancangan dan pemilihan metode pembelajaran mesin yang digunakan dalam penelitian ini.

3.1.2 Simulasi Numerik

Data numerik diperoleh melalui simulasi sistem uji IEEE 30-bus menggunakan perangkat lunak MATLAB. Simulasi dilakukan dengan menjalankan metode NRLF pada kondisi normal dan pada setiap skenario kontingensi N-1 berupa pemadaman satu saluran transmisi.

Nilai *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV) dihitung untuk berbagai variasi tingkat pembebanan sistem dari 50% hingga 100%. Data kondisi operasi sistem digunakan sebagai variabel masukan, sedangkan nilai PIP dan PIV hasil perhitungan NRLF digunakan sebagai variabel keluaran untuk membentuk dataset pelatihan dan pengujian model RBFN dan XGBoost. Proses pelatihan dan pengujian model machine learning dilakukan pada lingkungan Google Colab berbasis Python.

3.2 Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan setelah tahap penyusunan proposal dan berlangsung hingga seluruh rangkaian kegiatan penelitian selesai. Proses penelitian meliputi studi literatur, pemodelan sistem tenaga listrik, simulasi aliran daya, perancangan model machine learning, analisis hasil, serta penyusunan laporan akhir. Penelitian dilaksanakan di Program Studi Teknik Elektro Universitas Bengkulu, dengan Laboratorium Teknik Elektro sebagai lokasi utama pelaksanaan penelitian.

3.3 Alat, Perangkat Lunak, dan Data Penelitian

3.3.1 Alat

Adapun alat yang penulis gunakan dalam penelitian kali ini dapat dilihat pada tabel 3.1

Tabel 3. 1 Alat

No	Alat	Jumlah	Fungsi
1	Laptop	1	Menyediakan kapasitas komputasi untuk simulasi dan pelatihan model RBFN dan XGBOOST.
2	Koneksi Internet	1	Mendukung penggunaan Google Colab dan akses literatur daring

3.3.2 Perangkat Lunak

Adapun Perangkat lunak yang penulis gunakan dalam penelitian kali ini dapat dilihat pada tabel 3.2

Tabel 3. 2 Perangkat Lunak Penelitian

No	Perangkat Lunak	Fungsi
1	MATLAB	Digunakan untuk menyusun data sistem IEEE 30-bus, membentuk matriks admitansi bus (<i>Ybus</i>), menjalankan metode <i>Newton Raphson Load Flow</i> (NRLF), mensimulasikan skenario kontingensi N-1, serta menghitung indeks keamanan statis berupa <i>Active Power</i>

		<i>Performance Index</i> (PIP) dan <i>Voltage Performance Index</i> (PIV).
2	Google Colab (Python)	Digunakan untuk membangun, melatih, dan menguji model <i>Radial Basis Function Network</i> (RBFN) dan XGBoost, mengimpor dataset hasil simulasi NRLF, serta melakukan evaluasi dan visualisasi hasil prediksi nilai PIP dan PIV terhadap nilai referensi.

3.3.3 Data Penelitian

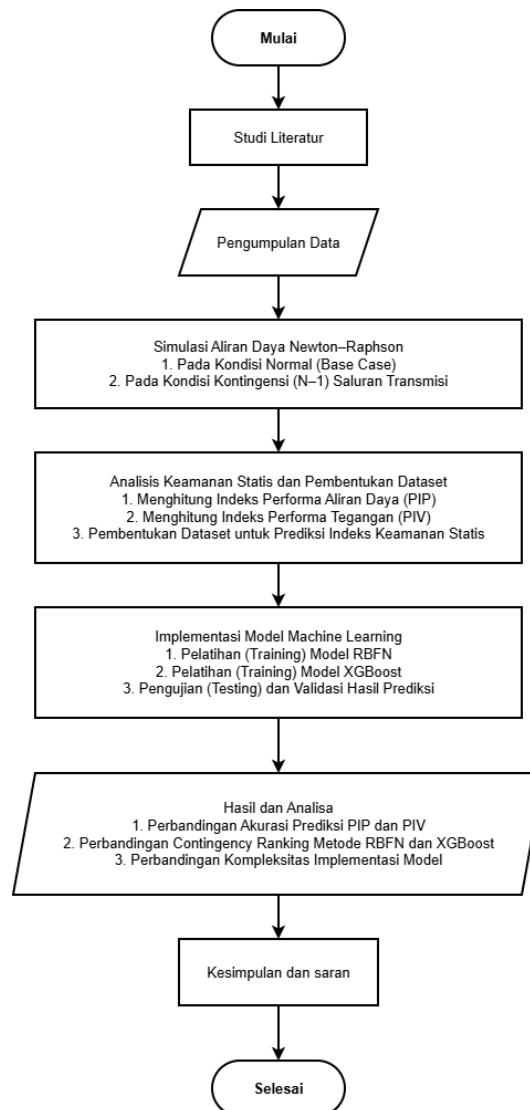
Adapun Data yang penulis gunakan dalam penelitian kali ini dapat dilihat pada tabel 3.3

Tabel 3. 3 Data Penelitian

No	Bahan/Data	Fungsi
1	Data sistem IEEE 30-bus (data bus dan saluran transmisi)	Digunakan sebagai model sistem tenaga listrik acuan untuk simulasi aliran daya dan analisis kontingensi N-1 .
2	Hasil perhitungan NRLF (nilai PIP dan PIV)	Digunakan sebagai <i>ground truth</i> dalam penilaian keamanan statis sistem tenaga listrik, perankingan kontingensi, serta sebagai target keluaran (<i>output</i>) pada proses pelatihan dan pengujian model RBFN dan XGBoost.
3	Dataset input-output RBFN dan XGBoost (format CSV)	Berisi pasangan variabel masukan berupa daya aktif dan reaktif beban (PL, QL), daya aktif dan reaktif pembangkit (PG, QG), magnitudo tegangan bus ($\ V\ $), serta indeks kontingensi (K_i), dan variabel keluaran berupa nilai PIP dan PIV yang digunakan dalam proses pelatihan dan pengujian model machine learning.

3.4 Alur Penelitian

Tahapan penelitian dalam menyelesaikan skripsi ini digambarkan dalam diagram alir pada Gambar 3.1. Diagram alir tersebut menunjukkan urutan proses penelitian secara sistematis, mulai dari tahap awal hingga tahap akhir.



Gambar 3. 1 Diagram Alir Penelitian

Diagram alir penelitian di atas menunjukkan tahapan pelaksanaan penelitian yang disusun secara sistematis dan runtut, dimulai dari tahap awal hingga tahap akhir sesuai dengan tujuan penelitian. Adapun tahapan penelitian tersebut adalah sebagai berikut,

1. Studi Literatur

Tahap awal penelitian diawali dengan studi literatur, yaitu mempelajari dan mengkaji berbagai buku teks, jurnal ilmiah, serta penelitian terdahulu yang berkaitan dengan sistem tenaga listrik, *static security assessment*, *contingency analysis*, metode aliran daya Newton-Raphson, serta metode *machine learning Radial Basis Function Network* (RBFN) dan XGBoost. Tahap ini bertujuan untuk memperoleh dasar teoritis dan metodologis sebagai landasan dalam perancangan dan pelaksanaan penelitian.

2. Pengumpulan Data

Tahap selanjutnya adalah pengumpulan data, di mana data yang digunakan berupa data sistem IEEE 30-bus yang meliputi data bus, data saluran transmisi, data pembangkit, dan data beban. Data tersebut digunakan sebagai dasar dalam pemodelan sistem tenaga listrik yang akan dianalisis pada penelitian ini.

3. Simulasi Aliran Daya

Setelah data sistem diperoleh, dilakukan simulasi aliran daya menggunakan metode Newton-Raphson. Simulasi dilakukan pada dua kondisi operasi sistem, yaitu kondisi normal (*base case*) dan kondisi kontingensi N-1 berupa pemadaman satu saluran transmisi secara bergantian. Simulasi ini bertujuan untuk memperoleh kondisi operasi sistem tenaga listrik pasca-kontingensi sebagai dasar dalam penilaian keamanan statis.

4. Analisis Keamanan Statis

Berdasarkan hasil simulasi aliran daya, dilakukan analisis keamanan statis sistem tenaga listrik dengan menghitung *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV). Nilai PIP dan PIV digunakan sebagai parameter kuantitatif untuk menggambarkan tingkat keparahan setiap kondisi kontingensi serta sebagai output referensi (*ground truth*) dalam penelitian ini.

5. Pembentukan Dataset

Berdasarkan nilai PIP dan PIV yang diperoleh, dilakukan pembentukan dataset yang terdiri dari variabel masukan sistem tenaga listrik dan variabel keluaran berupa indeks keamanan statis (PIP dan PIV).

6. Implementasi Model Machine Learning

Dataset yang telah dibentuk digunakan dalam implementasi dua metode *machine learning*, yaitu *Radial Basis Function Network* (RBFN) dan XGBoost. Pada tahap ini dilakukan proses pelatihan (*training*) dan pengujian (*testing*) untuk mengevaluasi kemampuan masing-masing model dalam memprediksi nilai PIP dan PIV pada kondisi sistem yang belum pernah dilihat sebelumnya.

7. Analisis Hasil

Tahap selanjutnya adalah analisis hasil, yang meliputi perbandingan akurasi prediksi nilai PIP dan PIV menggunakan metrik evaluasi, perbandingan hasil *contingency ranking* yang dihasilkan oleh metode RBFN dan XGBoost terhadap hasil *Newton Raphson Load Flow* (NRLF), serta perbandingan kompleksitas implementasi kedua model dalam penilaian keamanan statis sistem tenaga listrik secara *online*. Analisis tersebut digunakan untuk mengevaluasi keunggulan dan keterbatasan masing-masing metode berdasarkan hasil pengujian yang telah dilakukan.

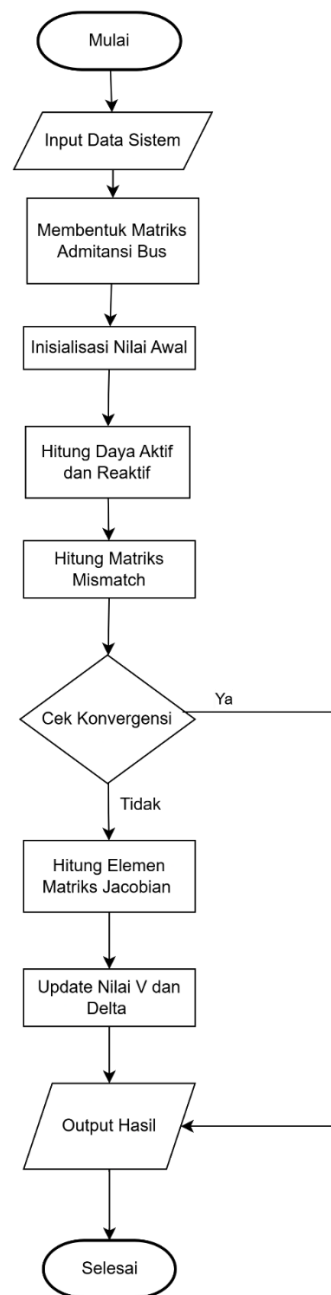
8. Penarikan Kesimpulan dan Saran

Tahap terakhir adalah penarikan kesimpulan dan pemberian saran berdasarkan hasil analisis yang telah diperoleh. Kesimpulan yang dihasilkan mencerminkan pencapaian tujuan penelitian, sedangkan saran diberikan sebagai masukan untuk pengembangan penelitian selanjutnya. Setelah seluruh tahapan tersebut dilaksanakan, maka penelitian dinyatakan selesai.

3.5 Simulasi Aliran Daya Metode *Newton Raphson*

Flowchart pada Gambar 3.2 menunjukkan alur perhitungan aliran daya menggunakan metode *Newton Raphson Load Flow*. Simulasi aliran daya dilakukan menggunakan metode *Newton Raphson* untuk memperoleh kondisi tegangan, sudut fasa, serta aliran daya pada setiap bus dan saluran. Proses perhitungan diawali dengan memasukkan data sistem dan membentuk matriks admitansi bus. Selanjutnya dilakukan inisialisasi nilai awal tegangan dan sudut fasa. Daya aktif dan reaktif dihitung untuk memperoleh nilai mismatch daya.

Perhitungan mismatch daya aktif dan reaktif dilakukan secara iteratif. Apabila nilai mismatch belum memenuhi kriteria konvergensi, maka matriks Jacobian disusun untuk memperbarui nilai tegangan dan sudut fasa. Proses ini dilakukan secara berulang hingga diperoleh hasil yang konvergen. Nilai tegangan dan aliran daya yang dihasilkan digunakan sebagai dasar analisis keamanan statis sistem tenaga listrik.



Gambar 3. 2 Flowchart Aliran Daya Metode *Newton Raphson*

3.5.1 Prosedur Simulasi Kontingensi N-1 pada Sistem IEEE 30-Bus

Penelitian ini tidak memodelkan proses transien terjadinya gangguan (fault) secara elektromagnetik, seperti bagaimana arus hubung singkat mengalir pada saat gangguan berlangsung. Yang disimulasikan adalah kondisi tunak (*steady-state*) sistem setelah satu elemen jaringan keluar dari operasi akibat kerja sistem proteksi. Pendekatan ini merepresentasikan skenario ketika suatu gangguan pada saluran transmisi sebagaimana dijelaskan jenis-jenisnya pada Subbab 2.2.2.3 telah berhasil diisolasi oleh pemutus tenaga, sehingga saluran tersebut lepas sepenuhnya dari sistem, dan yang dianalisis selanjutnya adalah dampak dari hilangnya saluran tersebut terhadap aliran daya dan profil tegangan pada sistem yang tersisa.

Secara konseptual, hilangnya suatu saluran transmisi pada skenario kontingensi N-1 dapat disebabkan oleh berbagai jenis gangguan, mulai dari hubung singkat satu fasa ke tanah, hubung singkat antar fasa, hingga gangguan tiga fasa simetris. Terlepas dari jenis gangguan awal yang terjadi, skema proteksi pada sistem transmisi bekerja dengan prinsip three-phase tripping, yaitu memutus ketiga fasa saluran secara bersamaan untuk menjaga sistem tetap seimbang setelah gangguan diisolasi, sebagaimana telah dijelaskan pada Subbab 2.2.2.4. Oleh karena itu, dalam representasi sistem tiga fasa seimbang yang digunakan pada penelitian ini dan digambarkan dalam bentuk single-line diagram, pelepasan satu saluran pada simulasi kontingensi N-1 merepresentasikan hilangnya seluruh sirkuit tiga fasa saluran tersebut, bukan hanya salah satu fasanya.

Sistem uji IEEE 30-Bus yang digunakan pada penelitian ini terdiri atas 30 bus dan 41 saluran transmisi (branch), sebagaimana ditunjukkan pada Tabel 3.5. Simulasi kontingensi N-1 dilakukan dengan cara melepas satu saluran transmisi secara bergantian, satu per satu, dari keseluruhan 41 saluran tersebut. Setiap kali satu saluran dilepas, saluran tersebut dihilangkan sepenuhnya dari matriks admitansi bus (Y_{bus}) yaitu baris dan kolom yang merepresentasikan koneksi antar dua bus terkait dihapus dari perhitungan, bukan sekadar diberi nilai impedansi yang sangat besar. Selanjutnya, perhitungan aliran daya menggunakan metode *Newton Raphson Load Flow* (NRLF) dijalankan ulang dari awal pada topologi jaringan yang baru tersebut untuk memperoleh nilai tegangan bus dan aliran daya saluran pasca-kontingensi.

Prosedur tersebut diulang secara berurutan untuk seluruh saluran transmisi yang ada, sehingga diperoleh 41 skenario kontingensi N-1 pada setiap tingkat pembebanan sistem yang diuji. Namun demikian, sebagaimana disebutkan pada Subbab 3.11.3, hanya 37 dari 41 skenario yang digunakan sebagai dasar perhitungan indeks keamanan statis dan pembentukan dataset penelitian, karena empat saluran sisanya menghasilkan kondisi topologi yang menyebabkan sistem terpecah (island) atau tidak relevan untuk dianalisis lebih lanjut dalam konteks penilaian keamanan statis pada penelitian ini.

Dari hasil NRLF pada setiap kondisi kontingensi, dihitung nilai Active Power Performance Index (PIP) dan Voltage Performance Index (PIV) sesuai dengan Persamaan (2.4) dan Persamaan (2.5) untuk menilai tingkat keparahan masing-masing skenario kontingensi terhadap keamanan statis sistem. Dengan demikian, alur simulasi kontingensi N-1 pada penelitian ini secara ringkas diawali dengan pelepasan satu saluran transmisi dari matriks admitansi bus, dilanjutkan dengan perhitungan ulang aliran daya menggunakan metode NRLF pada topologi baru, verifikasi konvergensi, perhitungan nilai PIP dan PIV berdasarkan hasil aliran daya pasca-kontingensi, serta diakhiri dengan penyimpanan hasil sebagai satu sampel data dalam dataset penelitian.

Penelitian ini berfokus pada dampak perubahan topologi yaitu hilangnya satu elemen jaringan terhadap keamanan statis sistem, sehingga tidak mencakup analisis rinci mengenai jenis gangguan penyebabnya maupun perhitungan arus hubung singkat (short circuit analysis), karena keduanya memerlukan pendekatan komponen simetris yang berada di luar ruang lingkup penelitian ini, sebagaimana telah dijelaskan pada Subbab 2.2.2.4.

3.6 Analisis Keamanan Statis Sistem Tenaga Listrik

Analisis keamanan statis sistem tenaga listrik pada penelitian ini dilakukan berdasarkan hasil simulasi aliran daya menggunakan metode *Newton Raphson*. Penilaian tingkat keparahan setiap kondisi kontingensi N-1 dilakukan dengan menggunakan *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV), yang masing-masing dirumuskan pada Persamaan (2.4) dan Persamaan (2.5).

Nilai PIP digunakan untuk menggambarkan tingkat pelanggaran aliran daya aktif pada saluran transmisi, sedangkan nilai PIV digunakan untuk menggambarkan tingkat pelanggaran batas tegangan pada bus. Perhitungan kedua indeks tersebut dilakukan dengan memanfaatkan hasil aliran daya pasca-kontingensi yang diperoleh dari simulasi Newton Raphson.

Nilai PIP dan PIV yang semakin besar menunjukkan tingkat keparahan kontingensi yang semakin tinggi. Nilai indeks ini selanjutnya digunakan sebagai parameter kuantitatif penilaian keamanan statis sistem tenaga listrik, serta sebagai nilai acuan (*ground truth*) dalam pembentukan dataset dan evaluasi hasil prediksi model RBFN dan XGBoost.

3.7 Pembentukan dan Pembagian Dataset

Dataset penelitian dibentuk berdasarkan hasil simulasi aliran daya dan perhitungan indeks keamanan statis yang dilakukan menggunakan perangkat lunak MATLAB. Simulasi aliran daya dijalankan pada sistem uji IEEE 30-Bus dengan memvariasikan tingkat pembebanan sistem mulai dari 50% hingga 100% terhadap kondisi beban dasar. Pada setiap tingkat pembebanan dilakukan simulasi kondisi normal (*base case*) serta simulasi 37 skenario kontingensi N-1 berupa pemadaman satu saluran transmisi secara bergantian.

Setiap skenario simulasi menghasilkan satu sampel data yang terdiri atas parameter kondisi operasi sistem tenaga listrik. Variabel masukan (*input*) meliputi daya aktif dan reaktif beban (PL, QL), daya aktif dan reaktif pembangkit (PG, QG), magnitudo tegangan bus ($|V|$), serta indeks kontingensi (K_i). Variabel keluaran (*output*) berupa nilai *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV) yang dihitung berdasarkan hasil simulasi aliran daya menggunakan metode *Newton Raphson Load Flow* (NRLF).

Seluruh hasil simulasi menghasilkan 222 sampel, yang selanjutnya digunakan dalam proses pembelajaran mesin. Data dari seluruh tingkat pembebanan dimanfaatkan sebagai data pelatihan untuk membangun model *Optimized Ensemble* RBFN dan XGBoost. Untuk memperoleh dasar perbandingan yang konsisten antara kedua metode, evaluasi kinerja model pada penelitian ini dilakukan menggunakan 37 sampel kontingensi N-1 pada tingkat pembebanan 100%. Pemilihan kondisi pembebanan tertinggi dilakukan karena kondisi tersebut

merepresentasikan kondisi operasi sistem yang paling kritis sehingga lebih sesuai untuk mengevaluasi kemampuan model dalam memprediksi indeks keamanan statis dan melakukan perankingan kontingensi.

Meskipun pada tahap pengembangan model XGBoost digunakan stratified train-test split sebagai bagian dari proses pelatihan dan validasi model, hasil evaluasi yang digunakan dalam pembahasan komparatif pada penelitian ini mengacu pada 37 sampel pengujian pada tingkat pembebanan 100% yang sama dengan evaluasi *Optimized Ensemble* RBFN. Dengan demikian, seluruh perbandingan akurasi prediksi, perankingan kontingensi, dan analisis kemampuan generalisasi pada Bab IV dilakukan menggunakan dataset pengujian yang identik sehingga hasil yang diperoleh dapat dibandingkan secara langsung.

3.8 Perancangan dan Pelatihan Model RBFN

Subbab ini menjelaskan proses perancangan dan pelatihan *Radial Basis Function Network* (RBFN) yang digunakan dalam penelitian. Alur proses pelatihan RBFN ditunjukkan pada diagram alir pelatihan RBFN Pada Gambar 3.3.

Diagram alir pelatihan RBFN menunjukkan tahapan proses pembelajaran jaringan saraf tiruan *Radial Basis Function Network* (RBFN) secara iteratif, yang dapat dijelaskan sebagai berikut,

1. Inisialisasi Jaringan

Proses pelatihan diawali dengan inisialisasi bobot jaringan, penentuan parameter jaringan, serta penetapan jumlah iterasi maksimum. Tahap ini bertujuan untuk menentukan kondisi awal jaringan sebelum proses pembelajaran dimulai.

2. Perhitungan Jarak *Euclidean* dan Fungsi Basis Radial

Pada tahap ini dilakukan perhitungan jarak *Euclidean* antara data input dan pusat fungsi basis radial. Hasil perhitungan jarak tersebut digunakan untuk menghitung keluaran dari fungsi basis radial pada lapisan tersembunyi.

3. Pembaruan Bobot Jaringan

Selanjutnya dilakukan pembaruan bobot jaringan menggunakan algoritma Least Mean Square (LMS). Algoritma ini bertujuan untuk meminimalkan galat antara nilai keluaran jaringan dan nilai target pada data pelatihan.

4. Perhitungan Keluaran Jaringan

Setelah bobot diperbarui, jaringan digunakan untuk menghitung nilai keluaran terestimasi berupa indeks kinerja untuk seluruh pola data pelatihan.

5. Perhitungan Galat (MSE)

Nilai keluaran jaringan dibandingkan dengan nilai target untuk menghitung galat kuadrat rata-rata (*Mean Square Error / MSE*) sebagai indikator kinerja proses pelatihan.

6. Pemeriksaan Kriteria Penghentian

Apabila nilai MSE yang diperoleh masih lebih besar dari nilai ambang batas kesalahan (ϵ), maka parameter jaringan diperbarui dan proses pelatihan diulang kembali mulai dari tahap perhitungan jarak *Euclidean*.

7. Penghentian Proses Pelatihan

Proses iterasi dihentikan apabila nilai MSE telah lebih kecil dari ϵ atau jumlah iterasi maksimum telah tercapai. Model RBFN yang telah terbentuk selanjutnya digunakan pada tahap pengujian dan analisis hasil penelitian.

3.9 Perancangan dan Pelatihan Model XGBoost

Metode XGBoost digunakan sebagai pendekatan machine learning berbasis ensemble decision tree untuk memprediksi nilai PIP dan PIV. Proses pelatihan diawali dengan penentuan rentang hyperparameter, seperti jumlah pohon, *learning rate*, dan kedalaman maksimum pohon. Pencarian kombinasi hyperparameter optimal dilakukan menggunakan metode grid search. Model dilatih menggunakan data pelatihan dan dievaluasi menggunakan data pengujian berdasarkan nilai error.

Diagram alir pelatihan XGBoost pada Gambar 3.4 menunjukkan tahapan proses pembelajaran model *Extreme Gradient Boosting* secara sistematis, yang dapat dijelaskan sebagai berikut,

1. Penentuan Rentang Hyperparameter

Proses pelatihan diawali dengan menentukan rentang nilai beberapa hyperparameter utama, seperti jumlah pohon keputusan, laju pembelajaran, kedalaman maksimum pohon, serta parameter regularisasi. Tahap ini

bertujuan untuk membatasi ruang pencarian kombinasi parameter agar proses optimasi lebih terarah dan efisien.

2. Pembangunan Ruang Grid Search

Pada tahap ini dibangun ruang pencarian menggunakan metode grid search. Seluruh kombinasi hyperparameter dalam rentang yang telah ditentukan diuji secara sistematis untuk memperoleh konfigurasi model dengan kinerja terbaik berdasarkan metrik evaluasi.

3. Pembagian Dataset (Training dan Testing)

Dataset dibagi menjadi dua bagian, yaitu data pelatihan dan data pengujian. Data pelatihan digunakan untuk membangun model, sedangkan data pengujian digunakan untuk mengevaluasi kemampuan generalisasi model terhadap kondisi sistem yang belum pernah dilatih sebelumnya.

4. Pembangunan Model XGBoost

Model dibangun menggunakan data pelatihan dan kombinasi hyperparameter yang sedang diuji. Proses pembelajaran dilakukan secara iteratif melalui mekanisme boosting, di mana setiap pohon keputusan baru dibentuk untuk memperbaiki kesalahan prediksi dari model sebelumnya.

5. Evaluasi Model XGBoost

Setelah model dilatih, dilakukan evaluasi menggunakan metrik kesalahan seperti Mean Square Error, Root Mean Square Error, Mean Absolute Error, dan koefisien determinasi. Tahap ini bertujuan untuk mengukur tingkat akurasi prediksi terhadap nilai referensi hasil perhitungan Newton-Raphson Load Flow.

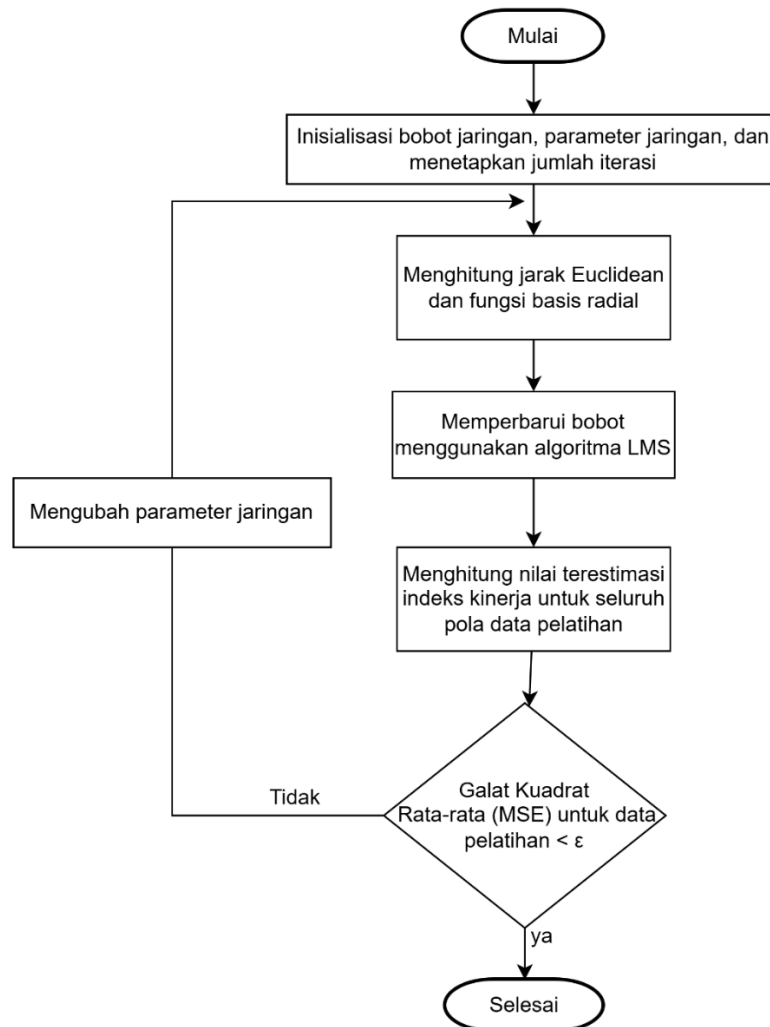
6. Pemeriksaan Kriteria Kinerja

Apabila nilai error yang diperoleh masih lebih besar dari ambang batas kesalahan (ϵ) atau belum memenuhi kriteria performa yang diinginkan, maka dilakukan penyesuaian kembali rentang hyperparameter dan proses grid search diulangi.

7. Penetapan Model Akhir XGBoost

Proses iterasi dihentikan apabila diperoleh kombinasi hyperparameter dengan nilai error minimum dan kinerja terbaik. Model tersebut ditetapkan sebagai model akhir XGBoost dan digunakan untuk memprediksi nilai

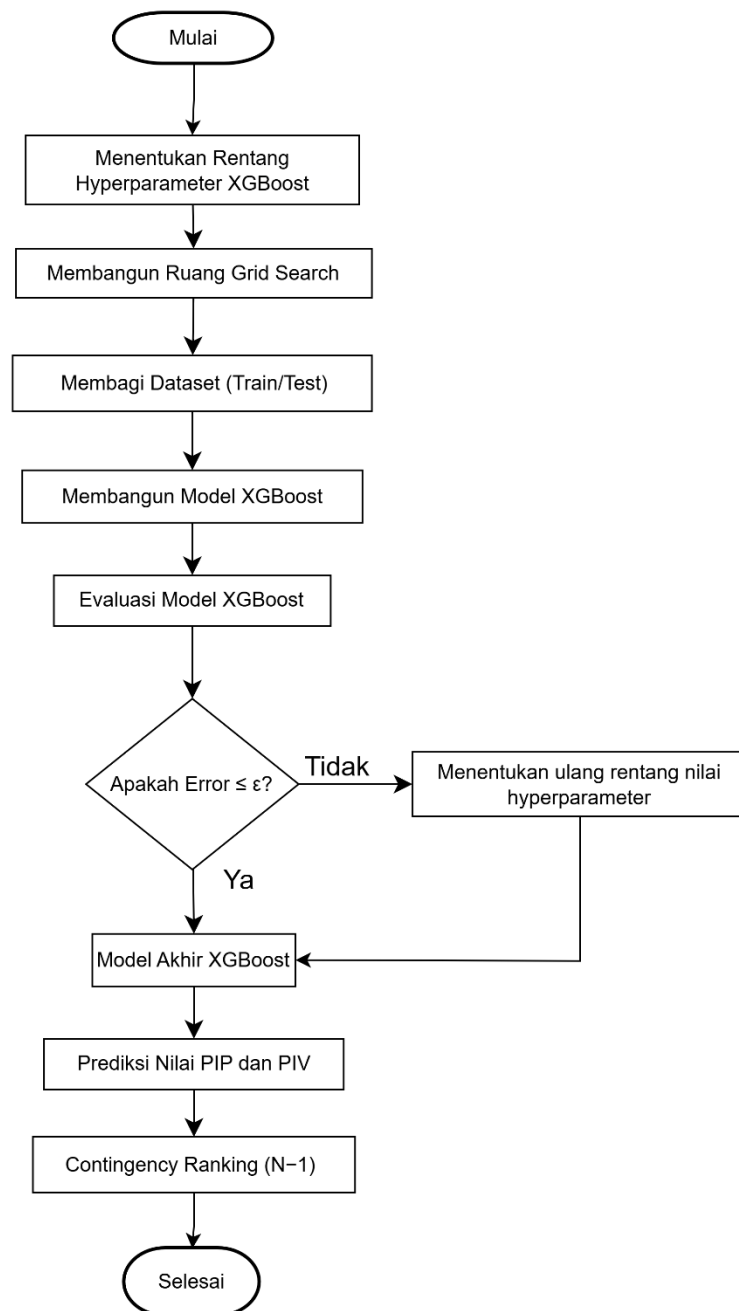
indeks keamanan statis pada seluruh skenario kontingensi N-1 serta melakukan proses perankingan kontingensi.



