Prediksi Risiko Keterlambatan Pengiriman Menggunakan Graph Convolutional Network dengan Optimasi Parameter

*GITHUB: https://github.com/agungibr/supply-chain

Himam Bashiran
Fakultas Informatika
2311110055
Telkom University
Purwokerto, Indonesia
himamb@student.telkomuniversity.ac.id

Agung Malik Ibrahim
Fakultas Informatika
2311110067
Telkom University
Purwokerto, Indonesia
agungibr@student.telkomuniversity.ac.id

Fito Satrio
Fakultas Informatika
2311110030
Telkom University
Purwokerto, Indonesia
fitosatrio@student.telkomuniversity.ac.id

***Abstract*—Dalam penelitian ini, kami mengembangkan model prediksi risiko keterlambatan pengiriman sebagai langkah strategis untuk meningkatkan efisiensi rantai pasok. Model yang kami gunakan berbasis *Graph Convolutional Network* (GCN), yang dirancang untuk menangkap hubungan kompleks antar entitas dalam jaringan logistik. Model ini dirancang untuk memanfaatkan representasi graf yang sulit dimodelkan secara efektif dengan pendekatan *machine learning* tradisional. Selain itu, kami juga membandingkan performa GCN dengan beberapa algoritma *machine learning* lain, termasuk *Linear Discriminant Analysis* (LDA). Hasil pengujian menunjukkan bahwa GCN memberikan metrik evaluasi yang kompetitif, dengan *Accuracy* sebesar 95.23%, *precision* sebesar 95.06%, *recall* sebesar 96.31%, dan *F1-score* sebesar 95.68%. Meskipun LDA menunjukkan akurasi lebih tinggi (98%), GCN lebih dipilih karena mampu menangkap pola struktural.**

Dengan pendekatan ini, penelitian ini menawarkan solusi berbasis GCN yang lebih adaptif dan relevan untuk memprediksi risiko keterlambatan pengiriman dalam sistem logistik modern yang kompleks.

***Kata Kunci*—supply chain, keterlambatan pengiriman, graph convolutional network, machine learning**

I. Pendahuluan

Supply chain atau rantai pasok adalah serangkaian aktivitas yang mencakup seluruh proses dari hulu ke hilir dalam suatu sistem produksi dan distribusi. Proses ini melibatkan pengadaan bahan baku, pembelian, produksi, hingga distribusi produk akhir ke tangan konsumen. *Supply chain* tidak hanya mengatur pergerakan barang atau bahan, tetapi juga mencakup aliran informasi dan keuangan yang diperlukan untuk memastikan kelancaran operasional. Dengan manajemen yang baik, *supply chain* dapat meningkatkan efisiensi penggunaan sumber daya, mengoptimalkan biaya, serta memperkuat hubungan antara berbagai pihak yang terlibat dalam rantai pasokan, seperti pemasok, produsen, distributor, dan konsumen [1].

Pengelolaan rantai pasok yang efektif sangat penting dalam memastikan kelancaran aliran barang, informasi, dan keuangan dari hulu ke hilir. Salah satu tantangan utama dalam manajemen rantai pasok adalah risiko keterlambatan pengiriman,

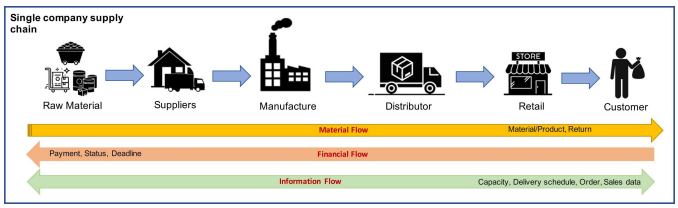

Fig. 1. Alur Proses Rantai Pasokan

yang dapat mempengaruhi kepuasan pelanggan dan efisiensi operasional. Keterlambatan pengiriman tidak hanya mempengaruhi kepuasan pelanggan, tetapi juga dapat menimbulkan berbagai dampak negatif lainnya dalam rantai pasok. Jika terjadi keterlambatan pengiriman suatu produk, ada kemungkinan gudang atau pusat distribusi akan kehabisan stok sehingga mengganggu kelancaran produksi dan distribusi [2]. Selain itu, keterlambatan pengiriman dapat menyebabkan peningkatan biaya operasional. Misalnya, jika bongkar muat kapal tertunda, ekspedisi akan dirugikan karena kapal yang bersandar di pelabuhan harus membayar sewa hariannya. Akibatnya, ekspedisi atau perusahaan pelayaran akan mengalami kerugian finansial yang signifikan [3]. Untuk mengatasi tantangan ini, kami mencoba untuk melakukan pendekatan prediktif menggunakan teknik *machine learning* yang diterapkan dalam memprediksi status pengiriman barang.

Kami menggunakan *dataset DataCo Smart Supply Chain for Big Data Analysis* yang tersedia di *Kaggle* dan *Mendeley Data*. Dataset ini mencakup data transaksi dalam rantai pasokan, mulai dari tahap pengadaan hingga distribusi akhir. Setiap transaksi mencakup berbagai variabel, seperti waktu pemesanan, waktu pengiriman, status pengiriman, lokasi gudang, serta informasi pelanggan. Selain itu, *dataset* ini juga mencatat atribut tambahan, seperti metode pengiriman, kategori produk, dan faktor-faktor lain yang dapat mempengaruhi keterlambatan pengiriman. Untuk mendukung analisis ini, kami menggunakan berbagai *tools* dalam *Python*, seperti

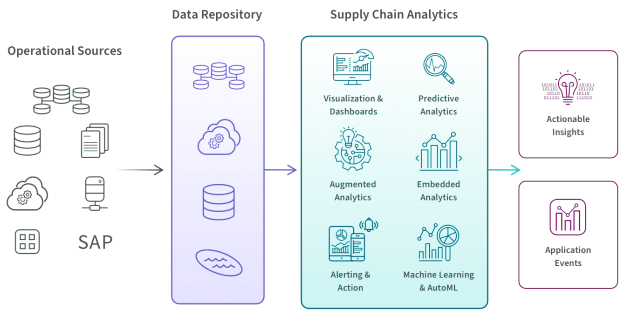

Fig. 2. *Supply Chain Analytic Architecture* [4]

Pandas dan *NumPy* untuk manipulasi data, *Matplotlib* dan *Seaborn* untuk visualisasi, serta *Scikit-learn* dan *torch* untuk pengembangan model prediktif. Dengan memanfaatkan *dataset* ini, kami akan menganalisis pola dalam rantai pasokan dan membangun model prediksi berbasis *machine learning* untuk mengidentifikasi potensi keterlambatan pengiriman.

Pada penelitian ini, kami mencoba untuk memprediksi keterlambatan pengiriman dalam rantai pasokan berdasarkan data transaksi dan logistik yang tersedia dalam *dataset DataCo Smart Supply Chain for Big Data Analysis*. Prediksi keterlambatan pengiriman dilakukan menggunakan model *machine learning* tradisional dengan mempertimbangkan faktor-faktor seperti waktu pemesanan, metode pengiriman, lokasi gudang, dan kategori produk. Apabila hasil prediksi menunjukkan potensi keterlambatan yang signifikan, data tersebut akan diproses lebih lanjut melalui model *deep learning* yang menggunakan arsitektur GNN untuk menganalisis pola keterlambatan dengan lebih akurat. Dengan memperoleh perkiraan waktu keterlambatan yang lebih presisi, kami dapat mengoptimalkan strategi mitigasi risiko dalam rantai pasokan, sehingga meningkatkan efisiensi operasional dan kepuasan pelanggan.

Dengan adanya sistem prediksi keterlambatan pengiriman ini, kami berharap dapat meminimalkan dampak negatif dalam rantai pasokan, seperti ketidakpuasan pelanggan, peningkatan biaya operasional, dan gangguan terhadap jadwal produksi. Dengan informasi yang lebih akurat mengenai kemungkinan keterlambatan, perusahaan dapat mengambil langkah mitigasi yang tepat, seperti mengoptimalkan rute pengiriman, menyesuaikan strategi persediaan, atau memberikan pemberitahuan dini kepada pelanggan. Hal ini diharapkan dapat meningkatkan efisiensi logistik serta memperkuat kepercayaan pelanggan terhadap layanan yang diberikan.

