Komparasi Algoritma Random Forest dan Gradient Boosting Untuk Prediksi Air Quality Index (AQI)

DOSEN PENGAMPU:
PRASETYO WIBOWO S.ST, M.KOM

DISUSUN OLEH:
1. ROBI'ATUL ADAWIYAH (3323600041)
2. MOCHAMAD ARIEL SULTON (3323600054)
3. JOGI FERGIO SCHUMACHER (3323600060)

Proyek Akhir
Mata Kuliah Praktikum Mesin Pembelajaran

DAFTAR ISI

BAB I
PENDAHULUAN

1.1.Latar Belakang

Kualitas udara merupakan salah satu faktor lingkungan yang sangat penting bagi kesehatan manusia dan keberlanjutan ekosistem. *Air Quality Index* (AQI) adalah indikator yang digunakan untuk menggambarkan tingkat pencemaran udara dan dampaknya terhadap kesehatan. Peningkatan urbanisasi dan aktivitas industri di Indonesia khususnya Jakarta telah menyebabkan peningkatan signifikan dalam polusi udara. Berdasarkan data dari Badan Lingkungan Hidup, Jakarta sering kali mencatat tingkat AQI yang tidak sehat, yang berpotensi menimbulkan risiko kesehatan bagi penduduknya, seperti penyakit pernapasan dan gangguan kardiovaskular.

Peningkatan pencemaran udara membutuhkan perhatian dari berbagai pihak dan dengan perkembangan teknologi yang pesat penelitian mengenai prediksi kualitas udara menggunakan *machine learning* menjadi sangat relevan. Algoritma Random Forest dan teknik ensemble learning seperti Boosting khususnya Gradient Boosting telah terbukti efektif dalam menangani masalah prediksi dalam berbagai domain. Random Forest merupakan algoritma berbasis pohon keputusan yang mampu menangani data kompleks dengan baik, sementara Gradient Boosting menawarkan pendekatan yang lebih adaptif dan mampu meningkatkan akurasi prediksi dengan mengurangi bias model. Oleh karena itu, komparasi antara kedua metode ini diharapkan dapat memberikan wawasan yang lebih mengenai efektivitas masing-masing algoritma dalam memprediksi AQI.

Data dalam penelitian ini diambil dari dataset yang tersedia di Kaggle yang mencakup informasi mengenai berbagai variabel atmosfer dan lingkungan yang berpengaruh terhadap kualitas udara di Jakarta dari tahun 2010 hingga 2021. Meskipun terdapat sejumlah penelitian yang telah dilakukan mengenai prediksi kualitas udara, masih kurang literatur yang membahas perbandingan langsung antara Random Forest dan teknik Boosting dalam konteks AQI. Sebagian besar penelitian sebelumnya lebih fokus pada penggunaan satu jenis algoritma tanpa mempertimbangkan komparasi antara berbagai metode.

Analisis masalah dalam penelitian ini berfokus pada bagaimana variabel-variabel lingkungan berinteraksi dan mempengaruhi kualitas udara, serta bagaimana algoritma machine learning dapat digunakan untuk mengidentifikasi pola-pola ini. Dengan memahami interaksi tersebut, diharapkan dapat dicapai model prediksi yang lebih akurat yang dapat digunakan untuk perencanaan dan pengelolaan kualitas udara di Jakarta.

Penelitian ini diharapkan dapat memberikan pemahaman yang mendalam mengenai faktor-faktor yang mempengaruhi kualitas udara dan bagaimana memprediksi perubahan AQI sehingga bisa memberikan rekomendasi kebijakan yang efektif dalam mengatasi polusi udara serta meningkatkan perhatian masyarakat

terhadap dampak kesehatan yang diakibatkan oleh kualitas udara. Selain itu, dengan meningkatnya perhatian terhadap perubahan iklim dan polusi udara global, penelitian ini dapat memberikan kontribusi terhadap pengembangan model prediksi yang lebih akurat dan efisien.

1.2.Rumusan Masalah

1.2.1. Bagaimana kinerja algoritma Random Forest dalam memprediksi *Air Quality Index* (AQI)?

1.2.2. Seberapa efektif algoritma Gradient Boosting dalam menghasilkan prediksi kualitas udara dibandingkan metode Random Forest?

1.2.3. Apakah terdapat perbedaan yang signifikan dalam akurasi, presisi, recall, dan F1-Score dalam algoritma Random Forest dan Gradient Boosting dalam prediksi *Air Quality Index* (AQI)?

1.2.4. Variabel independen manakah yang memiliki pengaruh paling dominan terhadap prediksi *Air Quality Index* (AQI) dalam model Random Forest dan Gradient Boosting?

1.2.5. Bagaimana performa kedua algoritma dalam menangani kompleksitas dan variabilitas data *Air Quality Index* (AQI)?

1.3.Tujuan

1.3.1. Mengukur akurasi prediksi kedua algoritma menggunakan matrik evaluasi standar (akurasi, presisi, recall, F1-score)

1.3.2. Mengidentifikasi variabel independen yang memiliki pengaruh signifikan terhadap prediksi AQI

1.3.3. Memberikan rekomendasi algoritma terbaik untuk prediksi AQI

BAB II

KAJIAN PUSTAKA

2.1. Penelitian Sebelumnya

Penelitian yang dilakukan oleh M. Ja'far Sodiq dengan judul penelitian "PERBANDINGAN METODE NAIVE BAYES DAN K-NEAREST NEIGHBOR PADA KLASIFIKASI KUALITAS UDARA DI DKI JAKARTA" dengan menggunakan 5 variabel independen dan 1 variabel dependen dengan tipe data kategorikal memiliki luaran hasil bahwa metode *K-Nearest Neighbor* dengan *Manhattan Distance* dan nilai K sebesar 7 memiliki tingkat akurasi yang lebih baik dibandingkan metode *Naïve Bayes* yakni sebesar 97.3396 % berbanding dengan 91.862 %. Pada metode *Naïve Bayes*, jumlah data latih dan keberagaman data berpengaruh pada akurasi yang dihasilkan. Sedangkan pada metode *K-Nearest Neighbor*, metode pencarian jarak dan nilai K berpengaruh pada akurasi yang dihasilkan.

Penelitian yang dilakukan oleh Indrawata Wardhana et al yang memiliki topik bahasan mengenai klasifikasi kacang kering menggunakan algoritma *Gradient Boosting Machine, Random Forest* dan *Light GBM*. Dalam penelitian ini digunakan 7 jenis kacang kering untuk dilakukan klasifikasi berdasarkan 17 fitur. Kesimpulan hasil yang didapat dari penelitian ini menunjukkan bahwa algoritma *Gradient Boosting Machine, Random Forest* dan *Light GBM* memiliki hasil prediksi klasifikasi kelas yang hampir sama. Namun dalam fase training, memiliki perbedaan yang cukup signifikan. *Random Forest* memiliki kemampuan akurasi training hingga 96 persen, *Gradient Boosting Machine* hanya mampu pada angka 93 persen, sedangkan *Light GBM* yang awalnya terjadi penurunan tajam pada parameter *learning rate* namun pada parameter estimator mampu mencapai 99,9 persen.

Penelitian selanjutnya, yaitu penelitian yang dilakukan oleh Eva Sapan Patasik dan Sri Yulianto dengan judul penelitian *"CLASSIFICATION OF REGIONAL LANGUAGES USING METHODS GRADIENT BOOST AND RANDOM FOREST"* memiliki luaran perbandingan yang dilakukan pada penelitian klasifikasi bahasa Jawa, Nias dan Toraja dengan menggunakan metode *gradient boost* dan *random forest*, kedua metode yang digunakan cukup bagus dalam melakukan klasifikasi bahasa maupun penerjemah bahasa dengan hasil tingkat akurasi 0.8 atau 80%. Dengan hasil akhir yang dilihat pada proses perhitungan *confusion matrix* mencakup nilai *Precision, Recall dan FI-score* yang menghasilkan nilai *accuracy* pada metode *gradient boost* lebih tinggi dibandingkan *random forest*. Perbandingan kedua metode dapat dilihat pada hasil penelitian *gradient boost* memiliki nilai *accuracy gradient boost* sebesar 0.8850 atau 88.5% dibandingkan dengan metode *random forest* memiliki nilai *accuracy random forest* lebih rendah yaitu 0.8794 atau 87.94%, sehingga pada penelitian ini dapat disimpulkan bahwa metode *gradient boost* merupakan metode yang lebih baik dalam melakukan klasifikasi bahasa daerah.

2.2. Dasar Teori

2.2.1. Pencemaran Udara

Menurut Keputusan Menteri Kesehatan Republik Indonesia Nomor 1407 Tahun 2002, pencemaran udara adalah masuknya atau dimasukkannya zat, energi, dan atau komponen lain kedalam udara oleh kegiatan manusia sehingga mutu udara turun sampai ke tingkat tertentu yang menyebabkan atau mempengaruhi kesehatan manusia.