Gambar 3. 2 Flowchart Perancangan RBFN

3.10 Metode Evaluasi Kinerja Model

Evaluasi kinerja model *Radial Basis Function Network* (RBFN) dan XGBoost dilakukan untuk menilai kemampuan kedua metode dalam memprediksi nilai *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV). Evaluasi dilakukan dengan membandingkan nilai hasil prediksi model terhadap nilai referensi yang diperoleh dari perhitungan *Newton Raphson Load Flow* (NRLF).



Gambar 3. 3 Flowchart Perancangan Model XGBOOST

Evaluasi kuantitatif dilakukan menggunakan beberapa metrik kesalahan, yaitu *Mean Square Error* (MSE), *Root Mean Square Error* (RMSE), *Mean Absolute Error* (MAE), dan Koefisien Determinasi (R^2). *Mean Square Error* (MSE) digunakan untuk mengukur rata-rata kuadrat selisih antara nilai prediksi dan nilai referensi, yang dirumuskan sebagai berikut,

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

Root Mean Square Error (RMSE) merupakan akar dari nilai MSE dan digunakan untuk memberikan interpretasi kesalahan prediksi dalam satuan yang sama dengan nilai PIP dan PIV, yang dinyatakan sebagai,

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

Selain itu, *Mean Absolute Error* (MAE) digunakan untuk mengukur rata-rata kesalahan absolut antara nilai prediksi dan nilai referensi sehingga memberikan gambaran langsung mengenai besarnya deviasi hasil prediksi. Persamaan MAE dituliskan sebagai berikut,

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

Selain ketiga metrik tersebut, penelitian ini juga menggunakan *Mean Absolute Percentage Error* (MAPE) untuk mengukur rata-rata persentase kesalahan prediksi terhadap nilai referensi. MAPE memberikan ukuran kesalahan dalam bentuk persentase sehingga memudahkan interpretasi tingkat akurasi model. Persamaan MAPE dinyatakan sebagai berikut,

$$MAPE = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

Untuk melengkapi evaluasi kinerja model, digunakan pula koefisien determinasi (R^2) yang menunjukkan kemampuan model dalam menjelaskan variasi data referensi. Koefisien determinasi dinyatakan dengan persamaan,

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2}$$

dengan keterangan,

y_i adalah nilai referensi PIP atau PIV hasil NRLF,

$\hat{y}_i$ adalah nilai PIP atau PIV hasil prediksi model,

$\bar{y}$ adalah nilai rata-rata data referensi, dan

N adalah jumlah data pengujian.

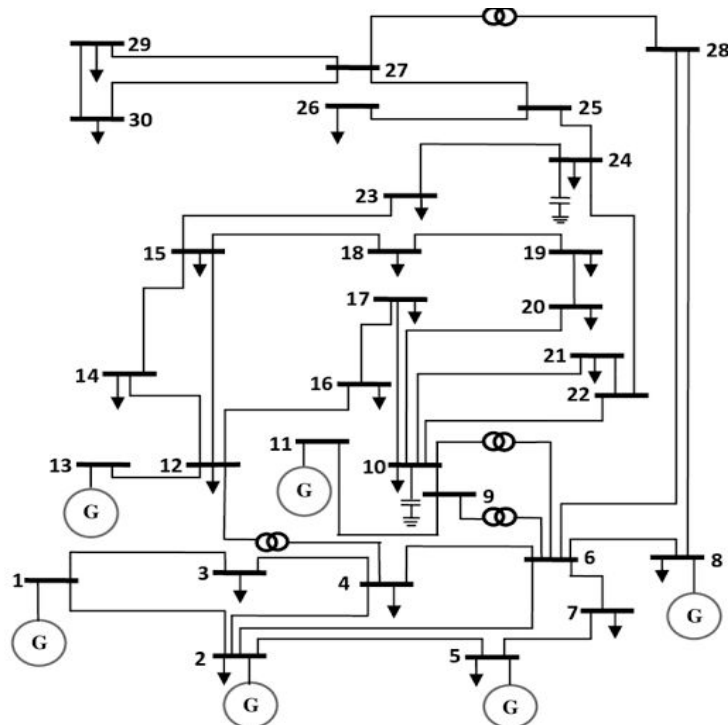
Nilai MSE, RMSE, MAE, dan MAPE yang semakin kecil serta nilai R^2 yang semakin mendekati satu menunjukkan bahwa model memiliki kemampuan prediksi yang semakin baik. Selain evaluasi akurasi prediksi, penelitian ini juga melakukan evaluasi terhadap hasil perankingan kontingensi N-1 yang dihasilkan oleh model RBFN dan XGBoost. Evaluasi dilakukan menggunakan *Capture Ratio* (CR) untuk mengukur kemampuan model dalam mengidentifikasi kontingensi kritis yang sesuai dengan hasil perankingan referensi berdasarkan *Newton Raphson Load Flow* (NRLF). Perhitungan *Capture Ratio* mengacu pada Persamaan (2.6) yang telah dijelaskan pada Bab II. Nilai *Capture Ratio* yang semakin tinggi menunjukkan bahwa model semakin mampu menghasilkan perankingan kontingensi yang konsisten dengan hasil referensi NRLF.

3.11 Data Pendukung

Pada penelitian ini, data pendukung yang digunakan adalah sistem tenaga listrik standar IEEE 30 Bus yang umum digunakan sebagai sistem uji dalam analisis aliran daya dan evaluasi keamanan sistem tenaga listrik. Sistem ini dipilih karena memiliki kompleksitas yang lebih tinggi dibandingkan IEEE 14 Bus serta mencerminkan karakteristik sistem tenaga skala menengah.

Sistem IEEE 30 Bus terdiri atas 30 bus, 41 saluran transmisi (branch), dan 6 unit generator yang terletak pada Bus 1, 2, 5, 8, 11, dan 13. Bus 1 berfungsi sebagai slack bus, sedangkan bus lainnya diklasifikasikan sebagai bus PV (generator bus) dan bus PQ (load bus). Sistem ini memiliki total beban aktif sebesar 283.4 MW dan total beban reaktif sebesar 126.2 MVAR. Beban pada sistem

diasumsikan konstan selama analisis aliran daya dilakukan. Visualisasi sistem dalam bentuk single line diagram IEEE 30 Bus dapat dilihat pada Gambar 3.4.



Gambar 3. 4 Single Line Diagram IEEE 30 Bus

3.11.1 Klasifikasi dan Parameter Bus

Pada analisis aliran daya, setiap bus pada sistem tenaga listrik memiliki empat parameter utama, yaitu tegangan (V), sudut fasa (θ), daya aktif (P), dan daya reaktif (Q). Berdasarkan parameter yang diketahui dalam perhitungan, bus diklasifikasikan menjadi tiga jenis.

Slack bus adalah bus referensi dengan nilai tegangan dan sudut fasa yang ditentukan. *PV bus* merupakan bus generator dengan daya aktif dan tegangan yang diketahui, sedangkan daya reaktif dan sudut fasa dihitung. *PQ bus* adalah bus beban dengan daya aktif dan daya reaktif yang diketahui, sementara tegangan dan sudut fasanya diperoleh melalui perhitungan *Newton Raphson Load Flow*. Klasifikasi dan parameter bus sistem IEEE 30 Bus ditunjukkan pada Tabel 3.4.

Tabel 3. 4 Data Bus IEEE 30 Bus

Bus	Type Bus	Pd (MW)	Qd (MVAR)	Vm (pu)	Va (°)
1	Slack	0	0	1,06	0
2	PV	21,7	12,7	1,043	-5,48
3	PQ	2,4	1,2	1,021	-7,96
4	PQ	7,6	1,6	1,012	-9,62
5	PV	94,2	19,0	1,01	-14,37
6	PQ	0	0	1,01	-11,34
7	PQ	22,8	10,9	1,002	-13,12
8	PV	30,0	30,0	1,01	-12,10
9	PQ	0	0	1,051	-14,38
10	PQ	5,8	2,0	1,045	-15,97
11	PV	0	0	1,082	-14,39
12	PQ	11,2	7,5	1,057	-15,24
13	PV	0	0	1,071	-15,24
14	PQ	6,2	1,6	1,042	-16,13
15	PQ	8,2	2,5	1,038	-16,22
16	PQ	3,5	1,8	1,045	-15,83
17	PQ	9,0	5,8	1,040	-16,14
18	PQ	3,2	0,9	1,028	-16,82
19	PQ	9,5	3,4	1,026	-17,00
20	PQ	2,2	0,7	1,030	-16,80

3.11.2 Data Saluran Transmisi (*Branch Data*)

Data saluran transmisi pada sistem IEEE 30 Bus meliputi parameter *resistansi* (R), *reaktansi* (X), dan *susceptansi* (B) dalam satuan per unit (pu). Parameter-parameter ini digunakan dalam pembentukan matriks admitansi sistem (Y-bus), yang menjadi dasar perhitungan aliran daya menggunakan metode *Newton Raphson Load Flow*. Nilai *resistansi* (R) dan *reaktansi* (X) merepresentasikan impedansi seri saluran transmisi, sedangkan nilai *susceptansi* (B) merepresentasikan efek kapasitif saluran (*line charging*). Dalam format MATPOWER, nilai B yang tercantum merupakan total *susceptansi* saluran. Sistem IEEE 30 Bus terdiri atas 41 saluran transmisi yang saling menghubungkan 30 bus dalam sistem. Tabel 3.5 berikut menunjukkan parameter beberapa branch sebagai representasi data saluran transmisi sistem IEEE 30 Bus.

Berdasarkan Tabel 3.5, sistem IEEE 30 Bus terdiri atas 41 saluran transmisi yang saling menghubungkan 30 bus dalam jaringan. Nilai *resistansi* (R) dan *reaktansi* (X) pada setiap branch menentukan impedansi saluran yang memengaruhi

distribusi daya aktif dan daya reaktif dalam sistem, sedangkan nilai susceptansi (B) merepresentasikan efek kapasitif saluran yang berkontribusi terhadap pembentukan matriks admitansi (Y_{bus}). Nilai PIP dihitung berdasarkan tingkat pembebanan saluran terhadap batas maksimumnya, sedangkan nilai PIV dihitung berdasarkan deviasi tegangan bus terhadap batas tegangan yang diizinkan

Tabel 3. 5 Data Branch IEEE 30 Bus

No	From Bus	To Bus	R (pu)	X (pu)	B (pu)
1	1	2	0,0192	0,0575	0,0528
2	1	3	0,0452	0,1652	0,0408
3	2	4	0,0570	0,1737	0,0368
4	3	4	0,0132	0,0379	0,0084
5	2	5	0,0472	0,1983	0,0418
6	2	6	0,0581	0,1763	0,0374
7	4	6	0,0119	0,0414	0,0090
8	5	7	0,0460	0,1160	0,0204
9	6	7	0,0267	0,0820	0,0170
10	6	8	0,0120	0,0420	0,0090
11	6	9	0,0000	0,2080	0,0000
12	6	10	0,0000	0,5560	0,0000
13	9	11	0,0000	0,2080	0,0000
14	9	10	0,0000	0,1100	0,0000
15	4	12	0,0000	0,2560	0,0000
16	12	13	0,0000	0,1400	0,0000
17	12	14	0,1231	0,2559	0,0000
18	12	15	0,0662	0,1304	0,0000
19	12	16	0,0945	0,1987	0,0000
20	14	15	0,2210	0,1997	0,0000
21	16	17	0,0524	0,1923	0,0000
22	15	18	0,1073	0,2185	0,0000
23	18	19	0,0639	0,1292	0,0000

No	From Bus	To Bus	R (pu)	X (pu)	B (pu)
24	19	20	0,0340	0,0680	0,0000
25	10	20	0,0936	0,2090	0,0000
26	10	17	0,0324	0,0845	0,0000
27	10	21	0,0348	0,0749	0,0000
28	10	22	0,0727	0,1499	0,0000
29	21	22	0,0116	0,0236	0,0000
30	15	23	0,1000	0,2020	0,0000
31	22	24	0,1150	0,1790	0,0000
32	23	24	0,1320	0,2700	0,0000
33	24	25	0,1885	0,3292	0,0000
34	25	26	0,2544	0,3800	0,0000
35	25	27	0,1093	0,2087	0,0000
36	28	27	0,0000	0,3960	0,0000
37	27	29	0,2198	0,4153	0,0000
38	27	30	0,3202	0,6027	0,0000
39	29	30	0,2399	0,4533	0,0000
40	8	28	0,0636	0,2000	0,0428
41	6	28	0,0169	0,0599	0,0130

3.11.3 Data Validasi Newthson Raphson Loadflow

Data validasi dalam penelitian ini diperoleh dari hasil perhitungan aliran daya menggunakan metode *Newton Raphson Load Flow* (NRLF). Metode ini digunakan untuk menentukan kondisi operasi sistem tenaga listrik secara akurat, meliputi tegangan bus dan aliran daya pada setiap saluran transmisi. Hasil perhitungan NRLF digunakan sebagai data acuan (*ground truth*) dalam penelitian ini. Nilai PIP dan PIV yang diperoleh dari NRLF selanjutnya digunakan sebagai target dalam proses pelatihan dan evaluasi model machine learning. Selain itu, hasil NRLF juga digunakan untuk melakukan perankingan kontingensi N-1, yang menjadi dasar dalam mengevaluasi tingkat keparahan gangguan pada sistem tenaga listrik.

Tabel 3. 6 Data Validasi Newthton Raphson Loadflow

Performance Index power		Performance Index Voltage	
Line-outage	PIP	Line-outage	PIV
L1-2	0,7646	L27-29	2,6823
L2-5	0,7448	L27-30	2,6548
L4-12	0,7409	L6-7	2,2505
L4-6	0,5459	L10-20	2,2259
L1-3	0,5418	L3-4	2,2135
L6-8	0,5412	L22-24	2,2097
L3-4	0,5363	L29-30	2,1834
L2-6	0,5165	L19-20	2,1778
L6-28	0,5029	L6-28	2,1621
L10-20	0,4896	L24-25	2,1493
L10-21	0,4892	L15-23	2,1321
L12-15	0,4851	L23-24	2,1250
L9-10	0,4805	L25-27	2,1088
L6-9	0,4772	L21-22	2,1078
L19-20	0,4680	L14-15	2,1039
L12-14	0,4669	L18-19	2,1024
L10-22	0,4605	L16-17	2,0985
L27-30	0,4599	L5-7	2,0604
L12-16	0,4585	L10-22	2,0559
L6-10	0,4565	L15-18	2,0527
L6-7	0,4531	L8-28	2,0519
L15-18	0,4517	L10-17	2,0471
L25-27	0,4493	L6-8	2,0212
L15-23	0,4484	L10-21	2,0189
L10-17	0,4480	L4-6	1,9580
L22-24	0,4474	L12-14	1,9341
L27-29	0,4474	L6-10	1,9144
L16-17	0,4426	L6-9	1,9086
L21-22	0,4401	L12-16	1,9077
L14-15	0,4397	L2-5	1,8739
L29-30	0,4397	L12-15	1,8700
L18-19	0,4386	L2-4	1,8661
L24-25	0,4382	L2-6	1,8646
L23-24	0,4381	L1-3	1,8293
L8-28	0,4372	L1-2	1,7405
L5-7	0,4315	L9-10	1,6579
L2-4	0,4255	L4-12	1,6146

BAB IV

HASIL DAN PEMBAHASAN

Penelitian ini disusun untuk menganalisis serta membandingkan unjuk kerja dua metode kecerdasan buatan, yaitu Radial Basis Function Network (RBFN) dan *Extreme Gradient Boosting* (XGBoost), dalam memprediksi indeks keamanan statis sistem tenaga listrik yang direpresentasikan oleh *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV). Kedua model dilatih menggunakan dataset yang dibangun dari hasil simulasi *Newton Raphson Load Flow* (NRLF) pada sistem uji IEEE 30-Bus, dengan mempertimbangkan seluruh skenario kontingensi N-1 pada enam tingkat pembebanan (50%-100%). Pemilihan rentang pembebanan tersebut dimaksudkan agar model yang dihasilkan mampu mengenali pola kritis sistem baik pada kondisi beban ringan maupun pada kondisi beban puncak, sehingga hasil prediksi yang diperoleh lebih representatif terhadap variasi operasi sistem tenaga listrik yang sesungguhnya. Nilai PIP dan PIV hasil perhitungan NRLF selanjutnya diperlakukan sebagai data acuan (ground truth) dalam mengevaluasi akurasi prediksi masing-masing model.

Evaluasi performa model dilakukan menggunakan empat parameter statistik, yaitu Mean Square Error (MSE), Mean Absolute Error (MAE), Root Mean Square Error (RMSE), dan koefisien determinasi (R^2). Keempat parameter ini dipilih karena saling melengkapi dalam menggambarkan kualitas prediksi, MSE dan RMSE peka terhadap galat besar sehingga dapat mengungkap apabila model gagal memprediksi skenario kontingensi yang ekstrem, MAE memberikan gambaran kesalahan rata-rata yang lebih tahan terhadap outlier, sedangkan R^2 menunjukkan seberapa besar variasi data referensi yang mampu dijelaskan oleh model. Dengan menggabungkan keempat indikator tersebut, perbandingan kinerja RBFN dan XGBoost dapat dilakukan secara lebih komprehensif dan tidak hanya bergantung pada satu ukuran kesalahan saja.

4.1 Hasil Simulasi Aliran Daya *Newton Raphson Load Flow* (NRLF)

Simulasi aliran daya pada sistem IEEE 30-Bus dilakukan terlebih dahulu untuk memperoleh kondisi operasi normal sistem (base case) sebelum analisis kontingensi N-1 dijalankan. Kondisi base case ini penting untuk ditetapkan lebih

awal karena seluruh perhitungan indeks keamanan statis pada tahap berikutnya bersifat relatif terhadap kondisi acuan tersebut, tanpa base case yang valid dan konvergen, nilai PIP dan PIV yang dihasilkan tidak akan merepresentasikan dampak kontingensi secara akurat.

Tabel 4. 1 Hasil Simulasi NRLF pada Kondisi *Base case*

Parameter	Nilai
Status simulasi	Converged
Jumlah iterasi	3
Total pembangkitan daya aktif	301,06 MW
Total pembebanan daya aktif	283,40 MW
Rugi-rugi daya aktif	17,66 MW
Total pembangkitan daya reaktif	134,89 MVar
Total pembebanan daya reaktif	126,20 MVar
Rugi-rugi daya reaktif	50,84 MVar

Berdasarkan Tabel 4.1, proses iterasi Newton Raphson mencapai kondisi *converged* hanya dalam tiga iterasi. Kecepatan konvergensi ini sesuai dengan karakteristik metode Newton Raphson yang memiliki laju konvergensi kuadratik apabila nilai tebakan awal (*flat start*) berada cukup dekat dengan solusi sebenarnya. Topologi sistem IEEE 30-Bus yang berbentuk jaring (*meshed*) dengan tingkat impedansi saluran yang relatif seimbang turut mendukung terbentuknya matriks Jacobian yang *well-conditioned*, sehingga proses iterasi dapat konvergen dengan cepat tanpa mengalami osilasi maupun divergensi numerik.

Total daya aktif yang dibangkitkan sebesar 301,06 MW, sedangkan total daya aktif yang diserap beban sebesar 283,40 MW, sehingga terdapat rugi-rugi daya aktif sebesar 17,66 MW atau setara dengan sekitar 6,2% dari total pembebanan. Besarnya rugi-rugi ini merupakan konsekuensi langsung dari daya yang harus dibangkitkan oleh *slack bus* (Bus 1) untuk menutup selisih antara total pembangkitan pada bus-bus PV dan total kebutuhan sistem, termasuk rugi-rugi itu sendiri. Rugi-rugi saluran timbul akibat arus yang mengalir berinteraksi dengan resistansi saluran, sehingga besarnya rugi-rugi tersebut meningkat tajam seiring

bertambahnya arus yang melewati saluran. Oleh karena itu, saluran-saluran yang memikul aliran daya besar dari Bus 1 secara alami menyumbang rugi-rugi terbesar, sebagaimana akan terlihat pada pembahasan Tabel 4.3.

Pada sisi daya reaktif, rugi-rugi yang terjadi jauh lebih besar secara proporsional, yaitu 50,84 MVar dari total kebutuhan 126,20 MVar. Hal ini wajar terjadi karena pada saluran transmisi tegangan tinggi, nilai reaktansi (X) jauh lebih dominan dibandingkan resistansi (R), sehingga rugi-rugi daya reaktif yang timbul akibat interaksi arus dengan reaktansi saluran cenderung lebih besar dibandingkan rugi-rugi daya aktif yang timbul akibat interaksi arus dengan resistansi saluran. Selain itu, sebagian daya reaktif yang dibangkitkan juga digunakan untuk menopang profil tegangan pada bus-bus beban yang letaknya jauh dari sumber pembangkitan. Selanjutnya, profil tegangan pada setiap bus hasil simulasi *base case* disajikan pada Tabel 4.2.

Tabel 4. 2 Profil Tegangan Setiap Bus pada Kondisi *Base case*

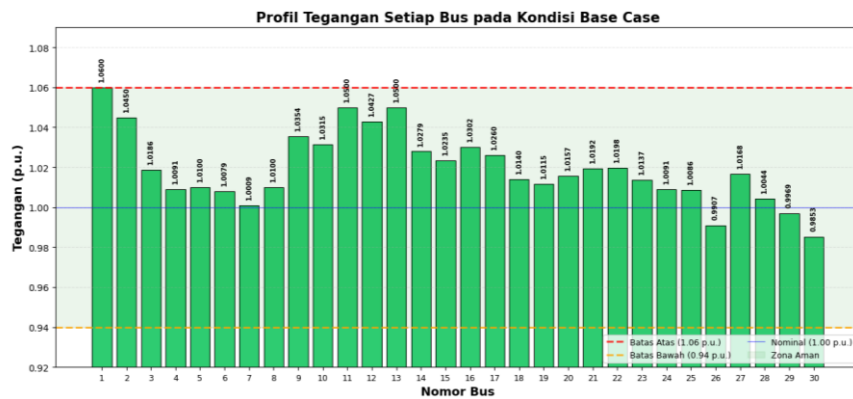
Bus No	Tegangan (p.u.)	Sudut (deg)
1	1,0600	0,00
2	1,0450	-5,39
3	1,0186	-7,49
4	1,0091	-9,24
5	1,0100	-14,18
6	1,0079	-11,04
7	1,0009	-12,85
8	1,0100	-11,83
9	1,0354	-14,12
10	1,0315	-15,74
11	1,0500	-14,12
12	1,0427	-14,96
13	1,0500	-14,96
14	1,0279	-15,88
15	1,0235	-15,98
16	1,0302	-15,57
17	1,0260	-15,91
18	1,0140	-16,61
19	1,0115	-16,79
20	1,0157	-16,59
21	1,0192	-16,20
22	1,0198	-16,18
23	1,0137	-16,38
24	1,0091	-16,57
25	1,0086	-16,16

Bus No	Tegangan (p.u.)	Sudut (deg)
26	0,9907	-16,59
27	1,0168	-15,64
28	1,0044	-11,67
29	0,9969	-16,89
30	0,9853	-17,78

Hasil pada Tabel 4.2 menunjukkan bahwa seluruh nilai tegangan berada pada rentang 0,9853-1,0600 p.u., sehingga masih memenuhi batas operasi normal 0,94-1,06 p.u. sesuai standar IEEE. Tegangan tertinggi terjadi pada Bus 1 sebesar 1,0600 p.u. dengan sudut fasa 0° , hal ini disebabkan Bus 1 berperan sebagai *slack bus* yang nilai magnitudo dan sudut tegangannya ditetapkan sebagai referensi sistem sekaligus penyeimbang daya. Sebaliknya, tegangan terendah terjadi pada Bus 30 sebesar 0,9853 p.u. Kondisi ini terjadi karena Bus 30 berada pada ujung jaringan yang paling jauh secara elektris dari sumber-sumber pembangkitan utama, sehingga akumulasi jatuh tegangan (*voltage drop*) di sepanjang saluran yang dilalui menjadi paling besar dibandingkan bus-bus lain.

Pola sudut fasa tegangan yang menurun secara monoton dari 0° pada Bus 1 hingga $-17,78^\circ$ pada Bus 30 sejalan dengan karakteristik aliran daya pada sistem transmisi, di mana daya aktif mengalir dari bus dengan sudut fasa tegangan yang lebih besar menuju bus dengan sudut fasa yang lebih kecil, dan besarnya aliran daya tersebut akan semakin besar apabila selisih sudut fasa antar-bus semakin besar atau reaktansi saluran penghubung semakin kecil. Penurunan sudut yang konsisten ini mengindikasikan bahwa daya aktif mengalir searah dari bus pembangkit menuju bus beban tanpa adanya pembalikan arah aliran yang tidak wajar, sehingga dapat disimpulkan bahwa sistem beroperasi pada kondisi yang stabil.

Pola pada Gambar 4.1 memperlihatkan penurunan tegangan yang lebih terlihat pada Bus 26, Bus 29, dan Bus 30. Ketiga bus tersebut secara topologi merupakan bus-bus yang berada pada percabangan akhir (ujung) jaringan sehingga hanya menerima pasokan daya melalui sejumlah kecil saluran dengan impedansi kumulatif yang besar. Karakteristik ini umum ditemukan pada sistem tenaga listrik radial-like di ujung jaringan meshed dan menjadi salah satu alasan mengapa bus-bus tersebut berulang kali muncul sebagai bus kritis pada hasil simulasi kontingensi, Data hasil simulasi tersebut ditunjukkan pada Tabel 4.3.



Gambar 4. 1 Profil Tegangan Setiap Bus pada Kondisi *Base case*

Berdasarkan Tabel 4.3, Aliran daya aktif terbesar terjadi pada saluran Bus 1-Bus 2 (173,57 MW) dan Bus 1-Bus 3 (87,50 MW). Hal ini terjadi karena Bus 1 merupakan slack bus yang menanggung seluruh selisih daya yang tidak dapat dipenuhi oleh generator pada bus-bus PV, sekaligus menutup rugi-rugi sistem, sehingga kedua saluran yang terhubung langsung dengannya secara alami memikul porsi aliran daya paling besar. Karena rugi-rugi daya sebanding dengan kuadrat arus (I^2R), kedua saluran tersebut juga tercatat memiliki rugi-rugi terbesar, yaitu 5,229 MW pada saluran 1-2 dan 3,107 MW pada saluran 1-3.

Tabel 4. 3 Aliran Daya pada Saluran Transmisi (*Base case*)

Dari	Ke	P (MW)	Q (MVar)	Loss P (MW)	Loss Q (MVar)
1	2	173,57	-24,76	5,229	9,810
1	3	87,50	5,94	3,107	6,948
2	4	43,65	6,66	1,034	0,733
3	4	81,99	-2,20	0,856	1,594
2	5	82,54	2,77	2,956	8,004
2	6	60,45	2,95	1,957	1,998
4	6	72,17	-17,09	0,641	1,315
5	7	-14,61	12,87	0,184	1,600
6	7	37,98	-4,11	0,382	0,542
6	8	29,69	-13,83	0,125	0,478
6	9	27,57	-1,67	0,000	1,494
6	10	15,84	2,27	0,000	1,315
9	11	0,00	-7,27	0,000	0,102
9	10	27,57	4,10	0,000	0,797

Dari	Ke	P (MW)	Q (MVar)	Loss P (MW)	Loss Q (MVar)
4	12	43,98	19,09	0,000	5,020
12	13	-0,00	-5,41	0,000	0,038
12	14	7,81	2,33	0,075	0,156
12	15	17,77	6,48	0,218	0,429
12	16	7,19	3,18	0,054	0,113
14	15	1,54	0,57	0,006	0,005
16	17	3,64	1,27	0,007	0,027
15	18	5,99	1,55	0,039	0,080
18	19	2,75	0,57	0,005	0,010
19	20	-6,75	-2,84	0,018	0,036
10	20	9,05	3,77	0,085	0,189
10	17	5,38	4,60	0,015	0,040
10	21	15,64	9,71	0,111	0,239
10	22	7,52	4,40	0,052	0,107
21	22	-1,97	-1,73	0,001	0,002
15	23	4,90	2,57	0,029	0,059
22	24	5,50	2,56	0,041	0,063
23	24	1,67	0,91	0,005	0,009
24	25	-1,58	1,08	0,007	0,012
25	26	3,55	2,37	0,045	0,068
25	27	-5,13	-1,30	0,030	0,057
28	27	18,44	6,09	0,000	1,387
27	29	6,19	1,67	0,087	0,165
27	30	7,09	1,67	0,164	0,310
29	30	3,70	0,61	0,034	0,064
8	28	-0,43	0,77	0,006	4,325
6	28	18,94	-0,12	0,060	1,104

Beberapa saluran seperti Bus 5-Bus 7, Bus 19-Bus 20, Bus 24-Bus 25, dan Bus 25-Bus 27 menunjukkan nilai daya aktif negatif. Nilai ini tidak menunjukkan kesalahan simulasi, melainkan menandakan bahwa arah aliran daya aktual berlawanan dengan arah penomoran 'dari-ke' yang ditetapkan pada data saluran, sehingga secara fisis daya justru mengalir dari bus tujuan menuju bus asal. Secara keseluruhan, tidak ditemukannya saluran dengan pembebanan berlebih maupun bus dengan tegangan di luar batas pada kondisi base case menegaskan bahwa hasil ini layak dijadikan kondisi acuan bagi analisis kontingensi N-1 pada subbab berikutnya.