II. *EXPLORATORY DATA ANALYSIS*

A. *Penjelasan Sederhana DataCo Smart Supply Chain*

Untuk penelitian ini, kami menggunakan *dataset DataCo Smart Supply Chain* karena jumlah data yang besar serta informasi yang cukup detail terkait setiap transaksi dalam rantai pasok. Dataset ini terdiri dari 180.519 baris dan 53 kolom, dengan setiap baris mewakili satu transaksi pesanan. *Dataset* ini mencakup berbagai aspek dalam rantai pasok, seperti waktu pengiriman, informasi pelanggan, status pesanan, serta keuntungan per transaksi.

Setiap transaksi dalam *dataset* memiliki beberapa variabel utama, seperti *Days for shipping (real)* dan *Days for shipment (scheduled)*, yang menunjukkan perbedaan antara waktu pengiriman aktual dan yang telah dijadwalkan. Selain itu, *dataset* ini juga mencatat *Shipping Mode*, yang merepresentasikan metode pengiriman yang digunakan dalam setiap transaksi. Informasi pelanggan, seperti *Customer City*, *Customer Country*, dan *Customer Segment*, juga tersedia untuk menganalisis bagaimana faktor lokasi dan jenis pelanggan mempengaruhi keterlambatan pengiriman.

Selain informasi pengiriman dan pelanggan, *dataset* ini juga mencakup variabel terkait produk dan penjualan, seperti *Sales per customer*, *Order Item Quantity*, *Order Item Discount*, dan *Order Profit Per Order*, yang dapat digunakan untuk menganalisis dampak volume pesanan terhadap keterlambatan pengiriman. Dengan adanya informasi yang lengkap dan terstruktur ini, *dataset DataCo Smart Supply Chain* memungkinkan analisis mendalam terhadap berbagai faktor yang berkontribusi terhadap keterlambatan pengiriman dalam rantai pasok.

B. *Informasi DataCo Smart Supply Chain*

Dataset ini memiliki empat jenis tipe data utama, yaitu *Datetime64* (2 fitur), *Float64* (15 fitur), *Int64* (14 fitur), dan *Object* (22 fitur), dengan total 180.519 baris dan 53 kolom. Beberapa kolom memiliki *missing values*, seperti *Order Zipcode* yang hanya berisi 24.840 nilai, sedangkan *Product Description* kosong sepenuhnya sehingga mungkin dapat dihapus.

	NaN_count	NaN_percentage
Product Description	180519	1.000000
Order Zipcode	155679	0.862397
Customer Zipcode	3	0.000017

Fig. 3. *Missing Values*

Variabel-variabel numerik dalam dataset menunjukkan pola yang menarik. *Days for Shipping (real)* berkisar antara 0 hingga 6 hari, dengan rata-rata sekitar 3,5 hari. Sementara itu, sekitar 55% transaksi mengalami keterlambatan, yang ditandai oleh nilai *Late_delivery_risk*.

Terdapat beberapa hubungan menarik antar variabel. Korelasi antara *Days for Shipping (real)* dan *Late Delivery Risk* sebesar 0.40 menunjukkan bahwa semakin lama waktu pengiriman, semakin tinggi risiko keterlambatan. Hubungan antara *Sales per Customer* dan *Order Profit Per Order* memiliki korelasi 0.55 yang berarti peningkatan penjualan per pelanggan cenderung meningkatkan profit, meskipun faktor lain juga berpengaruh.

Selain itu, terdapat indikasi redundansi fitur dalam data ini. Data redudansi merupakan beberapa salinan dari informasi

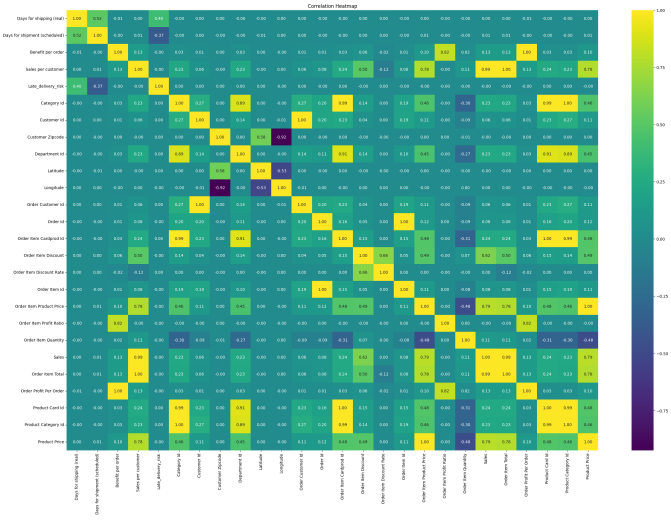

Fig. 4. *Heatmap*

yang sama disimpan di lebih dari satu tempat sekaligus [5]. Redundansi terjadi ketika dua atau lebih variabel memiliki korelasi yang sangat tinggi (1 atau mendekati -1), sehingga informasi yang diberikan oleh satu variabel dapat direpresentasikan oleh variabel lainnya dan terhindar dari multikolinearitas. Misalnya, *Benefit per order* dan *Order Profit per order* memiliki korelasi 1, yang menunjukkan bahwa salah satu dari kedua variabel ini dapat dihilangkan tanpa kehilangan banyak informasi.

	Benefit per order	Order Profit Per Order
0	91.250000	91.250000
1	-249.089996	-249.089996
2	-247.779999	-247.779999
3	22.860001	22.860001
4	134.210007	134.210007

Fig. 5. *Feature Redudancy*

Selanjutnya, deteksi pencilan (outlier). Outlier atau pencilan adalah pengamatan yang secara signifikan menyimpang dari mayoritas data dalam suatu *dataset* [6]. Untuk mendeteksi nilai outlier, terdapat berbagai metode yang bisa digunakan seperti IQR Filter (quantitative), Z-Score Filter (quantitative), atau Boxplot (qualitative). Kehadiran outlier dapat disebabkan oleh transaksi yang tidak umum, kesalahan pencatatan data, atau variasi yang ekstrem dalam proses bisnis. Rumus yang digunakan untuk menghitung *Outlier* atas dan bawah:

$$Q3 + (1.5 \times IQR) < \text{Outlier atas} \leq Q3 + (3 \times IQR) \quad (1)$$

$$Q1 - (1.5 \times IQR) > \text{Outlier bawah} \geq Q1 - (3 \times IQR) \quad (2)$$

Beberapa fitur dalam *dataset*, seperti *Customer Email*, *Customer Password*, dan *Product Image*, mengandung informasi yang tidak relevan untuk analisis rantai pasokan, sehingga perlu dihapus untuk menjaga fokus analisis. Selain itu, kolom seperti *Customer Fname*, *Customer Lname*, dan *Customer Street* memberikan detail yang terlalu spesifik dan tidak memberikan kontribusi signifikan terhadap prediksi keterlambatan pengiriman. Fitur seperti *Product Status* yang tidak menunjukkan variasi juga sebaiknya diabaikan karena tidak menambah nilai pada model analisis.