2.2.2. Indeks Standar Pencemaran Udara

Menurut Keputusan Menteri Negara Lingkungan Hidup Nomor KEP-45 Tahun 1997, indeks standar pencemaran udara adalah angka yang tidak mempunyai satuan yang menggambarkan kondisi kualitas udara ambien di lokasi dan waktu tertentu yang didasarkan kepada dampak terhadap kesehatan manusia, nilai estetika dan makhluk hidup lainnya. Parameter yang digunakan dalam perhitungan indeks standar pencemaran udara ada 5, diantaranya:

a. Partikel Debu (PM10)

   Partikel debu (PM) adalah istilah untuk partikel padat atau cair yang ditemukan di udara. Partikel dengan ukuran besar atau cukup padat biasanya disebut asap, sedangkan partikel yang sangat kecil dapat dilihat dengan bantuan mikroskop elektron.

b. Sulfur Dioksida (SO2)

   Pencemaran oleh sulfur oksida terutama disebabkan oleh dua komponen sulfur berbentuk gas yang tidak berwarna, yaitu sulfur dioksida (SO2) dan sulfur trioksida (SO3) kemudian keduanya menjadi sulfur oksida (SOx). Sulfur Dioksida memiliki karakteristik bau yang tajam dan tidak mudah terbakar diudara, sedangkan sulfur trioksida merupakan komponen yang tidak reaktif.

c. Karbon Monoksida (CO)

   Karbon monoksida merupakan senyawa yang tidak berbau, tidak berasa dan pada suhu udara normal berbentuk gas yang tidak berwarna. Tidak seperti senyawa lain, CO mempunyai potensi bersifat racun yang berbahaya karena dapat menghalangi hemoglobin mengangkut Oksigen yang dibutuhkan oleh tubuh manusia. Hal ini disebabkan karena hemoglobin lebih mudah mengikat Karbon Monoksida dibandingkan Oksigen.

d. Ozon (O3)

   Ozon (O3) merupakan bentuk oksigen yang sangat reaktif. Ozon memiliki ciri berwarna biru pucat dan memiliki bau yang menyengat. Ozon menjadi berbahaya ketika muncul di

permukaan tanah dengan konsentrasi diatas 50 ppm. Ozon menjadi berbahaya karena Ozon merupakan oksidan yang sangat kuat, sehingga ketika terhirup Ozon dapat merusak jaringan mukosa dan pernapasan manusia, binatang dan tumbuhan.

2.2.3. Random Forest

Random Forest adalah algoritma *machine learning* yang bekerja dengan menggunakan banyak pohon keputusan, menggabungkan metode *bagging* dan *random subspaces*. Algoritma ini telah terbukti memiliki tingkat keberhasilan yang tinggi dalam melakukan prediksi dan klasifikasi, sehingga menjadi salah satu algoritma terbaik yang dapat diterapkan di berbagai bidang. Kelebihan *Random Forest* meliputi kemampuannya untuk meningkatkan akurasi meskipun terdapat data yang hilang, serta kemampuannya dalam mengatasi *outlier* dan menghasilkan *error* yang relatif rendah. Selain itu, *Random Forest* juga memiliki proses seleksi fitur yang efektif, memungkinkan pemilihan fitur terbaik untuk meningkatkan performa model klasifikasi. Dengan kelebihan tersebut, *Random Forest* dapat bekerja dengan baik pada *big data* yang memiliki parameter kompleks secara efektif.

2.2.4. Gradient Boosting

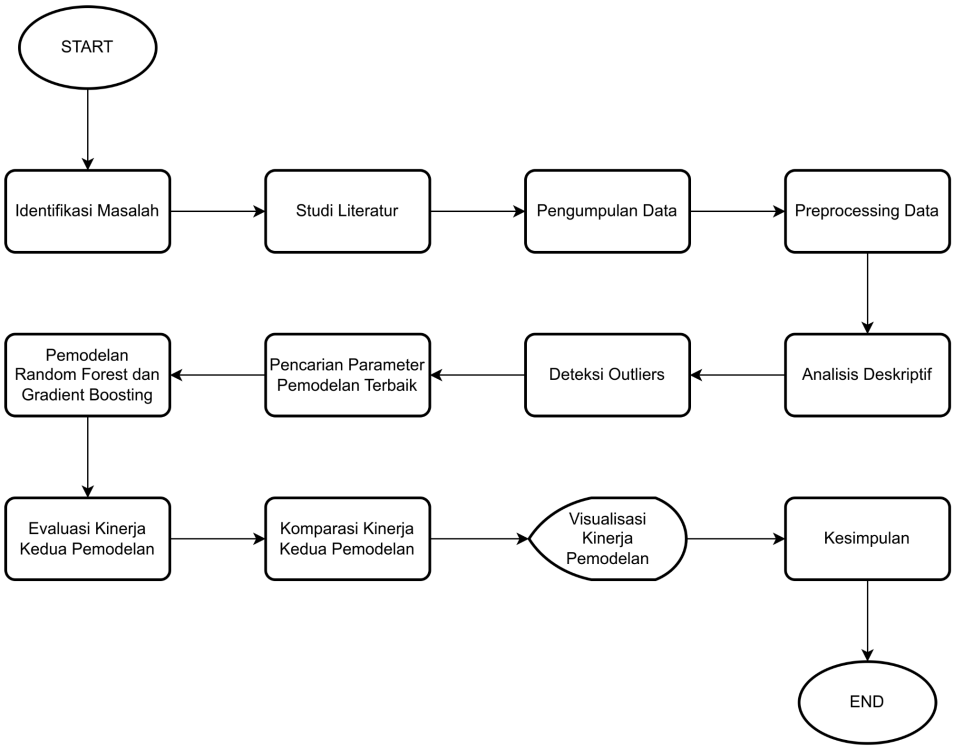
Gradient Boosting merupakan algoritma yang menggunakan teknik *ensemble* dari *decision tree*, yang dapat menyelesaikan persoalan klasifikasi dan prediksi data. Teknik *boosting* ini dapat dilihat sebagai metode model rata-rata yang awalnya dirancang untuk metode klasifikasi tetapi juga dapat diaplikasikan pada metode regresi, seperti dengan metode *bagging* yang memanfaatkan voting untuk tujuan mengklasifikasikan atau menghitung rata-rata estimasi numerik untuk output model individu tunggal. Persamaan lainnya yaitu menggabungkan model yang mempunyai jenis yang sama, seperti pada metode *decision tree.*

BAB III
METODOLOGI PENELITIAN

3.1. Dataset

Dataset yang digunakan pada penelitian ini adalah dataset sekunder *Air Quality Index* (AQI) di Jakarta dari website kaggle (Air Quality Index in Jakarta) yang didalamnya terdapat Indeks Standar Pencemar Udara (ISPU) dari 5 stasiun pemantauan kualitas udara di DKI Jakarta dari tahun 2010 sampai 2023 dengan total 4625 *record*.

3.2. Metodologi



Gambar 3.2 Flowchart Metodologi

Tahap awal dimulai dengan identifikasi masalah yang akan diangkat. Tahap kedua melakukan studi literatur dimana studi literatur merupakan suatu penyelesain untuk memecahkan persoalan dengan melakukan atau menelusuri berbagai informasi atau sumber-sumber tulisan atau jurnal yang pernah dibuat sebelumnya, buku, beserta referensi dan sumber yang terpercaya. Tahap ketiga pengumpulan data, dimana dilakukan dengan mengumpulkan data dari sumber internet. Tahap keempat melakukan preprocessing data untuk mempersiapkan data agar siap diolah lebih lanjut. Tahap kelima melakukan analisis deskriptif pada data untuk mengetahui ringkasan statistik dari data. Tahap keenam melakukan deteksi outliers atau pencilan menggunakan visualisasi boxplot. Tahap ketujuh melakukan pencarian parameter terbaik untuk pemodelan yang akan dilatih menggunakan metode grid search. Tahap

kedelapan melakukan pemodelan random forest dan gradient boosting dengan menggunakan parameter terbaik yang telah ditemukan sebelumnya. Tahap kesembilan melakukan evaluasi kinerja pada kedua pemodelan dengan luaran yang berbentuk sebuah laporan klasifikasi atau classification report. Tahap kesepuluh dan kesebelas membuat visualisasi bar chart untuk mengetahui komparasi dari kedua pemodelan untuk menentukan pemodelan yang memiliki hasil prediksi klasifikasi yang terbaik. Tahap akhir yakni membuat kesimpulan yang telah didapat dari penelitian yang telah dilakukan.

3.3. Variabel Penelitian

Variabel penelitian adalah suatu atribut atau sifat atau nilai dari orang, obyek, organisasi, atau kegiatan yang mempunyai variasi tertentu yang ditetapkan oleh peneliti untuk dipelajari dan kemudian ditarik kesimpulannya (Sugiyono, 2016 :68). Variabel dalam penelitian ini terdiri dari variabel independen (variabel bebas) dan variabel dependen (variabel terikat) yang lebih jelasnya bisa dilihat pada Tabel 1.

Tabel 1. Variabel Penelitian

Nama Variabel	Tipe Variabel	Keterangan
PM10	Float	Konsentrasi partikel dengan diameter 10 mikrometer atau kurang (PM10), diukur dalam mikrogram per meter kubik ($\mu g/m^3$).
SO2	Float	Konsentrasi sulfur dioksida (SO2), diukur dalam bagian per juta (ppm).
CO	Float	Konsentrasi karbon monoksida (CO), diukur dalam bagian per juta (ppm)
O3	Float	Konsentrasi ozon (O3), diukur dalam bagian per juta (ppm).
NO2	Float	Konsentrasi nitrogen dioksida (NO2), diukur dalam bagian per juta (ppm).
category_encoded	Integer	Kategori kualitas udara dengan keterangan kategori sebagai berikut: ● 0=Baik ● 1=Sedang

		• 2=Tidak sehat • 3=Sangat tidak sehat • 4=Berbahaya.