4.2 Hasil Simulasi Kontingensi N-1

Analisis kontingensi N-1 dilakukan dengan melepaskan satu saluran transmisi secara bergiliran untuk mengetahui respons sistem terhadap kehilangan salah satu elemen jaringan. Simulasi ini diulang pada enam tingkat pembebanan (50%-100%) agar pengaruh tingkat pembebanan terhadap tingkat keparahan kontingensi dapat diamati secara konsisten.

Tabel 4. 4 Hasil Simulasi Kontingensi N-1

Level Beban	Jumlah Kontingensi	Berhasil Konvergen	Gagal Konvergen
50%	37	37	0
60%	37	37	0
70%	37	37	0
80%	37	37	0
90%	37	37	0
100%	37	37	0
Total	222	222	0

Berdasarkan Tabel 4.4, Seluruh 222 skenario kontingensi (37 saluran $\times$ 6 tingkat pembebanan) berhasil mencapai kondisi converged tanpa satu pun kegagalan. Hasil ini dapat dijelaskan oleh karakteristik topologi IEEE 30-Bus yang berbentuk jaringan meshed dengan redundansi saluran yang memadai, sehingga pelepasan satu saluran pada umumnya tidak menyebabkan bus menjadi terisolasi (islanding) maupun membuat matriks Jacobian menjadi singular. Sistem uji ini memang lazim digunakan dalam studi keamanan statis karena dirancang untuk tetap N-1 secure pada rentang pembebanan normal hingga puncak, sehingga seluruh skenario dapat digunakan tanpa perlu membuang data akibat kegagalan konvergensi..Ringkasan hasil simulasi pada kondisi pembebanan 100% disajikan pada Tabel 4.5.

Berdasarkan Tabel 4.5, Setiap kontingensi menghasilkan pola redistribusi daya yang berbeda-beda bergantung pada posisi saluran yang dilepas terhadap topologi jaringan. Kontingensi L1-2 menghasilkan loading tertinggi pada saluran L4-6, yaitu 62,3%. Hal ini terjadi karena L1-2 merupakan salah satu jalur utama penyaluran daya dari slack bus, ketika saluran ini dilepas, sebagian besar daya yang semula mengalir melaluinya harus dialihkan melewati jalur alternatif L1-3-L4-6,

sehingga saluran L4-6 mengalami peningkatan pembebanan paling signifikan dibandingkan skenario lainnya. Pola serupa juga terlihat pada kontingensi L2-6, L1-3, dan L3-4 yang seluruhnya menyebabkan saluran L1-2 atau L4-6 menjadi kritis, menegaskan bahwa saluran-saluran di sekitar slack bus merupakan elemen paling sensitif dalam menjaga keandalan sistem.

Tabel 4. 5 Perubahan Aliran Daya Saluran Pasca Kontingensi

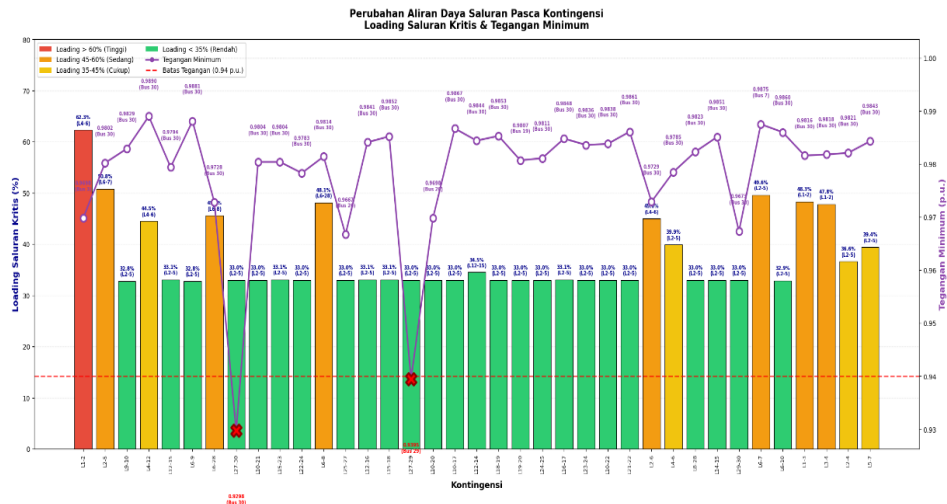
No	Kontingensi	Saluran Kritis	Loading (%)	Tegangan Min	Bus Kritis
1	L1 - 2	L4-6	62,3	0,9698	Bus 3
2	L2 - 5	L6-7	50,8	0,9802	Bus 30
3	L9 - 10	L2-5	32,8	0,9829	Bus 30
4	L4 - 12	L4-6	44,5	0,9890	Bus 30
5	L12 - 15	L2-5	33,1	0,9794	Bus 30
6	L6 - 9	L2-5	32,8	0,9881	Bus 30
7	L6 - 28	L6-8	45,6	0,9728	Bus 30
8	L27 - 30	L2-5	33,0	0,9298	Bus 30
9	L10 - 21	L2-5	33,0	0,9804	Bus 30
10	L15 - 23	L2-5	33,1	0,9804	Bus 30
11	L22 - 24	L2-5	33,0	0,9783	Bus 30
12	L6 - 8	L6-28	48,1	0,9814	Bus 30
13	L25 - 27	L2-5	33,0	0,9667	Bus 26
14	L12 - 16	L2-5	33,1	0,9841	Bus 30
15	L15 - 18	L2-5	33,1	0,9852	Bus 30
16	L27 - 29	L2-5	33,0	0,9395	Bus 29
17	L10 - 20	L2-5	33,0	0,9698	Bus 20
18	L10 - 17	L2-5	33,0	0,9867	Bus 30
19	L12 - 14	L12-15	34,5	0,9844	Bus 30
20	L18 - 19	L2-5	33,0	0,9853	Bus 30
21	L19 - 20	L2-5	33,0	0,9807	Bus 19
22	L24 - 25	L2-5	33,0	0,9811	Bus 30
23	L16 - 17	L2-5	33,1	0,9848	Bus 30
24	L23 - 24	L2-5	33,0	0,9836	Bus 30
25	L10 - 22	L2-5	33,0	0,9838	Bus 30

No	Kontingensi	Saluran Kritis	Loading (%)	Tegangan Min	Bus Kritis
26	L21 - 22	L2-5	33,0	0,9861	Bus 30
27	L2 - 6	L4-6	45,0	0,9729	Bus 30
28	L4 - 6	L2-5	39,9	0,9785	Bus 30
29	L8 - 28	L2-5	33,0	0,9823	Bus 30
30	L14 - 15	L2-5	33,0	0,9851	Bus 30
31	L29 - 30	L2-5	33,0	0,9673	Bus 30
32	L6 - 7	L2-5	49,6	0,9875	Bus 7
33	L6 - 10	L2-5	32,9	0,9860	Bus 30
34	L1 - 3	L1-2	48,3	0,9816	Bus 30
35	L3 - 4	L1-2	47,8	0,9818	Bus 30
36	L2 - 4	L2-5	36,6	0,9821	Bus 30
37	L5 - 7	L2-5	39,4	0,9843	Bus 30

Berdasarkan Tabel 4.5, Setiap kontingensi menghasilkan pola redistribusi daya yang berbeda-beda bergantung pada posisi saluran yang dilepas terhadap topologi jaringan. Kontingensi L1-2 menghasilkan loading tertinggi pada saluran L4-6, yaitu 62,3%. Hal ini terjadi karena L1-2 merupakan salah satu jalur utama penyaluran daya dari slack bus, ketika saluran ini dilepas, sebagian besar daya yang semula mengalir melaluinya harus dialihkan melewati jalur alternatif L1-3-L4-6, sehingga saluran L4-6 mengalami peningkatan pembebanan paling signifikan dibandingkan skenario lainnya. Pola serupa juga terlihat pada kontingensi L2-6, L1-3, dan L3-4 yang seluruhnya menyebabkan saluran L1-2 atau L4-6 menjadi kritis, menegaskan bahwa saluran-saluran di sekitar slack bus merupakan elemen paling sensitif dalam menjaga keandalan sistem.

Dari sisi tegangan, dua kontingensi menyebabkan pelanggaran batas operasi (undervoltage), yaitu L27-30 dengan tegangan minimum 0,9298 p.u. pada Bus 30, dan L27-29 dengan tegangan minimum 0,9395 p.u. pada Bus 29. Kedua saluran tersebut merupakan satu-satunya jalur pasokan menuju kelompok bus di ujung jaringan (Bus 26, 29, dan 30) yang telah teridentifikasi sejak Tabel 4.2 memiliki profil tegangan terendah pada kondisi base case. Ketika salah satu dari kedua saluran ini dilepas, dukungan daya reaktif menuju area tersebut praktis hilang atau

harus ditempuh melalui jalur yang jauh lebih panjang, sehingga jatuh tegangan yang terjadi menjadi jauh lebih besar dibandingkan kontingensi lain. Temuan ini konsisten dengan prinsip bahwa bus-bus yang secara elektrik terpencil (memiliki sedikit jalur pasokan alternatif) akan lebih rentan terhadap gangguan kontingensi.



Gambar 4. 2 Perbandingan Profil Tegangan Sebelum dan Sesudah Kontingensi

Secara umum, sebagian besar skenario menghasilkan loading saluran kritis pada kisaran 33-50%, yang menunjukkan bahwa mayoritas kontingensi tunggal masih dapat ditoleransi oleh sistem tanpa menimbulkan pembebanan berlebih. Namun demikian, ditemukannya beberapa kontingensi dengan loading di atas 45% serta dua kasus undervoltage menegaskan pentingnya proses perankingan kontingensi, agar operator sistem dapat memprioritaskan pengawasan pada saluran-saluran yang secara konsisten muncul sebagai elemen kritis.

4.3 Hasil Perhitungan Indeks Keamanan Statis (PIP dan PIV)

Nilai *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV) dihitung untuk seluruh 222 skenario kontingensi yang telah konvergen, guna mengukur secara kuantitatif tingkat keparahan setiap kontingensi terhadap kondisi pembebanan saluran dan profil tegangan sistem.

Berdasarkan Tabel 4.6, Nilai PIP menunjukkan kecenderungan meningkat secara konsisten seiring bertambahnya tingkat pembebanan, dari rata-rata 0,1130 pada pembebanan 50% menjadi 0,4904 pada pembebanan 100%. Kenaikan ini

terjadi karena PIP dirumuskan sebagai fungsi dari rasio pembebanan aktual saluran terhadap kapasitas termalnya, pada kondisi beban rendah, margin antara aliran daya pasca-kontingensi dengan batas termal saluran masih besar, sehingga nilai indeks relatif kecil. Sebaliknya, pada pembebanan tinggi, aliran daya dasar pada tiap saluran sudah mendekati kapasitasnya, sehingga redistribusi daya akibat pelepasan satu saluran menyebabkan sejumlah saluran lain terdorong lebih dekat atau bahkan melewati batas termalnya, yang secara langsung menaikkan nilai PIP. Meningkatnya standar deviasi dari 0,0171 pada 50% menjadi 0,0849 pada 100% turut menunjukkan bahwa sebaran tingkat keparahan antar-kontingensi juga semakin lebar pada beban tinggi, karena tidak semua saluran mengalami peningkatan pembebanan dengan proporsi yang sama.

Tabel 4. 6 Statistik Nilai PIP pada Setiap Tingkat Pembebanan

Level Beban	Minimum	Maksimum	Rata-rata	Std Dev	Median
50%	0,0992	0,1744	0,1130	0,0171	0,1066
60%	0,1453	0,2547	0,1661	0,0257	0,1564
70%	0,2007	0,3511	0,2300	0,0365	0,2163
80%	0,2657	0,4639	0,3051	0,0498	0,2864
90%	0,3405	0,5937	0,3918	0,0657	0,3671
100%	0,4255	0,7646	0,4904	0,0849	0,4585

Tabel 4. 7 Statistik Nilai PIV pada Setiap Tingkat Pembebanan

Level Beban	Minimum	Maksimum	Rata-rata	Std Dev	Median
50%	4,9988	6,8141	5,8265	0,3367	5,8102
60%	4,0321	5,4098	4,7547	0,2696	4,7771
70%	3,1829	4,2664	3,8201	0,2358	3,8655
80%	2,4762	3,4092	3,0423	0,2254	3,1091
90%	1,9297	2,8748	2,4438	0,2211	2,4978
100%	1,6146	2,6823	2,0523	0,2157	2,0559

Berbeda dengan PIP, nilai rata-rata PIV justru menurun secara konsisten dari 5,8265 pada pembebanan 50% menjadi 2,0523 pada pembebanan 100%. Kecenderungan ini merupakan konsekuensi dari formulasi PIV yang digunakan dalam penelitian ini, yaitu berbanding terbalik dengan besarnya deviasi tegangan bus terhadap batas operasinya. Pada pembebanan rendah, arus yang mengalir pada

saluran relatif kecil sehingga jatuh tegangan di sepanjang saluran juga kecil, membuat deviasi tegangan terhadap nilai nominal semakin kecil dan nilai PIV menjadi besar. Sebaliknya, pada pembebanan tinggi, arus saluran meningkat sehingga jatuh tegangan yang timbul pasca-kontingensi menjadi lebih besar, deviasi tegangan terhadap batas operasi pun membesar, dan nilai PIV menurun. Dengan kata lain, nilai PIV yang lebih kecil pada beban tinggi tetap merepresentasikan kondisi tegangan yang lebih tertekan, sejalan dengan makna fisis bahwa sistem semakin mendekati batas keamanan operasinya.

Perbandingan antara Tabel 4.6 dan Tabel 4.7 menegaskan bahwa PIP dan PIV menggambarkan dua aspek keamanan statis yang berbeda dan tidak selalu bergerak searah. PIP lebih peka terhadap perubahan pembebanan termal saluran, sedangkan PIV lebih peka terhadap perubahan kondisi tegangan bus. Oleh karena itu, kedua indeks tersebut perlu digunakan secara bersamaan agar penilaian tingkat keparahan kontingensi tidak bias hanya pada satu aspek keamanan sistem saja.

4.4 Hasil Perankingan Kontingensi N-1 Berdasarkan NRLF

Perankingan kontingensi dilakukan menggunakan hasil simulasi pada tingkat pembebanan 100%, dengan pertimbangan bahwa kondisi tersebut merepresentasikan pembebanan nominal sekaligus kondisi terberat yang dikaji dalam penelitian ini, sehingga pengaruh setiap kontingensi terhadap keamanan statis sistem dapat teridentifikasi secara paling jelas.

Tabel 4. 8 Perankingan 37 Kontingensi N-1 Berdasarkan PIP dan PIV

Performance Index power		Performance Index Voltage	
Line-outage	PIP	Line-outage	PIV
L1-2	0,7646	L27-29	2,6823
L2-5	0,7448	L27-30	2,6548
L4-12	0,7409	L6-7	2,2505
L4-6	0,5459	L10-20	2,2259
L1-3	0,5418	L3-4	2,2135
L6-8	0,5412	L22-24	2,2097
L3-4	0,5363	L29-30	2,1834
L2-6	0,5165	L19-20	2,1778

Performance Index power		Performance Index Voltage	
Line-outage	PIP	Line-outage	PIV
L6-28	0,5029	L6-28	2,1621
L10-20	0,4896	L24-25	2,1493
L10-21	0,4892	L15-23	2,1321
L12-15	0,4851	L23-24	2,1250
L9-10	0,4805	L25-27	2,1088
L6-9	0,4772	L21-22	2,1078
L19-20	0,4680	L14-15	2,1039
L12-14	0,4669	L18-19	2,1024
L10-22	0,4605	L16-17	2,0985
L27-30	0,4599	L5-7	2,0604
L12-16	0,4585	L10-22	2,0559
L6-10	0,4565	L15-18	2,0527
L6-7	0,4531	L8-28	2,0519
L15-18	0,4517	L10-17	2,0471
L25-27	0,4493	L6-8	2,0212
L15-23	0,4484	L10-21	2,0189
L10-17	0,4480	L4-6	1,9580
L22-24	0,4474	L12-14	1,9341
L27-29	0,4474	L6-10	1,9144
L16-17	0,4426	L6-9	1,9086
L21-22	0,4401	L12-16	1,9077
L14-15	0,4397	L2-5	1,8739
L29-30	0,4397	L12-15	1,8700
L18-19	0,4386	L2-4	1,8661
L24-25	0,4382	L2-6	1,8646
L23-24	0,4381	L1-3	1,8293
L8-28	0,4372	L1-2	1,7405
L5-7	0,4315	L9-10	1,6579
L2-4	0,4255	L4-12	1,6146

Berdasarkan Tabel 4.8, kontingensi L1-2 menempati peringkat pertama pada indeks PIP (0,7646), diikuti L2-5 (0,7448) dan L4-12 (0,7409). Ketiganya konsisten dengan temuan pada Tabel 4.5, di mana ketiga saluran tersebut

merupakan jalur utama penyaluran daya dari slack bus menuju pusat-pusat beban, sehingga pelepasannya memicu redistribusi daya paling besar pada sistem.

Urutan peringkat berdasarkan PIV tidak sejalan dengan urutan berdasarkan PIP. Kontingensi L27-29 menempati peringkat pertama pada PIV (2,6823) namun hanya berada di peringkat ke-27 pada PIP, sedangkan L1-2 yang menjadi peringkat pertama pada PIP justru berada di peringkat ke-35 pada PIV. Perbedaan ini dapat dijelaskan karena kedua indeks merepresentasikan mekanisme fisik yang berbeda, L1-2 berada dekat dengan slack bus sehingga pelepasannya sangat memengaruhi pembebanan termal saluran (PIP tinggi), namun karena berada di area dengan dukungan tegangan yang kuat dari beberapa generator terdekat, dampaknya terhadap profil tegangan relatif kecil (PIV rendah). Sebaliknya, L27-29 berada di area ujung jaringan dengan dukungan tegangan yang terbatas, sehingga pelepasannya sangat memengaruhi profil tegangan (PIV tinggi) meskipun pengaruhnya terhadap pembebanan termal saluran lain relatif kecil (PIP sedang).

Temuan ini menegaskan bahwa penilaian keamanan statis sistem tenaga listrik tidak dapat hanya mengandalkan satu indeks saja. Kombinasi PIP dan PIV memberikan gambaran yang lebih menyeluruh mengenai kontingensi mana yang perlu mendapat prioritas pengawasan, baik dari sisi risiko overload saluran maupun risiko pelanggaran tegangan. Hasil perankingan berdasarkan NRLF pada subbab ini selanjutnya dijadikan sebagai data acuan (ground truth) untuk mengevaluasi konsistensi hasil perankingan yang dihasilkan oleh model RBFN dan XGBoost pada subbab berikutnya.

4.5 Pembentukan dan Pembagian Dataset

Dataset penelitian dibentuk dari hasil simulasi *Newton Raphson Load Flow* (NRLF) pada enam tingkat pembebanan (50%-100%, interval 10%), masing-masing diuji terhadap 37 skenario kontingensi N-1 yang merepresentasikan pelepasan setiap saluran transmisi, sehingga diperoleh total 222 sampel. Variasi enam tingkat pembebanan dipilih agar model dapat mempelajari pola PIP dan PIV mulai dari kondisi beban ringan hingga puncak, bukan hanya pada satu titik operasi, sedangkan penggunaan seluruh 37 kontingensi pada tiap level sejalan dengan kriteria keamanan N-1 yang mencakup seluruh kemungkinan kontingensi tunggal.

Rincian dataset ditunjukkan pada Tabel 4.9. Variabel masukan berupa daya aktif dan reaktif beban (PL, QL), daya aktif dan reaktif pembangkit (PG, QG), magnitudo tegangan bus ($|V|$), serta indeks kontingensi (K_i) dipilih karena dihasilkan langsung oleh simulasi NRLF tanpa perhitungan tambahan, sehingga berpotensi diterapkan pada kondisi mendekati waktu nyata (near real time). Adapun PIP dan PIV digunakan sebagai variabel keluaran karena merupakan indeks keamanan statis standar yang merepresentasikan tingkat pembebanan saluran dan penyimpangan tegangan akibat kontingensi.

Tabel 4. 9 Ringkasan Dataset Penelitian

Keterangan	Nilai
Jumlah tingkat pembebanan	6
Jumlah skenario kontingensi	37
Total sampel	222
Variabel masukan	Daya aktif beban (PL), daya reaktif beban (QL), daya aktif pembangkit (PG), daya reaktif pembangkit (QG), magnitudo tegangan bus ($ V $), dan indeks kontingensi (K_i)
Variabel keluaran	<i>Active Power Performance Index</i> (PIP) dan <i>Voltage Performance Index</i> (PIV)

Dataset selanjutnya dibagi menggunakan `train_test_split()` menjadi 185 sampel (83,33%) data pelatihan dan 37 sampel (16,67%) data pengujian (Tabel 4.10). Alokasi 37 sampel setara satu replikasi penuh 37 kontingensi dinilai cukup untuk evaluasi yang representatif, sementara 185 sampel memberi model, terutama XGBoost dengan banyak fitur hasil rekayasa fitur, data yang memadai untuk belajar tanpa terlalu rawan overfitting.

Tabel 4. 10 Pembagian Dataset

Jenis Dataset	Jumlah Sampel	Persentase
Data Pelatihan (Training Set)	185	83,33%
Data Pengujian (Testing Set)	37	16,67%
Total	222	100%

Berdasarkan Tabel 4.10, dataset dibagi menjadi 185 sampel sebagai data pelatihan dan 37 sampel sebagai data pengujian. Pembagian data menerapkan stratified sampling berdasarkan tingkat pembebanan (*load_bins*), bukan pembagian acak murni, agar proporsi setiap level beban tetap seimbang pada data pelatihan maupun pengujian. Tanpa stratifikasi, data pengujian berisiko didominasi oleh level beban tertentu saja sehingga hasil evaluasi menjadi bias. Dengan pendekatan ini, hasil evaluasi model pada Subbab 4.6 dan 4.7 dapat merepresentasikan performa model secara menyeluruh pada seluruh rentang kondisi operasi.

4.5.1 Normalisasi Data

Sebelum digunakan pada proses pelatihan model, seluruh dataset terlebih dahulu melalui tahap normalisasi untuk menyeragamkan skala data. Pada penelitian ini, variabel masukan (*input features*) dinormalisasi menggunakan *RobustScaler*, sedangkan variabel keluaran (target output), yaitu *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV), dinormalisasi menggunakan *StandardScaler*. Setelah proses normalisasi selesai, dataset kemudian dibagi menjadi data pelatihan dan data pengujian sesuai dengan implementasi yang digunakan pada penelitian.

```
scaler_X = RobustScaler()
X_scaled = scaler_X.fit_transform(X_enhanced)

scaler_y = StandardScaler()
y_scaled = scaler_y.fit_transform(y_all)

total_load = np.sum(X_all[:, 0:30], axis=1)
load_bins = pd.qcut(total_load, q=5, labels=False, duplicates='drop')

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_scaled, test_size=37, random_state=42, stratify=load_bins
)
```

Gambar 4. 3 Implementasi Normalisasi Data Menggunakan *RobustScaler* dan *StandardScaler*

Berdasarkan Gambar 4.3, proses normalisasi diawali dengan penerapan *RobustScaler* pada variabel masukan menggunakan fungsi *fit_transform()*, sedangkan pada model RBFN variabel keluaran dinormalisasi menggunakan *StandardScaler*. Setelah seluruh data berada pada skala yang sesuai, dilakukan

pembagian dataset menggunakan fungsi *train_test_split()* dengan parameter *test_size = 37*, *random_state = 42*, dan parameter *stratify* yang menjaga proporsi data pada setiap tingkat pembebanan sehingga distribusi data pelatihan dan data pengujian tetap seimbang. Hasil dari proses ini menghasilkan 185 sampel sebagai data pelatihan dan 37 sampel sebagai data pengujian, yang selanjutnya digunakan pada proses pelatihan dan evaluasi model.

Meskipun kerangka umum normalisasi dan pembagian data pada kedua model mengikuti pola yang sama, terdapat beberapa perbedaan implementasi antara model *Radial Basis Function Network* (RBFN) dan *Extreme Gradient Boosting* (XGBoost), sebagaimana dirangkum pada Tabel 4.11

Tabel 4. 11 Perbandingan Implementasi Normalisasi dan Pembagian Data pada Model RBFN dan XGBoost

Aspek	RBFN	XGBoost
Normalisasi input (X)	RobustScaler, <i>fit_transform()</i>	RobustScaler, <i>fit_transform()</i>
Normalisasi output (y, PIP, PIV)	StandardScaler, <i>fit_transform()</i>	Tidak dinormalisasi (nilai PIP/PIV asli langsung digunakan)
Dasar stratifikasi (stratify)	<i>load_bins</i> 5 bin kuantil dari total beban sistem (<i>pd.qcut(sys_load_P_MW, q=5)</i>)	<i>level_pct</i> kategori tingkat pembebanan asli dari dataset (tanpa proses binning kuantil)
test_size	37	37
random_state	42	42
Jumlah data latih , uji	185 , 37	185 , 37 (185 data latih selanjutnya dibagi lagi 80,20 menjadi data latih-akhir dan data validasi untuk <i>early stopping</i>)

Berdasarkan Tabel 4.11, perbedaan utama terletak pada normalisasi variabel keluaran. Model RBFN menormalisasi PIP dan PIV menggunakan *StandardScaler* sebelum pelatihan dan mengembalikannya ke skala asli (*inverse_transform*) saat evaluasi, sedangkan model XGBoost dilatih langsung pada nilai PIP dan PIV asli tanpa normalisasi. Perbedaan lain terletak pada variabel dasar stratifikasi, RBFN menggunakan bin kuantil dari total beban sistem, sedangkan XGBoost menggunakan kategori *level_pct* yang sudah tersedia pada dataset. Selain itu, pada

model XGBoost terdapat tahap pembagian data tambahan (validation split) yang digunakan untuk mekanisme *early stopping* pada proses training. Perbedaan-perbedaan ini tetap menjaga konsistensi proporsi data pelatihan dan data pengujian sebesar 185 , 37 sampel pada kedua model.

4.6 Hasil Implementasi Model *Radial Basis Function Network* (RBFN)

Pada penelitian ini, implementasi model *Radial Basis Function Network* (RBFN) mengacu pada [7] yang menjadi rujukan metodologis dalam perankingan kontingensi N-1 untuk penilaian keamanan statis sistem tenaga listrik. Untuk menjaga konsistensi terhadap paper rujukan sekaligus mengevaluasi keterbatasannya pada data hasil simulasi penelitian ini, implementasi RBFN dilakukan dalam dua tahap. Tahap pertama merupakan replikasi murni menggunakan parameter optimal yang dilaporkan pada paper rujukan ($m1 = 38$, $g2 = 1,8$, $g1 = 1,8$), dan tahap kedua merupakan pengembangan model dalam bentuk *Optimized Ensemble RBFN* yang mengombinasikan *K-Means++*, *Ridge Regression*, *Least Mean Squares (LMS)*, dan *Bootstrap Aggregating (Bagging)* untuk mengatasi keterbatasan yang ditemukan pada tahap pertama.

4.6.1 Implementasi RBFN dengan Parameter [7]

Sebagai tahap awal, penelitian ini mengimplementasikan *Radial Basis Function Network* (RBFN) menggunakan konfigurasi parameter yang mengacu pada penelitian [7] sebagai model acuan (*baseline*). Implementasi ini bertujuan untuk mengevaluasi sejauh mana parameter yang dilaporkan pada penelitian referensi dapat diterapkan pada dataset hasil simulasi IEEE 30-Bus yang digunakan dalam penelitian ini. Selain itu, model *baseline* digunakan sebagai pembanding terhadap model *Optimized Ensemble RBFN* yang dikembangkan pada penelitian ini sehingga peningkatan performa yang diperoleh dapat dianalisis secara lebih objektif.

Arsitektur model mengikuti prosedur pelatihan RBFN yang ditunjukkan pada penelitian[7]. Tahapan pelatihan diawali dengan inisialisasi bobot dan parameter jaringan, dilanjutkan dengan perhitungan jarak *Euclidean* antara data masukan dan setiap pusat (*center*), pembentukan fungsi basis radial Gaussian, serta pembaruan bobot menggunakan algoritma *Least Mean Squares (LMS)*. Proses

iterasi dilakukan hingga diperoleh nilai *Mean Square Error* (MSE) minimum atau jumlah iterasi maksimum tercapai. Parameter implementasi yang digunakan pada model *baseline* ditunjukkan pada Tabel 4.12.

Tabel 4. 12 Parameter Implementasi RBFN [7]

Parameter	Nilai
Jumlah Center (m1)	38
Sigma Factor (g2)	1,8
<i>Learning rate</i> (g1)	1,8
Maximum Iteration	100
Target Output	PIP dan PIV

Berdasarkan parameter pada Tabel 4.12, model RBFN diimplementasikan tanpa mekanisme regularisasi maupun *ensemble* sehingga seluruh proses pelatihan mengikuti konfigurasi dasar sebagaimana dijelaskan pada penelitian referensi [7]. Dengan konfigurasi tersebut, model hanya menggunakan satu jaringan RBFN untuk mempelajari hubungan antara variabel masukan dengan target keluaran *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV). Jumlah 38 center digunakan untuk membentuk fungsi basis radial, sedangkan proses pembelajaran bobot dilakukan menggunakan algoritma *Least Mean Square* (LMS) dengan *learning rate* sebesar 1,8 dan maksimum 100 iterasi. Konfigurasi ini digunakan sebagai model *baseline* sehingga hasilnya dapat dijadikan acuan dalam mengevaluasi peningkatan performa yang diperoleh melalui pendekatan *Optimized Ensemble RBFN*.

Tabel 4. 13 Perkembangan Nilai Average MSE Selama Pelatihan Model RBFN [7]

Epoch	Average MSE
20	1.424,43
40	898,28
60	611,14
80	477,61
100	666,63

Berdasarkan Tabel 4.13, nilai Average MSE mengalami penurunan dari 1.424,43 pada epoch ke-20 menjadi 477,61 pada epoch ke-80, yang menunjukkan bahwa model secara bertahap mampu mengurangi galat selama proses pembelajaran. Namun, pada epoch ke-100 nilai Average MSE meningkat menjadi 666,63. Fluktuasi tersebut mengindikasikan bahwa proses pembaruan bobot menggunakan algoritma LMS belum sepenuhnya stabil untuk dataset yang digunakan pada penelitian ini, sehingga diperlukan pengembangan melalui optimasi parameter dan penerapan pendekatan *Optimized Ensemble* RBFN. Evaluasi model selanjutnya dilakukan menggunakan 37 skenario kontingensi N-1 pada tingkat pembebanan 100% sebagai data uji baru yang sepenuhnya terpisah dari proses pelatihan. Hasil evaluasi model baseline pada data uji terbaru tersebut ditunjukkan pada Tabel 4.14.