	Customer Email	Customer Password	Product Status	Customer Fname	Customer Lname	Customer Street	Product Image
0	XXXXXXXXX	XXXXXXXXX	0	Cally	Holloway	5365 Noble Nectar Island	http://images.acmesports.sports/Smart+watch
1	XXXXXXXXX	XXXXXXXXX	0	Irene	Luna	2679 Rustic Loop	http://images.acmesports.sports/Smart+watch
2	XXXXXXXXX	XXXXXXXXX	0	Gillian	Maldonado	8510 Round Bear Gate	http://images.acmesports.sports/Smart+watch
3	XXXXXXXXX	XXXXXXXXX	0	Tana	Tate	3200 Amber Bend	http://images.acmesports.sports/Smart+watch
4	XXXXXXXXX	XXXXXXXXX	0	Orli	Hendricks	8671 Iron Anchor Corners	http://images.acmesports.sports/Smart+watch
5	XXXXXXXXX	XXXXXXXXX	0	Kimberly	Flowers	2122 Hazy Corner	http://images.acmesports.sports/Smart+watch
6	XXXXXXXXX	XXXXXXXXX	0	Constance	Terrell	1879 Green Pine Bank	http://images.acmesports.sports/Smart+watch
7	XXXXXXXXX	XXXXXXXXX	0	Erica	Stevens	7595 Cotton Log Row	http://images.acmesports.sports/Smart+watch
8	XXXXXXXXX	XXXXXXXXX	0	Nichole	Olsen	2051 Dusty Route	http://images.acmesports.sports/Smart+watch
9	XXXXXXXXX	XXXXXXXXX	0	Oprah	Delacruz	9139 Blue Blossom Court	http://images.acmesports.sports/Smart+watch

Fig. 6. Fitur yang Mengganggu

III. *PRE-PROCESSING*

Data *pre-processing* adalah proses mengubah data mentah menjadi bentuk yang sesuai untuk analisis lebih lanjut. Langkah-langkah dalam *pre-processing* meliputi pembersihan data, penghilangan noise, dan pengorganisasian data agar sesuai dengan kebutuhan analisis. Proses ini penting untuk meningkatkan kualitas data, sehingga model *machine learning* dapat menghasilkan kinerja yang lebih baik [7].

A. *Membagi Data Latih dan Data Uji*

Dalam *machine learning*, pembagian *dataset* menjadi data latih dan data uji adalah langkah krusial untuk mengevaluasi kinerja model secara objektif. Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk menguji kemampuan model dalam memprediksi data yang belum pernah dilihat sebelumnya [8]. Salah satu teknik yang umum digunakan untuk memastikan evaluasi model yang lebih stabil adalah *k-fold cross-validation*. Dengan menggunakan *5-fold*, dataset dibagi menjadi lima lipatan yang masing-masing berperan sebagai data latih dan data uji secara bergantian.

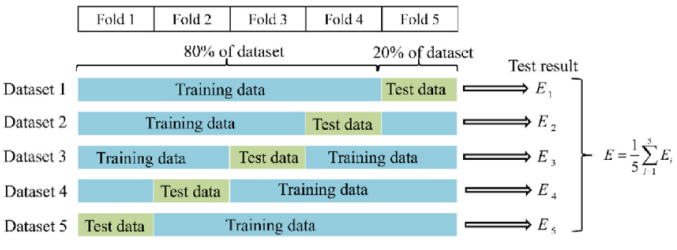

Fig. 7. *5-fold cross-validation*

Pada implementasi ini, digunakan nilai *random state* 42 untuk memastikan hasil yang konsisten dalam proses pembagian

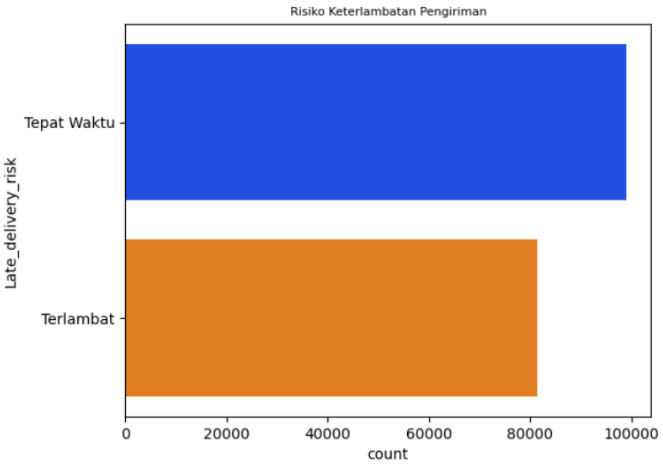

Fig. 8. *Target Variable*

dataset. *Random state* berfungsi sebagai penentu nilai awal dari generator angka acak yang digunakan untuk membagi data menjadi data latih dan data uji [9]. Dengan menentukan *random state*, pembagian *dataset* menjadi data latih dan data uji akan menghasilkan hasil yang sama setiap kali kode dijalankan, sehingga memungkinkan proses replikasi eksperimen dan evaluasi yang lebih dapat diandalkan.

TABLE I
CONTOH JUMLAH DATA PADA FOLD PERTAMA

Dataset	Shape
X_{train}	(162.467, 36)
y_{train}	(162.467, 1)
X_{test}	(18.052, 36)
y_{test}	(18.052, 1)

Selain itu, pembagian *dataset* menjadi data latih dan data uji dilakukan di awal proses analisis untuk mencegah terjadinya *data leakage*. *Data leakage* terjadi ketika informasi dari data uji, yang seharusnya tidak terlihat oleh model, secara tidak sengaja digunakan selama proses pelatihan [11]. Hal ini dapat menyebabkan model mendapatkan hasil evaluasi yang terlalu optimis karena secara tidak langsung memanfaatkan informasi yang tidak tersedia di situasi dunia nyata. Dengan memisahkan *dataset* sejak awal, risiko *data leakage* dapat diminimalkan, sehingga evaluasi kinerja model menjadi lebih akurat dan mencerminkan performa pada data baru yang belum pernah dilihat sebelumnya.

B. Memisahkan Fitur Kategorikal dengan Fitur Numerikal

Pemisahan antara fitur kategorikal dan numerikal merupakan langkah penting untuk memastikan bahwa setiap tipe data diproses sesuai dengan karakteristiknya. Data kategorikal biasanya terdiri dari variabel yang merepresentasikan kategorik, seperti *Customer Segment* atau *Order Status*, sementara data numerikal mencakup variabel yang bersifat kuantitatif, seperti *Sales* atau *Days for Shipping (real)*. Selain itu, format *datetime* pada kolom seperti *order date (DateOrders)* dan *shipping date (DateOrders)* juga perlu diubah ke format standar untuk mempermudah pengolahan lebih lanjut, seperti ekstraksi informasi tahun, bulan, dan hari. Pemisahan ini mempermudah penerapan teknik praproses, seperti *label encoding* untuk data kategorikal dan standarisasi untuk data numerikal, sehingga menghasilkan representasi data yang lebih optimal untuk model *machine learning*.

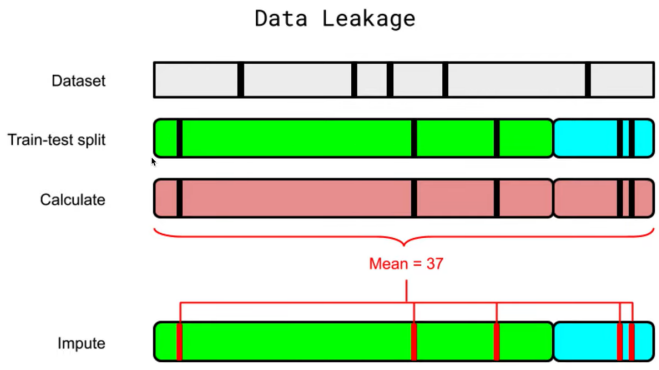

Fig. 9. Kebocoran Data

C. Mengolah Data Kategorikal dengan Label Encoding

Data kategorikal harus diubah menjadi format numerik agar dapat digunakan dalam model *machine learning*. Salah satu metode yang umum digunakan adalah *encoding*. *Encoding* merupakan proses merubah tipe data kategorik ke numerik sebelum diproses dengan algoritma *machine learning* [10]. Terdapat berbagai macam metode *encoding* seperti *one-hot encoding*, *ordinal encoding*, atau *label encoding*. Pada penelitian ini, kami menggunakan metode *label encoding* dengan mempertimbangkan banyaknya fitur kategorikal yang terdapat dalam *dataset* serta efisiensi waktu proses. Dengan *label encoding*, setiap kategori unik dalam suatu fitur akan diberi representasi numerik, seperti ”Completed”, ”Pending”, dan ”Cancelled” diubah menjadi 0, 1, dan 2.

TABLE II
CONTOH PROSES *Label Encoding* PADA FITUR KATEGORIKAL

Kategori Asli	Hasil *Label Encoding*
Completed	0
Pending	1
Cancelled	2

Setelah proses *label encoding* selesai, data kategorikal yang telah diubah menjadi numerik kemudian digabungkan (*concat*) dengan data numerik asli untuk membentuk satu *dataset* utuh yang siap digunakan dalam proses selanjutnya yaitu imputasi.