BAB IV

HASIL DAN PEMBAHASAN

4.1. Pra-Pemrosesan Data

4.1.1. Analisis Sebaran dan Karakteristik Data

Dataset memiliki total entri 4625 baris dengan rentang indeks 0 hingga 4624 dan total kolom 6, mayoritas variabel menggunakan tipe data float hanya terdapat variabel dengan tipe data integer yaitu 'categori_encoded'. Perbandingan komparatif rata-rata variabel dari yang memiliki tingkat risiko rendah ke tinggi adalah NO2, SO2, CO, PM10, dan 03.

`pm10`: 4466 non-null (159 hilang), float64
`so2`: 4607 non-null (18 hilang), float64
`co`: 4618 non-null (7 hilang), float64
`o3`: 4621 non-null (4 hilang), float64
`no2`: 4618 non-null (7 hilang), float64
`categori_encoded`: 4625 non-null (tidak ada yang hilang), int64

Kualitas udara seringkali fokus pada PM10 karena partikel ini memiliki ukuran yang cukup kecil (kurang dari 10 mikron) dan dapat menembus saluran pernapasan manusia, sehingga berpotensi menimbulkan dampak kesehatan yang signifikan.PM10 dapat berasal dari berbagai sumber, termasuk emisi kendaraan, pembakaran bahan bakar, dan debu, yang membuatnya relevan untuk dipantau dalam konteks polusi udara. Analisis statistik juga menunjukkan bahwa nilai rata-rata (mean) yang tinggi pada kolom pm10 mencerminkan tingkat partikel pm10 yang umumnya tinggi, sementara nilai standar deviasi yang juga tinggi menunjukkan adanya variasi signifikan dalam pengukuran pm10, yang mengindikasikan bahwa terdapat pengukuran yang jauh lebih tinggi atau lebih rendah dari rata-rata, sehingga informasi ini sangat penting untuk memahami kualitas udara di area yang diukur.

4.1.2. Penanganan *Missing Value* dengan Metode Mahalanobis

Metode Mahalanobis merupakan pendekatan yang efektif untuk menangani Keberadaan nilai yang hilang (missing value), terutama dalam konteks data multivariat. Metode ini menggunakan jarak Mahalanobis untuk mengidentifikasi outlier dan mengestimasi nilai

yang hilang berdasarkan hubungan antar variabel, sehingga dapat mempertahankan struktur korelasi dalam dataset.

Jarak Mahalanobis dihitung dengan rumus:

$$[D^2 = (X - \mu)T S^{-1} (X - \mu)]$$

Dimana (D^2) adalah jarak Mahalanobis kuadrat, (X) adalah vektor data, (μ) adalah vektor rata-rata, dan (S) adalah matriks kovarians. Dengan metode ini, peneliti dapat memperkirakan nilai yang hilang dengan mempertimbangkan pola dalam data, terutama ketika data yang hilang tidak bersifat acak.

Proses penerapan metode ini dimulai dengan menghitung matriks kovarians dari data yang tersedia. Estimasi nilai yang hilang dilakukan melalui model regresi, yang dapat dinyatakan sebagai:

$$[Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta X + \epsilon]$$

Dimana (Y) adalah variabel dependen (nilai yang hilang) dan (X_1, X_2, \ldots, X) adalah variabel independen. Pendekatan ini menghasilkan estimasi yang lebih representatif dan dapat mendeteksi outlier yang berpotensi mempengaruhi akurasi

Metode Mahalanobis efektif dalam menangani missing value dengan mempertimbangkan hubungan antar variabel. Imputasi yang baik akan meningkatkan kualitas analisis data.

4.2. Evaluasi Kinerja Model Random Forest

4.2.1. Hasil Cross-Validation

Random Forest Cross-Validation Scores: [0.96756757 0.96756757 0.96756757 0.95945946 0.97027027]
Mean CV Score: 0.9664864864864866

Berdasarkan hasil dari prosedur cross-validation menunjukkan bahwa model Random Forest yang diterapkan pada dataset ini mencapai akurasi rata-rata sebesar 96,65%, dengan skor yang konsisten di sekitar 96% pada setiap fold, yaitu [0.96756757, 0.96756757, 0.96756757, 0.95945946, 0.97027027]. Nilai-nilai ini menjelaskan bahwa model tidak hanya memiliki kemampuan prediktif yang tinggi, tetapi juga menunjukkan stabilitas yang baik, yang berarti bahwa model tersebut tidak terlalu sensitif terhadap variasi dalam data train. Dengan demikian, hasil ini menunjukkan bahwa model Random Forest memiliki potensi yang kuat untuk melakukan generalisasi dengan baik pada data baru yang tidak terlihat, sehingga memberikan keyakinan bahwa model ini dapat diandalkan dalam aplikasi praktis di masa depan. Penelitian selanjutnya dapat mempertimbangkan untuk

melakukan analisis kesalahan dan tuning hyperparameter guna mengeksplorasi kemungkinan peningkatan performa lebih lanjut.

4.2.2. Analisis Metrik Performa Model

===== Random Forest Metrics =====

Accuracy: 0.9751351351351352

Precision: 0.975134323328498

Recall: 0.9751351351351352

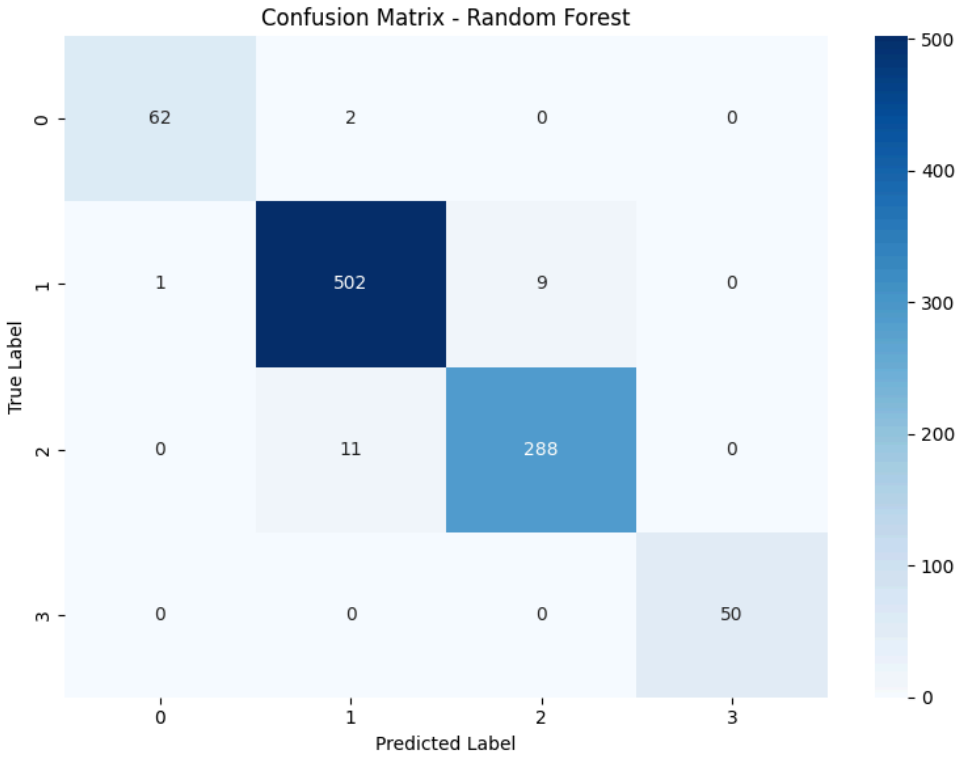
F1 Score: 0.9751224053174996

Hasil dari metrik performa model dengan menggunakan algoritma Random Forest menunjukkan hasil yang sangat memuaskan. Tingkat akurasi yang diperoleh adalah 97,51%, yang mencerminkan kemampuan model dalam mengklasifikasikan data dengan tingkat kesalahan yang minimal. Angka akurasi yang tinggi ini menandakan bahwa model ini efektif dalam memberikan prediksi yang tepat pada sebagian besar data yang diuji.

Metrik presisi yang dicapai sebesar 97,51% menunjukkan bahwa proporsi prediksi positif yang dihasilkan oleh model sebagian besar adalah benar-benar positif. Hal ini mengindikasikan bahwa model tidak hanya akurat dalam klasifikasi, tetapi juga memiliki kemampuan yang baik dalam mengurangi jumlah false positive. Dengan demikian, presisi yang tinggi ini sangat penting, terutama dalam konteks di mana kesalahan dalam klasifikasi positif dapat memiliki dampak yang signifikan.

Selain itu, nilai recall yang juga mencapai 97,51% menunjukkan bahwa model mampu mengidentifikasi sebagian besar kasus positif yang terdapat dalam data. F1 Score, yang merupakan kombinasi antara presisi dan recall, juga menunjukkan nilai yang tinggi, yakni 97,51%. Hasil ini menegaskan bahwa model Random Forest yang digunakan memiliki keseimbangan yang baik antara sensitivitas dan spesifisitas. Secara keseluruhan, hasil analisis ini menunjukkan bahwa model ini memiliki kinerja yang sangat baik dan dapat diandalkan untuk aplikasi yang memerlukan akurasi tinggi dalam klasifikasi.