Tabel 4. 14 Metrik Evaluasi Model RBFN [7]

Metrik	PIP Test	PIV Test
R ²	-29,4240	-15,9230
MSE	0,213542	0,765833
RMSE	0,462106	0,875119
MAE	0,454448	0,848869
MAPE	92,52%	40,75%
Max Error	0,728700	1,478900

Berdasarkan hasil evaluasi pada Tabel 4.14, model RBFN [7] menunjukkan performa yang baik pada data pelatihan (R² sebesar 0,9820 untuk PIP dan 0,9914 untuk PIV), namun mengalami kegagalan generalisasi yang sangat signifikan ketika diuji pada 37 skenario kontingensi baru dengan tingkat pembebanan 100% yang sepenuhnya terpisah dari data pelatihan. Pada prediksi *Active Power Performance Index* (PIP), model menghasilkan nilai koefisien determinasi (R²) sebesar -29,4240 dengan RMSE sebesar 0,462106, MAE sebesar 0,454448, dan MAPE mencapai 92,52%. Nilai tersebut jauh lebih buruk dibandingkan performa pada data pelatihan, mengindikasikan bahwa model tidak mampu mempertahankan pola pembelajarannya pada data yang belum pernah dilihat sebelumnya.

Kondisi serupa juga teramati pada prediksi *Voltage Performance Index* (PIV), dengan nilai R^2 sebesar -15,9230, RMSE sebesar 0,875119, MAE sebesar 0,848869, dan MAPE sebesar 40,75%. Investigasi lebih lanjut terhadap hasil prediksi menunjukkan bahwa model menghasilkan nilai keluaran yang hampir konstan untuk seluruh 37 sampel data uji, yaitu PIP sebesar 0,0359 dan PIV sebesar 1,2034, terlepas dari variasi kondisi sistem pada tiap kontingensi. Nilai prediksi yang hampir konstan ini mengindikasikan bahwa bobot keluaran (output weights) hasil pelatihan mengalami kolaps atau saturasi, sehingga fungsi basis radial Gaussian kehilangan sensitivitasnya terhadap perbedaan pola input antar-kontingensi. Kondisi ini konsisten dengan fluktuasi nilai Average MSE yang telah diamati pada Tabel 4.13, di mana pembaruan bobot melalui algoritma LMS tanpa mekanisme regularisasi maupun early stopping menyebabkan proses pelatihan menjadi tidak stabil, sehingga bobot akhir yang diperoleh tidak mampu mempertahankan kemampuan generalisasi model terhadap data baru.

Secara keseluruhan hasil evaluasi menunjukkan bahwa model belum mampu memberikan performa prediksi yang memadai pada skenario kontingensi baru, yang ditunjukkan oleh nilai koefisien determinasi (R^2) yang sangat rendah disertai peningkatan nilai galat prediksi. Kondisi tersebut mengindikasikan bahwa model belum memiliki kemampuan generalisasi yang baik sehingga diperlukan pengembangan melalui optimasi parameter dan penerapan pendekatan Optimized Ensemble RBFN. Pengembangan model tersebut selanjutnya dibahas pada Subbab 4.6.2.

4.6.2 Analisis Implementasi [7] pada Dataset Penelitian

Hasil implementasi pada Subbab 4.6.1 menunjukkan bahwa penggunaan parameter Radial Basis Function Network (RBFN) yang mengacu pada penelitian [7] menghasilkan performa yang sangat baik pada data pelatihan ($R^2 > 0,98$) namun gagal secara signifikan ketika diuji pada data baru yang sepenuhnya terpisah (R^2 PIP = -29,4240, R^2 PIV = -15,9230). Kesenjangan performa yang sangat besar tersebut menjadi indikasi kuat terjadinya overfitting pada model baseline.

Perbedaan performa tersebut tidak menunjukkan bahwa parameter pada penelitian referensi kurang baik, melainkan mengindikasikan bahwa parameter

optimal suatu model sangat dipengaruhi oleh karakteristik dataset, konfigurasi fitur, serta lingkungan implementasi. Oleh karena itu, dilakukan analisis terhadap beberapa faktor yang diduga memengaruhi hasil implementasi parameter referensi pada penelitian ini.

1. Perbedaan Karakteristik Dataset. Parameter jumlah center (m_1), sigma factor (g_2), dan learning rate (g_1) yang digunakan pada penelitian Sekhar dan Mohanty (2016) diperoleh melalui proses optimasi terhadap dataset yang dibentuk menggunakan skenario simulasi pada penelitian tersebut. Sementara itu, dataset pada penelitian ini dibangun secara mandiri menggunakan simulasi Newton-Raphson Load Flow (NRLF) pada sistem IEEE 30-Bus dengan 37 skenario kontingensi N-1 dan enam variasi tingkat pembebanan, sehingga menghasilkan total 222 sampel data. Perbedaan proses pembentukan dataset menyebabkan distribusi data yang dipelajari oleh model juga berbeda, sehingga parameter yang optimal pada penelitian referensi belum tentu memberikan performa terbaik pada dataset penelitian ini.
2. Pengaruh Dimensi Fitur Masukan. Model menerima 191 fitur masukan hasil simulasi NRLF setelah melalui normalisasi dan feature engineering, jauh lebih besar dibandingkan implementasi pada penelitian referensi. Ketika dimensi fitur meningkat, penggunaan satu nilai sigma global pada fungsi basis radial Gaussian menjadi kurang mampu merepresentasikan penyebaran data pada seluruh dimensi secara merata, sehingga sensitivitas fungsi basis radial terhadap pola lokal data berkurang dan model lebih rentan mengalami overfitting terhadap data pelatihan.
3. Stabilitas Proses Pelatihan dan Kegagalan Generalisasi. Selama proses pelatihan, nilai MSE menunjukkan kecenderungan menurun dari 1.424,43 pada epoch ke-20 menjadi 477,61 pada epoch ke-80, kemudian meningkat menjadi 666,63 pada epoch ke-100 (Tabel 4.13), yang menunjukkan bahwa pembaruan bobot menggunakan algoritma LMS belum sepenuhnya stabil. Ketidakstabilan ini berdampak nyata pada kemampuan generalisasi model, sebagaimana ditunjukkan oleh hasil pengujian pada data uji terbaru (Tabel 4.14) yang menghasilkan prediksi hampir konstan

dan nilai R^2 yang sangat negatif. Temuan ini menjadi dasar utama dilakukannya optimasi parameter dan penerapan *Optimized Ensemble RBFN*.

4. Tidak Adanya Regularisasi dan Mekanisme Ensemble. Implementasi parameter referensi menggunakan satu model RBFN tanpa mekanisme regularisasi maupun ensemble, sehingga bobot keluaran dibentuk hanya berdasarkan satu proses pelatihan dan lebih rentan terhadap kolaps bobot maupun overfitting ketika dihadapkan pada data di luar distribusi data latih, sebagaimana teramati pada Tabel 4.14. Kondisi ini lebih terlihat pada prediksi PIV, yang memiliki rentang nilai relatif sempit dibandingkan PIP, sehingga kesalahan prediksi yang relatif kecil dapat memberikan pengaruh yang besar terhadap nilai koefisien determinasi (R^2).
5. Perbedaan Lingkungan Implementasi. Penelitian [7] dikembangkan menggunakan MATLAB, sedangkan penelitian ini diimplementasikan menggunakan Python pada lingkungan Google Colab, dengan perbedaan pustaka numerik, proses inisialisasi parameter, serta tahapan normalisasi data, pembentukan fitur, dan pembagian dataset. Perbedaan tersebut turut memengaruhi proses pembelajaran model sehingga hasil yang diperoleh tidak identik dengan penelitian acuan.

Berdasarkan hasil analisis tersebut, dapat disimpulkan bahwa parameter RBFN yang diperoleh pada penelitian Sekhar dan Mohanty (2016) belum dapat langsung digeneralisasi pada dataset penelitian ini, bahkan cenderung mengalami overfitting dan kegagalan generalisasi yang signifikan pada data yang benar-benar baru. Perbedaan karakteristik data, jumlah fitur, proses pembentukan dataset, ketidakstabilan pelatihan, serta lingkungan implementasi menyebabkan model memerlukan penyesuaian parameter agar mampu menghasilkan performa yang optimal dan stabil. Oleh karena itu, pada penelitian ini dilakukan pengembangan model RBFN melalui optimasi parameter dan penerapan pendekatan Ensemble RBFN, sebagaimana dijelaskan pada Subbab 4.6.3.

4.6.3 Arsitektur Implementasi Model RBFN

Arsitektur *Optimized Ensemble* RBFN dirangkum pada Tabel 4.15. Model terdiri atas 5 model RBFN yang dilatih secara paralel (bagging) dari 135 fitur input (PL, QL, PG, QG, V, dan Ki hasil normalisasi dan feature engineering). Pada hidden layer digunakan 38 center hasil algoritma K-Means++ dengan fungsi aktivasi Gaussian RBF (sigma factor 0,5), dilatih menggunakan *Least Mean Squares* (LMS) dengan learning rate 1,8, regularisasi Ridge ($\lambda = 0,001$), serta early stopping (patience 30 epoch, maksimum 100 iterasi) untuk meningkatkan generalisasi. Model menghasilkan prediksi PIP dan PIV secara bersamaan.

Tabel 4. 15 Parameter Implementasi Model RBFN

Parameter	Nilai
Jumlah Model Ensemble	5
Jumlah Center (n_centers)	38
Learning Rate	1,8
Maximum Iteration	100
Sigma Factor	0,5
Regularization (λ)	0,001
Patience	30 Epoch
Total Fitur Input	135 (fitur bus + fitur turunan + fitur agregat saluran)
Target Output	PIP dan PIV

4.6.4 Hasil Evaluasi Model RBFN

Evaluasi model dilakukan pada 37 skenario kontingensi N-1 tingkat pembebanan 100%, menggunakan metrik R^2 , RMSE, MAE, dan MAPE. Rincian perhitungan galat per saluran untuk PIP dan PIV ditunjukkan pada Tabel 4.16 dan Tabel 4.17, Dimana sebagian besar prediksi PIP mendekati nilai aktual, dengan galat terbesar pada L2-5 (0,0982), L6-8 (0,0793), L4-6 (0,0546), dan L2-4 (0,0445), mengindikasikan redistribusi aliran daya pada saluran tersebut masih cukup sulit dimodelkan secara akurat.

Berdasarkan Tabel 4.17, hasil prediksi PIV menunjukkan kesesuaian lebih baik daripada PIP, dengan galat terbesar hanya pada L9-10 (0,1368) dan sebagian besar saluran lain di bawah 0,10, menunjukkan perubahan profil tegangan lebih mudah dipelajari model dibandingkan redistribusi pembebanan saluran.

Tabel 4. 16 Tabel Metrik Evaluasi Simulasi Ensemble RBFN untuk PIP

No.	Saluran	Aktual (X)	Prediksi RBFN (Y)	$ X - Y $	$(X - Y)^2$	$(X - \bar{X})^2$
1	L 1-2	0,7646	0,7319	0,032700	0,00106929	0,07520787
2	L 2-5	0,7448	0,6466	0,098200	0,00964324	0,06473999
3	L 4-12	0,7409	0,6872	0,053700	0,00288369	0,06277056
4	L 4-6	0,5459	0,4913	0,054600	0,00298116	0,00308475
5	L 1-3	0,5418	0,5350	0,006800	0,00004624	0,00264613
6	L 6-8	0,5412	0,4619	0,079300	0,00628849	0,00258476
7	L 3-4	0,5363	0,5388	0,002500	0,00000625	0,00211053
8	L 2-6	0,5165	0,4951	0,021400	0,00045796	0,00068333
9	L 6-28	0,5029	0,4697	0,033200	0,00110224	0,00015727
10	L 10-20	0,4896	0,4756	0,014000	0,00019600	0,00000058
11	L 10-21	0,4892	0,4658	0,023400	0,00054756	0,00000134
12	L 12-15	0,4851	0,4861	0,001000	0,00000100	0,00002766
13	L 9-10	0,4805	0,4943	0,013800	0,00019044	0,00009721
14	L 6-9	0,4772	0,4620	0,015200	0,00023104	0,00017317
15	L 19-20	0,4680	0,4639	0,004100	0,00001681	0,00049995
16	L 12-14	0,4669	0,4696	0,002700	0,00000729	0,00055035
17	L 10-22	0,4605	0,4566	0,003900	0,00001521	0,00089159
18	L 27-30	0,4599	0,4574	0,002500	0,00000625	0,00092778
19	L 12-16	0,4585	0,4574	0,001100	0,00000121	0,00101503
20	L 6-10	0,4565	0,4655	0,009000	0,00008100	0,00114646
21	L 6-7	0,4531	0,4512	0,001900	0,00000361	0,00138827
22	L 15-18	0,4517	0,4543	0,002600	0,00000676	0,00149455
23	L 25-27	0,4493	0,4594	0,010100	0,00010201	0,00168588
24	L 15-23	0,4484	0,4552	0,006800	0,00004624	0,00176060
25	L 10-17	0,4480	0,4686	0,020600	0,00042436	0,00179432
26	L 27-29	0,4474	0,4601	0,012700	0,00016129	0,00184552
27	L 22-24	0,4474	0,4612	0,013800	0,00019044	0,00184552
28	L 16-17	0,4426	0,4494	0,006800	0,00004624	0,00228097
29	L 21-22	0,4401	0,4503	0,010200	0,00010404	0,00252601
30	L 29-30	0,4397	0,4506	0,010900	0,00011881	0,00256638
31	L 14-15	0,4397	0,4518	0,012100	0,00014641	0,00256638
32	L 18-19	0,4386	0,4505	0,011900	0,00014161	0,00267904
33	L 24-25	0,4382	0,4505	0,012300	0,00015129	0,00272061
34	L 23-24	0,4381	0,4500	0,011900	0,00014161	0,00273105

No.	Saluran	Aktual (X)	Prediksi RBFN (Y)	X - Y	(X - Y) ²	(X - $\bar{X}$) ²
35	L 8-28	0,4372	0,4578	0,020600	0,00042436	0,00282593
36	L 5-7	0,4315	0,4582	0,026700	0,00071289	0,00346444
37	L 2-4	0,4255	0,4700	0,044500	0,00198025	0,00420675
	Total			0,709500	0,03067459	0,25969851

$$\bar{X} = 0.4904$$

$$RMSE = \sqrt{\frac{0,030675}{37}} = 0,028793$$

$$MAE = \frac{0,709500}{37} = 0,019176$$

$$R^2 = 1 - \frac{0,030675}{0,259699} = 0,8819$$

$$MAPE = 3,64\%$$

Tabel 4. 17 Tabel Metrik Evaluasi Simulasi Ensemble RBFN untuk PIV

No.	Saluran	Aktual (X)	Prediksi RBFN (Y)	X - Y	(X - Y) ²	(X - $\bar{X}$) ²
1	L 27-29	2,6823	2,6147	0,067600	0,00456976	0,39688297
2	L 27-30	2,6548	2,6343	0,020500	0,00042025	0,36298997
3	L 6-7	2,2505	2,2699	0,019400	0,00037636	0,03927788
4	L 10-20	2,2259	2,1445	0,081400	0,00662596	0,03013227
5	L 3-4	2,2135	2,1801	0,033400	0,00111556	0,02598108
6	L 22-24	2,2097	2,1728	0,036900	0,00136161	0,02477051
7	L 29-30	2,1834	2,1658	0,017600	0,00030976	0,01718367
8	L 19-20	2,1778	2,1152	0,062600	0,00391876	0,01574686
9	L 6-28	2,1621	2,1716	0,009500	0,00009025	0,01205307
10	L 24-25	2,1493	2,1325	0,016800	0,00028224	0,00940638
11	L 15-23	2,1321	2,1300	0,002100	0,00000441	0,00636588
12	L 23-24	2,1250	2,1074	0,017600	0,00030976	0,00528333
13	L 25-27	2,1088	2,1096	0,000800	0,00000064	0,00319072
14	L 21-22	2,1078	2,0845	0,023300	0,00054289	0,00307875
15	L 14-15	2,1039	2,0812	0,022700	0,00051529	0,00266117

No.	Saluran	Aktual (X)	Prediksi RBFN (Y)	X - Y	(X - Y) ²	(X - $\bar{X}$) ²
16	L 18-19	2,1024	2,0892	0,013200	0,00017424	0,00250866
17	L 16-17	2,0985	2,0800	0,018500	0,00034225	0,00213319
18	L 5-7	2,0604	2,0913	0,030900	0,00095481	0,00006539
19	L 10-22	2,0559	2,0697	0,013800	0,00019044	0,00001286
20	L 15-18	2,0527	2,0630	0,010300	0,00010609	0,00000015
21	L 8-28	2,0519	2,0711	0,019200	0,00036864	0,00000017
22	L 10-17	2,0471	2,0392	0,007900	0,00006241	0,00002718
23	L 6-8	2,0212	2,0464	0,025200	0,00063504	0,00096805
24	L 10-21	2,0189	2,0600	0,041100	0,00168921	0,00111646
25	L 4-6	1,9580	2,0004	0,042400	0,00179776	0,00889504
26	L 12-14	1,9341	1,9940	0,059900	0,00358801	0,01397443
27	L 6-10	1,9144	1,9704	0,056000	0,00313600	0,01902014
28	L 6-9	1,9086	1,9608	0,052200	0,00272484	0,02065357
29	L 12-16	1,9077	1,9922	0,084500	0,00714025	0,02091307
30	L 2-5	1,8739	1,9334	0,059500	0,00354025	0,03183138
31	L 12-15	1,8700	1,9068	0,036800	0,00135424	0,03323822
32	L 2-4	1,8661	1,9370	0,070900	0,00502681	0,03467547
33	L 2-6	1,8646	1,9189	0,054300	0,00294849	0,03523636
34	L 1-3	1,8293	1,9216	0,092300	0,00851929	0,04973503
35	L 1-2	1,7405	1,7725	0,032000	0,00102400	0,09722767
36	L 9-10	1,6579	1,7947	0,136800	0,01871424	0,15556202
37	L 4-12	1,6146	1,6080	0,006600	0,00004356	0,19159312
	Total			1,396500	0,08452437	1,67439214

$$\bar{X} = 2.0523$$

$$RMSE = \sqrt{\frac{0,084524}{37}} = 0,047796$$

$$MAE = \frac{1,396500}{37} = 0,037743$$

$$R^2 = 1 - \frac{0,084524}{1,674392} = 0,9495$$

$$MAPE = 1.91\%$$

Tabel 4. 18 Metrik Evaluasi Model Ensemble RBFN

Metrik	PIP Train	PIP Test	PIV Train	PIV Test
R ²	0,9709	0,8818	0,9951	0,9495
RMSE	0,023224	0,028801	0,092544	0,047799
MAE	0,014830	0,019180	0,068399	0,037748
MAPE	5,51%	3,64%	1,93%	1,91%
Max Error	0,138793	0,098200	0,326535	0,136800
R ²	0,9709	0,8818	0,9951	0,9495

Berdasarkan Tabel 4.18, model Ensemble RBFN memberikan hasil baik pada kedua indeks, PIP memperoleh $R^2=0,8818$, $RMSE=0,0288$, $MAE=0,0192$, $MAPE=3,64\%$, PIV memperoleh $R^2=0,9495$, $RMSE=0,0478$, $MAE=0,0377$, $MAPE=1,91\%$. Performa PIV sedikit lebih baik daripada PIP, namun selisih antar-keduanya jauh lebih kecil dibandingkan model baseline RBFN [7], menandakan bahwa kombinasi K-Means++, Ridge Regression, LMS, bagging, dan fitur agregat saluran berhasil meningkatkan generalisasi model secara signifikan pada kedua indeks, bukan hanya salah satunya. Dibandingkan baseline yang menghasilkan R^2 sangat negatif pada data uji, *Optimized Ensemble* RBFN menunjukkan peningkatan performa yang sangat signifikan, sehingga layak digunakan sebagai pembanding terhadap XGBoost pada subbab berikutnya

4.6.5 Perbandingan Implementasi RBFN Referensi dan *Optimized Ensemble* RBFN

Ringkasan perbandingan antara RBFN Referensi [7] dan *Optimized Ensemble* RBFN disajikan pada Tabel 4.19. Kedua implementasi menggunakan prinsip dasar yang sama, yaitu fungsi basis radial Gaussian, namun berbeda pada strategi pembentukan dan pelatihan model, penelitian ini menambahkan penentuan center dengan K-Means++, adaptive sigma, regularisasi Ridge, early stopping, dan Bootstrap Aggregating (5 model, ensemble averaging).

Pengembangan yang diterapkan meningkatkan performa secara signifikan dan konsisten pada kedua indeks, terutama PIV (R^2 dari -0,6903 pada RBFN referensi menjadi 0,9495 pada *Optimized Ensemble* RBFN) dan PIP (R^2 dari -29,4240 menjadi 0,8818 dengan $MAPE$ hanya 3,64%). Peningkatan drastis ini menegaskan bahwa kombinasi K-Means++, adaptive sigma, Ridge Regression, early stopping, dan Bagging berhasil mengatasi kegagalan generalisasi yang

dialami konfigurasi referensi murni, sehingga model jauh lebih robust dan stabil ketika menghadapi data di luar distribusi pelatihan. Berdasarkan hasil tersebut, *Optimized Ensemble* RBFN dipilih sebagai representasi metode RBFN pada penelitian ini dan digunakan sebagai pembanding terhadap *Extreme Gradient Boosting* (XGBoost) pada tahap evaluasi berikutnya.

Tabel 4. 19 Perbandingan Implementasi RBFN Referensi dan *Optimized Ensemble* RBFN

Aspek	RBFN [7]	<i>Optimized Ensemble</i> RBFN (Penelitian)
Penentuan Center	Menggunakan jumlah center tetap (38 center)	Center ditentukan menggunakan K-Means++ (150 center)
Fungsi Basis	Gaussian	Gaussian dengan adaptive sigma
Pembaruan Bobot	<i>Least Mean Squares</i> (LMS)	Ridge Regression + LMS
Regularisasi	Tidak digunakan	Ridge Regularization
Ensemble	Model tunggal	Bagging (5 model)
Penghentian Pelatihan	Maximum iteration	Early stopping + maximum iteration
Target Prediksi	PIP dan PIV	PIP dan PIV
Nilai R ² PIP	0,7089	0,8818
Nilai R ² PIV	-0,6903	0,9495

4.7 Implementasi Model *Extreme Gradient Boosting* (XGBoost)

Model *Extreme Gradient Boosting* (XGBoost) diimplementasikan pada penelitian ini sebagai metode pembanding terhadap *Optimized Ensemble* RBFN dalam memprediksi *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV). Implementasi menggunakan Python (Google Colab) dengan pustaka XGBoost, terhadap 222 sampel dari 37 saluran unik pada enam level pembebanan (50%-100%) dan 64 fitur hasil *feature engineering* (ringkasan pembebanan saluran, profil tegangan, dan indikator kontingensi). Berbeda dari RBFN yang memakai satu jaringan untuk dua keluaran, PIP dan PIV dimodelkan menggunakan dua regressor XGBoost terpisah agar setiap model dapat berfokus pada karakteristik targetnya masing-masing.

4.7.1 Konfigurasi Parameter Model XGBoost

Hiperparameter dioptimasi dengan *RandomizedSearchCV* (40 kombinasi) menggunakan skema *GroupKFold* 5-fold yang mengelompokkan data berdasarkan saluran kontingensi bukan *Grid Search* konvensional maupun k-fold acak agar data dari saluran yang sama tidak muncul bersamaan pada subset latih dan validasi. Pilihan ini penting karena tanpa pengelompokan per-saluran, model dapat “mengingat” karakteristik saluran tertentu alih-alih mempelajari pola yang benar-benar dapat digeneralisasi ke saluran baru, sehingga estimasi performa selama tuning menjadi terlalu optimistis. Parameter terbaik hasil optimasi terpisah untuk PIP dan PIV ditunjukkan pada Tabel 4.20 dan Tabel 4.21.

Tabel 4. 20 Parameter Model XGBoost untuk Prediksi PIP

Parameter	Nilai
n_estimators	300
max_depth	3
learning_rate	0,1
min_child_weight	3
subsample	0,70
colsample_bytree	0,80
gamma	0,0
reg_alpha	0,01
reg_lambda	1,0
objective	reg,squarederror

Tabel 4. 21 Parameter Model XGBoost untuk Prediksi PIV

Parameter	Nilai
n_estimators	300
max_depth	3
learning_rate	0,1
min_child_weight	3
subsample	0,70
colsample_bytree	0,80
gamma	0,0
reg_alpha	0,01
reg_lambda	1,0
objective	reg,squarederror
tree_method	hist

Kedua model memperoleh konfigurasi optimal yang identik, kedalaman pohon dangkal ($max_depth = 3$) dan $learning\ rate$ moderat (0,1) yang dikendalikan oleh regularisasi $subsample$, $colsample_bytree$, reg_alpha , dan reg_lambda . Kesamaan ini konsisten dengan rasio fitur terhadap sampel yang tinggi (64,222), pohon yang dangkal cukup untuk menangkap pola dominan tanpa membuka ruang bagi *overfitting* pada dimensi fitur yang besar relatif terhadap jumlah data latih.

4.7.2 Hasil Evaluasi Model XGBoost

Evaluasi model XGBoost dilakukan menggunakan data pengujian yang terdiri atas 37 skenario kontingensi N-1 pada kondisi pembebanan 100%. Pengujian dilakukan secara terpisah untuk prediksi *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV). Nilai aktual hasil simulasi *Newton Raphson Load Flow* (NRLF) dibandingkan dengan nilai prediksi XGBoost menggunakan metrik *Root Mean Square Error* (RMSE), *Mean Absolute Error* (MAE), koefisien determinasi (R^2), dan *Mean Absolute Percentage Error* (MAPE).

Tabel 4. 22 Tabel Metrik Evaluasi Simulasi XGBoost untuk PIP

No.	Saluran	Aktual (X)	Prediksi XGBoost (Y)	$ X - Y $	$(X - Y)^2$	$(X - \bar{X})^2$
1	L 1-2	0,7646	0,751326	0,013274	0,00017620	0,07520787
2	L 2-5	0,7448	0,746180	0,001380	0,00000190	0,06473999
3	L 4-12	0,7409	0,742299	0,001399	0,00000196	0,06277056
4	L 4-6	0,5459	0,548516	0,002616	0,00000684	0,00308475
5	L 1-3	0,5418	0,548167	0,006367	0,00004054	0,00264613
6	L 6-8	0,5412	0,537107	0,004093	0,00001675	0,00258476
7	L 3-4	0,5363	0,537547	0,001247	0,00000156	0,00211053
8	L 2-6	0,5165	0,512916	0,003584	0,00001285	0,00068333
9	L 6-28	0,5029	0,499826	0,003074	0,00000945	0,00015727
10	L 10-20	0,4896	0,490334	0,000734	0,00000054	0,00000058
11	L 10-21	0,4892	0,488872	0,000328	0,00000011	0,00000134
12	L 12-15	0,4851	0,488965	0,003865	0,00001494	0,00002766
13	L 9-10	0,4805	0,481805	0,001305	0,00000170	0,00009721
14	L 6-9	0,4772	0,476271	0,000929	0,00000086	0,00017317

No.	Saluran	Aktual (X)	Prediksi XGBoost (Y)	X - Y	(X - Y) ²	(X - $\bar{X}$) ²
15	L 19-20	0,4680	0,467618	0,000382	0,00000015	0,00049995
16	L 12-14	0,4669	0,465776	0,001124	0,00000126	0,00055035
17	L 10-22	0,4605	0,453577	0,006923	0,00004793	0,00089159
18	L 27-30	0,4599	0,459422	0,000478	0,00000023	0,00092778
19	L 12-16	0,4585	0,460145	0,001645	0,00000271	0,00101503
20	L 6-10	0,4565	0,453528	0,002972	0,00000883	0,00114646
21	L 6-7	0,4531	0,453438	0,000338	0,00000011	0,00138827
22	L 15-18	0,4517	0,452305	0,000605	0,00000037	0,00149455
23	L 25-27	0,4493	0,449492	0,000192	0,00000004	0,00168588
24	L 15-23	0,4484	0,448864	0,000464	0,00000022	0,00176060
25	L 10-17	0,4480	0,447664	0,000336	0,00000011	0,00179432
26	L 27-29	0,4474	0,450402	0,003002	0,00000901	0,00184552
27	L 22-24	0,4474	0,447907	0,000507	0,00000026	0,00184552
28	L 16-17	0,4426	0,443631	0,001031	0,00000106	0,00228097
29	L 21-22	0,4401	0,438980	0,001120	0,00000125	0,00252601
30	L 29-30	0,4397	0,441955	0,002255	0,00000509	0,00256638
31	L 14-15	0,4397	0,439678	0,000022	0,00000000	0,00256638
32	L 18-19	0,4386	0,439678	0,001078	0,00000116	0,00267904
33	L 24-25	0,4382	0,437670	0,000530	0,00000028	0,00272061
34	L 23-24	0,4381	0,439561	0,001461	0,00000213	0,00273105
35	L 8-28	0,4372	0,438489	0,001289	0,00000166	0,00282593
36	L 5-7	0,4315	0,432211	0,000711	0,00000051	0,00346444
37	L 2-4	0,4255	0,428727	0,003227	0,00001041	0,00420675
	Total			0,075887	0,00038098	0,25969851

$$\bar{X} = 0.4904$$

$$RMSE = \sqrt{\frac{0,000381}{37}} = 0,003209$$

$$MAE = \frac{0,075887}{37} = 0,002051$$

$$R^2 = 1 - \frac{0,000381}{0,259699} = 0,9985$$

$$MAPE = 0,40\%$$

Berdasarkan Tabel 4.22, hasil prediksi XGBoost terhadap nilai PIP pada data pengujian terbaru menunjukkan tingkat kesesuaian yang sangat tinggi terhadap nilai aktual. Seluruh kontingensi memiliki galat prediksi yang sangat kecil, dengan galat terbesar terjadi pada saluran L1-2 sebesar 0,013274, jauh lebih kecil dibandingkan galat pada evaluasi sebelumnya. Hasil evaluasi menghasilkan RMSE sebesar 0,003209, MAE sebesar 0,002051, R^2 sebesar 0,9985, dan MAPE sebesar 0,40%. Nilai R^2 yang sangat mendekati satu menunjukkan bahwa hampir seluruh variasi nilai PIP pada data pengujian mampu dijelaskan oleh model XGBoost, sementara nilai MAPE yang jauh di bawah 1% mengindikasikan tingkat kesalahan prediksi yang sangat rendah.