D. Imputasi Kolom yang Hilang

Imputasi merupakan salah satu pendekatan yang dapat digunakan untuk menangani data yang hilang dengan asumsi bahwa pola kehilangan data bersifat acak *(missing at random)* [12]. Teknik imputasi dapat dilakukan dengan berbagai metode, seperti menggunakan mean, median, modus, atau

pendekatan berbasis algoritma seperti *K-Nearest Neighbors Imputation (KNN)*, regresi, dan model probabilistik.

Fig. 10. *Imputation* [13]

Pada analisis ini, kami akan menggunakan metode *K-Nearest Neighbors Imputation* (KNN Imputer) untuk imputasi nilai yang hilang pada kolom *Customer Zipcode* yang memiliki 3 nilai *missing values*. Mengapa demikian, karena metode ini mampu mengestimasi nilai yang hilang berdasarkan pola dan kedekatan dengan nilai-nilai lainnya dalam *dataset*. KNN Imputer bekerja dengan cara mencari sejumlah tetangga terdekat (k), dalam hal ini 5 tetangga, berdasarkan jarak Euclidean, kemudian menghitung rata-rata dari nilai-nilai tetangga tersebut untuk menggantikan nilai yang hilang. Pendekatan ini lebih efektif dibandingkan metode sederhana seperti imputasi dengan mean atau median, terutama pada data yang memiliki variasi atau distribusi yang kompleks. Rumus untuk menghitung jarak Euclidean antara dua vektor x dan y, yang masing-masing memiliki n komponen:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3}$$

Keterangan:

- d adalah jarak Euclidean.
- x_i adalah komponen ke-i dari vektor x.
- y_i adalah komponen ke-i dari vektor y.
- n adalah jumlah dimensi dari vektor.

E. Standarisasi Data

Standarisasi data adalah proses transformasi data numerik agar memiliki nilai rata-rata (μ) sebesar 0 dan standar deviasi (σ) sebesar 1. Proses ini bertujuan untuk menghilangkan skala asli dari data sehingga semua fitur memiliki kontribusi yang setara dalam proses pelatihan model *machine learning*. Sebuah penelitian menunjukkan bahwa standarisasi data dapat meningkatkan kinerja algoritma *machine learning* dalam tugas-tugas klasifikasi dan prediksi [15]. Dalam standarisasi, setiap nilai data diubah menggunakan rumus berikut:

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

di mana x adalah nilai asli, μ adalah rata-rata, dan σ adalah standar deviasi dari fitur tersebut.

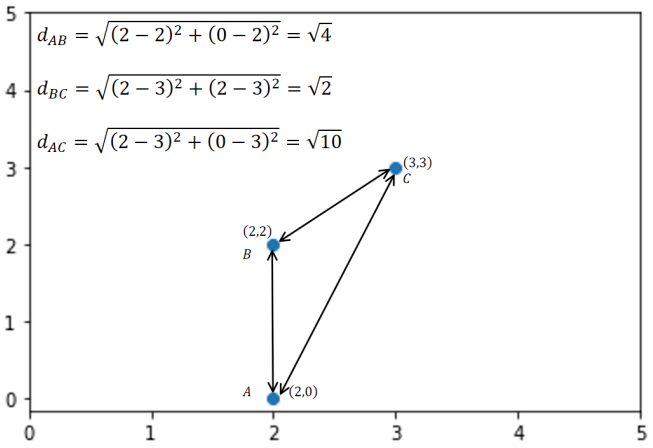

Fig. 11. *KNN Imputer* [14]

Pada penelitian ini, kami menggunakan *Library Standard-Scaler* dari *scikit-learn* untuk memastikan bahwa semua fitur numerik berada pada skala yang sama sebelum dilatih dengan model. Meskipun algoritma seperti *Decision Tree*, *Random Forest*, dan beberapa model berbasis pohon keputusan (*LightGBM*, *CatBoost*, dan *XGBoost*) tidak sensitif terhadap skala data, beberapa algoritma lain seperti *K-Nearest Neighbors (KNN)*, *SVM*, dan *Linear Discriminant Analysis (LDA)* sangat bergantung pada skala fitur untuk memberikan hasil yang optimal.

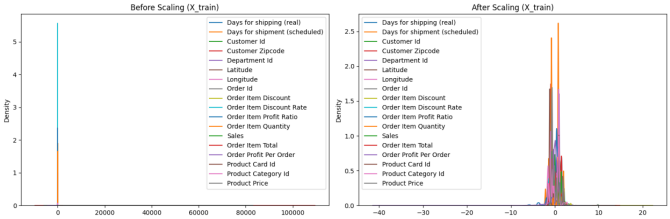

Fig. 12. *Xtrain Scaling*

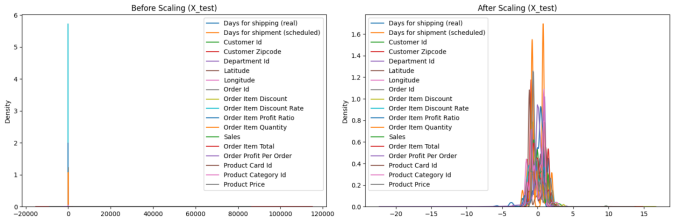

Fig. 13. *Xtest Scaling*

Sebelum dilakukan penskalaan fitur, data pada *X_train* dan *X_test* menunjukkan distribusi yang sangat tidak merata dengan rentang nilai yang besar, seperti terlihat pada sumbu X yang mencapai lebih dari 100.000. Hal ini mengindikasikan adanya fitur dengan skala yang sangat berbeda, yang dapat menyebabkan algoritma pemodelan menjadi bias terhadap fitur dengan nilai lebih besar.

Setelah dilakukan penskalaan fitur, distribusi data pada kedua set menjadi lebih terpusat di sekitar nol, dengan rentang nilai lebih seragam (sekitar -40 hingga 20). Proses *scaling* ini memastikan semua fitur berada pada skala yang sebanding, sehingga dapat meningkatkan kinerja model dan mempercepat konvergensi selama pelatihan.

F. Modelling

Tahap *modelling* merupakan langkah krusial dalam membangun sistem prediksi yang akurat dan andal. Proses ini melibatkan pemilihan algoritma yang paling sesuai dengan karakteristik data, evaluasi performa menggunakan metrik seperti *Accuracy*, *Precision*, *Recall*, dan *F1-Score* untuk memastikan model tidak hanya mampu membuat prediksi yang benar secara keseluruhan, tetapi juga seimbang dalam mengenali kategori minoritas maupun mayoritas. Pendekatan ini penting untuk menghindari ketidakseimbangan yang dapat merugikan, terutama pada data dengan distribusi kelas yang tidak merata. Kombinasi metrik dapat memberikan evaluasi yang lebih komprehensif dibandingkan penggunaan akurasi secara tunggal [16].

Sebelum tahap pemodelan, *baseline* akurasi dihitung untuk memberikan tolak ukur awal performa prediksi tanpa menggunakan algoritma pembelajaran mesin. Dalam kasus ini, *baseline* akurasi diperoleh dengan memprediksi semua data sebagai kelas mayoritas, yang menghasilkan akurasi sebesar 54%. Hal ini menggambarkan kinerja minimum yang harus dilampaui oleh model yang dibangun.

$$\textit{Baseline Accuracy} = \frac{\text{Jumlah Sampel Kelas Mayoritas}}{\text{Total Jumlah Sampel}}$$

```
Late_delivery_risk
1                    0.548293
0                    0.451707
Name: proportion, dtype: float64
```

Fig. 14. *Baseline Accuracy*

Untuk meningkatkan performa prediksi, kami menggunakan delapan algoritma *machine learning* yang dikelompokkan berdasarkan karakteristiknya. *Tree-based models*, seperti *Decision Tree*, *Random Forest*, *LightGBM*, *CatBoost*, dan *XGBoost*, memanfaatkan prinsip pembagian data berbasis pohon keputusan untuk membangun prediksi yang akurat. Model-model ini sangat efektif dalam menangani data yang besar dan kompleks serta dapat menangani fitur numerik maupun kategorikal. Selanjutnya, *distance-based models*, seperti *K-Nearest Neighbors* (KNN), memanfaatkan jarak antar data untuk memprediksi kelas target berdasarkan tetangga terdekat. Kami juga menggunakan *linear models*, seperti *Support Vector Machine* (SVM) dan *Linear Discriminant Analysis* (LDA), yang berfokus pada pemisahan kelas dengan pendekatan berbasis garis lurus atau *hyperplane*. Masing-masing algoritma dievaluasi untuk mengidentifikasi model dengan performa terbaik berdasarkan *Accuracy*, *Precision*, *Recall*, dan *F1-Score*.