4.2.3. Interpretasi Matriks Konfusi

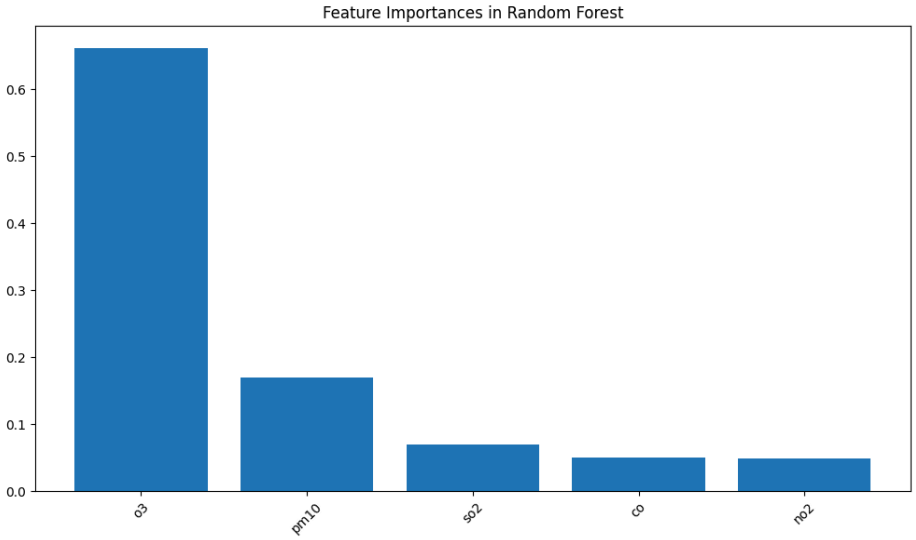


Gambar 4.2.3 Matriks Konfusi Algoritma Random Forest

Berdasarkan hasil analisis Confusion Matrix, dapat disimpulkan bahwa model Random Forest secara keseluruhan menunjukkan kinerja yang baik. Model mampu mengklasifikasikan sebagian besar sampel dengan benar, terutama pada kelas 1. Namun, terdapat beberapa kasus kesalahan klasifikasi, terutama antara kelas 1 dan kelas 2. Hal ini mengindikasikan adanya overlap pada fitur-fitur kedua kelas tersebut. Selain itu, kinerja model pada kelas minoritas (kelas 3) perlu diperhatikan lebih lanjut karena jumlah sampel yang terbatas dapat mempengaruhi generalisasi hasil. Hasil analisis ini memiliki beberapa implikasi penting. Pertama, model Random Forest dapat menjadi pilihan yang baik untuk permasalahan klasifikasi yang serupa. Kedua, untuk meningkatkan kinerja model, perlu dilakukan beberapa perbaikan seperti penyeimbangan data, penyesuaian hyperparameter, dan eksplorasi fitur yang lebih mendalam. Selain itu, membandingkan kinerja model dengan algoritma klasifikasi lain juga dapat memberikan wawasan tambahan.

Secara keseluruhan, Confusion Matriks merupakan alat yang sangat berguna untuk mengevaluasi kinerja model klasifikasi. Melalui analisis yang cermat, kita dapat memperoleh pemahaman yang komprehensif mengenai kekuatan dan kelemahan model, serta mengidentifikasi area-area yang perlu ditingkatkan.

4.2.4. Analisis *Feature Importance*



Gambar 4.2.4 Visualisasi Bar Chart mengenai Fitur Penting

Analisis fitur penting dalam model Random Forest ini menunjukkan bahwa konsentrasi ozon (O3) memiliki pengaruh yang sangat dominan terhadap hasil prediksi. Artinya, perubahan pada kadar ozon akan sangat signifikan mengubah hasil yang diberikan oleh model. Hal ini mengindikasikan bahwa ozon merupakan faktor kunci yang perlu diperhatikan dalam memahami fenomena yang sedang dipelajari.

Selain ozon, beberapa faktor lain seperti PM10, SO2, CO, dan NO2 juga memberikan kontribusi terhadap prediksi, namun dengan tingkat pengaruh yang lebih kecil. Urutan pentingnya fitur-fitur ini dapat dilihat dari tinggi rendahnya batang pada grafik.

4.3. Evaluasi Kinerja Model Gradient Boosting

4.3.1. Hasil Cross-Validation

Gradient Boosting Cross-Validation Scores: [0.96756757 0.97162162 0.97162162 0.97297297 0.97162162]

Mean CV Score: 0.971081081081081

Hasil dari prosedur cross-validation menunjukkan bahwa model Gradient Boosting yang diterapkan pada dataset ini mencapai akurasi rata-rata sebesar 97,11%, dengan skor yang konsisten di sekitar 97% pada setiap fold, yaitu [0.96756757 0.97162162 0.97162162 0.97297297 0.97162162]. Nilai-nilai ini menjelaskan bahwa model tidak hanya memiliki kemampuan prediktif yang tinggi, tetapi juga menunjukkan stabilitas yang baik, yang berarti bahwa model tersebut tidak terlalu sensitif terhadap variasi dalam data train. Dengan demikian, hasil ini menunjukkan bahwa model Gradien Boosting memiliki potensi yang kuat untuk melakukan generalisasi dengan baik

pada data baru yang tidak terlihat, sehingga memberikan keyakinan bahwa model ini dapat diandalkan dalam aplikasi praktis di masa depan. Penelitian selanjutnya dapat mempertimbangkan untuk melakukan analisis kesalahan dan tuning hyperparameter guna mengeksplorasi kemungkinan peningkatan performa lebih lanjut.

4.3.2. Analisis Metrik Performa Model

===== Gradient Boosting Metrics =====

Accuracy: 0.9686486486486486

Precision: 0.9686996450393901

Recall: 0.9686486486486486

F1 Score: 0.9686620064339219

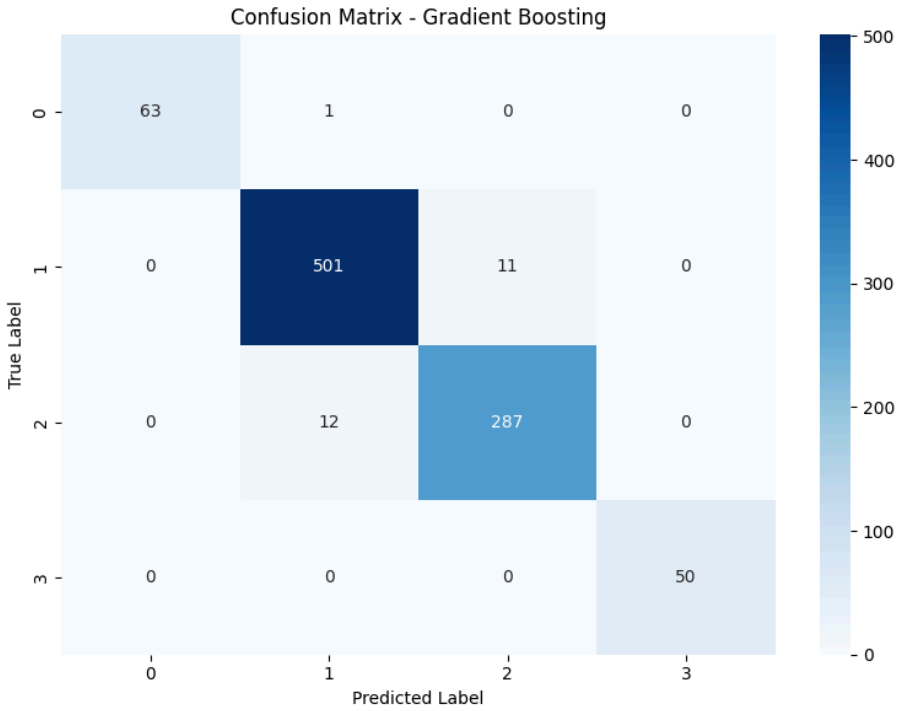
Analisis metrik performa model dengan menggunakan algoritma Gradient Boosting menunjukkan hasil yang memuaskan. Tingkat akurasi yang diperoleh adalah 96,86%, yang mencerminkan kemampuan model dalam mengklasifikasikan data dengan tingkat kesalahan yang rendah. Akurasi ini menandakan bahwa model memiliki performa yang baik dalam memberikan prediksi yang tepat pada sebagian data yang di uji

Presisi sebesar 96,87% menunjukkan bahwa sebagian besar prediksi positif yang dihasilkan oleh model adalah benar-benar positif. Hal ini mengindikasikan bahwa model tidak hanya akurat, tetapi juga efektif dalam mengurangi jumlah false positive. Presisi yang tinggi sangat penting, terutama dalam situasi di mana kesalahan dalam klasifikasi positif dapat membawa dampak yang signifikan.

Recall yang dicapai yaitu 96,86%, mengindikasikan kemampuan model untuk mendeteksi hampir semua kasus positif dalam data. Nilai F1 Score sebesar 96,87%, yang merupakan rata-rata harmonis antara presisi dan recall, menunjukkan bahwa model ini mampu menjaga keseimbangan yang baik antara sensitivitas dan spesifisitas.

Secara keseluruhan, hasil analisis menunjukkan bahwa algoritma Gradient Boosting memberikan kinerja yang sangat baik. Model ini memiliki akurasi tinggi, presisi yang kuat, serta kemampuan deteksi yang baik, sehingga dapat diandalkan dalam aplikasi yang membutuhkan performa klasifikasi yang konsisten dan tepat.

4.3.3. Interpretasi Metrik Konfusi



Gambar 4.3.3 Matriks Konfusi Algoritma Gradient Boosting

Berdasarkan hasil analisis Confusion Matrix, dapat disimpulkan bahwa model Gradient Boosting secara keseluruhan menunjukkan kinerja yang baik. Model mampu mengklasifikasikan sebagian besar sampel dengan benar, terutama pada kelas 1. Namun, terdapat beberapa kasus kesalahan klasifikasi pada kelas 1 dan 2.