Tabel 4. 23 Tabel Metrik Evaluasi Simulasi XGBoost untuk PIV

No.	Saluran	Aktual (X)	Prediksi XGBoost (Y)	$ X - Y $	$(X - Y)^2$	$(X - \bar{X})^2$
1	L 27-29	2,6823	2,660335	0,021965	0,00048246	0,39688297
2	L 27-30	2,6548	2,640487	0,014313	0,00020486	0,36298997
3	L 6-7	2,2505	2,261056	0,010556	0,00011143	0,03927788
4	L 10-20	2,2259	2,242281	0,016381	0,00026834	0,03013227
5	L 3-4	2,2135	2,197691	0,015809	0,00024992	0,02598108
6	L 22-24	2,2097	2,217364	0,007664	0,00005874	0,02477051
7	L 29-30	2,1834	2,177294	0,006106	0,00003728	0,01718367
8	L 19-20	2,1778	2,173461	0,004339	0,00001883	0,01574686
9	L 6-28	2,1621	2,164109	0,002009	0,00000404	0,01205307
10	L 24-25	2,1493	2,145829	0,003471	0,00001205	0,00940638
11	L 15-23	2,1321	2,141104	0,009004	0,00008107	0,00636588
12	L 23-24	2,1250	2,106606	0,018394	0,00033834	0,00528333
13	L 25-27	2,1088	2,104695	0,004105	0,00001685	0,00319072
14	L 21-22	2,1078	2,113493	0,005693	0,00003241	0,00307875
15	L 14-15	2,1039	2,101853	0,002047	0,00000419	0,00266117
16	L 18-19	2,1024	2,102021	0,000379	0,00000014	0,00250866
17	L 16-17	2,0985	2,098667	0,000167	0,00000003	0,00213319
18	L 5-7	2,0604	2,067848	0,007448	0,00005547	0,00006539
19	L 10-22	2,0559	2,069838	0,013938	0,00019427	0,00001286

No.	Saluran	Aktual (X)	Prediksi XGBoost (Y)	X - Y	(X - Y) ²	(X - $\bar{X}$) ²
20	L 15-18	2,0527	2,043634	0,009066	0,00008219	0,00000015
21	L 8-28	2,0519	2,052626	0,000726	0,00000053	0,00000017
22	L 10-17	2,0471	2,049340	0,002240	0,00000502	0,00002718
23	L 6-8	2,0212	2,025981	0,004781	0,00002286	0,00096805
24	L 10-21	2,0189	2,010032	0,008868	0,00007864	0,00111646
25	L 4-6	1,9580	1,965023	0,007023	0,00004932	0,00889504
26	L 12-14	1,9341	1,933150	0,000950	0,00000090	0,01397443
27	L 6-10	1,9144	1,911994	0,002406	0,00000579	0,01902014
28	L 6-9	1,9086	1,908007	0,000593	0,00000035	0,02065357
29	L 12-16	1,9077	1,922058	0,014358	0,00020615	0,02091307
30	L 2-5	1,8739	1,878429	0,004529	0,00002051	0,03183138
31	L 12-15	1,8700	1,848122	0,021878	0,00047865	0,03323822
32	L 2-4	1,8661	1,863283	0,002817	0,00000794	0,03467547
33	L 2-6	1,8646	1,869895	0,005295	0,00002804	0,03523636
34	L 1-3	1,8293	1,829311	0,000011	0,00000000	0,04973503
35	L 1-2	1,7405	1,744540	0,004040	0,00001632	0,09722767
36	L 9-10	1,6579	1,655063	0,002837	0,00000805	0,15556202
37	L 4-12	1,6146	1,628795	0,014195	0,00020150	0,19159312
	Total			0,270401	0,00338347	1,67439214

$$\bar{X} = 2.0523$$

$$RMSE = \sqrt{\frac{0,003383}{37}} = 0,009563$$

$$MAE = \frac{0,270401}{37} = 0,007308$$

$$R^2 = 1 - \frac{0,003383}{1,674392} = 0,9980$$

$$MAPE = 0,35\%$$

Berdasarkan Tabel 4.23, hasil prediksi XGBoost terhadap nilai PIV pada data pengujian terbaru menunjukkan tingkat kesesuaian yang sangat baik dengan nilai aktual. Galat terbesar terjadi pada saluran L27-29 sebesar 0,021965, diikuti

oleh L12-15 sebesar 0,021878 dan L23-24 sebesar 0,018394. Hasil evaluasi menghasilkan RMSE sebesar 0,009563, MAE sebesar 0,007308, R^2 sebesar 0,9980, dan MAPE sebesar 0,35%. Nilai koefisien determinasi yang mendekati satu menunjukkan bahwa hampir seluruh variasi nilai PIV dapat direpresentasikan dengan sangat baik oleh model XGBoost.

Berdasarkan Tabel 4.24, model XGBoost menunjukkan kemampuan prediksi yang sangat baik terhadap kedua indeks keamanan statis pada data pengujian yang terdiri atas 37 skenario kontingensi N-1 pada tingkat pembebanan 100%. Untuk prediksi PIP, diperoleh R^2 sebesar 0,9985 dengan MAPE sebesar 0,40%, menunjukkan bahwa model mampu mengikuti pola perubahan indeks pembebanan saluran dengan tingkat kesalahan yang sangat rendah. Pada prediksi PIV, model menghasilkan R^2 sebesar 0,9980 dan MAPE sebesar 0,35%, yang mengindikasikan tingkat kesalahan prediksi yang sangat rendah pada kedua target.

Tabel 4. 24 Metrik Evaluasi Model XGBoost

Metrik	PIP Train	PIP Test	PIV Train	PIV Test
R^2	1,0000	0,9985	0,9998	0,9980
RMSE	0,000598	0,003209	0,016337	0,009563
MAE	0,000443	0,002051	0,006134	0,007308
MAPE	0,21%	0,40%	0,18%	0,35%
Max Error	0,002447	0,013274	0,189092	0,021965
R^2	1,0000	0,9985	0,9998	0,9980

Dibandingkan dengan hasil evaluasi sebelumnya, model XGBoost pada pengujian terbaru ini menunjukkan peningkatan performa yang signifikan pada kedua target prediksi, baik dari sisi koefisien determinasi maupun galat prediksi (RMSE, MAE, dan MAPE). Hasil prediksi ini selanjutnya digunakan pada tahap perbandingan dengan model *Optimized Ensemble* RBFN untuk mengevaluasi akurasi prediksi, kemampuan perankingan kontingensi, serta kompleksitas implementasi kedua metode.

4.8 Perbandingan Kinerja Model *Extreme Gradient Boosting* (XGBoost) dan *Optimized Ensemble Radial Basis Function Network* (RBFN)

Subbab ini menyajikan analisis komparatif antara model *Extreme Gradient Boosting* (XGBoost) dan *Optimized Ensemble Radial Basis Function Network* (RBFN) yang telah diimplementasikan pada Subbab 4.6 dan Subbab 4.7. Perbandingan dilakukan untuk menjawab tujuan penelitian, yaitu membandingkan kemampuan kedua metode dalam memprediksi PIP dan PIV, mengevaluasi kesesuaian hasil perankingan kontingensi terhadap metode *Newton Raphson Load Flow* (NRLF) sebagai *ground truth*, serta meninjau kemampuan generalisasi masing-masing model.

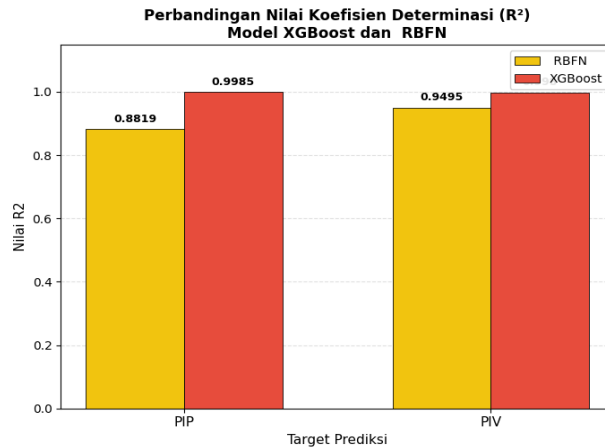
Sebagai representasi metode RBFN, penelitian ini menggunakan *Optimized Ensemble* RBFN, karena hasil pada Subbab 4.6 menunjukkan bahwa implementasi RBFN berdasarkan parameter referensi [7] belum memberikan performa yang optimal pada dataset penelitian. Seluruh proses evaluasi dilakukan menggunakan 37 skenario kontingensi N-1 pada kondisi pembebanan 100%, sehingga kedua model dibandingkan menggunakan dataset pengujian yang sama.

4.8.1 Perbandingan Akurasi Prediksi PIP dan PIV

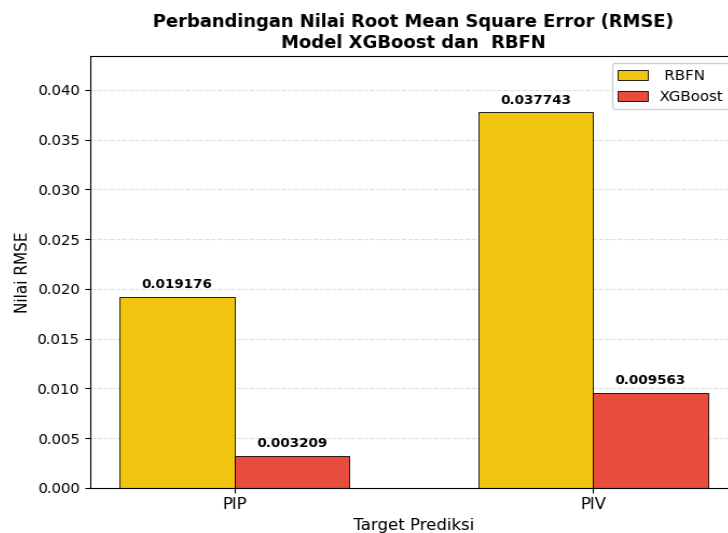
Untuk mengevaluasi kemampuan prediksi masing-masing model, digunakan empat metrik evaluasi, yaitu Koefisien Determinasi (R^2), (RMSE), (MAE), dan (MAPE). Hasil evaluasi kedua model pada data pengujian disajikan pada Tabel 4.25.

Tabel 4. 25 Perbandingan Metrik Evaluasi XGBoost dan *Optimized Ensemble* RBFN

Metrik	<i>Optimized Ensemble</i> RBFN (PIP)	XGBoost (PIP)	<i>Optimized Ensemble</i> RBFN (PIV)	XGBoost (PIV)
R^2	0,8819	0,9985	0,9495	0,9980
MSE	0,028793	0,0000103	0,047796	0,0000915
RMSE	0,019176	0,003209	0,037743	0,009563
MAE	0,0243	0,002051	0,0374	0,007308
MAPE	4,48%	0,40%	1,86%	0,35%



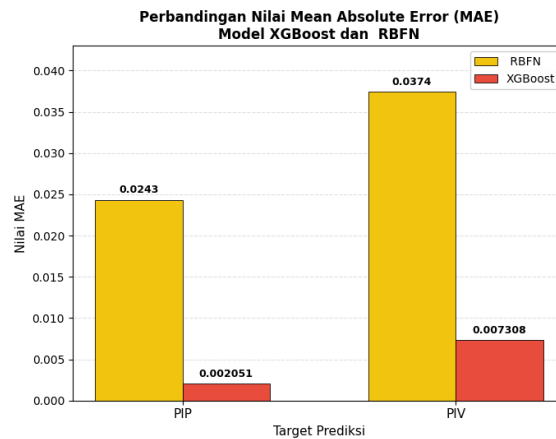
Gambar 4. 4 Perbandingan Nilai Koefisien Determinasi (R^2) Model XGBoost dan *Optimized Ensemble RBFN*



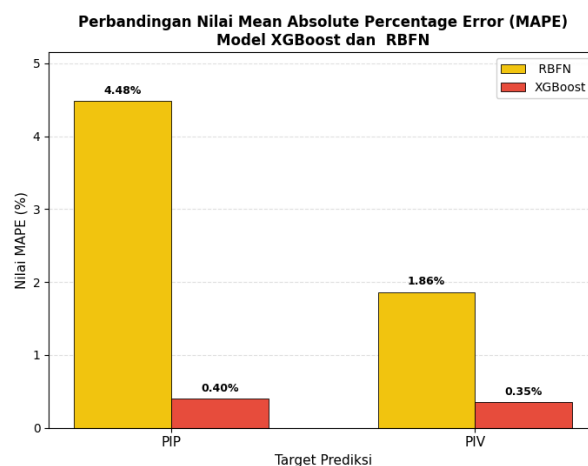
Gambar 4. 5 Perbandingan Nilai *Root Mean Square Error* (RMSE) Model XGBoost dan *Optimized Ensemble RBFN*

Berdasarkan Tabel 4.25, XGBoost unggul konsisten pada seluruh metrik dan kedua target, RMSE turun 83% (PIP) dan 75% (PIV) relatif terhadap RBFN, sedangkan MAPE turun dari 4,48% menjadi 0,40% (PIP) dan dari 1,86% menjadi 0,35% (PIV). Perbedaan ini bersumber dari mekanisme pembelajaran yang berlawanan sifatnya. *Gradient boosting* membangun pohon secara *supervised* setiap pemisahan dipilih untuk memaksimalkan pengurangan galat terhadap label sehingga mampu menangkap ambang batas nonlinier tajam, seperti titik saat

pembebanan saluran mendekati batas termal. Sebaliknya, RBFN membentuk *center* secara *unsupervised* melalui K-Means++ (tidak mempertimbangkan label PIP/PIV), lalu menghaluskan prediksi melalui kombinasi fungsi basis Gaussian dan rata-rata *Bagging* atas lima model. Efek *smoothing* berlapis inilah yang menjelaskan mengapa galat RBFN pada PIP (MAPE 4,48%) jauh lebih besar daripada pada PIV (MAPE 1,86%), PIP terkait pembebanan saluran yang berperilaku hampir seperti ambang (mendekati batas termal secara tajam pada saluran tertentu), sedangkan PIV berupa deviasi tegangan yang berubah lebih kontinu dan lebih sesuai dengan sifat interpolasi mulus fungsi basis radial.



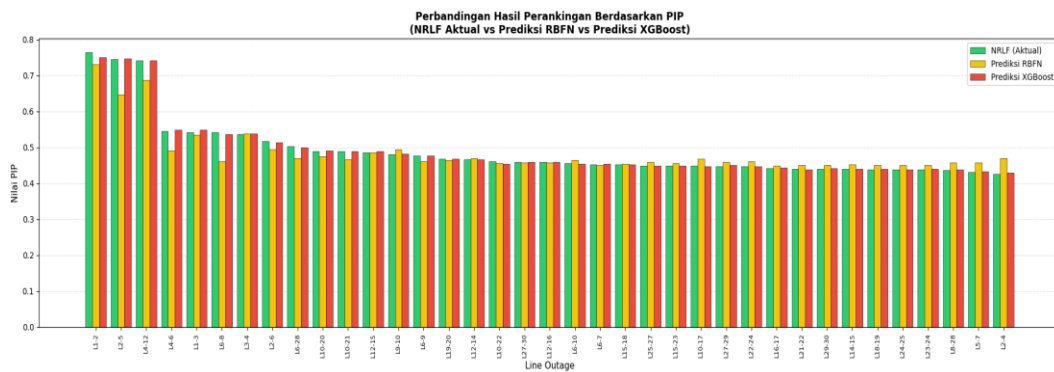
Gambar 4. 6 Perbandingan Nilai *Mean Absolute Error* (MAE) Model XGBoost dan *Optimized Ensemble RBFN*



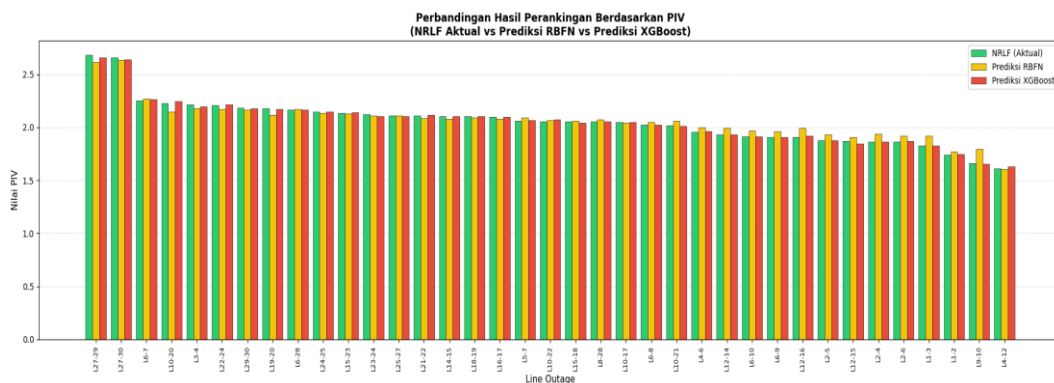
Gambar 4. 7 Perbandingan Nilai *Mean Absolute Percentage Error* (MAPE) Model XGBoost dan *Optimized Ensemble RBFN*

4.8.2 Kesesuaian Perankingan Kontingensi N-1 terhadap NRLF Menggunakan Capture Ratio

Selain mengevaluasi akurasi prediksi menggunakan metrik regresi, penelitian ini juga mengevaluasi kemampuan *Optimized Ensemble Radial Basis Function Network* (RBFN) dan *Extreme Gradient Boosting* (XGBoost) dalam mempertahankan urutan prioritas kontingensi terhadap hasil simulasi *Newton Raphson Load Flow* (NRLF) sebagai *ground truth*. Evaluasi dilakukan pada seluruh 37 skenario kontingensi N-1 pada kondisi pembebanan 100%. Perbandingan nilai *Active Power Performance Index* (PIP) dan *Voltage Performance Index* (PIV) antara hasil NRLF dengan kedua model ditunjukkan pada Gambar 4.8 dan Gambar 4.9.



Gambar 4. 8 Perbandingan Hasil Prediksi PIP terhadap NRLF Menggunakan *Optimized Ensemble* RBFN dan XGBoost



Gambar 4. 9 Perbandingan Hasil Prediksi PIV terhadap NRLF Menggunakan *Optimized Ensemble* RBFN dan XGBoost

Berdasarkan kedua gambar tersebut, baik Optimized Ensemble RBFN maupun XGBoost mampu mengikuti pola perubahan nilai Performance Index yang dihasilkan oleh NRLF. Namun, secara visual terlihat bahwa prediksi XGBoost lebih konsisten terhadap nilai aktual, sedangkan Optimized Ensemble RBFN masih mengalami penyimpangan pada beberapa kontingensi sehingga menyebabkan perubahan urutan prioritas. Untuk mengevaluasi kemampuan model dalam mempertahankan urutan kontingensi kritis secara kuantitatif, digunakan metrik *Capture Ratio* (CR) sebagaimana ditunjukkan pada Tabel 4.26.

Tabel 4. 26 Perbandingan *Capture Ratio* Perankingan Kontingensi terhadap NRLF

Target	Model	N=5	N=10
PIP	<i>Optimized Ensemble RBFN</i>	80% (4/5)	80% (8/10)
PIP	XGBoost	100% (5/5)	100% (10/10)
PIV	<i>Optimized Ensemble RBFN</i>	80% (4/5)	90% (9/10)
PIV	XGBoost	80% (4/5)	100% (10/10)

Berdasarkan Tabel 4.26, XGBoost berhasil mempertahankan seluruh kontingensi kritis pada evaluasi Top-10 baik untuk target PIP maupun PIV, sedangkan Optimized Ensemble RBFN masih kehilangan dua kontingensi pada target PIP dan satu kontingensi pada target PIV. Hasil tersebut menunjukkan bahwa XGBoost lebih konsisten dalam mempertahankan prioritas kontingensi yang dihasilkan oleh simulasi NRLF.

Hasil penelitian menunjukkan bahwa kegagalan tersebut dipengaruhi oleh dua kondisi, yaitu karakteristik sistem tenaga setelah terjadinya line outage dan kemampuan model dalam merepresentasikan perubahan tersebut. Pada target PIP, hasil perbandingan sepuluh kontingensi teratas ditunjukkan pada Tabel 4.27. Berdasarkan Tabel 4.27, kontingensi L6–8 merupakan kontingensi yang mengalami perubahan peringkat paling signifikan. Berdasarkan hasil simulasi NRLF, kontingensi ini berada pada peringkat ke-6 dengan nilai PIP sebesar 0,5412, sedangkan hasil prediksi Optimized Ensemble RBFN menempatkannya pada peringkat ke-19 dengan nilai PIP sebesar 0,4619. Saluran L6–8 menghubungkan

Bus 6 dengan Bus 8, di mana Bus 8 merupakan bus generator (PV). Pemutusan saluran tersebut menyebabkan redistribusi aliran daya ke beberapa lintasan alternatif sehingga pembebanan pada saluran lain meningkat dan menghasilkan nilai PIP yang tinggi berdasarkan hasil simulasi NRLF.

Namun, hasil prediksi Optimized Ensemble RBFN menunjukkan nilai PIP yang lebih rendah dibandingkan hasil NRLF sehingga kontingensi L6–8 kehilangan prioritas pada kelompok Top-10. Kondisi ini mengindikasikan bahwa model mengestimasi tingkat keparahan kontingensi lebih rendah (*underestimation*) dibandingkan kondisi aktual. Salah satu faktor yang diduga memengaruhi kondisi tersebut adalah karakteristik representasi RBFN yang dibentuk menggunakan fungsi basis Gaussian, sehingga hubungan antara pola kondisi sistem dan nilai PIP pada kontingensi L6–8 belum dapat dipetakan secara optimal. Akibatnya, nilai PIP hasil prediksi menjadi lebih rendah dibandingkan hasil simulasi NRLF dan menyebabkan peringkat kontingensi bergeser dari urutan ke-6 menjadi ke-19.

Tabel 4. 27 Perbandingan Sepuluh Kontingensi Teratas Berdasarkan Nilai PIP

Peringkat	NRLF (Aktual) PIP	Prediksi Optimized Ensemble RBFN	Prediksi XGBoost
1	L1-2 (0,7646)	L1-2 (0,7319)	L1-2 (0,751326)
2	L2-5 (0,7448)	L4-12 (0,6872)	L2-5 (0,746180)
3	L4-12 (0,7409)	L2-5 (0,6466)	L4-12 (0,742299)
4	L4-6 (0,5459)	L3-4 (0,5388)	L4-6 (0,548516)
5	L1-3 (0,5418)	L1-3 (0,5350)	L1-3 (0,548167)
6	L6-8 (0,5412)	L2-6 (0,4951)	L3-4 (0,537547)
7	L3-4 (0,5363)	L9-10 (0,4943)	L6-8 (0,537107)
8	L2-6 (0,5165)	L4-6 (0,4913)	L2-6 (0,512916)
9	L6-28 (0,5029)	L12-15 (0,4861)	L6-28 (0,499826)
10	L10-20 (0,4896)	L10-20 (0,4756)	L10-20 (0,490334)

Berbeda dengan L6–28, hasil penelitian menunjukkan bahwa perubahan peringkat tidak disebabkan oleh *underestimation* yang signifikan, melainkan oleh kedekatan nilai PIP antar kontingensi. Berdasarkan hasil simulasi NRLF, L6–28 berada pada peringkat ke-9, sedangkan hasil prediksi Optimized Ensemble RBFN menempatkannya pada peringkat ke-12. Selisih nilai PIP pada kelompok peringkat

8 hingga 14 relatif kecil sehingga galat prediksi sekitar 6,61% sudah cukup mengubah urutan peringkat dan menggeser L6–28 keluar dari kelompok Top-10. Dengan demikian, perubahan peringkat pada kontingensi ini lebih dipengaruhi oleh distribusi nilai Performance Index yang saling berdekatan dibandingkan oleh keterbatasan model.

Tabel 4. 28 Perbandingan Sepuluh Kontingensi Teratas Berdasarkan Nilai PIV

Peringkat	NRLF (Aktual) PIV	Prediksi Optimized Ensemble RBFN	Prediksi XGBoost
1	L27-29 (2,6823)	L27-30 (2,6343)	L27-29 (2,660335)
2	L27-30 (2,6548)	L27-29 (2,6147)	L27-30 (2,640487)
3	L6-7 (2,2505)	L6-7 (2,2699)	L6-7 (2,261056)
4	L10-20 (2,2259)	L3-4 (2,1801)	L10-20 (2,242281)
5	L3-4 (2,2135)	L22-24 (2,1728)	L22-24 (2,217364)
6	L22-24 (2,2097)	L6-28 (2,1716)	L3-4 (2,197691)
7	L29-30 (2,1834)	L29-30 (2,1658)	L29-30 (2,177294)
8	L19-20 (2,1778)	L10-20 (2,1445)	L19-20 (2,173461)
9	L6-28 (2,1621)	L24-25 (2,1325)	L6-28 (2,164109)
10	L24-25 (2,1493)	L15-23 (2,1300)	L24-25 (2,145829)

Pada target PIV, Optimized Ensemble RBFN tidak berhasil mempertahankan kontingensi L19–20 pada kelompok Top-10. Berdasarkan hasil simulasi NRLF, kontingensi tersebut berada pada peringkat ke-8, sedangkan hasil prediksi Optimized Ensemble RBFN menempatkannya pada peringkat ke-11. Saluran L19–20 menghubungkan Bus 19 dengan Bus 20 pada bagian hilir sistem sehingga pemutusannya memengaruhi distribusi daya reaktif dan profil tegangan di wilayah tersebut. Perubahan tersebut menyebabkan nilai PIV meningkat sehingga kontingensi ini termasuk dalam kelompok prioritas berdasarkan hasil simulasi NRLF.

Berbeda dengan kasus L6–8, hasil prediksi pada kontingensi L19–20 masih cukup mendekati nilai aktual dengan galat sebesar 2,88%. Pergeseran peringkat terjadi karena nilai PIV pada kelompok peringkat 8 hingga 11 memiliki selisih yang sangat kecil, sehingga perbedaan prediksi yang relatif kecil sudah cukup mengubah urutan prioritas. Dengan demikian, perubahan peringkat pada L19–20 lebih

dipengaruhi oleh karakteristik distribusi nilai PIV hasil NRLF dibandingkan oleh keterbatasan kemampuan prediksi model.

Sementara itu, pada model XGBoost, Capture Ratio belum mencapai 100% pada evaluasi Top-5 PIV karena kontingensi L3–4 bertukar posisi dengan L22–24. Berdasarkan hasil simulasi NRLF, kedua kontingensi tersebut memiliki nilai PIV yang hampir identik dengan selisih sekitar 0,0038, sehingga perbedaan prediksi yang sangat kecil sudah cukup menyebabkan perubahan urutan peringkat. Namun, ketika evaluasi diperluas menjadi Top-10, kedua kontingensi tetap berada dalam kelompok yang sama sehingga Capture Ratio XGBoost kembali mencapai 100%. Temuan ini menunjukkan bahwa perubahan peringkat lebih dipengaruhi oleh kedekatan nilai aktual daripada keterbatasan kemampuan model.

Secara keseluruhan, hasil evaluasi Capture Ratio menunjukkan bahwa XGBoost lebih konsisten dalam mempertahankan urutan kontingensi kritis dibandingkan Optimized Ensemble RBFN. Hasil penelitian juga menunjukkan bahwa perubahan peringkat kontingensi tidak selalu mencerminkan rendahnya kemampuan prediksi model. Pada kontingensi L6–8, perubahan peringkat terutama disebabkan oleh underestimation nilai PIP sehingga tingkat keparahan kontingensi diprediksi lebih rendah dibandingkan hasil simulasi NRLF. Sebaliknya, pada kontingensi L6–28, L19–20, dan L3–4, perubahan peringkat lebih dipengaruhi oleh kedekatan nilai Performance Index di sekitar batas evaluasi Top-*N*. Akibatnya, perbedaan prediksi yang relatif kecil sudah cukup mengubah urutan prioritas meskipun nilai prediksi masih mendekati hasil simulasi NRLF. Temuan ini menjelaskan bahwa Capture Ratio dipengaruhi tidak hanya oleh akurasi prediksi model, tetapi juga oleh distribusi nilai Performance Index antar kontingensi.

4.8.3 Perbandingan Kemampuan Generalisasi dan Waktu Komputasi

Kemampuan generalisasi menunjukkan sejauh mana suatu model mampu mempertahankan performanya ketika digunakan untuk memprediksi data pada kondisi operasi yang sama sekali tidak dilibatkan selama proses pelatihan. Pengujian generalisasi pada penelitian ini dilakukan menggunakan 37 skenario kontingensi pada kondisi pembebanan 85%, yaitu level pembebanan yang tidak diikutsertakan pada data pelatihan maupun data pengujian utama model (yang

mencakup level 50%-100% dengan interval tertentu, tidak termasuk 85%). Dengan demikian, pengujian ini merupakan pengujian generalisasi yang lebih ketat dibandingkan evaluasi pada Subbab 4.6, 4.7, dan 4.8.1-4.8.2, karena model diuji pada kondisi pembebanan yang benar-benar baru. Hasil evaluasi kedua model pada data uji 85% ini disajikan pada Tabel 4.28.

Berdasarkan Tabel 4.28, pada prediksi *Active Power Performance Index* (PIP), model *Optimized Ensemble* RBFN menghasilkan performa yang lebih baik dibandingkan XGBoost pada kondisi pembebanan 85% yang belum pernah dilihat model sebelumnya. RBFN memperoleh R^2 sebesar 0,8978 dengan MAPE sebesar 3,43%, sedangkan XGBoost memperoleh R^2 sebesar 0,8314 dengan MAPE sebesar 5,14%. Hasil ini berbeda dari hasil evaluasi pada 37 kontingensi level 100% (Tabel 4.24), di mana XGBoost menunjukkan performa yang lebih baik. Perbedaan ini mengindikasikan bahwa pada kondisi pembebanan yang terletak di antara rentang data pelatihan (interpolasi), model RBFN yang berbasis fungsi basis radial memiliki kemampuan generalisasi yang lebih stabil dibandingkan XGBoost yang berbasis pohon keputusan.