- **Tree-Based Models.** Model berbasis pohon keputusan (tree-based models) adalah algoritma pembelajaran mesin bekerja dengan memecah *dataset* menjadi subset yang lebih kecil berdasarkan fitur-fitur tertentu, sehingga membentuk struktur seperti pohon yang terdiri dari node dan cabang.
  Tree-Based Model yang pertama yaitu *Decision Tree*. Salah satu algoritma pembelajaran mesin yang sederhana namun sangat efektif untuk tugas klasifikasi dan regresi. Algoritma ini memiliki kemampuan untuk menangkap hubungan nonlinear antara fitur dan target dengan cara membagi *dataset* menjadi subset berdasarkan aturan keputusan. Decision Tree digunakan dalam dataset ini karena algoritma ini mampu menangani data dengan berbagai tipe fitur, termasuk fitur numerik maupun kategorikal, yang cukup banyak dalam kasus ini.

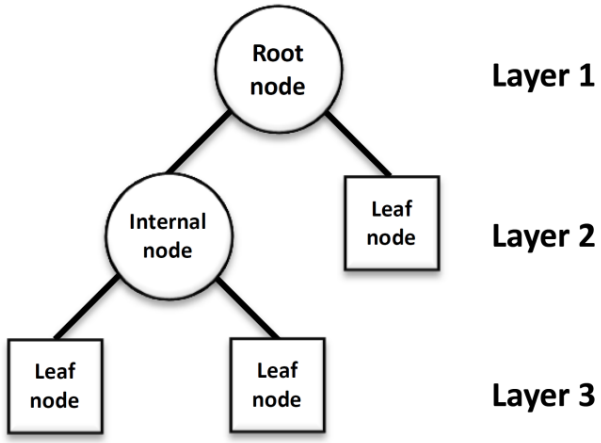

Fig. 15. *Decision Tree Model*

Tree-Based Model kedua yaitu *Random Forest*. Model ini digunakan untuk meningkatkan akurasi prediksi dengan cara menggabungkan banyak pohon keputusan (*ensemble learning*). Algoritma ini bekerja dengan membuat sejumlah pohon keputusan independen, di mana setiap pohon dilatih pada subset acak dari data dan fitur. Pendekatan ini sangat bermanfaat dalam analisis prediksi keterlambatan pengiriman karena mampu menangani data yang mungkin memiliki *outlier* atau pola non-linear.
Model *Tree-Based* ketiga adalah *LightGBM* (*Light Gradient Boosting Machine*), yang dirancang untuk menangani dataset besar dengan efisiensi tinggi. LightGBM menggunakan metode pembagian pohon berbasis *leaf-wise* alih-alih *level-wise*, sehingga memungkinkan proses pelatihan lebih cepat tanpa mengorbankan akurasi. LightGBM sangat cocok untuk memprediksi keterlambatan pengiriman karena mampu menangani fitur dengan nilai kategorikal, menangkap pola interaksi antar variabel, serta memberikan performa yang baik pada data dengan distribusi kelas tidak seimbang.

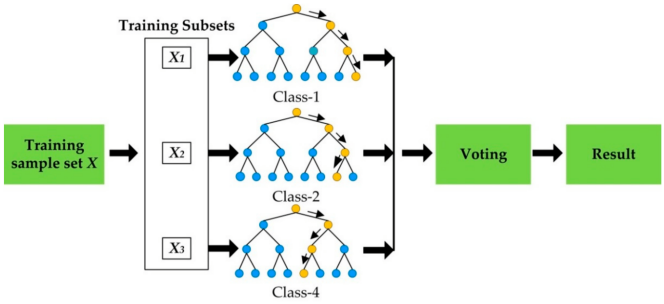

Fig. 16. *Random Forest Model*

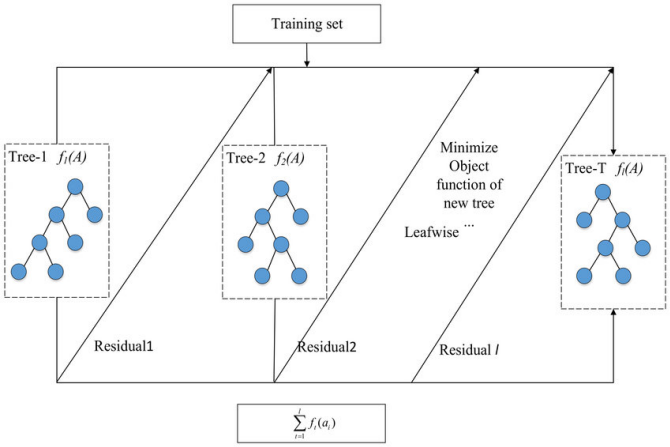

Fig. 17. *LightGBM Model*

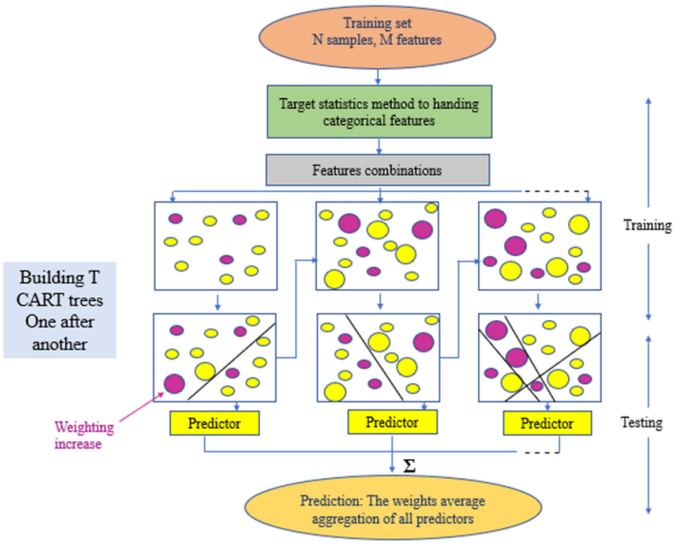

Fig. 18. *CatBoost Model*

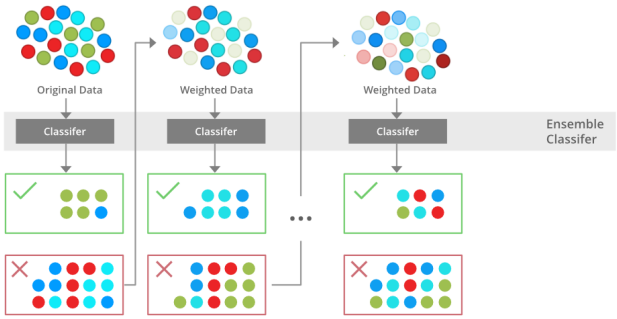

Fig. 19. *XGBoost Model*

Model *Tree-Based* keempat adalah *CatBoost* (*Categorical Boosting*), yang dirancang untuk menangani data kategorikal secara lebih efisien. CatBoost menggunakan teknik khusus, seperti *ordered boosting*, untuk mencegah *target leakage* selama pelatihan. Kemampuan CatBoost untuk memproses data kategorikal tanpa memerlukan *one-hot encoding* membantu meningkatkan efisiensi proses pelatihan dan mengurangi kompleksitas data. Selain itu, model ini dapat menangkap pola kompleks yang relevan untuk memprediksi keterlambatan pengiriman.