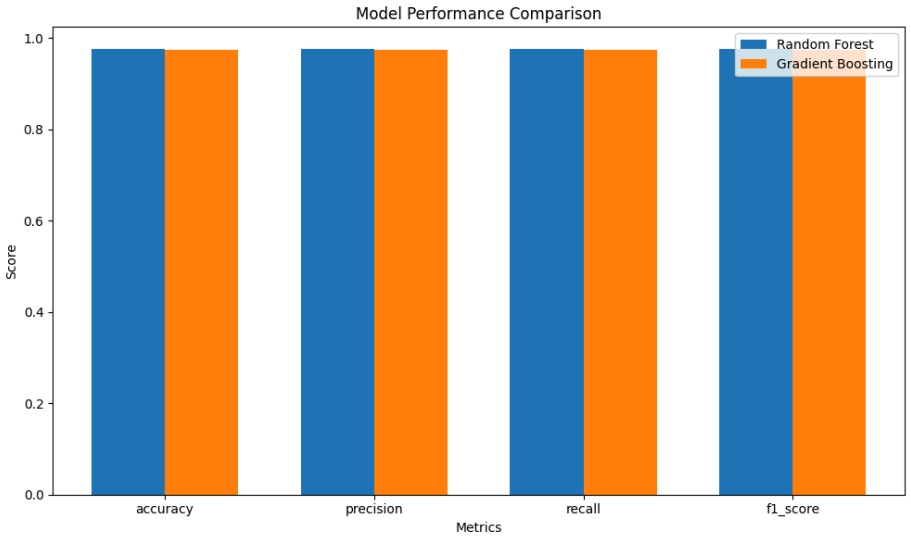
4.4. Perbandingan Kinerja Algoritma

4.4.1. Komparasi Akurasi, Presisi, Recall, dan F1-Score

Untuk mengetahui komparasi performa antara kedua model, dibutuhkan luaran sebuah laporan klasifikasi atau classification report. Classification report yang dihasilkan pada penelitian ini pada masing-masing model memiliki luaran performa masing-masing pemodelan. Dari luaran ini dapat diambil kesimpulan bahwa performa dari pemodelan Random Forest memiliki performa yang sedikit lebih baik daripada performa yang dimiliki oleh pemodelan Gradient Boost. Hal ini dapat dibuktikan oleh luaran laporan masing-masing pemodelan. Untuk pemodelan Random Forest memiliki luaran classification report ['accuracy': 0.9751351351351352, 'precision': 0.975134323328498, 'recall': 0.9751351351351352, 'f1_score': 0.9751224053174996], sedangkan untuk pemodelan Gradient Boost memiliki luaran classification report ['accuracy': 0.9740540540540541, 'precision': 0.9740688334117943, 'recall': 0.9740540540540541, 'f1_score': 0.9740542614307212].

4.4.2. Visualisasi Perbandingan Kinerja Model

Untuk mengetahui perbandingan kinerja model menggunakan visualisasi, pada penelitian ini digunakan visualisasi bar chart untuk mengetahui hal tersebut. Visualisasi bar chart dapat dilihat pada Gambar 4.4.2.



Gambar 4.4.2 Visualisasi Bar Chart Perbandingan Kinerja Pemodelan

BAB V

KESIMPULAN

Berdasarkan perbandingan yang dilakukan pada penelitian prediksi indeks kualitas udara dengan menggunakan metode random forest dan gradient boosting, kedua metode yang digunakan cukup bagus dalam melakukan klasifikasi dengan hasil tingkat akurasi 0.96-0.97 atau 96-97%. Dengan hasil akhir yang dilihat pada proses perhitungan confusion matrix mencakup nilai Precision, Recall dan FI-score yang menghasilkan nilai accuracy pada metode random forest lebih tinggi dibandingkan gradient boosting. Perbandingan kedua metode dapat dilihat pada hasil penelitian algoritma random forest memiliki nilai akurasi sebesar 0.975 atau 97.5% dibandingkan dengan algoritma gradient boosting memiliki nilai akurasi lebih rendah yaitu 0.974 atau 97.4%, sehingga pada penelitian ini dapat disimpulkan bahwa algoritma random forest memiliki kekuatan klasifikasi dan prediksi lebih baik daripada algoritma gradient boosting pada indeks kualitas udara Jakarta tahun 2010 sampai 2023.

DAFTAR PUSTAKA

Qonitan, Fatimah Dinan, Fikri Abdurrahman Haidar, and Nurulbaiti L. Zahra. "Overview of the air pollution standard index and associated health risk in DKI Jakarta during the 2019 dry season." *ICONIC-RS 2022: Proceedings of the 1st International Conference on Contemporary Risk Studies, ICONIC-RS 2022*. Vol. 31. 2022.

Sumantri, H. Arif, and M. Kes SKM. *Kesehatan Lingkungan-Edisi Revisi*. Prenada Media, 2017.

Sodiq, M. *Perbandingan Metode Naive Bayes Dan K-Nearest Neighbor Pada Klasifikasi Kualitas Udara Di Dki Jakarta*. Diss. University of Technology Yogyakarta, 2020.

Wardhana, Indrawata, et al. "Gradient Boosting Machine, Random Forest dan Light GBM untuk Klasifikasi Kacang Kering." *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)* 6.1 (2022): 92-99.

Patasik, Eva Sapan, and Sri Yulianto. "Classification of Regional Languages Using Methods Gradient Boots and Random Forest." *Jurnal Teknik Informatika (JUTIF)* 4.5 (2023): 1249-1255.

Safitri, Lina. "Pengendalian Pencemaran Udara."

Hermawan, Asep, Miko Hananto, and Doni Lasut. "Peningkatan Indeks Standar Pencemaran Udara (ISPU) dan kejadian gangguan saluran pernapasan di kota Pekanbaru." *Indonesian Journal of Health Ecology* 15.2 (2016): 76-86.

Hermawan, Iwan, and M. Pd. *Metodologi penelitian pendidikan (kualitatif, kuantitatif dan mixed method)*. Hidayatul Quran, 2019.

Patasik, Eva Sapan, and Sri Yulianto. "Classification of Regional Languages Using Methods Gradient Boots and Random Forest." *Jurnal Teknik Informatika (JUTIF)* 4.5 (2023): 1249-1255.

Siregar, Amril Mutoi. "Klasifikasi Untuk Prediksi Cuaca Menggunakan Esemble Learning." *Petir* 13.2 (2020): 522607.

Greenstone, Michael, and Qing Claire Fan. "Kualitas udara Indonesia yang memburuk dan dampaknya terhadap harapan hidup." *Chicago: Energy Policy Institute At The University of Chicago* (2019).

Agista, Putri Imas, Ninin Gusdini, and Maya Dewi Dyah Maharani. "Analisis Kualitas Udara Dengan Indeks Standar Pencemar Udara (Ispu) Dan Sebaran Kadar Polutannya Di Provinsi Dki Jakarta." *Sustainable Environmental and Optimizing Industry Journal* 2.2 (2020): 39-57.

Lampiran

```
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import chi2
from scipy.spatial import distance

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix,
    classification_report
)
```

Dataset dan Analisis Deskriptif

```
data = pd.read_csv('air_data.csv')

data['categori_encoded'] = data['categori'].map({
    'BAIK': 0,
    'SEDANG': 1,
    'TIDAK SEHAT': 2,
    'SANGAT TIDAK SEHAT': 3,
    'BERBAHAYA': 4
})

data = data.drop(columns=["tanggal", "stasiun", "max", "critical", "pm25",
"categori"])

data
```

	pm10	so2	co	o3	no2	categori_encoded
0	60.0	4.0	73.0	27.0	14.0	1
1	32.0	2.0	16.0	33.0	9.0	0
2	27.0	2.0	19.0	20.0	9.0	0
3	22.0	2.0	16.0	15.0	6.0	0
4	25.0	2.0	17.0	15.0	8.0	0
...	...	...	...	...	...	...
4620	55.0	43.0	15.0	15.0	25.0	1
4621	54.0	56.0	13.0	27.0	16.0	1
4622	62.0	45.0	15.0	29.0	34.0	1
4623	71.0	30.0	19.0	22.0	14.0	2

```
4624  38.0  43.0  12.0  34.0  34.0                1

[4625 rows x 6 columns]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4625 entries, 0 to 4624
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   pm10              4466 non-null   float64
 1   so2               4607 non-null   float64
 2   co                4618 non-null   float64
 3   o3                4621 non-null   float64
 4   no2               4618 non-null   float64
 5   categori_encoded  4625 non-null   int64  
dtypes: float64(5), int64(1)
memory usage: 216.9 KB
```

```
data.describe()
```

```
              pm10          so2           co           o3          no2  \
count  4466.000000  4607.000000  4618.000000  4621.000000  4618.000000   
mean     65.086879    28.183851    30.263534    86.586670    18.120832   
std      19.243150    13.822107    13.640344    50.295879     8.451605   
min       0.000000     0.000000     0.000000     0.000000     0.000000   
25%      54.000000    18.000000    21.000000    51.000000    13.000000   
50%      64.000000    27.000000    29.000000    78.000000    17.000000   
75%      75.000000    34.000000    37.000000   110.000000    21.000000   
max     179.000000   126.000000   134.000000   314.000000   134.000000   

       categori_encoded  
count       4625.000000  
mean           1.366486  
std            0.664830  
min            0.000000  
25%            1.000000  
50%            1.000000  
75%            2.000000  
max            4.000000  
```

```
print("Missing Values:")
print(data.isnull().sum())
```