Tabel 4. 29 Perbandingan Metrik Evaluasi Model pada 37 Kontingensi Level Beban 85%

Metrik	Optimized Ensemble RBFN (PIP)	XGBoost (PIP)	Optimized Ensemble RBFN (PIV)	XGBoost (PIV)
R^2	0,8978	0,8314	0,9173	0,9142
RMSE	0,018092	0,023239	0,063366	0,064572
MAE	0,011905	0,018482	0,050878	0,050552
MAPE	3,43%	5,14%	1,91%	1,86%

Pada prediksi *Voltage Performance Index* (PIV), kedua model menunjukkan performa yang relatif setara pada data uji 85%. *Optimized Ensemble* RBFN memperoleh R^2 sebesar 0,9173 dengan MAPE sebesar 1,91%, sedangkan XGBoost memperoleh R^2 sebesar 0,9142 dengan MAPE sebesar 1,86%. Nilai R^2 yang sedikit lebih tinggi diperoleh RBFN, namun nilai MAPE yang sedikit lebih rendah diperoleh XGBoost, sehingga kedua model dapat dikatakan memiliki kemampuan

generalisasi yang setara untuk prediksi PIV pada kondisi pembebanan yang tidak dilibatkan dalam pelatihan.

Tabel 4. 30 *Capture Ratio* Model pada 37 Kontingensi Level Beban 85%

Target	Model	N=5	N=10
PIP	<i>Optimized Ensemble</i> RBFN	60% (3/5)	80% (8/10)
PIP	XGBoost	60% (3/5)	80% (8/10)
PIV	<i>Optimized Ensemble</i> RBFN	100% (5/5)	90% (9/10)
PIV	XGBoost	80% (4/5)	100% (10/10)

Berdasarkan Tabel 4.29, pada prediksi PIP kedua model memperoleh *Capture Ratio* yang identik, yaitu 60% pada Top-5 (3 dari 5 kontingensi) dan 80% pada Top-10 (8 dari 10 kontingensi). Pada prediksi PIV, *Optimized Ensemble* RBFN berhasil mengidentifikasi seluruh 5 kontingensi teratas dengan tepat (*Capture Ratio* 100%) namun sedikit menurun menjadi 90% pada Top-10, sedangkan XGBoost memperoleh *Capture Ratio* sebesar 80% pada Top-5 dan justru mencapai 100% pada Top-10. Hasil ini menunjukkan bahwa pada data uji 85%, kedua model tetap memiliki kemampuan yang baik dalam mengidentifikasi kontingensi kritis meskipun diuji pada kondisi pembebanan yang tidak dilibatkan dalam pelatihan, dengan keunggulan yang bervariasi tergantung pada target prediksi dan jumlah kontingensi teratas yang ditinjau.

Kemampuan generalisasi tersebut didukung oleh mekanisme pembelajaran yang berbeda pada masing-masing model. XGBoost memanfaatkan mekanisme gradient boosting yang dipadukan dengan parameter regularisasi (reg_alpha, reg_lambda), subsample, dan colsample_bytree hasil grid search bertahap (Tabel 4.19 dan Tabel 4.20), serta early stopping berbasis validation set untuk mencegah overfitting. Sementara itu, *Optimized Ensemble* RBFN meningkatkan kemampuan generalisasi melalui kombinasi inisialisasi center dengan K-Means++, adaptive sigma, regularisasi Ridge pada pembaruan bobot, early stopping berbasis patience, serta teknik Bootstrap Aggregating (Bagging) atas 5 model. Kedua pendekatan

pengembangan tersebut terbukti efektif menjaga performa model pada data pembebanan 85% yang sama sekali tidak dilibatkan dalam pelatihan.

Dari sisi kompleksitas implementasi, kedua metode memiliki karakteristik yang berbeda. Model XGBoost membangun dua model regresi secara terpisah untuk prediksi PIP dan PIV, masing-masing melalui proses grid search bertahap (Tabel 4.19 dan Tabel 4.20). Sementara itu, *Optimized Ensemble* RBFN membangun lima model RBFN untuk setiap target melalui mekanisme Bagging, kemudian menggabungkan hasil prediksi menggunakan ensemble averaging. Implementasi *Optimized Ensemble* RBFN juga melibatkan tahapan tambahan, seperti penentuan center menggunakan K-Means++, perhitungan adaptive sigma, serta pelatihan bobot menggunakan kombinasi *Least Mean Squares* (LMS) dan regularisasi Ridge. Oleh karena itu, dari sisi struktur implementasi, *Optimized Ensemble* RBFN memiliki proses pembentukan model yang lebih kompleks dibandingkan XGBoost.

Tabel 4. 31 Perbandingan Waktu Komputasi Model *Optimized Ensemble* RBFN dan XGBoost

Aspek Waktu Komputasi	Optimized Ensemble RBFN	XGBoost
Waktu Pelatihan (Training Time)	49 detik	60 detik
Waktu Prediksi Total (37 Sampel Uji, <i>Inference time</i>)	44 detik	21 detik
Waktu Prediksi Rata-rata per Sampel	$\approx 1,19$ detik/sampel	$\approx 0,57$ detik/sampel

Selain kemampuan generalisasi, efisiensi komputasi juga menjadi pertimbangan penting dalam pemilihan model untuk aplikasi praktis, khususnya bila model akan digunakan pada studi keamanan sistem tenaga secara near real time. Pengukuran waktu komputasi dilakukan pada perangkat yang sama untuk kedua model, meliputi waktu pelatihan (training time) dan waktu pengujian atau prediksi (*inference time*) terhadap 37 sampel data uji. Hasil pengukuran waktu komputasi kedua model disajikan pada Tabel 4.30.

Berdasarkan Tabel 4.30, XGBoost membutuhkan waktu pelatihan yang sedikit lebih lama dibandingkan *Optimized Ensemble* RBFN (60 detik berbanding 49 detik), yang wajar mengingat XGBoost menjalankan proses grid search bertahap dan early stopping pada dua model regresi terpisah (PIP dan PIV). Sebaliknya, pada tahap prediksi (inference), XGBoost jauh lebih cepat dibandingkan RBFN, dengan waktu prediksi total sebesar 21 detik berbanding 44 detik untuk 37 sampel uji, atau setara dengan rata-rata 0,57 detik per sampel untuk XGBoost berbanding 1,19 detik per sampel untuk RBFN. Hal ini disebabkan oleh proses prediksi *Optimized Ensemble* RBFN yang melibatkan lima model RBFN sekaligus (Bootstrap Aggregating) yang harus dijalankan dan digabungkan (ensemble averaging) untuk setiap sampel, sedangkan XGBoost hanya perlu melakukan forward pass pada satu struktur pohon keputusan per target prediksi. Dengan demikian, jika aplikasi yang dituju mengutamakan kecepatan prediksi secara near real time, XGBoost lebih unggul dari sisi efisiensi komputasi, meskipun *Optimized Ensemble* RBFN menunjukkan kemampuan generalisasi yang setara atau bahkan lebih baik pada beberapa skema evaluasi, sebagaimana dibahas sebelumnya pada subbab ini.

Secara keseluruhan, hasil pengujian generalisasi pada 37 kontingensi level beban 85% menunjukkan bahwa kedua model, baik *Optimized Ensemble* RBFN maupun XGBoost, mampu mempertahankan performa yang baik pada kondisi pembebanan yang sama sekali tidak dilibatkan dalam proses pelatihan. *Optimized Ensemble* RBFN menunjukkan keunggulan pada prediksi PIP dan kemampuan *Capture Ratio* yang lebih tinggi pada Top-5 PIV, sedangkan XGBoost unggul dari sisi efisiensi waktu prediksi (*inference time*) dan *Capture Ratio* Top-10 PIV. Pemilihan model yang tepat untuk implementasi praktis perlu mempertimbangkan trade-off antara akurasi prediksi pada kondisi pembebanan baru dan efisiensi komputasi sesuai dengan kebutuhan aplikasi.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil simulasi *Newton Raphson Load Flow* (NRLF), implementasi metode *Extreme Gradient Boosting* (XGBoost) dan *Optimized Ensemble Radial Basis Function Network* (RBFN), serta analisis yang telah dilakukan pada sistem IEEE 30-Bus menggunakan 37 skenario kontingensi N-1 pada kondisi pembebanan 100%, diperoleh kesimpulan sebagai berikut.

1. Pada pembebanan 100%, XGBoost lebih unggul dari *Optimized Ensemble RBFN* dalam memprediksi PIP dan PIV. R^2 XGBoost mencapai 0,9985 (PIP) dan 0,9980 (PIV), dibandingkan RBFN sebesar 0,8819 dan 0,9495. MAPE XGBoost juga jauh lebih rendah, yaitu 0,40% (PIP) dan 0,35% (PIV), dibandingkan RBFN sebesar 4,48% dan 1,86%.
2. XGBoost lebih baik dalam mempertahankan perankingan kontingensi terhadap NRLF. Pada PIP, *Capture Ratio* XGBoost mencapai 100% (Top-5 dan Top-10), dibandingkan RBFN yang hanya 80%. Pada PIV, XGBoost memperoleh 80% (Top-5, setara RBFN) dan 100% (Top-10), sedangkan RBFN mencapai 90% pada Top-10.
3. Pengujian generalisasi pada pembebanan 85% (di luar data pelatihan dan pengujian utama) menunjukkan hasil berbeda, RBFN justru unggul untuk PIP (R^2 0,8978 berbanding 0,8314, MAPE 3,43% berbanding 5,14%) dan setara untuk PIV (R^2 0,9173 berbanding 0,9142). Hasil ini menunjukkan RBFN memiliki kemampuan interpolasi yang lebih stabil di antara level beban yang dilatih, sehingga keunggulan XGBoost tidak berlaku mutlak pada seluruh kondisi operasi. Dari sisi kompleksitas implementasi, *Optimized Ensemble RBFN* membutuhkan tahapan yang lebih banyak (penentuan center dengan K-Means++, perhitungan adaptive sigma, serta Bagging atas lima model), sedangkan XGBoost cukup membangun dua model regresi terpisah untuk PIP dan PIV. Dari sisi waktu komputasi, XGBoost sedikit lebih lambat pada tahap namun jauh lebih cepat pada tahap inferensi, sehingga lebih unggul untuk kebutuhan prediksi *near real-time*.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, beberapa saran untuk pengembangan penelitian selanjutnya adalah sebagai berikut.

1. Penelitian selanjutnya disarankan menggunakan sistem tenaga listrik dengan skala yang lebih besar, seperti IEEE 57-Bus, IEEE 118-Bus, atau sistem tenaga listrik nyata, sehingga kinerja metode dapat dievaluasi pada jaringan yang lebih kompleks.
2. Penelitian selanjutnya dapat mengembangkan skenario pengujian dengan mempertimbangkan variasi kondisi operasi sistem, seperti perubahan pola pembebanan, integrasi pembangkit energi terbarukan, maupun kontingensi majemuk (N-2), untuk meningkatkan kemampuan generalisasi model.
3. Penelitian selanjutnya dapat membandingkan XGBoost dan RBFN dengan metode pembelajaran mesin lainnya, seperti LightGBM, CatBoost, Random Forest, atau model Deep Learning, sehingga dapat diperoleh metode yang paling efektif untuk implementasi *Static Security Assessment* secara online.

DAFTAR PUSTAKA

- [1] J. J. Grainger dan W. D. Stevenson, *Power System Analysis*. McGraw-Hill, 2019.
- [2] J. M. H. Arteaga, F. Hancharou, F. Thams, dan S. Chatzivasileiadis, "Deep learning for power system security assessment," dalam 2019 IEEE Milan PowerTech, PowerTech 2019, Institute of Electrical and Electronics Engineers Inc., Jun 2019. doi: 10.1109/PTC.2019.8810906.
- [3] W. D. Stevenson, *Elements of Power System Analysis*. McGraw-Hill, 2020.
- [4] A. A. Abdulrazzaq, "Contingency ranking of power systems using a performance index," *International Research Journal of Engineering and Technology*, Mei 2015. [Daring]. Tersedia pada: www.irjet.net
- [5] Y. Li, Y. Li, dan Y. Sun, "Online static security assessment of power systems based on Lasso algorithm," *Applied Sciences (Switzerland)*, vol. 8, no. 9, Agu 2018, doi: 10.3390/app8091442.
- [6] M. Gholami, M. J. Sanjari, M. Safari, M. Akbari, dan M. R. Kamali, "Static security assessment of power systems: A review," *International Transactions on Electrical Energy Systems*, vol. 30, no. 9, Sep 2020, doi: 10.1002/2050-7038.12432.
- [7] P. Sekhar dan S. Mohanty, "An online power system static security assessment module using multi-layer perceptron and radial basis function network," *International Journal of Electrical Power and Energy Systems*, vol. 76, hlm. 165-173, Mar 2016, doi: 10.1016/j.ijepes.2015.11.009.
- [8] Y. Wang dkk., "Short-term load forecasting of industrial customers based on SVM and XGBoost," *International Journal of Electrical Power and Energy Systems*, vol. 129, Jul 2021, doi: 10.1016/j.ijepes.2021.106830.
- [9] T. Chen dan C. Guestrin, "XGBoost: A scalable tree boosting system," dalam *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Agu 2016, hlm. 785-794, doi: 10.1145/2939672.2939785.
- [10] K. K. Patra, *Contingency Analysis in Power System using Load Flow Solution*. 2015.

- [11] S. Putatunda dan K. Rama, "A comparative analysis of hyperopt as against other approaches for hyper-parameter optimization of XGBoost," dalam ACM International Conference Proceeding Series, Association for Computing Machinery, Nov 2018, hlm. 6-10, doi: 10.1145/3297067.3297080.
- [12] I. S. Saeh dan M. W. Mustafa, "Artificial Neural Network for Power System Static Security Assessment: A Survey," Jurnal Teknologi, 2014. [Daring]. Tersedia pada: www.jurnalteknologi.utm.my
- [13] D. L. Alvarez, M. Gaha, J. Prévost, A. Côté, G. Abdul-Nour, dan T. J. M. Meango, "N – k Static Security Assessment for Power Transmission System Planning Using Machine Learning," Energies (Basel), vol. 17, no. 2, Jan 2024, doi: 10.3390/en17020292.
- [14] Y. Zhang, X. Tian, dan L. Zhang, "A stochastic power flow-based static security assessment under uncertain scenarios," Front. Energy Res., vol. 12, 2024, doi: 10.3389/fenrg.2024.1429160.
- [15] International Energy Agency, "Strengthening Power System Security in Kyrgyzstan: A Roadmap." [Daring]. Tersedia pada: www.iea.org/t&c/
- [16] N. Yaabari, O. Patrick Otubu, dan O.-S. Ojah, "Impact of High Voltage Transmission on I²R Losses Using A Simplified ETAP Model," American Journal of Engineering Research (AJER), vol. 10, hlm. 208-213, 2021. [Daring]. Tersedia pada: www.ajer.org
- [17] A. Y. Abdelaziz, A. T. M. Taha, M. A. Mostafa, dan A. M. Hassan, "Fuzzy Logic Based Power System Contingency Ranking," International Journal of Intelligent Systems and Applications, vol. 5, no. 3, hlm. 1-12, Feb 2013, doi: 10.5815/ijisa.2013.03.01.
- [18] Electrical Technology, "Electric Power System - Generation, Transmission & Distribution of Electricity." [Daring].
- [19] N. Rahmawati, "Analisis Kontingensi Sistem Tenaga Listrik dengan Metode Bounding," Jurusan Teknik Elektro, Fakultas Teknik, 2012.
- [20] B. A. Hemad, N. M. A. Ibrahim, S. A. Fayad, dan H. E. A. Talaat, "Hierarchical Clustering-Based Framework for Interconnected Power

- System Contingency Analysis,” *Energies* (Basel), vol. 15, no. 15, Agu 2022, doi: 10.3390/en15155631.
- [21] J. V. Canto Dos Santos, I. F. Costa, dan T. Nogueira, “New genetic algorithms for contingencies selection in the static security analysis of electric power systems,” *Expert Syst. Appl.*, vol. 42, no. 6, hlm. 2849-2856, Apr 2015, doi: 10.1016/j.eswa.2014.11.041.
- [22] J. M. Da Silva, I. Costa, J. V. C. Dos Santos, J. L. V. Barbosa, T. Braun, dan G. Pessin, “Toward a more reliable system for contingency selection in static security analysis of electric power systems,” *IEEE Syst. J.*, vol. 14, no. 1, hlm. 1183-1194, Mar 2020, doi: 10.1109/JSYST.2019.2938607.
- [23] E. A. Hailu, G. N. Nyakoe, dan C. M. Muriithi, “Techniques of power system static security assessment and improvement: A literature survey,” *Heliyon*, Mar 2023, doi: 10.1016/j.heliyon.2023.e14524.
- [24] J. H. Friedman, “Greedy Function Approximation: A Gradient Boosting Machine,” 2001. [Daring]. Tersedia pada: <http://www.jstor.org/stable/2699986>
- [25] L. Zhou, Q. Zhu, Q. Chen, P. Wang, dan H. Huang, “Predicting hospital outpatient volume using XGBoost: a machine learning approach,” *Sci. Rep.*, vol. 15, no. 1, Des 2025, doi: 10.1038/s41598-025-01265-y.
- [26] N. Alamsyah, B. Budiman, T. P. Yoga, dan R. Y. R. Alamsyah, “XGBoost Hyperparameter Optimization Using RandomizedSearchCV for Accurate Forest Fire Drought Condition Prediction,” *Jurnal Pilar Nusa Mandiri*, vol. 20, no. 2, hlm. 103-110, Sep 2024, doi: 10.33480/pilar.v20i2.5569.
- [27] S. Hakkal dan A. A. Lahcen, “XGBoost To Enhance Learner Performance Prediction,” *Computers and Education: Artificial Intelligence*, vol. 7, Des 2024, doi: 10.1016/j.caeai.2024.100254.
- [28] A. F. L. Ptr, M. M. Siregar, dan I. Daniel, “Analysis of Gradient Boosting, XGBoost, and CatBoost on Mobile Phone Classification,” *Journal of Computer Networks, Architecture and High Performance Computing*, vol. 6, no. 2, hlm. 661-670, Apr 2024, doi: 10.47709/cnahpc.v6i2.3790.
- [29] C. Sekhar dan P. S. Meghana, “A Study on Backpropagation in Artificial Neural Networks,” *Asia-Pacific Journal of Neural Networks and Its*

- Applications, vol. 4, no. 1, hlm. 21-28, Agu 2020, doi: 10.21742/AJNNIA.2020.4.1.03.
- [30] D. A. Matang, M. B. Jhala, P. Ashok, dan L. Vaghamshi, "Review Paper on Power System Contingency Analysis using Artificial Neural Network," 2015. [Daring]. Tersedia pada: www.ijserd.com
- [31] Y. Wu, H. Wang, B. Zhang, dan K.-L. Du, "Using Radial Basis Function Networks for Function Approximation and Classification," *ISRN Applied Mathematics*, vol. 2012, hlm. 1-34, Mar 2012, doi: 10.5402/2012/324194.

LAMPIRAN

Lampiran 1. Kode Program *Newton Raphson Load Flow*

```
clear; clc; close all;

fprintf('=====\n');
fprintf('GENERATE DATASET OSSA - FINAL\n');
fprintf('=====\n\n');

fclose('all');

RATING_G1 = 560; RATING_G2 = 250; RATING_G3 = 100;
RATING_G4 = 250; RATING_G5 = 70;
DELTA_V_MAX = 0.07;
V_ref = 1.0; baseMVA = 100;
load_levels = [0.5, 0.6, 0.7, 0.8, 0.9, 1.0];

contingency_list = [
    1,2; 2,5; 9,10; 4,12; 12,15; 6,9; 6,28;
    27,30; 10,21; 15,23; 22,24; 6,8; 25,27; 12,16;
    15,18; 27,29; 10,20; 10,17; 12,14; 18,19; 19,20;
    24,25; 16,17; 23,24; 10,22; 21,22; 2,6; 4,6;
    8,28; 14,15; 29,30; 6,7; 6,10; 1,3; 3,4;
    2,4; 5,7;
];

mpc = case_ieee30_ossa(RATING_G1, RATING_G2, RATING_G3, RATING_G4, RATING_G5);
gen_bus = mpc.gen(:,1);
find_br = @(fb,tb,br) find((br(:,1)==fb&br(:,2)==tb) | (br(:,1)==tb&br(:,2)==fb),1);

ts = datestr(now, 'yyyymmdd_HHMMSS');
txt_file = sprintf('system_summary %s.txt', ts);
csv_file = sprintf('dataset_ossa %s.csv', ts);

if exist(txt_file, 'file'), delete(txt_file); end
if exist(csv_file, 'file'), delete(csv_file); end

fid_txt = fopen(txt_file, 'w');
if fid_txt == -1
    error('GAGAL membuat file TXT. Coba ganti folder atau restart MATLAB.');
```

```
end

fid_csv = fopen(csv_file, 'w');
if fid_csv == -1
    fclose(fid_txt);
    error('GAGAL membuat file CSV. TUTUP EXCEL terlebih dahulu!');
end

fprintf('File output:\n');
fprintf('  TXT: %s\n', txt_file);
fprintf('  CSV: %s\n\n', csv_file);

fprintf(fid_txt,
'=====\n');
fprintf(fid_txt, '          DATASET OSSA - SYSTEM SUMMARY\n');
fprintf(fid_txt, '  G1=%d G2=%d G3=%d G4=%d G5=%d | dV=%2f | V11=V13=1.05\n', ...
    RATING_G1, RATING_G2, RATING_G3, RATING_G4, RATING_G5, DELTA_V_MAX);
fprintf(fid_txt, '  6 Level x 37 Kontingensi = 222 Sampel\n');
fprintf(fid_txt,
'=====\n');

fprintf(fid_csv, 'Sample_ID,Load_Level_pct,Contingency_Label,Ki');
for i=1:30, fprintf(fid_csv, ',PL_bus%d', i); end
for i=1:30, fprintf(fid_csv, ',QL_bus%d', i); end
for i=1:6, fprintf(fid_csv, ',PG_gen%d', gen_bus(i)); end
for i=1:6, fprintf(fid_csv, ',QG_gen%d', gen_bus(i)); end
for i=1:30, fprintf(fid_csv, ',Vm_bus%d', i); end
fprintf(fid_csv, ',PIP,PIV\n');

sc = 0; all_PIP = []; all_PIV = [];
fprintf('%-8s %-6s %-16s %-16s\n', 'Level', 'Valid', 'PIP Range', 'PIV Range');
fprintf('%-8s %-6s %-16s %-16s\n', '-----', '-----', '-----', '-----');
tic;

for li = 1:length(load_levels)
    lf = load_levels(li); lp = lf 100;
    mpc_lf = mpc; mpc_lf.bus(:,3)=mpc.bus(:,3) lf; mpc_lf.bus(:,4)=mpc.bus(:,4) lf;
    rb = runpf(mpc_lf, mpooption('verbose',0,'out.all',0));
    if ~rb.success, fprintf('%-8.0f%% %-6s\n',lp,'GAGAL'); continue; end

    Vm_base = rb.bus(:,8); Pd_base = rb.bus(:,3)/baseMVA; Qd_base = rb.bus(:,4)/baseMVA;
    Pg_base = zeros(6,1); Qg_base = zeros(6,1);
    for g=1:size(rb.gen,1), idx=find(gen_bus==rb.gen(g,1),1); if ~isempty(idx),
    Pg_base(idx)=rb.gen(g,2)/baseMVA; Qg_base(idx)=rb.gen(g,3)/baseMVA; end, end

    vc = 0; lPIP = []; lPIV = [];

    for k = 1:size(contingency_list,1)
        fb = contingency_list(k,1); tb = contingency_list(k,2);
```

```

bri = find_br(fb, tb, mpc_lf.branch); if isempty(bri), continue; end

mc = mpc_lf; mc.branch(bri,11) = 0;
rc = runpf(mc, mption('verbose',0,'out.all',0));
if ~rc.success, continue; end

PIP = 0;
for l=1:size(rc.branch,1)
    if rc.branch(l,11)>0
        Pf=abs(rc.branch(l,14))/baseMVA; Pm=rc.branch(l,6)/baseMVA;
        if Pm>0, PIP=PIP+0.5 (Pf/Pm)^2; end
    end
end

PIV = 0;
for i=1:size(rc.bus,1)
    dev=(rc.bus(i,8)-V_ref)/DELTA_V_MAX;
    PIV=PIV+0.5 dev^2;
end

sc=sc+1; vc=vc+1; cl=sprintf('L%d-%d',fb,tb);
all_PIP=[all_PIP;PIP]; all_PIV=[all_PIV;PIV];
lPIP=[lPIP;PIP]; lPIV=[lPIV;PIV];

fprintf(fid_csv, '%d,%.0f,%.6f,%.6f', sc, lp, cl, k);
for i=1:30, fprintf(fid_csv, ',%.6f', Pd_base(i)); end
for i=1:30, fprintf(fid_csv, ',%.6f', Qd_base(i)); end
for i=1:6, fprintf(fid_csv, ',%.6f', Pg_base(i)); end
for i=1:6, fprintf(fid_csv, ',%.6f', Qg_base(i)); end
for i=1:30, fprintf(fid_csv, ',%.6f', Vm_base(i)); end
fprintf(fid_csv, ',%.6f,%.6f\n', PIP, PIV);

print_summary(fid_txt, rc, mc, cl, lp, PIP, PIV);
end

if vc>0
    fprintf('%-8.0f%% %-6d %-16s %-16s\n', lp, vc, ...
        sprintf('%.4f-%.4f', min(lPIP), max(lPIP)), sprintf('%.4f-%.4f', min(lPIV), max(lPIV)));
end
end

fclose(fid_csv); fclose(fid_txt); et = toc;

fprintf('\n=====');
fprintf('SELESAI! %d sampel (%.1fs)\n', sc, et);
fprintf('TXT: %s\n', txt_file);
fprintf('CSV: %s\n', csv_file);
fprintf('PIP: %.4f-%.4f | PIV: %.4f-%.4f\n', min(all_PIP), max(all_PIP), min(all_PIV),
max(all_PIV));
fprintf('=====');

function print_summary(fid, res, mpc, cont_label, load_pct, PIP, PIV)
    fprintf(fid, '\n');
    fprintf(fid,
'=====
| LEVEL: %3.0f%% | %-10s | PIP: %.4f | PIV: %.4f | \n', load_pct,
cont_label, PIP, PIV);
    fprintf(fid,
'=====
\n');
end

mpopt = mption('verbose', 0, 'out.all', 1);
printpf(res, fid, mption);
end

```

Lampiran 2. Codingan Mesin Learning RBFN referensi

```

# - - coding: utf-8 - -
# - - coding: utf-8 - -

# =====
# CELL 1: IMPORT LIBRARY & UPLOAD FILE
# =====

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re
import os
import warnings
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import pairwise_distances
from google.colab import files

warnings.filterwarnings('ignore')

RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)

print("=" 80)
print("RBFN - PAPER SEKHAR (2016) DENGAN STABILITAS NUMERIK")
print("Parameter: m1=38, g2=1.8, g1=1.8 (dengan decay)")
print("=" 80)

```

```

print("\nSilakan upload file CSV/TXT Anda...")
uploaded = files.upload()
filename = list(uploaded.keys())[0]
print(f"✓ File '{filename}' berhasil diupload")

# =====
# CELL 2: DAFTAR LINE IEEE 30-BUS & FUNGSI PARSING
# =====

ALL_LINE_NAMES = [
    '1-2', '1-3', '2-4', '3-4', '2-5', '2-6',
    '4-6', '5-7', '6-7', '6-8', '6-9', '6-10',
    '9-10', '4-12', '12-13', '12-14', '12-15', '12-16',
    '14-15', '16-17', '15-18', '18-19', '19-20', '10-20',
    '10-17', '10-21', '10-22', '21-22', '15-23', '22-24',
    '23-24', '24-25', '25-26', '25-27', '28-27', '27-29',
    '27-30', '29-30', '8-28', '6-28', '9-11'
]

def parse_line_name(line_str):
    """Membersihkan dan menstandarisasi nama line"""
    line_str = str(line_str).strip()
    if line_str.upper().startswith('L'):
        line_str = line_str[1:].strip()

    if line_str in ALL_LINE_NAMES:
        return line_str

    for fmt_line in ALL_LINE_NAMES:
        if fmt_line.replace('-', ' ') == line_str.replace('-', ' ').replace(' ', ' '):
            return fmt_line
        if fmt_line in line_str or line_str in fmt_line:
            return fmt_line

    return line_str

# =====
# CELL 3: MEMBACA FILE CSV
# =====

def read_csv_dataset(filename):
    """Membaca dataset dalam format CSV"""
    print("\nMembaca file CSV...")

    df = pd.read_csv(filename)
    print(f"Dimensi data: {df.shape}")
    print(f"Kolom: {list(df.columns)[:15]}...")

    level_col = None
    line_col = None
    pip_col = None
    piv_col = None

    for col in df.columns:
        col_upper = col.upper().strip()
        if 'LEVEL' in col_upper:
            level_col = col
        elif 'LINE' in col_upper or 'OUTAGE' in col_upper:
            line_col = col
        elif col_upper == 'PIP' or 'PIP_' in col_upper:
            pip_col = col
        elif col_upper == 'PIV' or 'PIV_' in col_upper:
            piv_col = col

    if level_col is None:
        level_col = df.columns[0]

    numeric_cols = df.select_dtypes(include=[np.number]).columns
    if pip_col is None and len(numeric_cols) >= 2:
        pip_col = numeric_cols[-2]
        piv_col = numeric_cols[-1]

    print(f"\nKolom terdeteksi:")
    print(f"  LEVEL: {level_col}")
    print(f"  LINE: {line_col}")
    print(f"  PIP: {pip_col}")
    print(f"  PIV: {piv_col}")

    v_mag_cols = [c for c in df.columns if 'V_MAG' in c.upper()]
    v_ang_cols = [c for c in df.columns if 'V_ANG' in c.upper()]
    p_gen_cols = [c for c in df.columns if 'P_GEN' in c.upper() or 'PGEN' in c.upper()]
    q_gen_cols = [c for c in df.columns if 'Q_GEN' in c.upper() or 'QGEN' in c.upper()]
    p_load_cols = [c for c in df.columns if 'P_LOAD' in c.upper() or 'PLOAD' in c.upper()]
    q_load_cols = [c for c in df.columns if 'Q_LOAD' in c.upper() or 'QLOAD' in c.upper()]

    print(f"\nKolom bus terdeteksi:")
    print(f"  V_mag: {len(v_mag_cols)} kolom")
    print(f"  V_ang: {len(v_ang_cols)} kolom")
    print(f"  P_gen: {len(p_gen_cols)} kolom")
    print(f"  Q_gen: {len(q_gen_cols)} kolom")
    print(f"  P_load: {len(p_load_cols)} kolom")
    print(f"  Q_load: {len(q_load_cols)} kolom")

    blocks = []

    for idx, row in df.iterrows():
        try:

```

```

        level_str = str(row[level_col])
        level_match = re.search(r'(\d+)', level_str)
        level = int(level_match.group(1)) if level_match else 100

        line_name = parse_line_name(str(row[line_col])) if line_col else f"Line_{idx}"

        pip_val = float(row[pip_col]) if pip_col else 0.0
        piv_val = float(row[piv_col]) if piv_col else 0.0

        bus_data = []

        if len(v_mag_cols) >= 30:
            for i in range(30):
                bus_data.append({
                    'bus': i + 1,
                    'V_mag': float(row[v_mag_cols[i]]) if i < len(v_mag_cols) else 1.0,
                    'V_ang': float(row[v_ang_cols[i]]) if i < len(v_ang_cols) else 0.0,
                    'P_gen': float(row[p_gen_cols[i]]) if i < len(p_gen_cols) else 0.0,
                    'Q_gen': float(row[q_gen_cols[i]]) if i < len(q_gen_cols) else 0.0,
                    'P_load': float(row[p_load_cols[i]]) if i < len(p_load_cols) else 0.0,
                    'Q_load': float(row[q_load_cols[i]]) if i < len(q_load_cols) else 0.0
                })
        else:
            for bus_num in range(1, 31):
                v_col = ang_col = pg_col = qg_col = pl_col = ql_col = None

                for col in df.columns:
                    col_upper = col.upper()
                    if f'BUS{bus_num}' in col_upper or f'BUS_{bus_num}' in col_upper:
                        if 'V_MAG' in col_upper or ('V_' in col_upper and 'ANG' not in
col_upper):
                            v_col = col
                        elif 'V_ANG' in col_upper or 'ANG' in col_upper:
                            ang_col = col
                        elif 'P_GEN' in col_upper or 'PG_' in col_upper:
                            pg_col = col
                        elif 'Q_GEN' in col_upper or 'QG_' in col_upper:
                            qg_col = col
                        elif 'P_LOAD' in col_upper or 'PL_' in col_upper:
                            pl_col = col
                        elif 'Q_LOAD' in col_upper or 'QL_' in col_upper:
                            ql_col = col

                if v_col:
                    bus_data.append({
                        'bus': bus_num,
                        'V_mag': float(row[v_col]) if v_col else 1.0,
                        'V_ang': float(row[ang_col]) if ang_col else 0.0,
                        'P_gen': float(row[pg_col]) if pg_col else 0.0,
                        'Q_gen': float(row[qg_col]) if qg_col else 0.0,
                        'P_load': float(row[pl_col]) if pl_col else 0.0,
                        'Q_load': float(row[ql_col]) if ql_col else 0.0
                    })

            if len(bus_data) == 30:
                blocks.append({
                    'block_num': idx,
                    'level': level,
                    'line_outage': line_name,
                    'PIP': pip_val,
                    'PIV': piv_val,
                    'bus_df': pd.DataFrame(bus_data)
                })
        except Exception as e:
            print(f" ⚠ Baris {idx}: Error - {str(e)[:100]}")
            continue

    return blocks

# =====
# CELL 4: MEMBACA FILE TXT (FALLBACK)
# =====

def read_txt_dataset(filename):
    """Membaca dataset dalam format TXT"""
    print("\nMembaca file TXT...")

    with open(filename, 'r') as f:
        content = f.read()

    header_pattern = r'LEVEL:\s (\d+)\s \|\s (L?\d+-\d+)\s \|\s PIP:\s ([\d.]+\s \|\s PIV:\s ([\d.]+)')
    blocks_raw = re.split(r'={70,}\n+={70,}', content)

    parsed_blocks = []
    start_index = 0

    if len(blocks_raw) > 0 and not re.search(header_pattern, blocks_raw[0]):
        start_index = 1

    for i, block_text in enumerate(blocks_raw[start_index:]):
        match = re.search(header_pattern, block_text)
        if match:
            line_outage = match.group(2)
            if line_outage.startswith('L'):
                line_outage = line_outage[1:]

            bus_section_pattern = r'Bus Data. \s [-]+\s \n(. ?)\n\s [-]+\s '

```

```

        bus_match = re.search(bus_section_pattern, block_text, re.DOTALL)

        bus_data = []
        if bus_match:
            bus_lines = bus_match.group(1).strip().split('\n')
            for line in bus_lines:
                parts = line.strip().split()
                if len(parts) >= 7 and parts[0].isdigit():
                    try:
                        bus_data.append({
                            'bus': int(parts[0]),
                            'V_mag': float(parts[1]),
                            'V_ang': float(parts[2]),
                            'P_gen': float(parts[3]) if parts[3] != '-' else 0.0,
                            'Q_gen': float(parts[4]) if parts[4] != '-' else 0.0,
                            'P_load': float(parts[5]) if parts[5] != '-' else 0.0,
                            'Q_load': float(parts[6]) if parts[6] != '-' else 0.0
                        })
                    except (ValueError, IndexError):
                        continue

            if len(bus_data) == 30:
                parsed_blocks.append({
                    'block_num': i + start_index,
                    'level': int(match.group(1)),
                    'line_outage': parse_line_name(line_outage),
                    'PIP': float(match.group(3)),
                    'PIV': float(match.group(4)),
                    'bus_df': pd.DataFrame(bus_data)
                })

        return parsed_blocks

# =====
# CELL 5: PROSES FILE DAN BANGUN DATASET
# =====

file_ext = filename.lower().split('.')[-1]
print(f"\nTipe file: {file_ext}")

if file_ext == 'csv':
    blocks = read_csv_dataset(filename)
elif file_ext == 'txt':
    blocks = read_txt_dataset(filename)
else:
    try:
        blocks = read_csv_dataset(filename)
    except:
        blocks = read_txt_dataset(filename)

print(f"\nBlok data valid: {len(blocks)}")

if len(blocks) == 0:
    raise ValueError("Tidak ada data valid!")

X_list, y_list, metadata_list = [], [], []

for blk in blocks:
    bus_df = blk['bus_df'].sort_values('bus').reset_index(drop=True)

    PL = bus_df['P_load'].values.astype(float)
    QL = bus_df['Q_load'].values.astype(float)
    PG = bus_df['P_gen'].values.astype(float)
    QG = bus_df['Q_gen'].values.astype(float)
    V_mag = bus_df['V_mag'].values.astype(float)

    Ki = np.zeros(len(ALL_LINE_NAMES))
    line_name = blk['line_outage']
    if line_name in ALL_LINE_NAMES:
        Ki[ALL_LINE_NAMES.index(line_name)] = 1
    else:
        parsed = parse_line_name(line_name)
        if parsed in ALL_LINE_NAMES:
            Ki[ALL_LINE_NAMES.index(parsed)] = 1
        else:
            Ki[blk['block_num'] % len(ALL_LINE_NAMES)] = 1

    X = np.concatenate([PL, QL, PG, QG, V_mag, Ki])
    y = np.array([blk['PIP'], blk['PIV']])

    X_list.append(X)
    y_list.append(y)
    metadata_list.append({
        'level': blk['level'],
        'line_outage': line_name,
        'PIP': blk['PIP'],
        'PIV': blk['PIV']
    })

X_all = np.array(X_list)
y_all = np.array(y_list)
metadata_df = pd.DataFrame(metadata_list)

print(f"Dataset: X={X_all.shape}, y={y_all.shape}")
print(f"Level beban: {sorted(metadata_df['level'].unique())}")

# =====
# CELL 6: SPLIT DATA (LEVEL 100% UNTUK TESTING)

```

```

# =====

total_pl_all = np.sum(X_all[:, 0:30], axis=1)
max_load = np.max(total_pl_all)

load_100_mask = (total_pl_all >= max_load * 0.80)
indices_100 = np.where(load_100_mask)[0]

if len(indices_100) > 37:
    sorted_idx = indices_100[np.argsort(total_pl_all[indices_100])[:-1]]
    indices_100 = sorted_idx[:37]

indices_train = np.setdiff1d(np.arange(len(X_all)), indices_100)

X_train = X_all[indices_train]
y_train = y_all[indices_train]
X_test = X_all[indices_100]
y_test = y_all[indices_100]
metadata_train = metadata_df.iloc[indices_train].reset_index(drop=True)
metadata_test = metadata_df.iloc[indices_100].reset_index(drop=True)

print(f"\nData Split:")
print(f" Training: {X_train.shape[0]} sampel")
print(f" Testing (100%): {X_test.shape[0]} sampel")

np.save('X_train.npy', X_train)
np.save('y_train.npy', y_train)
np.save('X_test.npy', X_test)
np.save('y_test.npy', y_test)
metadata_train.to_csv('metadata_train.csv', index=False)
metadata_test.to_csv('metadata_test.csv', index=False)
print("✓ Data saved: X_train.npy, y_train.npy, X_test.npy, y_test.npy")

# =====
# CELL 7: NORMALISASI
# =====

print("\n" + "=" * 80)
print("NORMALISASI DATA")
print("=" * 80)

scaler_X = StandardScaler()
X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)

scaler_y = StandardScaler()
y_train_scaled = scaler_y.fit_transform(y_train)
y_test_scaled = scaler_y.transform(y_test)

print(f"X_train_scaled: mean={X_train_scaled.mean():.6f}, std={X_train_scaled.std():.6f}")
print(f"y_train_scaled: mean={y_train_scaled.mean():.6f}, std={y_train_scaled.std():.6f}")

# =====
# CELL 8: RBFN PAPER SEKHAR DENGAN STABILITAS NUMERIK
# =====

class RBFN_Paper_Stable:
    """
    RBFN Paper Sekhar dengan stabilitas numerik
    Parameter: m1=38, g2=1.8, g1=1.8 (dengan decay)
    """

    def __init__(self, n_centers=38, learning_rate=1.8, max_iter=100,
                  patience=30, sigma_factor=1.8, reg_lambda=0.001):
        self.n_centers = n_centers
        self.learning_rate = learning_rate
        self.max_iter = max_iter
        self.patience = patience
        self.sigma_factor = sigma_factor
        self.reg_lambda = reg_lambda
        self.centers = None
        self.d_max = None
        self.weights = None
        self.train_mse = []
        self.val_mse = []

    def _init_centers(self, X):
        n_centers = min(self.n_centers, max(2, len(X) - 1))
        kmeans = KMeans(n_clusters=n_centers, init='k-means++', n_init=20,
                        max_iter=300, random_state=42)
        kmeans.fit(X)
        self.n_centers = n_centers
        return kmeans.cluster_centers_

    def _calculate_dmax(self):
        if self.n_centers < 2:
            return 1.0
        D = pairwise_distances(self.centers)
        tri = np.triu_indices_from(D, 1)
        d_max = np.max(D[tri]) if len(D[tri]) > 0 else 1.0
        return d_max

    def _rbf_matrix(self, X):
        X = np.asarray(X, float)
        centers = np.asarray(self.centers, float)

        X2 = np.sum(X * X, axis=1).reshape(-1, 1)
        C2 = np.sum(centers * centers, axis=1).reshape(1, -1)

```

```

XC = X.dot(centers.T)
d2 = np.clip(X2 + C2 - 2 * XC, 0, None)

G = np.exp(-d2 / (self.d_max ** 2 + 1e-10))
G = G / (np.max(G, axis=1, keepdims=True) + 1e-10)
return G

def fit(self, X_train, y_train, X_val=None, y_val=None, verbose=False):
    n_samples = X_train.shape[0]
    n_outputs = y_train.shape[1]

    self.centers = self._init_centers(X_train)
    self.d_max = self._calculate_dmax()
    G_train = self._rbf_matrix(X_train)
    G_val = self._rbf_matrix(X_val) if X_val is not None else None

    I = np.eye(self.n_centers)
    self.weights = np.linalg.solve(
        G_train.T @ G_train + self.reg_lambda * I,
        G_train.T @ y_train
    )

    best_weights = self.weights.copy()
    best_val_mse = float('inf')
    patience_counter = 0

    if verbose:
        print(f"    N={n_samples}, M={self.n_centers}, d_max={self.d_max:.4f}")

    for iteration in range(self.max_iter):
        total_mse = 0
        lr = self.learning_rate / (1 + 0.01 * iteration)

        indices = np.random.permutation(n_samples)
        for i in indices:
            G_i = G_train[i, :].reshape(1, self.n_centers)
            y_i = y_train[i, :].reshape(1, n_outputs)
            y_pred_i = G_i @ self.weights
            error = y_i - y_pred_i
            grad = G_i.T @ error
            grad = np.clip(grad, -1, 1)
            self.weights = self.weights + lr * grad
            total_mse += np.sum(error ** 2)

        avg_mse = total_mse / n_samples
        self.train_mse.append(avg_mse)

        if verbose and ((iteration + 1) % 20 == 0):
            if not np.isnan(avg_mse):
                print(f"    Epoch {iteration+1}/{self.max_iter}, MSE: {avg_mse:.6e}")

        if G_val is not None and y_val is not None:
            y_pred_val = G_val @ self.weights
            mse_val = mean_squared_error(y_val, y_pred_val)
            self.val_mse.append(mse_val)

            if mse_val < best_val_mse:
                best_val_mse = mse_val
                best_weights = self.weights.copy()
                patience_counter = 0
            else:
                patience_counter += 1

            if patience_counter >= self.patience:
                if verbose:
                    print(f"    Early stopping at epoch {iteration+1}")
                break

        self.weights = best_weights
    return self

def predict(self, X):
    G = self._rbf_matrix(X)
    pred = G @ self.weights
    return np.nan_to_num(pred, nan=0.0, posinf=0.0, neginf=0.0)

# =====
# CELL 9: FUNGSI METRIK EVALUASI
# =====

def calc_metrics_detailed(y_true, y_pred):
    """
    Menghitung metrik evaluasi lengkap untuk multi-output
    y_true dan y_pred shape: (n_samples, 2) -> [PIP, PIV]
    """
    results = {}
    for i, name in enumerate(['PIP', 'PIV']):
        yt = y_true[:, i]
        yp = y_pred[:, i]

        mse = mean_squared_error(yt, yp)
        rmse = np.sqrt(mse)
        mae = mean_absolute_error(yt, yp)
        r2 = r2_score(yt, yp)
        mape = np.mean(np.abs((yt - yp) / (np.abs(yt) + 1e-10))) * 100
        max_err = np.max(np.abs(yt - yp))
        min_err = np.min(np.abs(yt - yp))

```

```

        results[name] = {
            'R²': r2,
            'MSE': mse,
            'RMSE': rmse,
            'MAE': mae,
            'MAPE': mape,
            'Max_Error': max_err,
            'Min_Error': min_err
        }
    return results

def print_metrics_table(train_metrics, test_metrics):
    """Menampilkan tabel metrik evaluasi lengkap"""
    print("\n" + "=" 90)
    print("TABEL METRIK EVALUASI LENGKAP")
    print("=" 90)

    header = f"{'Metrik':<15} {'PIP Train':>15} {'PIP Test':>15} | {'PIV Train':>15} {'PIV Test':>15}"
    print(header)
    print("-" 90)

    metrics_display = [
        ('R²', '{:.4f}'),
        ('MSE', '{:.6f}'),
        ('RMSE', '{:.6f}'),
        ('MAE', '{:.6f}'),
        ('MAPE', '{:.2f}%'),
        ('Max_Error', '{:.6f}'),
        ('Min_Error', '{:.6f}')
    ]

    for metric_name, fmt in metrics_display:
        pip_train = train_metrics['PIP'][metric_name]
        pip_test = test_metrics['PIP'][metric_name]
        piv_train = train_metrics['PIV'][metric_name]
        piv_test = test_metrics['PIV'][metric_name]

        if metric_name == 'MAPE':
            print(f"{metric_name:<15} {pip_train:>14.2f}% {pip_test:>14.2f}% | {piv_train:>14.2f}% {piv_test:>14.2f}%")
        elif metric_name == 'R²':
            print(f"{metric_name:<15} {pip_train:>15.4f} {pip_test:>15.4f} | {piv_train:>15.4f} {piv_test:>15.4f}")
        else:
            print(f"{metric_name:<15} {pip_train:>15.6f} {pip_test:>15.6f} | {piv_train:>15.6f} {piv_test:>15.6f}")

    print("-" 90)

# =====
# CELL 10: TRAINING RBFN
# =====

print("\n" + "=" 80)
print("TRAINING RBFN - PAPER SEKCHAR (2016)")
print("=" 80)

model = RBFN_Paper_Stable(
    n_centers=38,
    learning_rate=1.8,
    max_iter=100,
    patience=30,
    sigma_factor=1.8,
    reg_lambda=0.001
)

print("\nParameter dari Paper Sekhar (2016):")
print(f"  m1 (n_centers)      = 38")
print(f"  g2 (sigma_factor)   = 1.8")
print(f"  g1 (learning_rate)  = 1.8 (dengan decay)")
print(f"  max_iter            = 100")
print(f"  reg_lambda          = 0.001")

print("\nMemulai training...")
model.fit(X_train_scaled, y_train_scaled, verbose=True)

# =====
# CELL 11: EVALUASI - TABEL METRIK LENGKAP
# =====

print("\n" + "=" 80)
print("EVALUASI MODEL")
print("=" 80)

print("\n[1] Prediksi pada Training Set...")
y_pred_train_scaled = model.predict(X_train_scaled)
y_pred_train = scaler_y.inverse_transform(y_pred_train_scaled)
y_train_orig = y_train

print("\n[2] Prediksi pada Testing Set (Level 100%)...")
y_pred_test_scaled = model.predict(X_test_scaled)
y_pred_test = scaler_y.inverse_transform(y_pred_test_scaled)
y_test_orig = y_test

print("\n[3] Menghitung metrik evaluasi...")
train_metrics = calc_metrics_detailed(y_train_orig, y_pred_train)
test_metrics = calc_metrics_detailed(y_test_orig, y_pred_test)

```

```

print_metrics_table(train_metrics, test_metrics)

# =====
# CELL 12: 37 KONTINGENSI LEVEL 100% - DETAIL
# =====

print("\n" + "=" 100)
print("37 KONTINGENSI - LEVEL BEBAN 100% (DETAIL)")
print("=" 100)

lines = metadata_test['line_outage'].values

# === PIP ===
print("\n" + "-" 100)
print("A. ACTIVE POWER PERFORMANCE INDEX (PIP)")
print("-" 100)
print(f"{'Rank':<6} {'Line':<14} {'PIP NRLF':>12} {'PIP RBFN':>12} {'Error':>12} {'Error %':>10}")
print("-" 100)

df_pip = pd.DataFrame({
    'Line': lines,
    'PIP_NRLF': y_test_orig[:, 0],
    'PIP_RBFN': y_pred_test[:, 0]
})
df_pip = df_pip.sort_values('PIP_NRLF', ascending=False).reset_index(drop=True)
df_pip['Error'] = np.abs(df_pip['PIP_NRLF'] - df_pip['PIP_RBFN'])
df_pip['Error_Pct'] = (df_pip['Error'] / (df_pip['PIP_NRLF'] + 1e-10)) * 100

for i in range(len(df_pip)):
    row = df_pip.iloc[i]
    print(f"{'i+1':<3} {'row['Line']':<14} {'row['PIP_NRLF']':>12.4f} "
          f"{'row['PIP_RBFN']':>12.4f} {'row['Error']':>12.6f} {'row['Error_Pct']':>9.2f}%")

print("-" 100)
print(f"STATISTIK PIP: R²={test_metrics['PIP']['R²']:.4f} | RMSE={test_metrics['PIP']['RMSE']:.6f} | "
      f"MAE={test_metrics['PIP']['MAE']:.6f} | MAPE={test_metrics['PIP']['MAPE']:.2f}%")

# === PIV ===
print("\n" + "-" 100)
print("B. VOLTAGE PERFORMANCE INDEX (PIV)")
print("-" 100)
print(f"{'Rank':<6} {'Line':<14} {'PIV NRLF':>12} {'PIV RBFN':>12} {'Error':>12} {'Error %':>10}")
print("-" 100)

df_piv = pd.DataFrame({
    'Line': lines,
    'PIV_NRLF': y_test_orig[:, 1],
    'PIV_RBFN': y_pred_test[:, 1]
})
df_piv = df_piv.sort_values('PIV_NRLF', ascending=False).reset_index(drop=True)
df_piv['Error'] = np.abs(df_piv['PIV_NRLF'] - df_piv['PIV_RBFN'])
df_piv['Error_Pct'] = (df_piv['Error'] / (df_piv['PIV_NRLF'] + 1e-10)) * 100

for i in range(len(df_piv)):
    row = df_piv.iloc[i]
    print(f"{'i+1':<3} {'row['Line']':<14} {'row['PIV_NRLF']':>12.4f} "
          f"{'row['PIV_RBFN']':>12.4f} {'row['Error']':>12.6f} {'row['Error_Pct']':>9.2f}%")

print("-" 100)
print(f"STATISTIK PIV: R²={test_metrics['PIV']['R²']:.4f} | RMSE={test_metrics['PIV']['RMSE']:.6f} | "
      f"MAE={test_metrics['PIV']['MAE']:.6f} | MAPE={test_metrics['PIV']['MAPE']:.2f}%")

# =====
# CELL 13: TOP 5 CRITICAL CONTINGENCIES
# =====

print("\n" + "=" 80)
print("TOP 5 CRITICAL CONTINGENCIES - LEVEL 100%")
print("=" 80)

print("\nTOP 5 PIP (Paling Kritis):")
print("-" 60)
print(f"{'Rank':<6} {'Line':<14} {'PIP NRLF':>12} {'PIP RBFN':>12} {'Error %':>10}")
print("-" 60)
for i in range(5):
    row = df_pip.iloc[i]
    print(f"{'i+1':<6} {'row['Line']':<14} {'row['PIP_NRLF']':>12.4f} {'row['PIP_RBFN']':>12.4f} "
          f"{'row['Error_Pct']':>9.2f}%")

print("\nTOP 5 PIV (Paling Kritis):")
print("-" 60)
print(f"{'Rank':<6} {'Line':<14} {'PIV NRLF':>12} {'PIV RBFN':>12} {'Error %':>10}")
print("-" 60)
for i in range(5):
    row = df_piv.iloc[i]
    print(f"{'i+1':<6} {'row['Line']':<14} {'row['PIV_NRLF']':>12.4f} {'row['PIV_RBFN']':>12.4f} "
          f"{'row['Error_Pct']':>9.2f}%")

# =====
# CELL 14: VISUALISASI
# =====

print("\n" + "=" 80)
print("VISUALISASI HASIL")
print("=" 80)

fig, axes = plt.subplots(2, 2, figsize=(14, 10))

```

```

ax = axes[0, 0]
ax.scatter(df_pip['PIP_NRLF'], df_pip['PIP_RBFN'], alpha=0.7, c='#2196F3', edgecolors='white', s=70)
lims = [min(df_pip['PIP_NRLF'].min(), df_pip['PIP_RBFN'].min()) 0.95,
        max(df_pip['PIP_NRLF'].max(), df_pip['PIP_RBFN'].max()) 1.05]
ax.plot(lims, lims, 'r--', lw=2, label='Perfect Fit')
ax.set_xlabel('PIP Actual (NRLF)', fontsize=11)
ax.set_ylabel('PIP Predicted (RBFN)', fontsize=11)
ax.set_title(f'PIP - Level 100%\nR2={test_metrics["PIP"]["R2"]:.4f} |
MAPE={test_metrics["PIP"]["MAPE"]:.1f}%', fontsize=13)
ax.legend()
ax.grid(True, alpha=0.3)

ax = axes[0, 1]
ax.scatter(df_piv['PIV_NRLF'], df_piv['PIV_RBFN'], alpha=0.7, c='#4CAF50', edgecolors='white', s=70)
lims = [min(df_piv['PIV_NRLF'].min(), df_piv['PIV_RBFN'].min()) 0.95,
        max(df_piv['PIV_NRLF'].max(), df_piv['PIV_RBFN'].max()) 1.05]
ax.plot(lims, lims, 'r--', lw=2, label='Perfect Fit')
ax.set_xlabel('PIV Actual (NRLF)', fontsize=11)
ax.set_ylabel('PIV Predicted (RBFN)', fontsize=11)
ax.set_title(f'PIV - Level 100%\nR2={test_metrics["PIV"]["R2"]:.4f} |
MAPE={test_metrics["PIV"]["MAPE"]:.1f}%', fontsize=13)
ax.legend()
ax.grid(True, alpha=0.3)

ax = axes[1, 0]
x = np.arange(len(df_pip))
width = 0.35
ax.bar(x - width/2, df_pip['PIP_NRLF'].values, width, label='NRLF', color='#2196F3', alpha=0.8)
ax.bar(x + width/2, df_pip['PIP_RBFN'].values, width, label='RBFN', color='#FF9800', alpha=0.8)
ax.set_xlabel('Contingency Index', fontsize=11)
ax.set_ylabel('PIP', fontsize=11)
ax.set_title('PIP - Actual vs Predicted (Level 100%)', fontsize=13)
ax.legend()
ax.grid(True, alpha=0.3, axis='y')

ax = axes[1, 1]
ax.bar(x - width/2, df_piv['PIV_NRLF'].values, width, label='NRLF', color='#4CAF50', alpha=0.8)
ax.bar(x + width/2, df_piv['PIV_RBFN'].values, width, label='RBFN', color='#FF5722', alpha=0.8)
ax.set_xlabel('Contingency Index', fontsize=11)
ax.set_ylabel('PIV', fontsize=11)
ax.set_title('PIV - Actual vs Predicted (Level 100%)', fontsize=13)
ax.legend()
ax.grid(True, alpha=0.3, axis='y')

plt.tight_layout()
plt.savefig('rbfn_paper_sekhar_results.png', dpi=150, bbox_inches='tight')
plt.show()
print("✓ Plot saved: rbfn_paper_sekhar_results.png")

# =====
# CELL 15: SIMPAN HASIL
# =====

print("\n" + "=" 80)
print("MENYIMPAN HASIL")
print("=" 80)

df_combined = pd.DataFrame({
    'Line_Outage': lines,
    'PIP_NRLF': y_test_orig[:, 0],
    'PIP_RBFN': y_pred_test[:, 0],
    'PIP_Error': np.abs(y_test_orig[:, 0] - y_pred_test[:, 0]),
    'PIP_Error_Pct': (np.abs(y_test_orig[:, 0] - y_pred_test[:, 0]) / (y_test_orig[:, 0] + 1e-10))
100,
    'PIV_NRLF': y_test_orig[:, 1],
    'PIV_RBFN': y_pred_test[:, 1],
    'PIV_Error': np.abs(y_test_orig[:, 1] - y_pred_test[:, 1]),
    'PIV_Error_Pct': (np.abs(y_test_orig[:, 1] - y_pred_test[:, 1]) / (y_test_orig[:, 1] + 1e-10))
100
})
df_combined.to_csv('rbfn_paper_sekhar_results.csv', index=False)
print("✓ Results saved: rbfn_paper_sekhar_results.csv")

metrics_summary = {
    'train': train_metrics,
    'test': test_metrics
}

import json

def convert_to_serializable(obj):
    if isinstance(obj, dict):
        return {k: convert_to_serializable(v) for k, v in obj.items()}
    elif isinstance(obj, np.floating):
        return float(obj)
    return obj

with open('rbfn_paper_sekhar_metrics.json', 'w') as f:
    json.dump(convert_to_serializable(metrics_summary), f, indent=2)
print("✓ Metrics saved: rbfn_paper_sekhar_metrics.json")

print("\n" + "=" 80)
print("SELESAI!")
print("=" 80)