Model *Tree-Based* yang terakhir adalah *XGBoost* (*Extreme Gradient Boosting*) yang terkenal dengan efisiensi, fleksibilitas, dan akurasinya yang tinggi. Algoritma ini menggunakan pendekatan *gradient boosting* yang dioptimalkan, memanfaatkan teknik seperti *regularization* untuk mencegah *overfitting* dan *weighted quantile sketch* untuk menangani data dengan skala besar. Kemampuan XGBoost untuk menangani data dengan jumlah besar dan fitur yang beragam menjadikannya pilihan unggul untuk menyelesaikan masalah prediktif dalam konteks ini.

- **Distance-Based Model.** Model ini memanfaatkan jarak antar data untuk memprediksi kelas target berdasarkan tetangga terdekat. Satu model yang kami gunakan yaitu *K-Nearest Neighbors* (KNN). Algoritma KNN bekerja dengan mengelompokkan data berdasarkan tetangga terdekatnya dalam ruang multidimensi, sehingga memberikan prediksi yang didasarkan pada perilaku data yang serupa. Keunggulan KNN adalah kesederhanaannya dalam implementasi dan kemampuan adaptasinya pada *dataset* yang tidak memerlukan asumsi distribusi tertentu. Namun, algoritma ini juga memiliki kelemahan seperti sensitif terhadap skala data dan cenderung lebih lambat pada *dataset* besar. Untuk memastikan performa optimal, kami terapkan *standardize* terlebih dahulu sebelum diterapkan ke model KNN.
- **Linear Models.** Model ini berfokus pada pemisahan kelas dengan pendekatan berbasis garis lurus atau *hyperplane*. Kami menggunakan dua model utama dalam kategori ini, yaitu Support Vector Machine (SVM) dan Linear Discriminant Analysis (LDA).

  Support Vector Machine (SVM) adalah algoritma linear yang bekerja dengan mencari *hyperplane* optimal untuk memisahkan data dari dua kelas atau lebih. Dalam kasus ini, SVM sangat efektif karena kemampuannya untuk menangani data berdimensi tinggi dan menangani masalah klasifikasi yang kompleks. Algoritma ini diran-

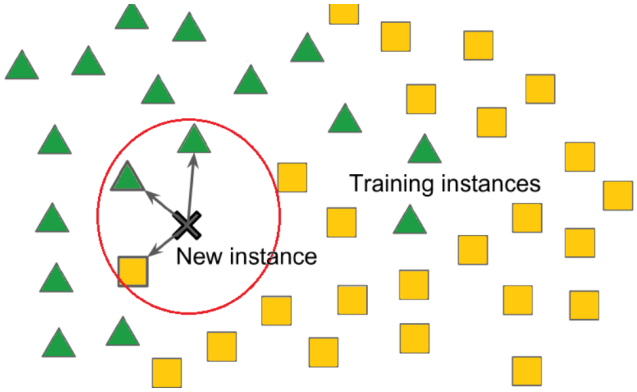

Fig. 20. *KNN Model*

cang untuk memaksimalkan margin, yaitu jarak antara *hyperplane* pemisah dengan data terdekat dari masing-masing kelas yang biasa disebut sebagai *support vectors*. Untuk memastikan performa optimal, kernel linear digunakan karena sifat data yang linear dalam domain *supply chain*.

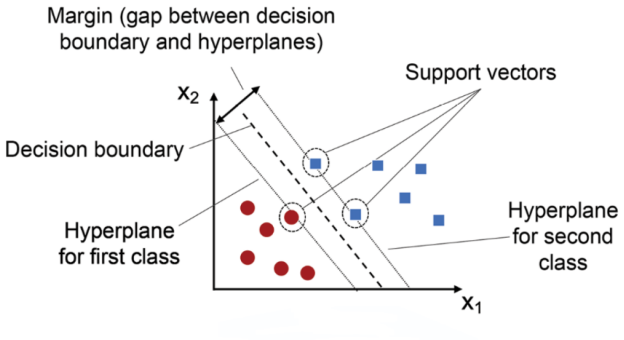

Fig. 21. *SVM Model*

Linear Discriminant Analysis (LDA) adalah algoritma berbasis statistik yang digunakan untuk klasifikasi dengan cara memaksimalkan separasi antar kelas. LDA bekerja dengan menemukan kombinasi linear dari fitur-fitur input yang paling memisahkan kelas target, sehingga menghasilkan proyeksi data yang optimal untuk klasifikasi. Dalam kasus ini, LDA sangat berguna karena kemampuannya untuk menangani data dengan distribusi Gaussian dan hubungan linier antar fitur.

Selain pendekatan *machine learning* klasik, kami juga menggunakan pendekatan *deep learning*. Pendekatan ini menggunakan arsitektur jaringan saraf tiruan (*artificial neural networks*) dengan banyak lapisan (*multilayer*), yang dikenal sebagai *deep neural networks*. Kami menggunakan salah satu model *deep learning* yang cukup hangat dibicarakan saat ini yaitu *Graph Neural Network* (GNN). Pemilihan GNN didasarkan pada kemampuan uniknya untuk menangkap pola relasional kompleks dalam data pengiriman, di mana hubungan

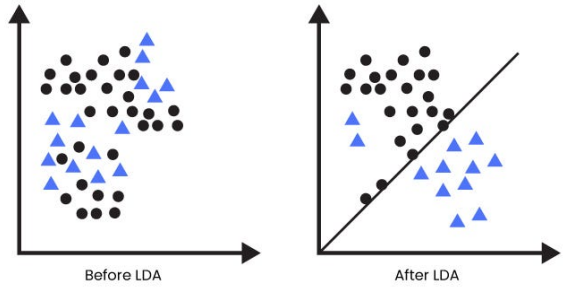

Fig. 22. *LDA Model*

antar entitas, seperti lokasi pengiriman, waktu pengiriman, dan pelanggan dapat direpresentasikan sebagai graf.

Convolution Formula:

$$S(i,j) = (K * I) = \sum_m \sum_n I(i+m, j+n) K(m,n) \quad (5)$$

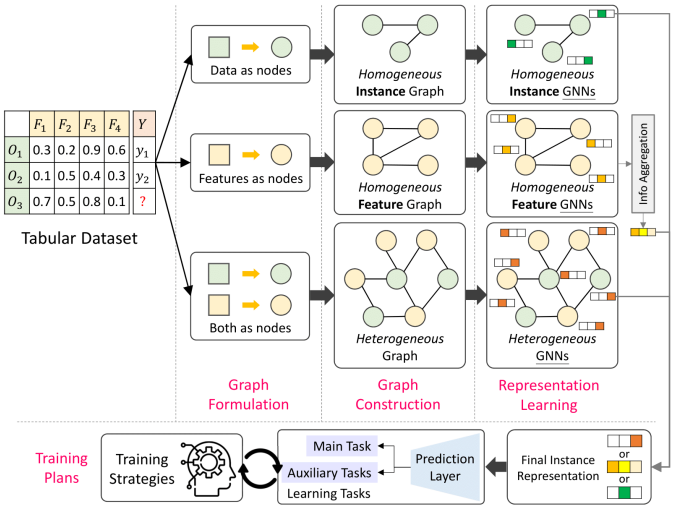

Fig. 23. *GNN Architecture*

Pada implementasi *Graph Neural Network* (GNN), kami memanfaatkan *Graph Convolutional Network* (GCN) sebagai subkelas utama untuk membangun model. GCN adalah salah satu jenis arsitektur GNN yang dirancang khusus untuk menangkap pola relasional antar node dalam graf melalui mekanisme propagasi pesan (*message passing*). Model GCN yang digunakan terdiri dari dua lapisan konvolusi. Lapisan pertama bertugas memproyeksikan fitur awal setiap node ke dimensi laten yang lebih representatif, sedangkan lapisan kedua digunakan untuk menghasilkan keluaran akhir berupa prediksi probabilitas untuk setiap kelas.