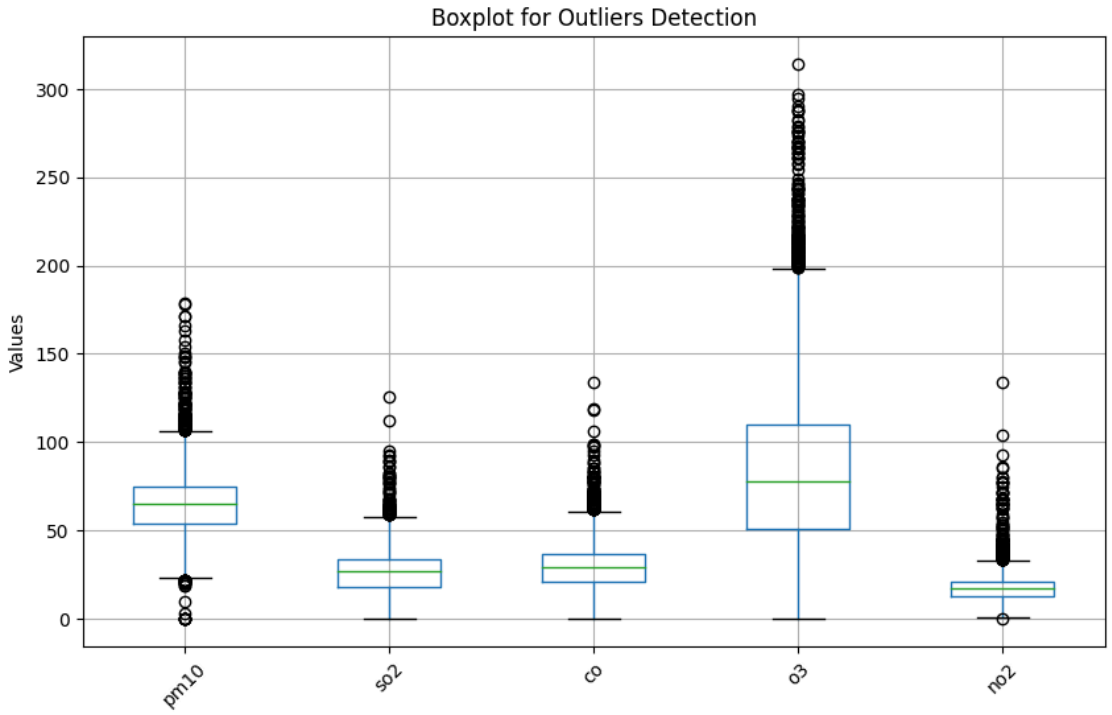
```
Missing Values:
pm10                159
so2                  18
co                    7
o3                    4
no2                   7
categori_encoded      0
dtype: int64
```

```
def mahalanobis_impute(data):
    mean_values = data.mean()
    cov_matrix = data.cov()
    inv_cov_matrix = np.linalg.inv(cov_matrix)

    for column in data.columns:
        if data[column].isnull().any():
            null_indices = data[data[column].isnull()].index

            for idx in null_indices:
                diff = data.loc[idx] - mean_values
                mahalanobis_dist = distance.mahalanobis(diff,
np.zeros(len(mean_values)), inv_cov_matrix)
                data.loc[idx, column] = mean_values[column]

    return data

def preprocess_data(data):
    data = mahalanobis_impute(data)

    X = data.drop(columns=['categori_encoded'])
    y = data['categori_encoded']

    return X, y, X.columns

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

Deteksi Outliers

```
X_df = pd.DataFrame(X, columns=feature_names)

numerical_columns = X_df.select_dtypes(include=[np.number]).columns
outlier_info = {}

for col in numerical_columns:
    Q1 = X_df[col].quantile(0.25)  # Kuartil pertama
    Q3 = X_df[col].quantile(0.75)  # Kuartil ketiga
    IQR = Q3 - Q1                  # Rentang antar-kuartil

    lower_bound = Q1 - 1.5 * IQR   # Batas bawah
    upper_bound = Q3 + 1.5 * IQR   # Batas atas

    # Outlier
    outliers = X_df[(X_df[col] < lower_bound) | (X_df[col] > upper_bound)]
    outlier_info[col] = {
        'lower_bound': lower_bound,
        'upper_bound': upper_bound,
        'num_outliers': len(outliers)
    }

plt.figure(figsize=(10, 6))
X_df[numerical_columns].boxplot(vert=True)
```

```
plt.title("Boxplot for Outliers Detection")
plt.ylabel("Values")
plt.xticks(rotation=45)
plt.show()

print("Outlier Information:")
for col, info in outlier_info.items():
    print(f"{col}: {info['num_outliers']} outliers (Bounds:
{info['lower_bound']} - {info['upper_bound']})")
```




```
Outlier Information:
pm10: 166 outliers (Bounds: 22.5 - 106.5)
so2: 77 outliers (Bounds: -6.0 - 58.0)
co: 129 outliers (Bounds: -3.0 - 61.0)
o3: 203 outliers (Bounds: -37.5 - 198.5)
no2: 190 outliers (Bounds: 1.0 - 33.0)
```

```
print("Distribusi kelas di y_train:")
print(y_train.value_counts())

print("Distribusi kelas di y_test:")
print(y_test.value_counts())
```

```
Distribusi kelas di y_train:
categori_encoded
1    2062
2    1269
0     216
3     152
4       1
Name: count, dtype: int64
Distribusi kelas di y_test:
```