```

Lampiran 3. Codingan Mesin Learning RBFN Optimized

```
# -*- coding: utf-8 -*-
"""RBFN - POWER SYSTEM STATIC SECURITY ASSESSMENT (VERSI CSV)"""

import numpy as np
import pandas as pd
import warnings
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler, StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.cluster import KMeans
from google.colab import files
import pickle
warnings.filterwarnings('ignore')

RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)

print("="*80)
print("RBFN - POWER SYSTEM STATIC SECURITY ASSESSMENT (VERSI CSV)")
print("IEEE 30-Bus Test System")
print("="*80)

print("\nSilakan upload file CSV Anda (ml_dataset_power_flow.csv)...")
uploaded = files.upload()
filename = list(uploaded.keys())[0]
print(f"File '{filename}' berhasil diupload")

ALL_LINE_NAMES = [
    '1-2', '1-3', '2-4', '3-4', '2-5', '2-6',
    '4-6', '5-7', '6-7', '6-8', '6-9', '6-10',
    '9-10', '4-12', '12-13', '12-14', '12-15', '12-16',
    '14-15', '16-17', '15-18', '18-19', '19-20', '10-20',
    '10-17', '10-21', '10-22', '21-22', '15-23', '22-24',
    '23-24', '24-25', '25-26', '25-27', '28-27', '27-29',
    '27-30', '29-30', '8-28', '6-28', '9-11'
]
N_BUS = 30
N_BRANCH = 41

# =====
# LOAD CSV & BUILD RAW FEATURE MATRICES
# =====
df = pd.read_csv(filename)
print(f"Dataset: {df.shape[0]} baris x {df.shape[1]} kolom")
print(f"Level beban: {sorted(df['level_pct'].unique())}")

PL = df[[f'bus{i:02d}_load_P' for i in range(1, N_BUS+1)]].values.astype(float)
QL = df[[f'bus{i:02d}_load_Q' for i in range(1, N_BUS+1)]].values.astype(float)
PG = df[[f'bus{i:02d}_gen_P' for i in range(1, N_BUS+1)]].values.astype(float)
QG = df[[f'bus{i:02d}_gen_Q' for i in range(1, N_BUS+1)]].values.astype(float)
V_mag = df[[f'bus{i:02d}_V_mag' for i in range(1, N_BUS+1)]].values.astype(float)

Ki = np.zeros((len(df), N_BRANCH))
for row_idx, cont in enumerate(df['contingency']):
    line_name = str(cont).replace('L', '')
    if line_name in ALL_LINE_NAMES:
        Ki[row_idx, ALL_LINE_NAMES.index(line_name)] = 1

X_bus_all = np.concatenate([PL, QL, PG, QG, V_mag, Ki], axis=1)

bus_feature_names = np.array(
    [f'PL_{i}' for i in range(1, N_BUS+1)] + [f'QL_{i}' for i in range(1, N_BUS+1)] +
    [f'PG_{i}' for i in range(1, N_BUS+1)] + [f'QG_{i}' for i in range(1, N_BUS+1)] +
    [f'V_{i}' for i in range(1, N_BUS+1)] + [f'Ki_{i}' for i in range(1, N_BRANCH+1)]
)

Pf = df[[f'br{i:02d}_from_P' for i in range(1, N_BRANCH+1)]].values.astype(float)
Qf = df[[f'br{i:02d}_from_Q' for i in range(1, N_BRANCH+1)]].values.astype(float)
Ploss = df[[f'br{i:02d}_loss_P' for i in range(1, N_BRANCH+1)]].values.astype(float)
Qloss = df[[f'br{i:02d}_loss_Q' for i in range(1, N_BRANCH+1)]].values.astype(float)
Pf = np.nan_to_num(Pf, nan=0.0)
Qf = np.nan_to_num(Qf, nan=0.0)
Ploss = np.nan_to_num(Ploss, nan=0.0)
Qloss = np.nan_to_num(Qloss, nan=0.0)

Pf_abs = np.abs(Pf)
Qf_abs = np.abs(Qf)
X_branch_agg = np.column_stack([
    Pf_abs.sum(axis=1), Pf_abs.max(axis=1), Pf_abs.mean(axis=1), Pf_abs.std(axis=1),
    Qf_abs.sum(axis=1), Qf_abs.max(axis=1),
    Ploss.sum(axis=1), Ploss.max(axis=1),
    Qloss.sum(axis=1), Qloss.max(axis=1)
])
branch_feature_names = np.array([
    'Total_Abs_Pflow', 'Max_Abs_Pflow', 'Mean_Abs_Pflow', 'Std_Abs_Pflow',
    'Total_Abs_Qflow', 'Max_Abs_Qflow', 'Total_Ploss', 'Max_Ploss',
    'Total_Qloss', 'Max_Qloss'
])

y_all = df[['label_PIP', 'label_PIV']].values.astype(float)

print(f"X_bus_all: {X_bus_all.shape} | X_branch_agg: {X_branch_agg.shape} | y_all: {y_all.shape}")

std_bus = np.std(X_bus_all, axis=0)
non_constant_mask_bus = std_bus > 1e-10
X_bus_filtered = X_bus_all[:, non_constant_mask_bus]
```

```

bus_names_filtered = bus_feature_names[non_constant_mask_bus]
print(f"Fitur bus dihapus (konstan): {np.sum(~non_constant_mask_bus)} | tersisa: {X_bus_filtered.shape[1]}")

std_branch = np.std(X_branch_agg, axis=0)
non_constant_mask_branch = std_branch > 1e-10
X_branch_filtered = X_branch_agg[:, non_constant_mask_branch]
branch_names_filtered = branch_feature_names[non_constant_mask_branch]

# =====
# FEATURE ENGINEERING
# =====
def create_engineered_features(X_filt, f_names):
    n_samples = X_filt.shape[0]
    names_list = list(f_names)
    new_features, new_names = [], []

    def get_indices(prefix):
        return [i for i, name in enumerate(f_names) if name.startswith(prefix)]

    pl_idx, ql_idx, pg_idx, qg_idx, v_idx = (get_indices(p) for p in ['PL_', 'QL_', 'PG_', 'QG_', 'V_'])

    if len(pl_idx) > 0:
        PL_data = X_filt[:, pl_idx]
        total_pl = np.sum(PL_data, axis=1)
        new_features.extend([total_pl, np.mean(PL_data, axis=1), np.max(PL_data, axis=1),
                             np.std(PL_data, axis=1)])
        new_names.extend(['Total_P_Load', 'Mean_P_Load', 'Max_P_Load', 'Std_P_Load'])

    if len(ql_idx) > 0:
        QL_data = X_filt[:, ql_idx]
        new_features.extend([np.sum(QL_data, axis=1), np.mean(QL_data, axis=1), np.max(QL_data,
axis=1)])
        new_names.extend(['Total_Q_Load', 'Mean_Q_Load', 'Max_Q_Load'])

    if len(pg_idx) > 0:
        PG_data = X_filt[:, pg_idx]
        total_pg = np.sum(PG_data, axis=1)
        new_features.append(total_pg)
        new_names.append('Total_P_Gen')
        if len(pl_idx) > 0:
            new_features.extend([total_pl / (total_pg + 1e-10), total_pg - total_pl])
            new_names.extend(['Load_Factor', 'P_Imbalance'])

    if len(qg_idx) > 0:
        new_features.append(np.sum(X_filt[:, qg_idx], axis=1))
        new_names.append('Total_Q_Gen')

    if len(v_idx) > 0:
        V_data = X_filt[:, v_idx]
        new_features.extend([
            np.mean(V_data, axis=1), np.std(V_data, axis=1),
            np.min(V_data, axis=1), np.max(V_data, axis=1),
            np.max(V_data, axis=1) - np.min(V_data, axis=1),
            np.mean(np.abs(V_data - 1.0), axis=1),
            np.sum(V_data < 0.95, axis=1), np.sum(V_data > 1.05, axis=1)
        ])
        new_names.extend(['V_Mean', 'V_Std', 'V_Min', 'V_Max', 'V_Range',
            'V_Deviation', 'V_UnderVoltage', 'V_OverVoltage'])

    if len(pl_idx) > 0 and len(v_idx) > 0:
        weighted_sum = np.zeros(n_samples)
        n_loaded = np.zeros(n_samples)
        for i in range(1, N_BUS + 1):
            pl_name, v_name = f'PL_{i}', f'V_{i}'
            if pl_name in names_list and v_name in names_list:
                pl_col = names_list.index(pl_name)
                v_col = names_list.index(v_name)
                weighted_sum += X_filt[:, pl_col] * X_filt[:, v_col]
                n_loaded += (X_filt[:, pl_col] > 0.01)
        new_features.extend([weighted_sum, n_loaded])
        new_names.extend(['Weighted_Load', 'N_Loaded_Buses'])

    return np.column_stack(new_features) if new_features else np.array([]).reshape(n_samples, 0),
    new_names

X_engineered, engineered_names = create_engineered_features(X_bus_filtered, bus_names_filtered)

X_enhanced = np.hstack([X_bus_filtered, X_engineered, X_branch_filtered])
all_feature_names = list(bus_names_filtered) + engineered_names + list(branch_names_filtered)

print(f"Fitur bus engineered : {len(engineered_names)}")
print(f"Fitur branch agregat : {X_branch_filtered.shape[1]}")
print(f"Total fitur final : {X_enhanced.shape[1]}")

scaler_X = RobustScaler()
X_scaled = scaler_X.fit_transform(X_enhanced)

scaler_y = StandardScaler()
y_scaled = scaler_y.fit_transform(y_all)

total_load = df['sys_load_P_MW'].values.astype(float)
load_bins = pd.qcut(total_load, q=5, labels=False, duplicates='drop')

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_scaled, test_size=37, random_state=42, stratify=load_bins
)

```

```

print(f"Training: {X_train.shape[0]} | Testing: {X_test.shape[0]}")
print(f"Features: {X_train.shape[1]}")

# =====
# MODEL RBFN
# =====
class OptimizedRBFN:
    def __init__(self, n_centers=38, learning_rate=1.8, max_iter=100,
                  patience=30, sigma_factor=0.5, reg_lambda=0.001):
        self.n_centers = n_centers
        self.learning_rate = learning_rate
        self.max_iter = max_iter
        self.patience = patience
        self.sigma_factor = sigma_factor
        self.reg_lambda = reg_lambda
        self.centers = None
        self.sigma = None
        self.weights = None
        self.train_mse = []
        self.val_mse = []

    def _init_centers(self, X):
        n_centers = min(self.n_centers, len(X) - 1)
        kmeans = KMeans(n_clusters=n_centers, init='k-means++', n_init=20,
                        max_iter=300, random_state=RANDOM_SEED)
        kmeans.fit(X)
        self.n_centers = n_centers
        return kmeans.cluster_centers_

    def _calculate_sigma(self, X=None):
        if self.n_centers < 2:
            return 1.0
        from sklearn.metrics.pairwise import pairwise_distances
        D = pairwise_distances(self.centers)
        tri = np.triu_indices_from(D, 1)
        median_dist = np.median(D[tri])
        sigma = max(median_dist * self.sigma_factor, 1e-6)
        return sigma

    def _rbf_matrix(self, X):
        X = np.asarray(X, float)
        centers = np.asarray(self.centers, float)
        X2 = np.sum(X**2, axis=1).reshape(-1, 1)
        C2 = np.sum(centers**2, axis=1).reshape(1, -1)
        XC = X.dot(centers.T)
        d2 = np.clip(X2 + C2 - 2*XC, 0, None)
        gamma = 1.0 / (2 * self.sigma * self.sigma + 1e-9)
        G = np.exp(-gamma * d2)
        G_sum = np.sum(G, axis=1, keepdims=True)
        return G / np.maximum(G_sum, 1e-12)

    def fit(self, X_train, y_train, X_val=None, y_val=None, verbose=False):
        n_samples = X_train.shape[0]
        n_outputs = y_train.shape[1]
        self.centers = self._init_centers(X_train)
        self.sigma = self._calculate_sigma(X_train)
        G_train = self._rbf_matrix(X_train)
        G_val = self._rbf_matrix(X_val) if X_val is not None else None
        I = np.eye(self.n_centers)
        self.weights = np.linalg.solve(
            G_train.T @ G_train + self.reg_lambda * I,
            G_train.T @ y_train
        )
        best_weights = self.weights.copy()
        best_val_mse = float('inf')
        patience_counter = 0
        if verbose:
            print(f"Start LMS training: N={n_samples}, M={self.n_centers}, L={n_outputs}")
        for iteration in range(self.max_iter):
            total_mse = 0
            indices = np.random.permutation(n_samples)
            for i in indices:
                G_i = G_train[i, :].reshape(1, self.n_centers)
                y_i = y_train[i, :].reshape(1, n_outputs)
                y_pred_i = G_i @ self.weights
                error = y_i - y_pred_i
                self.weights = self.weights + self.learning_rate * G_i.T @ error
                total_mse += np.sum(error**2)
            avg_mse = total_mse / n_samples
            self.train_mse.append(avg_mse)
            if verbose and ((iteration + 1) % 20 == 0 or iteration == self.max_iter - 1):
                print(f"Epoch {iteration+1}/{self.max_iter}, Avg. MSE: {avg_mse:.6e}")
            if G_val is not None and y_val is not None:
                y_pred_val = G_val @ self.weights
                mse_val = mean_squared_error(y_val, y_pred_val)
                self.val_mse.append(mse_val)
                if mse_val < best_val_mse:
                    best_val_mse = mse_val
                    best_weights = self.weights.copy()
                    patience_counter = 0
                else:
                    patience_counter += 1
                    if patience_counter >= self.patience:
                        break
            self.weights = best_weights
        if verbose:
            final_mse = self.train_mse[-1] if self.train_mse else 0

```

```

        print(f"    Final MSE: {final_mse:.6f}, sigma: {self.sigma:.6f}, centers:
{self.n_centers}")
        return self

    def predict(self, X):
        return self._rbf_matrix(X) @ self.weights

class EnsembleRBFN:
    def __init__(self, n_models=5, **kwargs):
        self.n_models = n_models
        self.models = []
        self.kwargs = kwargs

    def fit(self, X_train, y_train, verbose=True):
        n_samples = len(X_train)
        for i in range(self.n_models):
            if verbose:
                print(f"\n Model {i+1}/{self.n_models}:")
            indices = np.random.choice(n_samples, n_samples, replace=True)
            oob_mask = np.ones(n_samples, dtype=bool)
            oob_mask[indices] = False
            model = OptimizedRBFN(**self.kwargs)
            model.fit(
                X_train[indices], y_train[indices],
                X_train[oob_mask] if np.any(oob_mask) else None,
                y_train[oob_mask] if np.any(oob_mask) else None,
                verbose=verbose
            )
            self.models.append(model)
        return self

    def predict(self, X):
        predictions = np.array([m.predict(X) for m in self.models])
        return np.mean(predictions, axis=0)

print("\n" + "="*80)
print("TRAINING ENSEMBLE RBFN (bus + branch agregat)")
print("="*80)

ensemble_rbf = EnsembleRBFN(
    n_models=5,
    n_centers=38,
    learning_rate=1.8,
    max_iter=100,
    patience=30,
    sigma_factor=0.5,
    reg_lambda=0.001
)

print("\nTraining Ensemble RBFN...")
ensemble_rbf.fit(X_train, y_train, verbose=True)
y_pred = ensemble_rbf.predict(X_test)

print("\n" + "-"*80)
print("HASIL TRAINING:")
best_mse = float('inf')
best_model_idx = -1
for i, model in enumerate(ensemble_rbf.models):
    final_mse = model.train_mse[-1] if model.train_mse else 0
    print(f" Model {i+1}: MSE = {final_mse:.6f}, sigma = {model.sigma:.6f}")
    if final_mse < best_mse:
        best_mse = final_mse
        best_model_idx = i
print("-"*80)
print(f"\nModel terbaik: Model {best_model_idx+1} dengan MSE = {best_mse:.6f}")

# =====
# EVALUASI PADA TEST SET
# =====
y_pred_orig = scaler_y.inverse_transform(y_pred)
y_test_orig = scaler_y.inverse_transform(y_test)

pip_true, piv_true = y_pred_orig[:, 0], y_pred_orig[:, 1]
pip_true, piv_true = y_test_orig[:, 0], y_test_orig[:, 1]

pip_r2 = r2_score(pip_true, pip_pred)
pip_rmse = np.sqrt(mean_squared_error(pip_true, pip_pred))
pip_mae = mean_absolute_error(pip_true, pip_pred)
pip_mape = np.mean(np.abs((pip_true - pip_pred) / (np.abs(pip_true) + 1e-10))) * 100

piv_r2 = r2_score(piv_true, piv_pred)
piv_rmse = np.sqrt(mean_squared_error(piv_true, piv_pred))
piv_mae = mean_absolute_error(piv_true, piv_pred)
piv_mape = np.mean(np.abs((piv_true - piv_pred) / (np.abs(piv_true) + 1e-10))) * 100

print("\n" + "="*80)
print("HASIL EVALUASI PADA TEST SET")
print("="*80)
print(f"PIP -> R2={pip_r2:.4f} | RMSE={pip_rmse:.6f} | MAE={pip_mae:.6f} | MAPE={pip_mape:.2f}%")
print(f"PIV -> R2={piv_r2:.4f} | RMSE={piv_rmse:.6f} | MAE={piv_mae:.6f} | MAPE={piv_mape:.2f}%")

# =====
# SIMPAN MODEL
# =====
model_data = {
    'ensemble_model': ensemble_rbf,
    'scaler_X': scaler_X,
    'scaler_y': scaler_y,

```

```

        'bus_names_filtered': bus_names_filtered,
        'branch_names_filtered': branch_names_filtered,
        'engineered_names': engineered_names,
        'non_constant_mask_bus': non_constant_mask_bus,
        'non_constant_mask_branch': non_constant_mask_branch,
        'ALL_LINE_NAMES': ALL_LINE_NAMES
    }

    with open('rbfn_psssa_csv_model.pkl', 'wb') as f:
        pickle.dump(model_data, f)

    print("\nModel saved: rbf_n_psssa_csv_model.pkl")
    print("="*80)
    print("RBFN TRAINING SELESAI!")
    print("="*80)

```

Lampiran 4. Codingan Mesin Learning Xgboost

```

import sys
import pickle
import warnings

import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import GroupKFold, RandomizedSearchCV, train_test_split
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

warnings.filterwarnings("ignore")

RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)

print("=" * 80)
print("XGBoost - POWER SYSTEM STATIC SECURITY ASSESSMENT (CSV) - v8 (ANTI-OVERFIT)")
print("IEEE 30-Bus Test System")
print("=" * 80)

# -----
# 0. BACA FILE CSV
# -----
try:
    from google.colab import files # noqa: F401
    IN_COLAB = True
except ImportError:
    IN_COLAB = False

if IN_COLAB:
    print("\nSilakan upload file CSV Anda...")
    uploaded = files.upload()
    filename = list(uploaded.keys())[0]
else:
    filename = sys.argv[1] if len(sys.argv) > 1 else "ml_dataset_power_flow.csv"

print(f"Membaca file '{filename}' ...")
df = pd.read_csv(filename)
print(f"File berhasil dibaca. Shape: {df.shape}")

ALL_LINE_NAMES = [
    '1-2', '1-3', '2-4', '3-4', '2-5', '2-6',
    '4-6', '5-7', '6-7', '6-8', '6-9', '6-10',
    '9-10', '4-12', '12-13', '12-14', '12-15', '12-16',
    '14-15', '16-17', '15-18', '18-19', '19-20', '10-20',
    '10-17', '10-21', '10-22', '21-22', '15-23', '22-24',
    '23-24', '24-25', '25-26', '25-27', '28-27', '27-29',
    '27-30', '29-30', '8-28', '6-28', '9-11'
]

N_BUS = 30
N_BRANCH = 41

# -----
# 1. BANGUN FITUR PER BARIS (sama seperti v7 dari sisi sumber data)
# -----
def build_input_output_from_row(row, all_line_names, n_bus=N_BUS, n_branch=N_BRANCH):
    PL, QL, PG, QG, V_mag, V_ang = [], [], [], [], [], []
    for i in range(1, n_bus + 1):
        b = f"bus{i:02d}"
        PL.append(row.get(f"{b}_load_P", 0.0))
        QL.append(row.get(f"{b}_load_Q", 0.0))
        PG.append(row.get(f"{b}_gen_P", 0.0))
        QG.append(row.get(f"{b}_gen_Q", 0.0))
        V_mag.append(row.get(f"{b}_V_mag", 0.0))
        V_ang.append(row.get(f"{b}_V_ang", 0.0))

    PL, QL, PG, QG = (np.array(a, dtype=float) for a in (PL, QL, PG, QG))
    V_mag, V_ang = np.array(V_mag, dtype=float), np.array(V_ang, dtype=float)

    S_from, S_loss = [], []
    for i in range(1, n_branch + 1):
        br = f"br{i:02d}"
        from_P, from_Q = row.get(f"{br}_from_P", 0.0), row.get(f"{br}_from_Q", 0.0)

```

```

        loss_P, loss_Q = row.get(f"{br}_loss_P", 0.0), row.get(f"{br}_loss_Q", 0.0)
        S_from.append(np.sqrt(float(from_P) ** 2 + float(from_Q) ** 2))
        S_loss.append(np.sqrt(float(loss_P) ** 2 + float(loss_Q) ** 2))
    S_from, S_loss = np.array(S_from, dtype=float), np.array(S_loss, dtype=float)

    Ki = np.zeros(len(all_line_names))
    line_name = str(row["contingency"]).replace("L", "")
    if line_name in all_line_names:
        Ki[all_line_names.index(line_name)] = 1
    else:
        return None, None

    raw = dict(PL=PL, QL=QL, PG=PG, QG=QG, V_mag=V_mag, V_ang=V_ang,
              S_from=S_from, S_loss=S_loss, Ki=Ki)
    y = np.array([row["label_PIP"], row["label_PIV"]])
    return raw, y

raw_list, y_list, meta_list = [], [], []
for _, row in df.iterrows():
    raw, y = build_input_output_from_row(row, ALL_LINE_NAMES)
    if raw is not None:
        raw_list.append(raw)
        y_list.append(y)
        meta_list.append({
            "level": row["level_pct"],
            "line_outage": str(row["contingency"]).replace("L", ""),
        })

y_all = np.array(y_list)
metadata_df = pd.DataFrame(meta_list)
groups = metadata_df["line_outage"].values # <-- dipakai utk GroupKFold
level_labels = metadata_df["level"].values

print(f"Total baris valid: {len(y_all)} | saluran unik: {metadata_df['line_outage'].nunique()}")
print(f" | level beban: {sorted(metadata_df['level'].unique())}")

# -----
# 2. FEATURE ENGINEERING RAMPING (fokus fitur yang benar2 relevan secara
#    fisik, JUMLAHNYA DIBATASI supaya rasio fitur:sampel sehat)
# -----
def make_feature_row(row):
    S_from, S_loss = row["S_from"], row["S_loss"]
    PL, QL, PG, QG = row["PL"], row["QL"], row["PG"], row["QG"]
    V_mag, V_ang = row["V_mag"], row["V_ang"]

    feats, names = [], []

    def add(val, name):
        feats.append(val); names.append(name)

    # --- ringkasan branch loading (paling relevan secara fisik utk PIP) ---
    add(S_from.mean(), "Mean_Branch_Loading")
    add(S_from.max(), "Max_Branch_Loading")
    add(S_from.std(), "Std_Branch_Loading")
    add(np.sum(S_from ** 2), "SumSq_Branch_Loading")
    add(S_from.max() / (S_from.mean() + 1e-10), "Loading_Peak_Ratio")
    add(np.sort(S_from)[-3:].mean(), "Top3_Mean_Branch_Loading")
    add(S_loss.sum(), "Total_Branch_Loss")
    add(S_loss.max(), "Max_Branch_Loss")

    # --- ringkasan tegangan (relevan utk PIV) ---
    add(V_mag.mean(), "V_Mean")
    add(V_mag.std(), "V_Std")
    add(V_mag.min(), "V_Min")
    add(V_mag.max(), "V_Max")
    add(V_mag.max() - V_mag.min(), "V_Range")
    add(np.mean(np.abs(V_mag - 1.0)), "V_Deviation")
    add(np.sum(V_mag < 0.95), "V_UnderVoltage")
    add(np.sum(V_mag > 1.05), "V_OverVoltage")
    add(V_ang.max() - V_ang.min(), "V_Ang_Range")
    add(V_ang.std(), "V_Ang_Std")

    # --- ringkasan beban/pembangkitan (konteks tingkat operasi) ---
    total_pl = PL.sum()
    total_pg = PG.sum()
    add(total_pl, "Total_P_Load")
    add(QL.sum(), "Total_Q_Load")
    add(total_pg, "Total_P_Gen")
    add(QG.sum(), "Total_Q_Gen")
    add(total_pl / (total_pg + 1e-10), "Load_Factor")

    return np.array(feats, dtype=float), names

X_rows, feat_names = [], None
for raw in raw_list:
    x, feat_names = make_feature_row(raw)
    X_rows.append(x)
X_enhanced = np.array(X_rows)

# tambahkan indikator saluran (Ki) -- one-hot 41 kolom TAPI ini penting:
# karena tiap saluran akan dipisah total via GroupKFold, Ki hanya berguna
# saat prediksi utk saluran yg SAMA sudah pernah dilihat model (skenario
# operasional yg wajar: semua 37 saluran ini memang saluran nyata di sistem).
Ki_matrix = np.array([row["Ki"] for row in raw_list])
Ki_names = [f"Ki_{i+1}" for i in range(len(ALL_LINE_NAMES))]
X_enhanced = np.hstack([X_enhanced, Ki_matrix])

```

```

feat_names = feat_names + Ki_names

print(f"Jumlah fitur (v8, dirampingkan): {X_enhanced.shape[1]} "
      f"(vs >300 di v7) | rasio fitur:sampel = "
      f"{X_enhanced.shape[1]:{len(y_all)}}")

y_pip, y_piv = y_all[:, 0], y_all[:, 1]

# -----
# 3. HYPERPARAMETER SPACE KONSERVATIF (sesuai ukuran data ~222 baris)
# -----
PARAM_DIST = {
    "max_depth": [2, 3, 4, 5],
    "n_estimators": [50, 100, 200, 300, 400],
    "learning_rate": [0.02, 0.05, 0.08, 0.1],
    "subsample": [0.6, 0.7, 0.8, 0.9],
    "colsample_bytree": [0.5, 0.6, 0.7, 0.8],
    "min_child_weight": [1, 3, 5, 7],
    "gamma": [0.0, 0.01, 0.05, 0.1],
    "reg_alpha": [0.01, 0.1, 0.5, 1.0],
    "reg_lambda": [1.0, 2.0, 5.0, 10.0],
}

def tune_with_group_cv(X, y, groups, target_name, n_iter=40, n_splits=5):
    """RandomizedSearchCV dgn CV yg di-group by contingency -> tidak ada
    saluran yang sama nongol di fold train & validasi CV sekaligus."""
    gkf = GroupKFold(n_splits=n_splits)
    cv_splits = list(gkf.split(X, y, groups=groups))

    base_model = xgb.XGBRegressor(
        objective="reg:squarederror", tree_method="hist",
        random_state=RANDOM_SEED, verbosity=0,
    )

    search = RandomizedSearchCV(
        estimator=base_model, param_distributions=PARAM_DIST,
        n_iter=n_iter, scoring="r2", cv=cv_splits,
        random_state=RANDOM_SEED, n_jobs=-1, refit=False,
    )
    search.fit(X, y)
    print(f"[{target_name}] Best params (RandomizedSearchCV, group-CV): "
          f"{search.best_params}")
    print(f"[{target_name}] Best mean CV R2 (grouped, saluran tak-terlihat) "
          f"f= {search.best_score:.4f}")
    return search.best_params_

scaler_X = RobustScaler()
X_scaled = scaler_X.fit_transform(X_enhanced)

print("\n" + "=" * 80)
print("TUNING HYPERPARAMETER (RandomizedSearchCV, GroupKFold by saluran)")
print("=" * 80)
best_params_pip = tune_with_group_cv(X_scaled, y_pip, groups, "PIP")
best_params_piv = tune_with_group_cv(X_scaled, y_piv, groups, "PIV")

# -----
# 4. EVALUASI UTAMA: GroupKFold(6) di SELURUH DATA -- estimasi generalisasi
# yang jujur ke saluran yang belum pernah dilihat model.
# -----
def evaluate_group_cv(X, y, groups, params, target_name, n_splits=6):
    gkf = GroupKFold(n_splits=n_splits)
    r2s, rmse, maes = [], [], []
    for tr_idx, te_idx in gkf.split(X, y, groups=groups):
        model = xgb.XGBRegressor(
            *params, objective="reg:squarederror", tree_method="hist",
            random_state=RANDOM_SEED, verbosity=0,
        )
        model.fit(X[tr_idx], y[tr_idx])
        pred = model.predict(X[te_idx])
        r2s.append(r2_score(y[te_idx], pred))
        rmse.append(np.sqrt(mean_squared_error(y[te_idx], pred)))
        maes.append(mean_absolute_error(y[te_idx], pred))
    print(f"\n[{target_name}] GroupKFold(6) - evaluasi jujur ke saluran BARU:")
    print(f"R2 per fold = {np.round(r2s, 4)} -> mean={np.mean(r2s):.4f}, std={np.std(r2s):.4f}")
    print(f"RMSE per fold = {np.round(rmse, 4)} -> mean={np.mean(rmse):.4f}")
    print(f"MAE per fold = {np.round(maes, 4)} -> mean={np.mean(maes):.4f}")
    return {"R2_mean": float(np.mean(r2s)), "R2_std": float(np.std(r2s)),
            "RMSE_mean": float(np.mean(rmse)), "MAE_mean": float(np.mean(maes))}

print("\n" + "=" * 80)
print("EVALUASI UTAMA (JUJUR): GroupKFold by saluran outage")
print("=" * 80)
gcv_pip = evaluate_group_cv(X_scaled, y_pip, groups, best_params_pip, "PIP")
gcv_piv = evaluate_group_cv(X_scaled, y_piv, groups, best_params_piv, "PIV")

# -----
# 5. UNTUK PERBANDINGAN: split random 185/37 ala v7 (TIDAK di-group)
# -- diberi label jelas bahwa ini rawan bocor & optimis-palsu.
# -----
print("\n" + "=" * 80)
print("PEMBANDING: split random 185/37 seperti v7 (stratify level saja)")
print("SALURAN YANG SAMA BISA MUNCUL DI TRAIN & TEST -> hasil rawan optimis")
print("=" * 80)

idx_all = np.arange(len(X_scaled))
train_idx, test_idx = train_test_split(
    idx_all, test_size=37, random_state=RANDOM_SEED, stratify=level_labels

```

```

)
overlap_lines = set(groups[train_idx]) & set(groups[test_idx])
print(f"Jumlah saluran yang muncul di TRAIN & TEST sekaligus: {len(overlap_lines)} "
      f"dari {metadata_df['line_outage'].nunique()} total saluran")

def fit_eval_naive_split(X, y, params, target_name):
    model = xgb.XGBRegressor(**params, objective="reg:squarederror",
                             tree_method="hist", random_state=RANDOM_SEED, verbosity=0)
    model.fit(X[train_idx], y[train_idx])
    pred_tr = model.predict(X[train_idx])
    pred_te = model.predict(X[test_idx])
    r2_tr, r2_te = r2_score(y[train_idx], pred_tr), r2_score(y[test_idx], pred_te)
    print(f"[{target_name}] (split naive) R2 train={r2_tr:.4f} | R2 test={r2_te:.4f} "
          f"| gap={r2_tr - r2_te:.4f}")
    return r2_tr, r2_te

fit_eval_naive_split(X_scaled, y_pip, best_params_pip, "PIP")
fit_eval_naive_split(X_scaled, y_piv, best_params_piv, "PIV")

print("\n>>> Bandingkan angka 'split naive' di atas dengan GroupKFold di atasnya.")
print(">>> Kalau split naive jauh lebih tinggi, itu konfirmasi kebocoran data v7.")

# -----
# 6. MODEL FINAL UNTUK DEPLOY: dilatih di SEMUA data dgn hyperparameter
# konservatif hasil tuning. Estimasi performa yang dipakai untuk
# dilaporkan ke pengguna = hasil GroupKFold(6) di atas, BUKAN skor
# training penuh (yang pasti tinggi krn model sudah lihat semua data).
# -----
xgb_pip_final = xgb.XGBRegressor(**best_params_pip, objective="reg:squarederror",
                                 tree_method="hist", random_state=RANDOM_SEED, verbosity=0)
xgb_pip_final.fit(X_scaled, y_pip)

xgb_piv_final = xgb.XGBRegressor(**best_params_piv, objective="reg:squarederror",
                                 tree_method="hist", random_state=RANDOM_SEED, verbosity=0)
xgb_piv_final.fit(X_scaled, y_piv)

xgb_pip_final.save_model("xgboost_pip_final_v8.json")
xgb_piv_final.save_model("xgboost_piv_final_v8.json")

model_data = {
    "xgb_pip": xgb_pip_final, "xgb_piv": xgb_piv_final, "scaler_X": scaler_X,
    "feature_names": feat_names, "ALL_LINE_NAMES": ALL_LINE_NAMES,
    "best_params_pip": best_params_pip, "best_params_piv": best_params_piv,
    "honest_generalization_metrics": {"PIP": gcv_pip, "PIV": gcv_piv},
}
with open("xgboost_psssa_model_v8.pkl", "wb") as f:
    pickle.dump(model_data, f)

print("\n" + "=" * 80)
print("RINGKASAN AKHIR")
print("=" * 80)
print(f"PIP -> GroupKFold R2 (saluran baru): {gcv_pip['R2_mean']:.4f} ± {gcv_pip['R2_std']:.4f}")
print(f"PIV -> GroupKFold R2 (saluran baru): {gcv_piv['R2_mean']:.4f} ± {gcv_piv['R2_std']:.4f}")
print("Model & metrik disimpan -> xgboost_psssa_model_v8.pkl")
print("=" * 80)

```

Lampiran 5. Datasheet Mesin Learning

Link drive :

https://drive.google.com/drive/folders/1scZEevpZljdWg79am6kjMz8HAnZ5JGfm?usp=drive_link