Sebagai bagian dari desain arsitektur *Graph Convolutional Network* (GCN) yang diterapkan pada data *supply chain*, kami menggunakan fungsi aktivasi *tanh* setelah setiap operasi konvolusi. Fungsi ini dipilih karena kemampuannya menangkap hubungan nonlinear yang kompleks dalam data, seperti pola keterlambatan pengiriman berdasarkan interaksi antara lokasi,

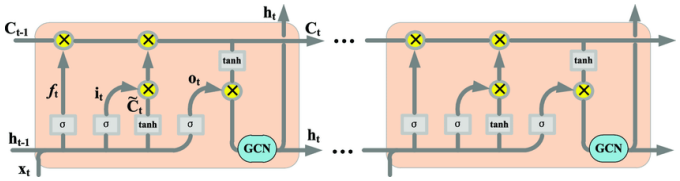

Fig. 24. *GCN Architecture*

waktu, and pelanggan. Aktivasi *tanh* menghasilkan nilai dalam rentang [-1,1], yang memungkinkan model untuk lebih sensitif terhadap variasi data, termasuk nilai-nilai ekstrem. 1,1], pendekatan ini masih masuk akal karena output jaringan dapat diinterpretasikan sebagai skor atau nilai aktivasi yang diubah menjadi probabilitas setelah mengaplikasikan ambang batas (*threshold*) untuk memutuskan apakah hasilnya adalah 0 atau 1.

Tanh Formula:

$$y = \tanh(x) = f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{6}$$

Sebagai bagian integral dari proses pelatihan model, kami menggunakan Binary Cross-Entropy (BCE) sebagai *Loss Function* atau fungsi kerugian. BCE dirancang khusus untuk tugas klasifikasi biner, di mana target kelas adalah 0 atau 1. Fungsi ini mengukur perbedaan antara probabilitas yang diprediksi oleh model dan nilai target sebenarnya, dengan memberikan penalti yang lebih besar pada prediksi yang jauh dari nilai sebenarnya. Dalam kasus ini, BCE membantu model mempelajari hubungan kompleks antar node pada graf *supply chain* dengan memberikan sinyal yang terarah untuk memperbarui bobot.

Binary Cross-Entropy Formula:

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\Big) \tag{7}$$

Penurunan nilai *loss* selama proses pelatihan menjadi indikator seberapa baik model mempelajari pola dari data. Dalam kasus ini, kami memantau perkembangan *training loss* menggunakan Binary Cross-Entropy (BCE) dengan logits selama 200 *epoch*. Hasil yang terlihat pada plot menunjukkan tren penurunan *loss*, meskipun terdapat beberapa fluktuasi pada tahap awal. Hal ini menunjukkan bahwa model secara bertahap mampu menyesuaikan parameter untuk menghasilkan prediksi yang lebih akurat sesuai target. Stabilitas nilai *loss* pada akhir pelatihan menunjukkan bahwa model telah mencapai titik konvergensi yang memadai untuk tugas klasifikasi biner.

G. Hyperparameter Tuning

Proses *hyperparameter tuning* dilakukan untuk menentukan kombinasi parameter terbaik yang dapat menghasilkan performa optimal pada model. *Hyperparameter tuning* adalah langkah eksplorasi nilai-nilai parameter eksternal model yang tidak diperbarui selama pelatihan, dilakukan untuk menentukan kombinasi parameter terbaik yang dapat menghasilkan

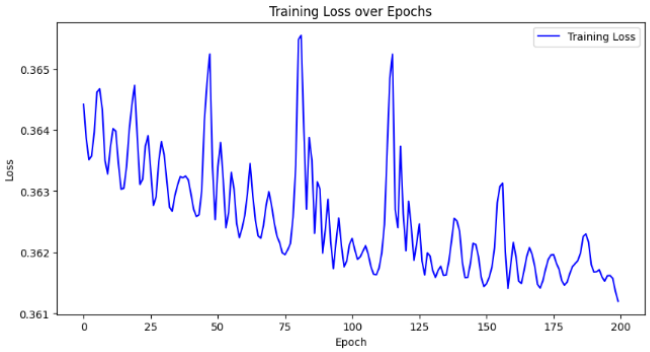

Fig. 25. *Training Loss BCEwithLogits*

performa optimal pada model. Dalam kasus ini, parameter-parameter seperti *learning rate* (LR), (*hidden dimension*), *weight decay* (WD), dan *dropout* telah diuji dengan berbagai nilai. Setelah serangkaian eksperimen, kombinasi terbaik diperoleh dengan nilai *learning rate* sebesar 0.05, *hidden dimension* sebesar 64, *weight decay* sebesar 0.0005, dan tanpa penerapan *dropout* (0.0). Konfigurasi ini dipilih karena menghasilkan *loss* yang stabil dan metrik evaluasi yang lebih baik, menunjukkan bahwa model mampu menangkap pola kompleks dari data dengan lebih efektif.

TABLE III
HASIL *Hyperparameter Tuning*

Hyperparameter	Nilai Terbaik
Learning Rate (LR)	0.05
Hidden Dimension	64
Weight Decay (WD)	0.0005
Dropout	0.0

H. Evaluasi Model

Tahap terakhir dalam penelitian ini adalah evaluasi model. Evaluasi model merupakan proses untuk menilai performa model yang telah dikembangkan dalam memprediksi *output* berdasarkan data masukan. Proses ini dilakukan menggunakan data uji, yaitu data yang tidak digunakan selama proses pelatihan untuk memastikan model dapat menggeneralisasi dengan baik pada data baru. Pada penelitian ini, evaluasi dilakukan dengan menggunakan metrik-metrik seperti *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Hasil evaluasi pada data uji menjadi indikator utama dalam menentukan keberhasilan pendekatan yang digunakan serta dasar untuk menyimpulkan efektivitas model yang telah dirancang. Evaluasi dibagi menjadi dua bagian, yaitu evaluasi untuk model *machine learning* (ML) dan model *deep learning* (DL).

- **Evaluasi Machine Learning.** Berdasarkan hasil evaluasi model pada data uji, performa berbagai algoritma *Machine Learning* dibandingkan menggunakan *Accuracy*, *Precision*, *Recall*, dan *F1-Score*.
  Confusion Matrix menunjukkan bahwa model *Decision Tree*, *Random Forest*, *LightGBM*, *CatBoost* dan *XGBoost*

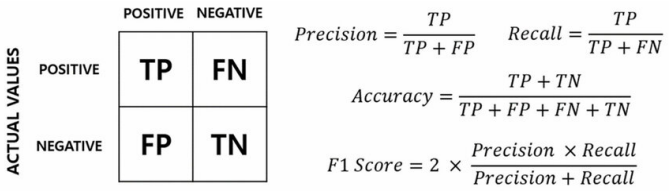

Fig. 26. *Confusion Matrix*

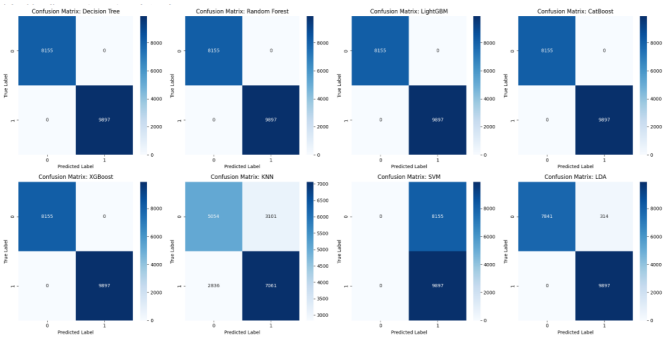

Fig. 27. *Confusion Matrix*

menghasilkan prediksi sempurna, ditandai dengan semua prediksi benar pada diagonal utama matriks dan elemen lainnya bernilai nol. Namun, prediksi sempurna ini mengindikasikan potensi *overfitting*, di mana model kemungkinan terlalu menyesuaikan dengan data latih sehingga kehilangan kemampuan generalisasi pada data baru.