```
categori_encoded
1    512
2    299
0     64
3     50
Name: count, dtype: int64
```

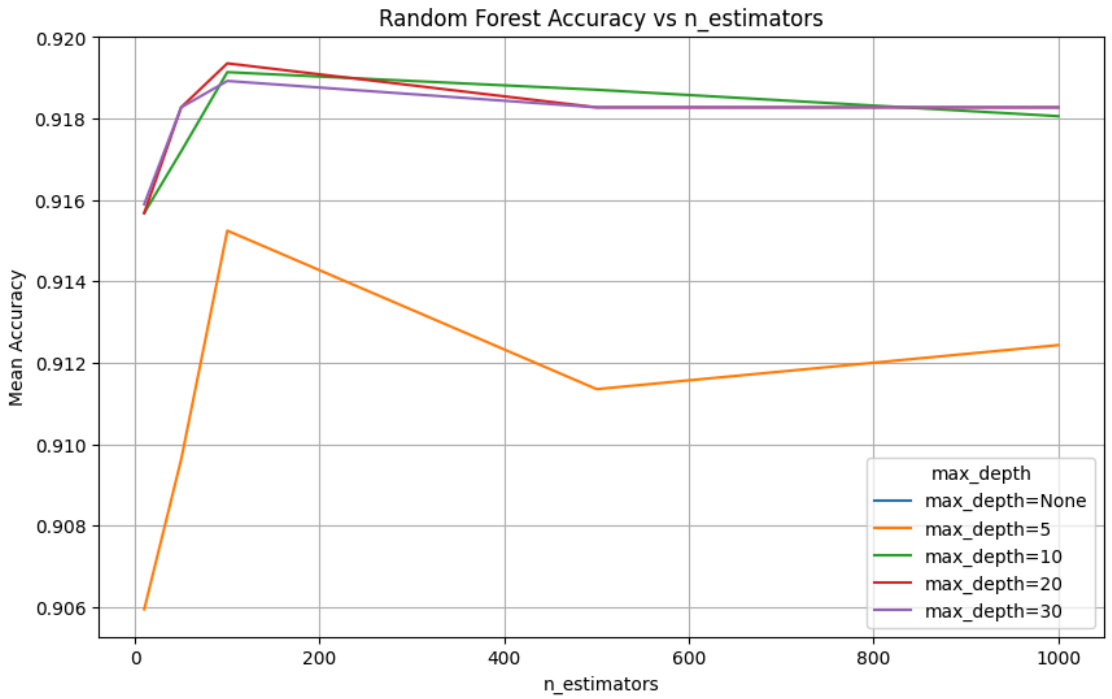
Pencarian Parameter Terbaik

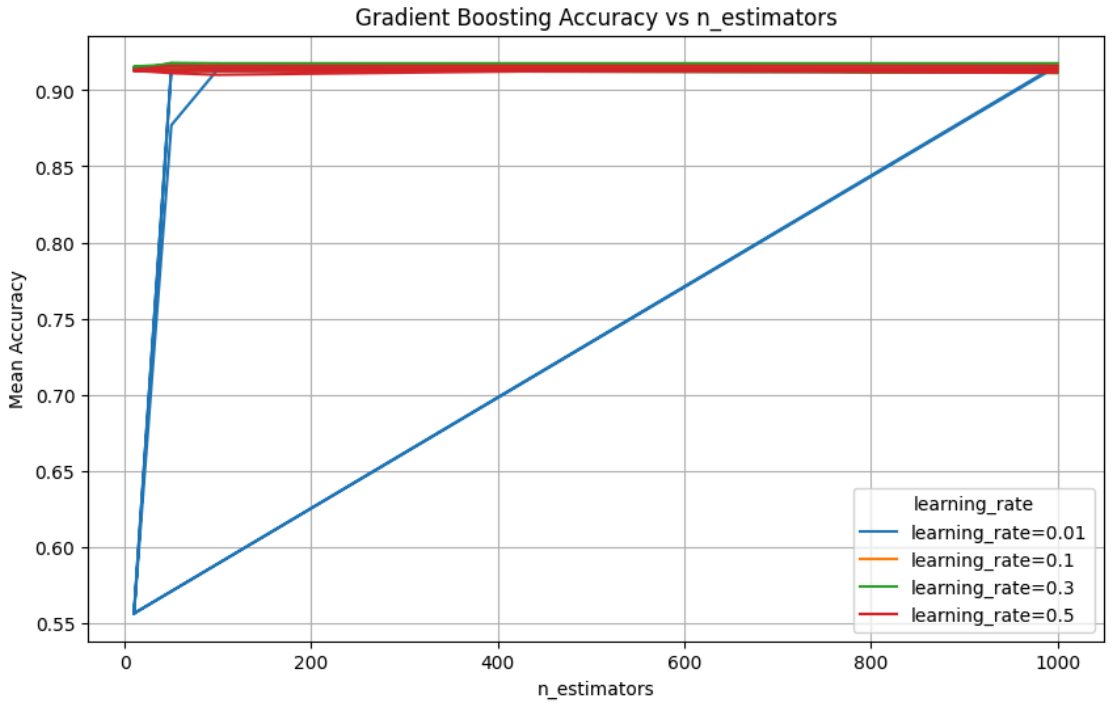
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier

rf_model = RandomForestClassifier(random_state=42)
gb_model = GradientBoostingClassifier(random_state=42)

rf_param_grid = {
    'n_estimators': [10, 50, 100, 500, 1000],
    'max_depth': [None, 5, 10, 20, 30]
}

gb_param_grid = {
    'n_estimators': [10, 50, 100, 500, 1000],
    'max_depth': [3, 5, 10],
    'learning_rate': [0.01, 0.1, 0.3, 0.5]
}

rf_grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=rf_param_grid,
    cv=5,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)

gb_grid_search = GridSearchCV(
    estimator=gb_model,
    param_grid=gb_param_grid,
    cv=5,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)

rf_grid_search.fit(X, y)
print("Best parameters for Random Forest:")
print(rf_grid_search.best_params_)
print(f"Best accuracy: {rf_grid_search.best_score_:.4f}")

gb_grid_search.fit(X, y)
print("\nBest parameters for Gradient Boosting:")
print(gb_grid_search.best_params_)
print(f"Best accuracy: {gb_grid_search.best_score_:.4f}")
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits

/home/arielsulton/.local/lib/python3.8/site-packages/sklearn/model_selecti
on/_split.py:737: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=5.
  warnings.warn(

Best parameters for Random Forest:
{'max_depth': 20, 'n_estimators': 100}
Best accuracy: 0.9194
Fitting 5 folds for each of 60 candidates, totalling 300 fits

/home/arielsulton/.local/lib/python3.8/site-packages/sklearn/model_selecti
on/_split.py:737: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=5.
  warnings.warn(


Best parameters for Gradient Boosting:
{'learning_rate': 0.3, 'max_depth': 10, 'n_estimators': 50}
Best accuracy: 0.9178
```

```
import matplotlib.pyplot as plt
import pandas as pd

# Random Forest: Ekstrak hasil tuning
rf_results = pd.DataFrame(rf_grid_search.cv_results_)

plt.figure(figsize=(10, 6))
for depth in rf_param_grid['max_depth']:
    subset = rf_results[rf_results['param_max_depth'] == depth]
    plt.plot(subset['param_n_estimators'], subset['mean_test_score'],
label=f"max_depth={depth}")

plt.title("Random Forest Accuracy vs n_estimators")
plt.xlabel("n_estimators")
plt.ylabel("Mean Accuracy")
plt.legend(title="max_depth")
plt.grid(True)
plt.show()

# Gradient Boosting: Ekstrak hasil tuning
gb_results = pd.DataFrame(gb_grid_search.cv_results_)

plt.figure(figsize=(10, 6))
for lr in gb_param_grid['learning_rate']:
    subset = gb_results[gb_results['param_learning_rate'] == lr]
    plt.plot(subset['param_n_estimators'], subset['mean_test_score'],
label=f"learning_rate={lr}")

plt.title("Gradient Boosting Accuracy vs n_estimators")
plt.xlabel("n_estimators")
plt.ylabel("Mean Accuracy")
plt.legend(title="learning_rate")
```

```
plt.grid(True)
plt.show()
```





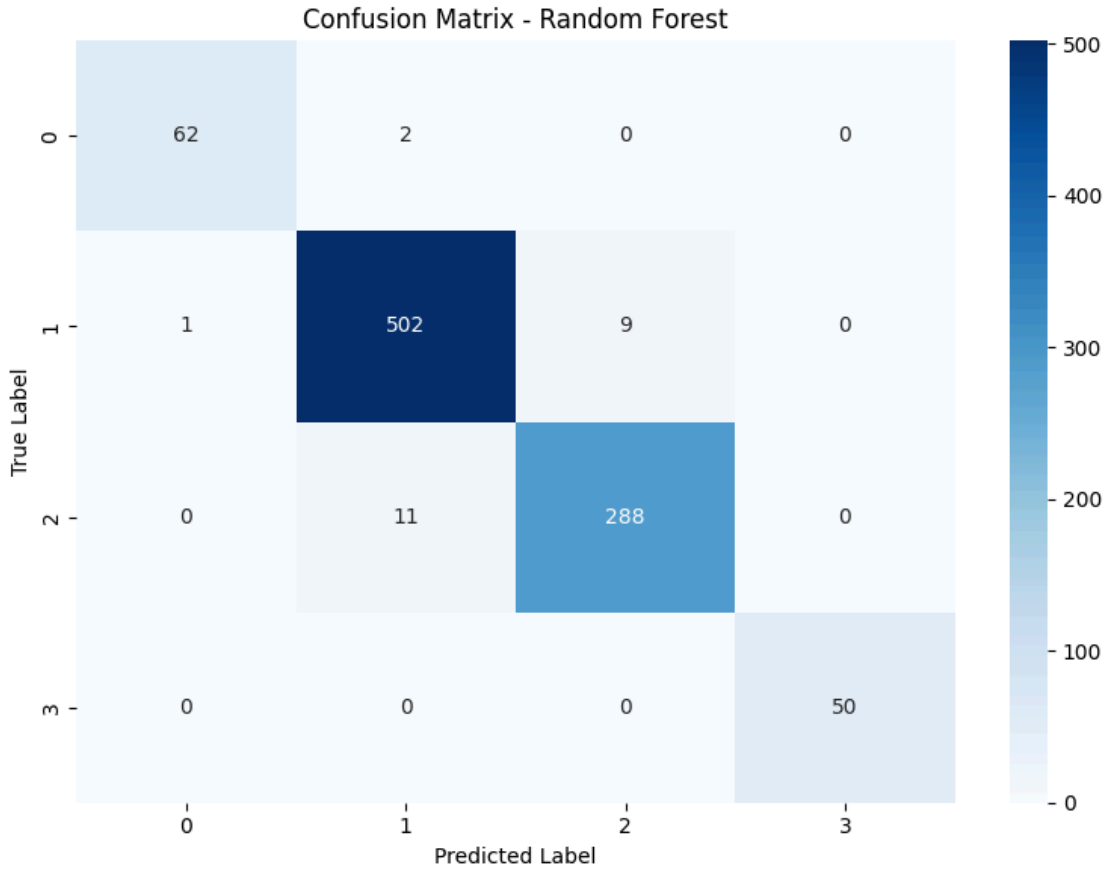
Pemodelan RF dan GB

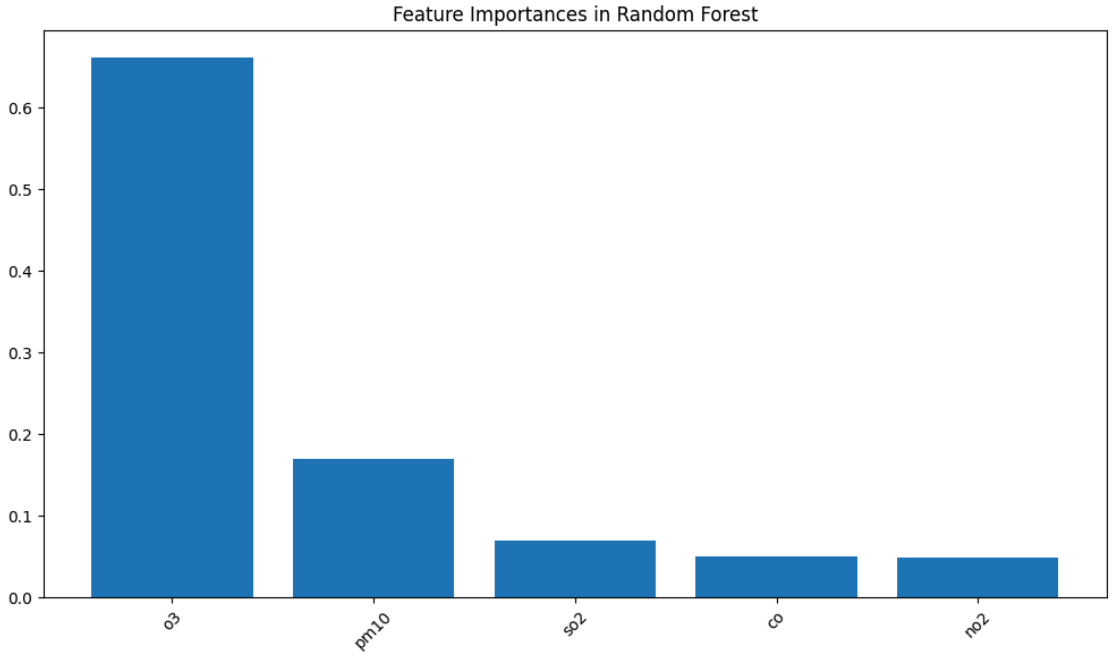
```
def evaluate_model(y_true, y_pred, model_name):
    print(f"===== {model_name} Metrics =====")
    print("Accuracy: ", accuracy_score(y_true, y_pred))
    print("Precision: ", precision_score(y_true, y_pred,
average='weighted'))
    print("Recall: ", recall_score(y_true, y_pred, average='weighted'))
```

```
    print("F1 Score: ", f1_score(y_true, y_pred, average='weighted'))

    plt.figure(figsize=(8, 6))
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix - {model_name}')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.tight_layout()
    plt.show()

    return {
        'accuracy': accuracy_score(y_true, y_pred),
        'precision': precision_score(y_true, y_pred, average='weighted'),
        'recall': recall_score(y_true, y_pred, average='weighted'),
        'f1_score': f1_score(y_true, y_pred, average='weighted')
    }

def random_forest_model():
    rf_classifier = RandomForestClassifier(
        n_estimators=100,
        random_state=42,
        max_depth=20,
        n_jobs=-1
    )

    cv_scores = cross_val_score(rf_classifier, X_train, y_train, cv=5)
    print("Random Forest Cross-Validation Scores:", cv_scores)
    print("Mean CV Score:", cv_scores.mean())

    rf_classifier.fit(X_train, y_train)
    y_pred_rf = rf_classifier.predict(X_test)

    rf_report = classification_report(y_test, y_pred_rf)
    rf_metrics = evaluate_model(y_test, y_pred_rf, "Random Forest")

    plt.figure(figsize=(10, 6))
    feature_imp = rf_classifier.feature_importances_
    indices = np.argsort(feature_imp)[::-1]

    plt.title("Feature Importances in Random Forest")
    plt.bar(range(len(feature_imp)), feature_imp[indices])
    plt.xticks(range(len(feature_imp)), [feature_names[i] for i in
indices], rotation=45)
    plt.tight_layout()
    plt.show()

    return rf_classifier, rf_report, rf_metrics

def gradient_boosting_model():
    gb_classifier = GradientBoostingClassifier(
        n_estimators=50,
        learning_rate=0.3,
        max_depth=10,
```

```
        random_state=42
    )

    cv_scores = cross_val_score(gb_classifier, X_train, y_train, cv=5)
    print("Gradient Boosting Cross-Validation Scores:", cv_scores)
    print("Mean CV Score:", cv_scores.mean())

    gb_classifier.fit(X_train, y_train)
    y_pred_gb = gb_classifier.predict(X_test)

    gb_report = classification_report(y_test, y_pred_gb)
    gb_metrics = evaluate_model(y_test, y_pred_gb, "Gradient Boosting")

    return gb_classifier, gb_report, gb_metrics

def compare_models(rf_metrics, gb_metrics):
    print("\n===== Model Comparison =====")
    metrics = ['accuracy', 'precision', 'recall', 'f1_score']

    plt.figure(figsize=(10, 6))
    x = np.arange(len(metrics))
    width = 0.35

    plt.bar(x - width/2, list(rf_metrics.values()), width, label='Random
Forest')
    plt.bar(x + width/2, list(gb_metrics.values()), width, label='Gradient
Boosting')

    plt.xlabel('Metrics')
    plt.ylabel('Score')
    plt.title('Model Performance Comparison')
    plt.xticks(x, metrics)
    plt.legend()
    plt.tight_layout()
    plt.show()

rf_model, rf_report, rf_metrics = random_forest_model()
```

```
/home/arielsulton/.local/lib/python3.8/site-packages/sklearn/model_selecti
on/_split.py:737: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=5.
  warnings.warn(

Random Forest Cross-Validation Scores: [0.96756757 0.96756757 0.96756757
0.95945946 0.97027027]
Mean CV Score: 0.9664864864864866
===== Random Forest Metrics =====
Accuracy:  0.9751351351351352
Precision:  0.975134323328498
Recall:  0.9751351351351352
F1 Score:  0.9751224053174996
```

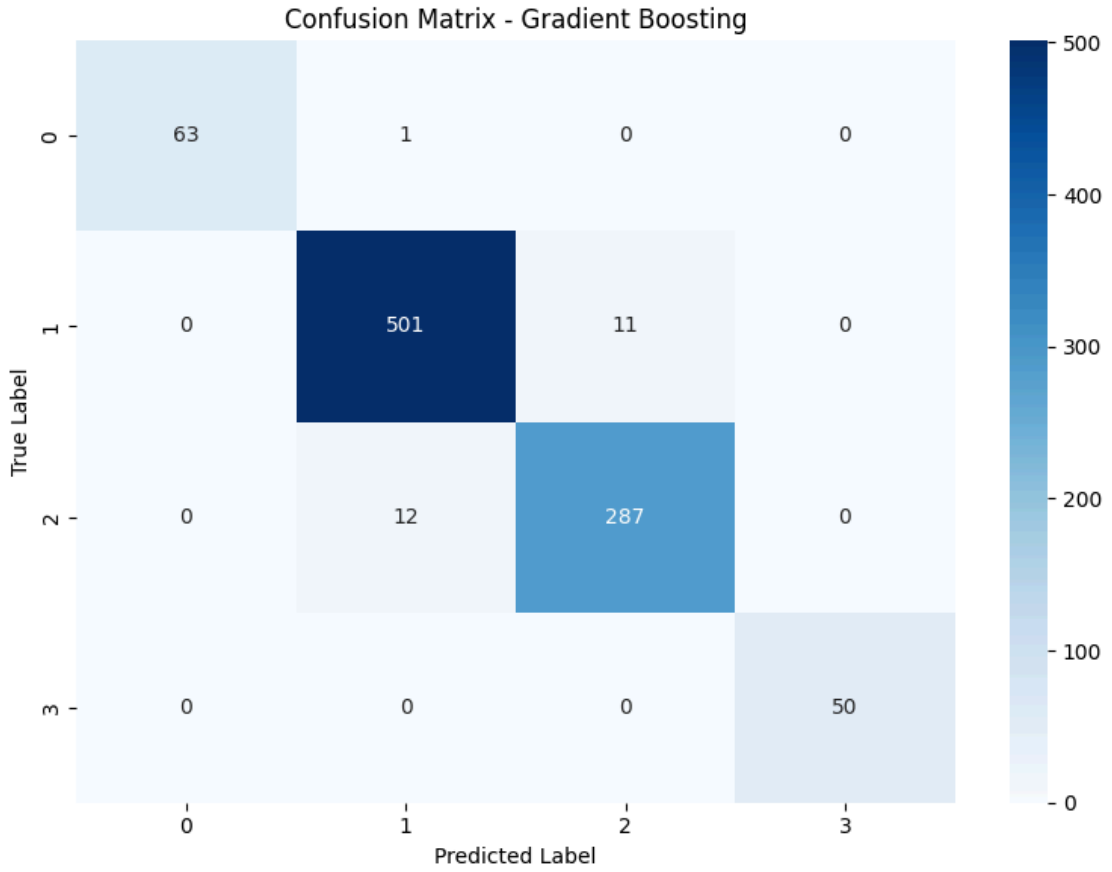





```
gb_model, gb_report, gb_metrics = gradient_boosting_model()
```

```
/home/arielsulton/.local/lib/python3.8/site-packages/sklearn/model_selecti
on/_split.py:737: UserWarning: The least populated class in y has only 1
members, which is less than n_splits=5.
  warnings.warn(
```

```
Gradient Boosting Cross-Validation Scores: [0.96756757 0.97162162
0.97162162 0.97297297 0.97162162]
Mean CV Score: 0.971081081081081
===== Gradient Boosting Metrics =====
Accuracy:  0.9740540540540541
Precision:  0.9740688334117943
Recall:  0.9740540540540541
F1 Score:  0.9740542614307212
```




```
print("Random Forest Classification Report:")
print(rf_report)
print(rf_metrics)
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98        64
           1       0.97      0.98      0.98       512
           2       0.97      0.96      0.97       299
           3       1.00      1.00      1.00        50

    accuracy                           0.98       925
   macro avg       0.98      0.98      0.98       925
weighted avg       0.98      0.98      0.98       925

{'accuracy': 0.9751351351351352, 'precision': 0.975134323328498, 'recall':
0.9751351351351352, 'f1_score': 0.9751224053174996}
```

```
print("Gradient Boosting Classification Report:")
print(gb_report)
print(gb_metrics)
```

```
Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        64
           1       0.97      0.98      0.98       512
           2       0.96      0.96      0.96       299
           3       1.00      1.00      1.00        50

    accuracy                           0.97       925
   macro avg       0.98      0.98      0.98       925
weighted avg       0.97      0.97      0.97       925

{'accuracy': 0.9740540540540541, 'precision': 0.9740688334117943,
'recall': 0.9740540540540541, 'f1_score': 0.9740542614307212}
```

```
compare_models(rf_metrics, gb_metrics)
```

```
===== Model Comparison =====
```