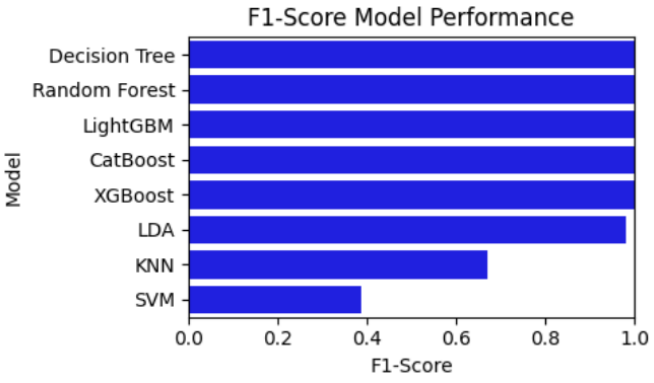

Fig. 28. *F-1 Score Chart*

Grafik *F1-Score* memperkuat temuan dari *Confusion Matrix*. Model *Decision Tree*, *Random Forest*, *LightGBM*, *CatBoost*, dan *XGBoost* mencapai nilai 1.0, yang menunjukkan keseimbangan sempurna antara presisi dan *recall*. Namun, hasil ini kemungkinan besar disebabkan oleh overfitting. Sementara itu, model seperti *LDA* memiliki nilai *F1-Score* mendekati 0.9, dan *KNN* serta *SVM* memiliki performa lebih rendah dengan nilai di bawah 0.7, tetapi mereka mungkin memiliki kemampuan generalisasi yang lebih baik.

TABLE IV
HASIL EVALUASI MODEL *Machine Learning*

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	1.000000	1.000000	1.000000	1.000000
Random Forest	1.000000	1.000000	1.000000	1.000000
LightGBM	1.000000	1.000000	1.000000	1.000000
CatBoost	1.000000	1.000000	1.000000	1.000000
XGBoost	1.000000	1.000000	1.000000	1.000000
KNN	0.671117	0.670320	0.671117	0.670573
SVM	0.548250	0.752328	0.548250	0.388280
LDA	0.982606	0.983141	0.982606	0.982571

Dari hasil ini, perlu dilakukan validasi lebih lanjut, seperti menggunakan data validasi yang benar-benar independen untuk memastikan bahwa model berbasis *ensemble learning* seperti *Random Forest*, *LightGBM*, *CatBoost* dan *XGBoost* benar-benar efektif dan tidak sekadar overfit terhadap data latih.

- **Evaluasi Deep Learning.**
  Evaluasi model ini dilakukan pertama kali disetel ke mode evaluasi, di mana fitur seperti *dropout* dinonaktifkan untuk menjaga konsistensi hasil. Selanjutnya, model memproses data uji dan menghasilkan nilai output berupa skor mentah dalam rentang $[-1, 1]$. Karena kami membutuhkan probabilitas dalam rentang $[0, 1]$, nilai tersebut diubah (*rescale*) dengan formula sederhana yang memetakan -1 menjadi 0 dan 1 menjadi 1.
  Rescale Formula:

$$\text{Rescale}(x) = \frac{x + 1}{2} \tag{8}$$

  Setelah itu, probabilitas yang dihasilkan diubah menjadi prediksi kelas dengan menetapkan batas (*threshold*) 0.5. Nilai di atas 0.5 dianggap kelas positif, sedangkan nilai di bawahnya dianggap negatif. Prediksi ini kemudian dibandingkan dengan label asli untuk menghitung metrik evaluasi, yaitu *Accuracy*, *Precision*, *Recall*, dan *F1 Score*.

TABLE V
HASIL EVALUASI MODEL DEEP LEARNING

Metrik	Nilai
Accuracy	0.9523
Precision	0.9506
Recall	0.9631
F1 Score	0.9568

I. Kesimpulan

Berdasarkan hasil pengujian model *Graph Convolutional Network* (GCN) dan algoritma *machine learning* lainnya, kami dapat memprediksi risiko keterlambatan pengiriman dengan akurasi yang tinggi. Model GCN berhasil menunjukkan metrik evaluasi yang kompetitif, dengan akurasi sebesar 95.23%, *precision* sebesar 95.06%, *recall* sebesar 96.31%, dan *F1-score* sebesar 95.68%. Meskipun demikian, terdapat beberapa kasus di mana model GCN belum mampu memprediksi dengan baik.

Kurang lebih 1% dari data pengujian menunjukkan prediksi keterlambatan yang salah secara signifikan. Hal ini

mengindikasikan bahwa model memprediksi pengiriman akan tiba tepat waktu, padahal kenyataannya terdapat keterlambatan yang cukup besar. Skenario seperti ini sangat krusial dalam konteks rantai pasok, terutama jika pengiriman tersebut berkaitan dengan barang yang mudah rusak atau bernilai tinggi. Keputusan operasional yang salah berdasarkan prediksi model dapat mengakibatkan kerugian finansial dan logistik yang besar.

Sebagai contoh, jika model memprediksi bahwa pengiriman akan tiba tepat waktu dan jadwal distribusi dilanjutkan berdasarkan asumsi tersebut, keterlambatan aktual dapat menyebabkan kekurangan stok di pusat distribusi atau bahkan kegagalan dalam memenuhi permintaan pelanggan. Sebaliknya, dalam skenario di mana model memprediksi keterlambatan yang terlalu tinggi (padahal pengiriman sebenarnya tepat waktu), pengalokasian ulang sumber daya dapat terjadi secara tidak efisien, yang juga berpotensi menambah biaya operasional.

Oleh karena itu, rekomendasi terbaik dalam situasi seperti ini adalah mengintegrasikan hasil prediksi model dengan pengambilan keputusan berbasis risiko yang lebih hati-hati, seperti menerapkan langkah mitigasi tambahan untuk pengiriman berisiko tinggi.

REFERENCES

[1] S. A. Musyarofah, A. E. Tontowi, N. A. Masruroh, and B. S. Wibowo, "Developing supply chain readiness measurement tool for the manufacturing industrial estates," vol. 9, Mar. 2023.

[2] A. S. Nugraha, M. A. Romadhon, N. Triningsih, and M. S. Widodo, "ANALISIS DAMPAK KEMACETAN LALU LINTAS TERHADAP EFISIENSI DISTRIBUSI LOGISTIK DI KAWASAN METROPOLITAN," Oct. 2024.

[3] Asbullah, D. Ginting, and Suparman, "Analisis Keterlambatan dan Efisiensi Kegiatan Bongkar Muat Petikemas Di Terminal PT. Prima Terminal Petikemas Belawan," vol. 4, 2024.

[4] Qlik, "Supply Chain Analytics." Accessed: Mar. 27, 2025. [Online]. Available: https://www.qlik.com/us/data-analytics/supply-chain-analytics

[5] Supardianto, L. Mutawalli, and W. Murniati, "PENERAPAN KNNIMPUTER DALAM MENGOLAH DATA MISSING VALUE UNTUK MEMBANTU MENINGKATKAN AKURASI SUPPORT VECTOR MACHINE KLASIFIKASI PENYAKIT TIROID," vol. 4, 2022.

[6] A. A. Dewayanti and H. Utami, "ESTIMASI ROBUST PADA MODEL REGRESI UNTUK MENANGANI OUTLIER DAN HETEROSKEDASTISITAS," vol. 3, 2021.

[7] K. Maharana, S. Mondal, and B. Nemade, "A review: Data preprocessing and data augmentation techniques," vol. 3, pp. 91–99, Jun. 2022.

[8] R. T. Handayanto, "Membagi Data Latih dan Uji Secara Otomatis Pada Python."

[9] geeksforgeeks, "What is Scikit-learn Random State in Splitting Dataset?"

[10] N. Alturayeif and J. Hassine, "Data leakage detection in machine learning code: transfer learning, active learning, or low-shot prompting?," 2025.

[11] Kristiawan and A. Widjaja, "Perbandingan Algoritma Machine Learning dalam Menilai Sebuah Lokasi Toko Ritel," vol. 7, Mar. 2021.

[12] D. B. Rubin, "Inference and missing data," vol. 63, no. 3, pp. 581–592, Dec. 1976.

[13] Kaushikrch, "KNNImputer: A robust way to impute missing values (using Scikit-Learn)."

[14] A. Ambarwari, Q. J. Adrian, and Y. Herdiyeni, "Analisis Pengaruh Data Scaling Terhadap Performa Algoritme Machine Learning untuk Identifikasi Tanaman," Feb. 2020.

[15] S. Kumar, "Evaluation Metrics For Classification Model."